# Data Science Al Workbench Documentation

Release 5.7.1

Anaconda Inc.

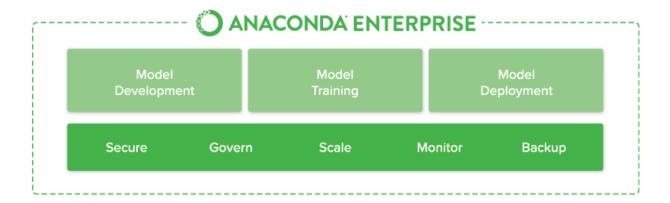
# **CONTENTS**

1	Preparing an environment for Workbench	3
2	Installing Workbench	39
3	Administering Workbench	73
4	Using Workbench	237
5	Troubleshooting	351
6	Reference materials	359

Workbench is an enterprise-ready, secure and scalable data science platform that empowers teams to govern data science assets, collaborate and deploy their data science projects.

With Workbench, you can do the following:

- Develop: ML/AI pipelines in a central development environment that scales from laptops to thousands of nodes
- Govern: Complete reproducibility from laptop to cluster with the ability to configure access control
- Automate: Model training and deployment on scalable, container-based infrastructure



CONTENTS 1

2 CONTENTS

# PREPARING AN ENVIRONMENT FOR WORKBENCH

This section provides guidance on preparing your environment before you install Data Science & AI Workbench.

Anaconda's instructions assume that the cluster itself has already been provisioned. Our preparation guide provides you with the details you need to verify that your cluster meets the basic requirements to operate Workbench, and to provision the storage and networking resources required to run the application.

Preparing your Kubernetes environment for Workbench.

Existing customers who rely on Anacodna's Gravity Kubernetes stack can find instructions for *preparing their envi*ronment for installation here.

# 1.1 Preparing a BYOK8s environment for Workbench

Determining the resource requirements for a Kubernetes cluster depends on a number of different factors, including what type of applications you are going to be running, the number of users that are active at once, and the workloads you will be managing within the cluster. *Data Science & AI Workbench's performance is tightly coupled with the health of your Kubernetes stack*, so it is important to allocate enough resources to manage your users workloads.

Anaconda's hardware recommendations ensure a reliable and performant Kubernetes cluster. However, most requirements are likely to be superseded by the requirements imposed by your existing Kubernetes cluster, whether that is an on-premise cluster that is configured to support multiple tenants or a cloud offering.

To install Workbench successfully, your systems must meet or exceed the requirements listed below. Anaconda has created a pre-installation checklist to help prepare you for installation. The checklist helps you verify that your cluster is ready to install Workbench, and that the necessary resources are reserved. Anaconda's Implementation team will review the checklist with you prior to your installation.

# 1.1.1 Supported Kubernetes Versions

Workbench is compatible with Kubernetes API versions 1.15-1.28. If your version of Kubernetes utilizes the API at these versions, you can install Workbench!

Workbench has been successfully installed on the following Kubernetes variants:

#### **On-Premise variants**

- · Vanilla Kubernetes
- · VMWare Tanzu
- · RedHat OpenShift
- · Google Anthos

#### **Cloud variants**

- Amazon Elastic Kubernetes Service (EKS)
- Microsoft Azure Kubernetes Service (AKS)
- Google Kubernetes Service (GKE)
- RedHat OpenShift on AWS (ROSA)

Aside from the basic requirements listed on this page, Anaconda also offers *environment-specific recommendations* for you to consider.

#### 1.1.2 Administration Server

Installation requires a machine with direct access to the target Kubernetes cluster and Docker registry. Anaconda refers to this machine as the Administration Server. Anaconda recommends that you identify a machine to be your Administration Server that will remain available for ongoing management of the application once installed. It is useful for this server to be able to mount the storage volumes as well.

The following software must be installed on the Administration Server:

- Helm version 3.2+
- The Kubernetes CLI tool kubect1
- (OpenShift only) The OpenShift oc CLI tool
- (Optional) The watch command line tool
- (Optional) The jq command line tool

#### **Administration Server setup**

You can obtain all of the tools you need for your Administration Server by installing the ae5-conda environment. This environment already contains helm, kubectl, oc, jq, and a number of other useful Workbench management utilities. To install the environment:

- 1. Download the environment.
- 2. If necessary, move the environment to the Administration Server.
- 3. Open a terminal shell and install the environment by running the following command:

```
bash ae5-conda-latest-Linux-x86_64.sh
```

- 4. Follow the prompts, then restart your terminal.
- 5. (Optional) Add the environment to your PATH.

## 1.1.3 CPU, memory, and nodes

Minimum node size: 8 CPU cores, 32GB RAM

• Recommended node size: 16 CPU cores, 64GB RAM (or more)

• Recommended oversubscription (limits/requests ratio) 4:1

• Minimum number of worker nodes: 3

Minimally sized nodes should be reserved for test environments with low user counts and small workloads. Any development or production environment should meet or exceed the recommended requirements.

Workbench utilizes node labels and taints and tolerations to ensure that workloads run on the appropriate nodes. Anaconda recommends identifying any necessary affinity (which nodes a workload runs on based on its applied labels) or toleration settings prior to installation.

# 1.1.4 Resource profiles

Resource profiles available to platform users for their sessions and deployments are created by the cluster administrator. Each resource profile can be customized for the amount of CPU, memory, and (optionally) GPU resources available. Anaconda recommends determining what resource profiles you will require prior to installation. For more information, see *Configuring workload resource profiles*.

# 1.1.5 Namespace, service account, RBAC

Workbench should be installed in a namespace that is *not occupied* by any other applications, including other instances of Workbench. Create a service account for the namespace with sufficient permissions to complete the helm installation and enable the dynamic resource provisioning Workbench performs during normal operation.

Workbench requires more permissions than would normally be given to an application that only requires read-only access to the Kubernetes API. However, with the exception of the ingress controller, all necessary permission grants are limited to the application namespace. Please speak with the Anaconda Implementation team about any questions you may have regarding these permissions.

#### **RBAC** template

The following Role and RoleBinding pair can be used to grant sufficient permissions to the Service Account.

```
- pods/log
    - events
- verbs:
   - create
   - delete
   - get
   - list
   - patch
   - update
    - watch
apiGroups:
   _ ''
resources:
   - configmaps
   - secrets
   - pods
    - persistentvolumeclaims
    - endpoints
   - services
- verbs:
   - create
    - delete
   - get
   - list
   - patch
   - update
    - watch
apiGroups:
    - apps
resources:

    deployments

    - replicasets
    - statefulsets
- verbs:
   - create
   - delete
   - get
   - list
   - patch
    - update
    - watch
apiGroups:
    - batch
resources:
    - jobs
   - cronjobs
- verbs:
    - create
   - delete
    - get
    - list
    - patch
```

```
- update
    - watch
apiGroups:
    - extensions
resources:
    - deployments
    - replicasets
- verbs:
   - create
    - delete
   - get
   - list
    - patch
    - update
    - watch
apiGroups:
    - networking.k8s.io
resources:
    - ingresses
- verbs:
   - create
    - delete
   - get
   - list
    - patch
   - update
    - watch
apiGroups:
    - route.openshift.io
resources:
   - routes
    - routes/custom-host
- verbs:
    - get
    - list
apiGroups:
    _ ''
resources:
   - serviceaccounts
    - roles
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
name: <SERVICE_ACCOUNT>
namespace: <NAMESPACE>
roleRef:
apiGroup: rbac.authorization.k8s.io
kind: Role
name: <SERVICE_ACCOUNT>
subjects:
- kind: ServiceAccount
```

```
name: <SERVICE_ACCOUNT>
```

**Note:** Recent versions of OpenShift no longer allow granting direct access to the anyuid Security Context Constraint (SCC), or any other default SCC. Instead, access grants are defined within the role.

## **Example anyuid SCC configuration**

```
- verbs:
    - use
apiGroups:
    - security.openshift.io
resources:
    - securitycontextconstraints
resourceNames:
    - anyuid
```

If you wish to use the Anaconda-supplied ingress, it is also necessary to grant a small number of additional, cluster-wide permissions. This is because the ingress controller expects to be able to monitor ingress-related resources across all namespaces.

#### Ingress controller permissions

The following is a minimal ClusterRole and ClusterRoleBinding pair that has grants the ingress controller sufficient permissions to run without warnings:

```
# Replace <SERVICE_ACCOUNT> with the name of your ingress service account
# Replace <NAMESPACE> with the namespace reserved for Workbench
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: <SERVICE_ACCOUNT>-ingress
rules:
    - verbs:
        _ '*'
    apiGroups:
        _ '*'
    resources:
        - ingressclasses
    - verbs:
        - patch
    apiGroups:
        _ '*'
    resources:
        - events
    - verbs:
        - list
        - watch
    apiGroups:
```

```
_ '*'
    resources:
        - secrets

    endpoints

        - ingresses
        - services
        - pods
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: <SERVICE_ACCOUNT>-ingress
roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: <SERVICE_ACCOUNT>-ingress
subjects:
    - kind: ServiceAccount
    name: <SERVICE_ACCOUNT>
    namespace: <NAMESPACE>
```

**Caution:** Please review these RBAC configurations with your Kubernetes administrator. While it is *possible* to further reduce these scopes, doing so is likely to prevent normal operation of Workbench.

# 1.1.6 Security

Preparing your Kubernetes cluster to install Workbench involves configuring your environment in a way that both supports the functionality of Workbench and adheres to security best practices.

Workbench containers can be run using any fixed, non-zero UID, making the application compatible with an OpenShift Container Platform (OCP) restricted SCC, or an equivalent non-permissive Kubernetes security context. This reduces the risk to your systems if the container is compromised.

However, in order to enable the Authenticated Network File System (NFS) capability, allowing user containers to access external, authenticated fileshares (storage servers), user pods must be permitted to run as root (UID 0).

**Note:** This configuration runs containers in a privileged state to determine and assign the authenticated group memberships of the user running the container only. Once authentication is complete, the container drops down to a non-privileged state for all further execution.

Please speak with your Anaconda Implementation team for more information, and to see if it is possible for your application to avoid this requirement.

# 1.1.7 Storage

A standard installation of Workbench requires two Persistent Volume Claims (PVCs) to be *statically provisioned and bound prior to installation*. Anaconda strongly recommends a premium performance tier for provisioning your PVCs if the option is available. Expand the following sections for more information about the necessary volumes:

#### anaconda-storage

This volume contains:

- · Anaconda's internal Postgres control database
- · Anaconda's internal Git storage mechanism
- · Anaconda's internal conda package repository

**Note:** If you are hosting conda packages outside of Workbench, the minimum size of your anaconda-storage volume must be at least *100GiB*.

However, if you intend to mirror conda packages into the Workbench repository, the anaconda-storage volume will need to be much larger to accommodate those packages. Anaconda recommends *at least 500GiB of storage*.

The anaconda-storage volume must support either the ReadWriteOnce or ReadWriteMany access mode.

**Caution:** The ReadWriteOnce configuration requires that the postgres, git-storage, and object-storage pods run on the same node.

#### anaconda-persistence

This volume hosts Anaconda's managed persistence storage. It contains:

- Custom sample projects
- · Custom conda environments
- · User code and data

The anaconda-persistence volume requires ReadWriteMany access.

**Caution:** The demands on the anaconda-persistence volume will continuously grow with usage. Anaconda recommends you provision *at least 1TiB of storage to start*.

It is possible to combine these into a single PersistentVolumeClaim to cover both needs, as long as that single volume simultaneously meets the performance needs demanded by both.

The root directories of these storage volumes must be writable by Workbench containers. This can be accomplished by configuring the volumes to be group writable by a single numeric GroupID (GID). Anaconda strongly recommends that this GID be 0. This is the default GID assigned to Kubernetes containers. If this is not possible, supply the GID within the Persistent Volume specification as a pv.beta.kubernetes.io/gid annotation.

**Caution:** To ensure that the data on these volumes is not lost if Workbench is uninstalled, do not change the ReclaimPolicy from its default value of Retain.

# 1.1.8 Ingress and firewall

Workbench is compatible with most ingress controllers that are commonly used with Kubernetes clusters. Because ingress controllers are a cluster-wide resource, Anaconda recommends that the controller be installed and configured prior to installing Workbench. For example, if your Kubernetes version falls within 1.19-1.26, any ingress controller with full support for the networking.k8s.io/v1 ingress API enables Workbench to build endpoints for user sessions and deployments.

**Note:** If your cluster is fully dedicated to Workbench, you can configure the Helm chart to install a version of the NGINX Ingress controller, which is compatible with multiple variants of Kubernetes, including OpenShift. Anaconda's only modification to the stock NGNIX container enables it to run without root privileges.

Your cluster configuration and firewall settings must allow all TCP traffic between nodes, particularly HTTP, HTTPS, and the standard Postgres ports.

**Caution:** Healthy clusters can block inter-node communication, which disrupts the pods that Workbench requires to provision user workloads.

External traffic to Workbench will be funneled entirely through the ingress controller, through the standard HTTPS port 443.

#### 1.1.9 **DNS/SSL**

Workbench requires the following:

- A valid, fully qualified domain name (FQDN) reserved for Workbench.
- A DNS record for the FQDN, as well as a wildcard DNS record for its subdomains.

**Note:** Both records must point to the IP address allocated by the ingress controller. If you are using an existing ingress controller, you may be able to obtain this address prior to installation. Otherwise, you must populate the DNS records with the address after the initial installation is complete.

• A valid *wildcard* SSL certificate covering the cluster FQDN and its subdomains. Installation requires both the public certificate and the private key.

**Note:** If the certificate chain includes an intermediate certificate, the public certificate for the intermediate is required. The *scope* of the wildcard only requires \*.anaconda.company.com to be covered, ensuring that all subdomains under this specific domain are included in the SSL certificate and DNS configuration.

• The public root certificate, if the above certificates were created with a private Certificate Authority (CA).

**Warning:** Wildcard DNS records and SSL certificates are *required* for correct operation of Workbench. If you have any objections to this requirement, speak with your Anaconda Implementation team.

# 1.1.10 Docker images

Anaconda strongly recommends that you copy the Workbench Docker images from our authenticated source repository on Docker Hub into your internal docker registry. This ensures their availability even if there is an interuption in connectivity to Docker Hub. This registry must be accessible from the Kubernetes cluster where Workbench is installed.

**Caution:** In an airgapped setting, this is *required*.

# Docker images from the aedev/ Docker Hub channel

Anaconda provides a more precise manifest, including version numbers and the credentials required to pull these images from our authenticated repository, prior to installation.

```
ae-app-proxy
ae-auth
ae-auth-api
ae-auth-escrow
ae-deploy
ae-docs
ae-editor
ae-git-proxy
ae-git-storage
ae-object-storage
ae-operation-controller
ae-operation-create-project
ae-repository
ae-storage
ae-sync
ae-ui
ae-workspace
ae-wagonwheel
ae-auth-keycloak
ae-nginx-ingress-v1
ae-nginx-ingress-v1beta1
postgres:12.17 # This is the only postgres image certified for use with.
--Workbench
```

**Note:** Docker images used by Workbench are larger than many Kubernetes administrators are accustomed to. For more background, see *Docker image sizes*.

# 1.1.11 GPU Information

This release of Workbench supports up to Compute Unified Device Architecture (CUDA) 11.6 DataCenter drivers.

Anaconda has directly tested the application with the following GPU cards:

- Tesla V100 (recommended)
- Tesla P100 (adequate)

Theoretically, Workbench will work with any GPU card compatible with the CUDA drivers, as long as they are properly installed. Other cards supported by CUDA 11.6:

- A-Series: NVIDIA A100, NVIDIA A40, NVIDIA A30, NVIDIA A10
- RTX-Series: RTX 8000, RTX 6000, NVIDIA RTX A6000, NVIDIA RTX A5000, NVIDIA RTX A4000, NVIDIA T1000, NVIDIA T600, NVIDIA T400
- HGX-Series: HGX A100, HGX-2
- T-Series: Tesla T4
- P-Series: Tesla P40, Tesla P6, Tesla P4
- K-Series: Tesla K80, Tesla K520, Tesla K40c, Tesla K40m, Tesla K40s, Tesla K40st, Tesla K40t, Tesla K20Xm, Tesla K20m, Tesla K20s, Tesla K20c, Tesla K10, Tesla K8
- M-Class: M60, M40 24GB, M40, M6, M4

Support for GPUs in Kubernetes is still a work in progress, and each cloud vendor provides different recommendations. For more information about GPUs, see *Understanding GPUs*.

#### 1.1.12 Helm chart

Workbench includes a values.yaml override file with the Helm chart. Make additions and modifications to this file with your cluster configurations now, then complete the helm install or upgrade process to update the default Helm chart.

#### Helm values.yaml template

**Note:** This template contains comments to guide you through the parameters that require modification or addition.

```
# The fully qualified domain name (FQDN) of the cluster
hostname: anaconda.example.com

# The name of the service account with the necessary
# permission to operate the cluster
serviceAccountName: anaconda-enterprise

# The UID under which to run the containers
uid: 1000

# Docker registry information
image:
    # Server address. This default points to Docker Hub
```

```
server: 'aedev/'
    # A pull secret name, or list of names
    pullSecrets:
# TLS / SSL secret management
# - generate: generate self-signed certificates
# - load: use the certificates in Anaconda-Enterprise/certs
# - skip: do nothing; assume the secrets already exist
# Existing secrets are always preserved during upgrades.
generateCerts: generate
# Keycloak LDAPS Settings
# truststore: path to your truststore file containing custom CA cert
# truststore_password: password of the truststore
# truststore_seret: name of secret used for the truststore such as anaconda-
→enterprise-truststore
keycloak:
    truststore: ""
   truststore_password: ""
    truststore_secret: ""
ingress:
    # If an existing ingress controller is being used, this
    # must match the ingress.className of that controller.
    className: nginx
    # - false: an existing ingress controller will be used
    # - true: install an ingress controller in this namespace
    install: false
    # - false: an existing IngressClass resource will be used
    # - true: create a new IngressClass in the global namespace
    # Ignored if ingress.install is false. Note that installing
    # the IngressClass requires cluster-level permissions.
    installClass: false
    # If your ingress controller requires custom labels to be
    # added to ingress entries, list them here as a dictionary
    # of key/value pairs.
    labels: {}
    # If your ingress requires custom annotations to be added
    # to ingress entries, they can be included here. These
    # will be added to any existing annotations in the chart.
    # For all ingress entries
    global: {}
    # For the master ingress only
    # For sessions and deployments only
    user: {}
# As discussed in the documentation, you may use the same
# persistent volume for both storage resources. If so, make
# sure to use the same same pvc: value in both locations.
storage:
   pvc: anaconda-storage
```

```
persistence:
    pvc: anaconda-persistence
# OPENSHIFT ONLY: uncomment this value for OCP platforms.
# dnsServer: dns-default.openshift-dns.svc.cluster.local
# TOLERATIONS / AFFINITY
# Please work with the Anaconda team for assistance
# to configure these settings if you need them.
tolerations:
    # For all pods
    global: []
    # For system pods, except the ingress
    system: []
    # For the ingress daemonset alone
    ingress: []
    # For user pods
   user: []
affinity:
    # For all pods
    global: {}
    # For system pods, except the ingress
    system: {}
    # For the ingress daemonset alone
    ingress: {}
    # For user pods
    user: {}
```

## 1.1.13 Pre-installation checklist

Anaconda has created this pre-installation checklist to help you verify that you have properly prepared your environment prior to installation.

Within this checklist, Anaconda provides some commands or command templates for you to run in order to verify a given requirement, along with a typical output from the command to give you an idea of the kind of information you should see. Run each of these commands, (modified as appropriate for your environment) and copy the outputs into a document. Send this document to your Anaconda implementation team so that they can verify your environment is ready before you begin the installation process.

#### BYOK8s pre-installation checklist

Verify that your *administration server* has been provisioned with appropriate versions of kubectl, helm, and other tools needed to perform installation and administration tasks by running the following command:

```
helm version
```

Here is an example response from the command:

Verify that the *API version* of the Kubernetes cluster is between 1.15 and 1.28 by running the following command:

```
kubectl version
```

Here is an example response from the command:

```
Client Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.

12", GitCommit:"e2a822d9f3c2fdb5c9bfbe64313cf9f657f0a725",

GitTreeState:"clean", BuildDate:"2020-05-06T05:17:59Z", GoVersion:

"go1.12.17", Compiler:"gc", Platform:"linux/amd64"}

Server Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.

12", GitCommit:"e2a822d9f3c2fdb5c9bfbe64313cf9f657f0a725",

GitTreeState:"clean", BuildDate:"2020-05-06T05:09:48Z", GoVersion:

"go1.12.17", Compiler:"gc", Platform:"linux/amd64"}
```

Verify all nodes nodes on which Workbench will be installed have *sufficient CPU and memory allocations* by running the following command:

Here is an example response from the command:

```
10.234.2.18 16 65806876Ki
10.234.2.19 16 65806876Ki
10.234.2.20 16 65806876Ki
10.234.2.21 16 65806876Ki
10.234.2.6 16 65974812Ki
```

Verify that the *namespace* where Workbench will be installed has been created by running the following command:

```
\# Replace <NAMESPACE> with the name
space you've reserved for Workbench kubectl describe name
space <NAMESPACE>
```

Here is an example response from the command:

```
Name: default
Labels: <none>
Annotations: <none>
Status: Active
No resource quota.
No resource limits.
```

Verify the *service account* that will be used by Workbench during installation and operation has been created by running the following command:

```
# Replace <SERVICE_ACCOUNT> with the name of the service account you
         've created for Workbench
kubectl describe sa <SERVICE_ACCOUNT>
```

Here is an example response from the command:

Name: anaconda-enterprise Namespace: default Labels: <none> Annotations: <none> Image pull secrets: <none> Mountable secrets: anaconda-enterprise-token-cdmnf anaconda-enterprise-token-cdmnf Tokens: Events: <none>

(*Openshift Only*) Verify the Security Context Constraint (SCC) associated with the service account contains all of the necessary permissions by running the following command:

```
oc describe scc <SCC_NAME>
```

Here is an example response from the command:

Name: anyuid
Priority: 10
Access:
Users: <none>
Groups: system:cluster-admins

**Note:** This example uses the anyuid SCC; however, the restricted SCC can also be used, as long as the uid range is known.

Verify the ClusterRole resource associated with the service account has the necessary permissions to facilitate installation and operation by running the following command:

```
# Replace <SERVICE_ACCOUNT> with the name of the service account you
→'ve created for Workbench
kubectl describe clusterrole <SERVICE_ACCOUNT>-ingress
```

Here is an example response from the command:

Name:	anaconda-enterpri	se-ingress	
Labels:	app.kubernetes.io/managed-by=Helm		
	skaffold.dev/run-id=8d38b94a-ab82-49d7-a6fd-0bc0fb549d1c		
Annotations:	meta.helm.sh/release-name: anaconda-enterprise		
	meta.helm.sh/release-namespace: default		
PolicyRule:		-	
Resources	Non-Resource URLs	Resource Names	Verbs
* *	[]	[]	[*]
	[*]		[*]

**Note:** The above example is fully permissive. See the *RBAC template* for more realistic configurations.

A numeric UID that will be used to run Workbench containers is reserved.

**Note:** Include the UID in your checklist results.

Verify that GID 0 is permitted by the security context.

Verify any *tolerations and/or node labels* required to permit Workbench to run on its assigned nodes have been identified by running the following command:

Verify that a Persistent Volume Claim (PVC) has been created within the application namespace, referencing a statically provisioned Persistent Volume that meets the *storage requirements* for the anaconda-storage volume.

Command: kubectl describe pvc anaconda-storage:

Name: anaconda-storage

Namespace: default

StorageClass: anaconda-storage

Status: Bound

Volume: anaconda-storage

Labels: <none>

Annotations: pv.kubernetes.io/bind-completed: yes

pv.kubernetes.io/bound-by-controller: yes

Finalizers: [kubernetes.io/pvc-protection]

Capacity: 500Gi Access Modes: RWO

VolumeMode: Filesystem

Mounted By: anaconda-enterprise-ap-git-storage-6658575d6f-vxj4s

anaconda-enterprise-ap-object-storage-76bcfc4d44-ctlhp

anaconda-enterprise-postgres-c76869799-cbqzq

Events: <none>

Verify a PVC has been created within the application namespace, referencing a statically provisioned Persistent Volume that meets the *storage requirements* for the anaconda-persistence volume by running the following command:

```
kubectl describe pvc anaconda-persistence
```

Here is an example response from the command:

Name: anaconda-persistence

Labels: <none>

Annotations: pv.kubernetes.io/bound-by-controller: yes

Finalizers: [kubernetes.io/pv-protection]

StorageClass:

Status: Bound

Claim: default/anaconda-persistence

Reclaim Policy: Retain
Access Modes: RWX
VolumeMode: Filesystem

Capacity: 500Gi

Node Affinity: <none>

Message: Source:

Type: NFS (an NFS mount that lasts the lifetime of a pod)

Server: 10.234.2.7

Path: /data/persistence

ReadOnly: false
Events: <none>

The cluster is sized appropriately (CPU / Memory) for user workload, including consideration for "burst" workloads. For more information, see *Understanding Workbench system requirements*.

Resource Profiles have been determined, and created in the values.yaml file.

A domain name for the Workbench application has been identified.

**Note:** Please include this domain name in your checklist output.

If you are supplying your own ingress controller, verify it has already been installed, and its master IP address and ingressClassName value have been identified.

Note: Please include both the IP address ingress class name in your checklist output.

Verify the DNS records for both anaconda.example.com and \*.anaconda.example.com have been created and point to the IP address of the ingress controller by running the following command:

```
ping test.anaconda.example.com
```

Here is an example response from the command:

```
PING test.anaconda.example.com (167.172.143.144): 56 data bytes
```

**Note:** If the ingress controller is to be installed with Workbench, this may not be possible. In such cases, it is sufficient to confirm that the networking team is prepared to establish these records immediately following installation.

A wildcard SSL secret for anaconda.example.com and \*.anaconda.example.com has been created. The public and private keys for the main certificate, as well as the full public certificate chain, are accessible from the administration server.

Note: Please share the public certificate chain in your checklist output.

If the SSL secret was created using a private CA, verify the public root certificate has been obtained.

If you are using a private Docker registry, verify the full set of Docker images have been transferred to this registry.

If a pull secret is required to access the Docker images (whether from the standard Workbench Docker channel or the private registry) verify the secret has been created in the application namespace by running the following command:

```
# Replace <NAMESPACE> with the namespace you've reserved for Workbench
# Replace <PULL_SECRET_NAME> with the name for your pull secret
kubectl get secret -n <NAMESPACE> <PULL_SECRET_NAME>
```

# 1.2 Environment-specific recommendations

Each of these sections augments our environment preparation guide with recommendations specific to the vendor's offered environments.

Anaconda is committed to keeping these recommendations current. If you deviate from these recommendations, please share your changes with Anaconda so that we can confirm they do not compromise the operation of Data Science & AI Workbench, and keep our recommendations up-to-date.

#### 1.2.1 Azure Kubernetes Service

This guide offers recommended configurations and settings unique to Azure Kubernetes Service (AKS). These should be used to augment the generic requirements offered on our *primary requirements page*.

## Instance types

• Minimum: Standard\_D8s\_v4

• Recommended: Standard\_D16s\_v4

## Storage

Unfortunately, we have found that Azure's built-in, managed Network File System (NFS) service, Azure Files NFS, does not provide an acceptable performance level for use with Data Science & AI Workbench. We have not yet had the opportunity to evaluate Azure NetApp Files.

For this reason, Anaconda recommends creating a separate Virtual Machine for hosting NFS storage. Follow the recommendations offered in *this document*, with these Azure-specific recommendations:

- The Standard\_D4s\_v3 machine type is suitable for this purpose.
- Azure tightly couples disk size and IOPS performance. Anaconda recommends a minimum disk size of 1 TiB to ensure good performance.

**Note:** This server can be the administration server as well.

#### **Network**

Azure offers two different networking options for AKS clusters. Both approaches are compatible with Workbench, so the determination depends upon your larger networking needs.

**Note:** AKS uses a LoadBalancer which by default sets TCP idle timeouts to 4 minutes, and enables TCP resets. This may affect any user workload that depends on a continuous TCP connection.

#### **GPUs**

Please see this Azure guide for adding GPU resources to your AKS cluster.

#### 1.2.2 Amazon Elastic Kubernetes Service

This guide offers recommended configurations and settings unique to AWS Elastic Kubernetes Service (EKS) in Data Science & AI Workbench. These should be used to augment the generic requirements offered on our *primary requirements page*.

#### Instance types

• Minimum: m5.2xlarge

Recommended: m5.4xlarge or larger

## **Storage**

EKS supports the use of both EBS and EFS storage for persistence. In theory, EBS can be employed for the anaconda-storage volume; but because EBS is limited to the ReadWriteOnce access mode, only EFS is acceptable for the anaconda-persistence volume. For this reason, Anaconda recommends provisioning a single volume that is large and performant enough to accommodate *both* storage requirements, to simplify management.

Please refer to the following pages for information on provisioning an EFS volume:

- Amazon EFS CSI driver
- Working with Amazon EFS Access Points
- CreationInfo

Anaconda recommends the following configuration parameters for this volume:

- OwnerUid: Anaconda recommends this be set to the same UID selected to run the Workbench containers.
- OwnerGid: Anaconda recommends a value of 0, which simplifies access from Kubernetes containers whose primary group is 0 by default. If you choose a different GID, it will be necessary to incorporate that into the PersistentVolume specification.
- Permissions: 770 or 775. It is important that the directory be group writable.

When defining the access controls for this volume, include both the EKS cluster and the administration server, so the latter can be used to manage the volume.

**Note:** You can create an EFS access point using the UID/GID defined above.



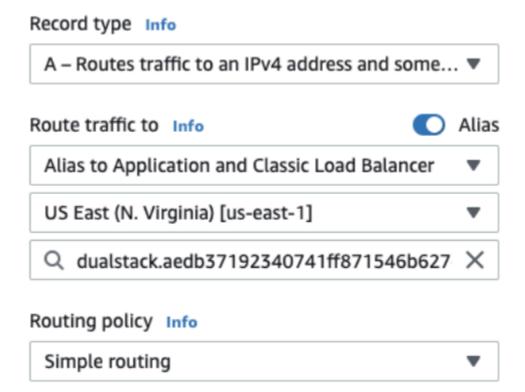
#### **Network**

If you are using the Ingress controller that ships with Workbench, *and* you are using an internal/private VPC or subnet you will need to add annotations to the ingress service:

kubectl edit svc/anaconda-enterprise-nginx-ingress

service.beta.kubernetes.io/aws-load-balancer-internal: 0.0.0.0/0
service.beta.kubernetes.io/aws-load-balancer-internal: "true"

Here is an example Route53 DNS configuration to the ingress.



# **GPU Support**

Please see this AWS guide for adding GPU resources to your EKS cluster.

# 1.2.3 Google Kubernetes Engine

This guide offers recommended configurations and settings unique to Google Kubernetes Engine (GKE) in Data Science & AI Workbench. These should be used to augment the generic requirements offered on our *primary requirements page*.

#### Instance types

• Minimum: n2-standard-8

• Recommended: n2-standard-16 or larger

#### **Storage**

We have found that the Basic SSD and High Scale SSD performance tiers for Google Cloud Filestore provide competent performance for use with Workbench. The minimum storage sizes are 2.5TiB for the Basic SSD tier and 10TiB for the High Scale SSD Tier. Because Filestore volumes supports the ReadWriteMany access mode, a single volume can serve both the anaconda-storage and anaconda-peristence storage requirements.

#### Recommendations:

- Do not change the default export options. In particular, squash-mode should remain as NO\_ROOT\_SQUASH.
- When granting access to this volume, include not just the GKE cluster itself, but to the Administration server as
  well. This will enable the server to be used to initialize relevant directories, set permissions, perform backups,
  and so forth.
- The ownership of the root directory should be set to <UID>: 0, where <UID> is the non-zero UID selected to run the Workbench containers.
- The permissions should be set to 770 or 775; group writability is required.

#### **GPU Support**

Please see this guide for adding GPU resources to your GKE cluster.

# 1.2.4 OpenShift Container Platform

This guide offers recommended configurations and settings unique to the Openshift Container Platform (OCP) in Data Science & AI Workbench. These should be used to augment the generic requirements offered on our *primary requirements page*.

#### **DNS/SSL**

OpenShift imposes a unique subdomain structure to the applications that run on its clusters; e.g.:

```
anaconda.apps.openshift.example.com
*.anaconda.apps.openshift.example.com
```

When creating the SSL certificate, make sure that they are constructed for this multi-level domain name.

**Note:** Note that even though your OCP cluster might already be configured to serve SSL, Workbench still requires its own SSL certificate due to the usage of wildcards.

#### Helm chart customization

OpenShift uses a different DNS server address than the one assumed by default in the helm chart. To ensure that the application uses the proper address, add the following lines, respecting indentation, to your values.yaml overrides file:

```
git:
    default:
    proxy:
        dns-server: dns-default.openshift-dns.svc.cluster.local
```

# 1.2.5 RedHat OpenShift Service on AWS (ROSA)

This guide offers recommended configurations and settings to install Data Science & AI Workbench onto a Red Hat OpenShift Service on AWS (ROSA) cluster.

#### Instance types

• Minimum: m5.2xlarge

• Recommended: m5.4xlarge or larger

## **Storage**

ROSA supports the use of both EBS and EFS storage for persistence. In theory, EBS can be employed for the anaconda-storage volume; but because EBS is limited to the ReadWriteOnce access mode, only EFS is acceptable for the anaconda-persistence volume. For this reason, Anaconda recommends provisioning a single volume that is large and performant enough to accommodate *both* storage requirements, to simplify management.

Please refer to the following pages for information on provisioning an EFS volume:

- Amazon EFS CSI driver
- · Working with Amazon EFS Access Points
- CreationInfo

Anaconda recommends the following configuration parameters for this volume:

• OwnerUid: Anaconda recommends this be set to the same UID selected to run the Workbench containers.

- OwnerGid: Anaconda recommends a value of 0, which simplifies access from Kubernetes containers whose primary group is 0 by default. If you choose a different GID, it will be necessary to incorporate that into the PersistentVolume specification.
- Permissions: 770 or 775. It is important that the directory be group writable.

When defining the access controls for this volume, include both the ROSA cluster and the administration server, so the latter can be used to manage the volume.

Note: You can create an EFS access point using the UID/GID defined above.



#### **DNS/SSL**

OpenShift imposes a unique subdomain structure to the applications that run on its clusters; e.g.:

```
anaconda.apps.openshift.example.com
*.anaconda.apps.openshift.example.com
```

When creating the SSL certificate, make sure that they are constructed for this multi-level domain name.

**Note:** Note that even though your ROSA cluster may already be configured to serve SSL, Workbench will still require its own SSL certificate due to the usage of wildcards.

#### **Helm chart customization**

OpenShift uses a different DNS server address than the one assumed by default in the helm chart. To ensure that the application uses the proper address, add the following lines, respecting indentation, to your values.yaml overrides file:

```
git:
    default:
    proxy:
        dns-server: dns-default.openshift-dns.svc.cluster.local
```

# 1.2.6 NFS Storage Recommendations

A common mechanism for provisioning the storage required for Data Science & AI Workbench persistence involves the use of a Network File Server (NFS). This includes many cloud offerings such as Amazon EFS and Google Filestore and many on-premise NAS/SAN implementations. In this section, we provide specific recommendations for server-and client-side configuration. These recommendations should be used to augment the *general storage requirements* offered on our *install requirements page*.

#### Server recommendations

- If you are building a new machine to serve as your NFS server:
  - It should have at least 4 cores and 16GiB of RAM.
  - Increase the number of threads created by the NFS daemon to at least 64, to reduce the likelihood of contention between multiple users. For information on how to do this see your operating system documentation; for instance, this RedHat article.
  - If possible, use this file server as your administration server as well. This is a great way to manage and
    administer this persistence. If this is not possible, make sure to export the volume to the administration
    server as well as the Kubernetes cluster.
  - If you are intending to use the same server for both anaconda-storage and anaconda-persistence, then you should consolidate to a single PersistentVolume, as discussed in the general storage requirements.
- The use of premium storage tiers, and SSD-based storage in particular, is strongly recommended.
- In many environments, the performance of the volume (e.g., IOPS) is tightly coupled to the size of the disk. For this reason, Anaconda recommends *over-provisioning* the size of the disk to take advantage of this. In some environments, IOPS can be provisioned separately, but it can still be cost-effective to over-provision size instead.
- Anaconda recommends the use of the async export option.
- Anaconda recommends *against* the use of the root\_squash option. While a seemingly sensible option for security reasons, in practice we find that it too often leads to unexpected permissions issues. That said, a similar and more reliable option is to use the all\_squash option along with anonuid and anonguid. This effectively forces *all* remote access to be translated to the same UID and GID on the server. In summary, in order of preference, Anaconda recommends:
  - no\_root\_squash for maximum administration flexibility, and to allow the containers to utilize GID 0, the Kubernetes default.
  - all\_squash / anon\_uid / anon\_gid for a reliable option that avoids UID 0 & GID 0;
  - root\_squash only if there is no other alternative.
- To improve both security and performance, locate the file server on the same private subnet as the Kubernetes cluster, and limit the exports to that subnet.

#### **Client recommendations**

- When mounting the NFS share, Anaconda recommends overriding the default read and write block sizes by
  using the options rsize=65536, wsize=65536. The reason smaller block sizes are preferred is because the
  creation of conda environments frequently involves the manipulation of thousands of smaller files. Large block
  sizes result in significant inefficiency.
- We also recommend the use of the noatime option. This eliminates the updating of file *access* times over NFS, further reducing network overhead. Note that file *modification* times are still preserved.

#### **Persistent Volume specifications**

Encapsulating the client recommendations into the PersistentVolume and PersistentVolumeClaim specifications is relatively simple.

Begin with the following template, called (for instance) pv.yaml:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: <NAME>
  annotations:
    pv.beta.kubernetes.io/gid: "<GID>"
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
 mountOptions:
    - rsize=65536
    - wsize=65536
    noatime
 nfs:
    server: <ADDRESS>
    path: <PATH>
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: <NAME>
spec:
  accessModes:

    ReadWriteMany

  volumeName: <NAME>
  storageClassName: ""
  resources:
    requests:
      storage: 100Gi
```

Perform the following replacements:

• <NAME>: you can give this any name you wish, or adhere to our conventions of anaconda-storage and/or anaconda-persistence. This name will ultimately be supplied to the Helm chart values. Note that <NAME> appears in three places; use the same value for all.

- <GID>: this is the group ID which has write access to the volume. As discussed above, the recommended value is 0; but if you are forced to use root\_squash or all\_squash, make sure this has the value of the selected GID. The quotes must be preserved.
- <ADDRESS>: the FQDN or numeric IP address of the NFS server.
- <PATH>: the exported path from the NFS server.

The size entry in both resources does *not* need to be changed, even if your volume is (as is likely) significantly larger. All that matters in this case is that the values are the same.

Once this template is properly populated, you can create the resources with the command:

kubectl create -f pv.yaml

If you have allocated two different NFS volumes for anaconda-storage and anaconda-persistence, repeat this template for each.

**Caution:** When creating the Persistent Volume (PV) using NFS as the provider, specify your NFS version under mountOptions to avoid performance issues within your Kubernetes platform. For example:

mountOptions:

- hard
- nfsvers=3
- rsize=65536
- wsize=65536
- noatime

# 1.3 Preparing a Gravity environment for Workbench

Determining the resource requirements for a Kubernetes cluster depends on a number of different factors, including what type of applications you are going to be running, the number of users that are active at once, and the workloads you will be managing within the cluster. *Data Science & AI Workbench's performance is tightly coupled with the health of your Kubernetes stack*, so it is important to allocate enough resources to manage your users workloads. Generally speaking, your system should contain at least 1 CPU, 1GB of RAM, and 5GB of disk space for each project session or deployment.

To install Workbench successfully, your systems must meet or exceed the requirements listed below. Anaconda has created a pre-installation checklist to help prepare you for installation. The checklist verifies that your cluster has the necessary resources are reserved and is ready to install Workbench. Anaconda's Implementation team will review the checklist with you prior to your installation.

You can initially install Workbench on up to five nodes. Once initial installation is complete, you can add or remove nodes as needed. Anaconda recommends having *one master and one worker node per cluster*. For more information, see *Adding and removing nodes*.

For historical information and details regarding Anaconda's policies related to Gravity, see our Gravity update policy.

# 1.3.1 Hardware requirements

Anaconda's hardware recommendations ensure a reliable and performant Kubernetes cluster.

The following are minimum specifications for the master and worker nodes, as well as the entire cluster.

Master node	Minimum
CPU	16 cores
RAM	64GB
Disk space in /opt/anaconda	500GB
Disk space in /var/lib/gravity	300GB
Disk space in /tmp or \$TMPDIR	50GB

#### Note:

- Disk space reserved for /var/lib/gravity is utilized as additional space to accommodate upgrades. Anaconda
  recommends having this available during installation.
- The /var/lib/gravity volume *must be mounted on local storage*. Core components of Kubernetes run from this directory, some of which are extremely intolerant of disk latency. Therefore, Network-Attached Storage (NAS) and Storage Area Network (SAN) solutions are not supported for this volume.
- Disk space reserved for /opt/anaconda is utilized for project and package storage (including mirrored packages).
- Anaconda recommends that you set up the /opt/anaconda and /var/lib/gravity partitions using Logical Volume Management (LVM) to provide the flexibility needed to accommodate easier future expansion.
- Currently /opt and /opt/anaconda must be an ext4 or xfs filesystem, and cannot be an NFS mountpoint.
   Subdirectories of /opt/anaconda may be mounted through NFS. For more information, see Mounting an external file share.

**Warning:** Installations of Workbench that utilize an xfs filesystem *must* support d\_type file labeling to work properly. To support d-type file labeling, set ftype=1 by running the following command prior to installing Workbench.

This command will erase all data on the specified device! Make sure you are targeting the correct device and that you have backed up any important data from it before proceeding.

Worker node	Minimum
CPU	16 cores
RAM	64GB
Disk space in /var/lib/gravity	300GB
Disk space in /tmp or \$TMPDIR	50GB

**Note:** When installing Workbench on a system with multiple nodes, verify that the clock of each node is in sync with the others prior to installation. Anaconda recommends using the Network Time Protocol (NTP) to synchronize computer system clocks automatically over a network. For step by step instructions, see How to Synchronize Time with Chrony NTP in Linux.

# 1.3.2 Disk IOPS requirements

Master and worker nodes require a minimum of 3000 concurrent Input/Output operations Per Second (IOPS).

**Note:** Hard disk manufacturers report *sequential* IOPS, which are different than *concurrent* IOPS. On-premise installations require servers with disks that support a minimum of 50 sequential IOPS. Anaconda recommends using Solid State Drive (SSD) or better.

# 1.3.3 Cloud performance requirements

Requirements for running Workbench in the cloud relate to compute power and disk performance. Make sure your chosen cloud platform meets these minimum specifications:

#### **Amazon Web Services (AWS)**

Anaconda recommends an instance type no smaller than m4.4xlarge for both master and worker nodes. You must have a minimum of 3000 IOPS.

#### **Microsoft Azure**

Anaconda recommends a VM size of Standard D16s v3 (16 VCPUs, 64 GB memory).

#### **Google Cloud Platform (GCP)**

There are no unique requirements for installing Workbench on the Google Cloud Platform.

# 1.3.4 Operating system requirements

Workbench currently supports the following Linux versions:

- RHEL/CentOS 7.x, 8.x
- Ubuntu 16.04
- SUSE 12 SP2, 12 SP3, 12 SP5 (Requires you set DefaultTasksMax=infinity in /etc/systemd/system. conf)

**Caution:** Some versions of the RHEL 8.4 AMI on AWS are bugged due to a combination of a bad ip rule and the networkmanager service. Remove the bad rule and disable the networkmanager service prior to installation.

# 1.3.5 Security requirements

- If your Linux system utilizes an antivirus scanner, make sure the scanner excludes the /var/lib/gravity volume from its security scans.
- Installation requires that you have sudo access.
- Nodes running CentOS or RHEL must make sure that Security Enhanced Linux (SELinux) set to either disabled or permissive mode in the /etc/selinux/config file.

**Tip:** Check the status of SELinux by running the following command:

getenforce

# **Configuring SELinux**

- 1. Open the /etc/selinux/config file using your preferred file editor.
- 2. Find the the line that starts with SELINUX= and set it to either disabled or permissive.
- 3. Save and close the file.
- 4. Reboot your system for changes to take effect.

# 1.3.6 Kernel module requirements

Kubernetes relies on certain functionalities provided by the Linux kernel. The Workbench installer verifies that the following kernel modules (required for Kubernetes to function properly) are present, and notifies you if any are not loaded.

Linux Distribution	Version	Required Modules
CentOS	7.2	bridge, ebtable_filter, ebtables, iptable_filter, iptable_nat, overlay
CentOS	7.3-7.7, 8.0	br_netfilter, ebtable_filter, ebtables, iptable_filter, iptable_nat, over-
		lay
RedHat Linux	7.2	bridge, ebtable_filter, ebtables, iptable_filter, iptable_nat
RedHat Linux	7.3-7.7, 8.0	br_netfilter, ebtable_filter, ebtables, iptable_filter, iptable_nat, over-
		lay
Ubuntu	16.04	br_netfilter, ebtable_filter, ebtables, iptable_filter, iptable_nat, over-
		lay
Suse	12 SP2, 12 SP3, 12	br_netfilter, ebtable_filter, ebtables, iptable_filter, iptable_nat, over-
	SP5	lay

Module Name	Purpose	
bridge	Enables Kubernetes iptables-based proxy to operate	
br_nerfilter	Enables Kubernetes iptables-based proxy to operate	
overlay	Enables the use of the overlay or overlay2 Docker storage driver	
ebtable_filter	Allows a service to communicate back to itself via internal load balancing when necessary	
ebtables	Allows a service to communicate back to itself via internal load balancing when necessary	
iptable_filter	Ensures the firewall rules set up by Kubernetes function properly	
iptable_nat	Ensures the firewall rules set up by Kubernetes function properly	

**Note:** Verify a module is loaded by running the following command:

```
# Replace <MODULE_NAME> with a module name
lsmod | grep <MODULE_NAME>
```

If the command produces a return, the module is loaded.

If necessary, run the the following command to load a module:

```
# Replace <MODULE_NAME> with a module name
sudo modprobe <MODULE_NAME>
```

**Caution:** If your system does not load modules at boot, you *must* run the following command—for each module—to ensure they are loaded on every reboot:

```
# Replace <MODULE_NAME> with a module name
sudo echo -e '<MODULE_NAME>' > /etc/modules-load.d/<MODULE_NAME>.conf
```

# 1.3.7 System control settings

Workbench requires the following Linux sysctl settings to function properly:

sysctl setting	Purpose	
net.bridge.bridge-	Communicates with bridge kernel module to ensure Kubernetes iptables-based proxy oper-	
nf-call-iptables	ates	
net.bridge.bridge-	Communicates with bridge kernel module to ensure Kubernetes iptables-based proxy oper-	
nf-call-ip6tables	ates	
fs.may_detach_mounts Allows the unmount operation to complete even if there are active references to the filesystem		
	remaining	
net.ipv4.ip_forward	Required for internal load balancing between servers to work properly	
fs.inotify.max_user_warshete 1048576 to improve cluster longevity		

**Note:** If necessary, run the following command to enable a system control setting:

```
# Replace <SYSCTL_SETTING> with a system control setting
sudo sysctl -w <SYSCTL_SETTING>=1
```

To persist system settings on boot, run the following for each setting:

```
# Replace <SYSCTL_SETTING> with a system control setting
sudo echo -e "<SYSCTL_SETTING> = 1" > /etc/sysctl.d/10-<SYSCTL_SETTING>.conf
```

## 1.3.8 GPU requirements

Workbench requires that you install a supported version of the NVIDIA Compute Unified Device Architecture (CUDA) driver *on the host OS of any GPU worked node*.

Currently, Workbench supports the following CUDA driver versions:

- CUDA 10.2
- CUDA 11.2
- CUDA 11.4
- CUDA 11.6

**Note:** Notify your Anaconda Implementation team member which CUDA version you intend to use, so they can provide the correct installer.

You can obtain the driver you need a few different ways.

- Use the package manager or the Nvidia runfile to download the file directly.
- For SLES, CentOS, and RHEL, you can get a supported driver using rpm (local) or rpm (network).
- For Ubunutu, you can get a driver using deb (local) or deb (network).

GPU deployments should use one of the following models:

- Tesla V100 (recommended)
- Tesla P100 (adequate)

Theoretically, Workbench will work with any GPU card compatible with the CUDA drivers, as long as they are properly installed. Other cards supported by CUDA 11.6:

- A-Series: NVIDIA A100, NVIDIA A40, NVIDIA A30, NVIDIA A10
- RTX-Series: RTX 8000, RTX 6000, NVIDIA RTX A6000, NVIDIA RTX A5000, NVIDIA RTX A4000, NVIDIA T1000, NVIDIA T600, NVIDIA T400
- HGX-Series: HGX A100, HGX-2
- T-Series: Tesla T4
- P-Series: Tesla P40, Tesla P6, Tesla P4
- K-Series: Tesla K80, Tesla K520, Tesla K40c, Tesla K40m, Tesla K40s, Tesla K40st, Tesla K40t, Tesla K20Xm, Tesla K20m, Tesla K20s, Tesla K20c, Tesla K10, Tesla K8
- M-Class: M60, M40 24GB, M40, M6, M4

Support for GPUs in Kubernetes is still a work in progress, and each cloud vendor provides different recommendations. For more information about GPUs, see *Understanding GPUs*.

# 1.3.9 Network requirements

Workbench requires the following network ports to be externally accessible:

## **External ports**

Port	Protocol	Description
80	TCP	Workbench UI (plaintext)
443	TCP	Workbench UI (encrypted)
32009	TCP	Operations Center Admin UI

These ports need to be externally accessible *during installation only*, and can be closed after completing the install process:

## **Install ports**

Port	Protocol	Description
4242	TCP	Bandwidth checker utility
61009	TCP	Install wizard UI access required during cluster installation
		Cluster mstanation
61008, 61010, 61022-61024	TCP	Installer agent ports

The following ports are used for cluster operation, and must be open internally, between cluster nodes:

## **Cluster communication ports**

Port	Protocol	Description
53	TCP and UDP	Internal cluster DNS
2379, 2380, 4001, 7001	TCP	Etcd server communication
3008-3012	TCP	Internal Workbench service
3022-3025	TCP	Teleport internal SSH control panel
3080	TCP	Teleport Web UI
5000	TCP	Docker registry
6443	TCP	Kubernetes API Server
6990	TCP	Internal Workbench service
7496, 7373	TCP	Peer-to-peer health check
7575	TCP	Cluster status gRPC API
8081, 8086-8091, 8095	TCP	Internal Workbench service
8472	UDP	Overlay network
9080, 9090, 9091	TCP	Internal Workbench service
10248-10250, 10255	TCP	Kubernetes components
30000-32767	TCP	Kubernetes internal services range

Make sure that the firewall is permanently set to keep the required ports open, and will save these settings across reboots. Then restart the firewall to load your changed settings.

**Tip:** There are various tools you can use to configure firewalls and open required ports, including iptables, firewall-cmd, susefirewall2, and more!

You'll also need to update your firewall settings to ensure that the 10.244.0.0/16 pod subnet and 10.100.0.0/16 service subnet are accessible to every node in the cluster, and grant all nodes the ability to communicate via their primary interface.

For example, if you're using iptables:

```
# Replace <NODE_IP> with the internal IP address(es) used by all nodes in the_cluster to connect to the master node
iptables -A INPUT -s 10.244.0.0/16 -j ACCEPT
iptables -A INPUT -s 10.100.0.0/16 -j ACCEPT
iptables -A INPUT -s <NODE_IP> -j ACCEPT
```

If you plan to use online package mirroring, allowlist the following domains in your network's firewall settings:

- · repo.anaconda.com
- · anaconda.org
- conda.anaconda.org
- binstar-cio-packages-prod.s3.amazonaws.com

To use Workbench in conjuctation with Anaconda Navigator in online mode, allowlist the following sites in your network's firewall settings as well:

- https://repo.anaconda.com (or for older versions of Navigator and conda)
- https://conda.anaconda.org if any users will use conda-forge and other channels on Anaconda.org
- google-public-dns-a.google.com (8.8.8.8:53) to check internet connectivity with Google Public DNS

# 1.3.10 TLS/SSL certificate requirements

Workbench uses certificates to provide transport layer security for the cluster. Self-signed certificates are generated during the initial installation. Once installation is complete, you can configure the platform to use your organizational TLS/SSL certificates.

You can purchase certificates commercially, or generate them using your organization's internal public key infrastructure (PKI) system. When using an internal PKI-signed setup, the CA certificate is inserted into the Kubernetes secret.

In either case, the configuration will include the following:

- A certificate for the root certificate authority (CA)
- An intermediate certificate chain
- · A server certificate
- A certificate private key

For more information about TLS/SSL certificates, see *Updating TLS/SSL certificates*.

### 1.3.11 DNS requirements

Workbench assigns unique URL addresses to deployments by combining a dynamically generated universally unique identifier (UUID) with your organization's domain name, like this: https://uuid001.anaconda.yourdomain.com.

This requires the use of wildcard DNS entries that apply to a set of domain names such as \*.anaconda.yourdomain.com.

For example, if you are using the domain name anaconda.yourdomain.com with a master node IP address of 12. 34.56.78, the DNS entries would be as follows:

```
anaconda.yourdomain.com IN A 12.34.56.78
*.anaconda.yourdomain.com IN A 12.34.56.78
```

**Note:** The wildcard subdomain's DNS entry points to the Workbench master node.

The master node's hostname and the wildcard domains must be resolvable with DNS from the master nodes, worker nodes, and the end user machines. To ensure the master node can resolve its own hostname, distribute any /etc/hosts entries to the gravity environment.

**Caution:** If dnsmasq is installed on the master node or any worker nodes, you'll need to remove it from all nodes *prior to installing Workbench*.

Verify dnsmasq is disabled by running the following command:

```
sudo systemctl status dnsmasq
```

If necessary, stop and disable dnsmasq, run the following commands:

```
sudo systemctl stop dnsmasq
sudo systemctl disable dnsmasq
```

# 1.3.12 Browser requirements

Workbench supports the following web browsers:

- Chrome 39+
- Firefox 49+
- Safari 10+

The minimum browser screen size for using the platform is 800 pixels wide and 600 pixels high.

# 1.3.13 Verifying system requirements

The installer performs pre-installation checks, and only allows installation to continue on nodes that are configured correctly, and include the required kernel modules. If you want to perform the system check yourself prior to installation, you can run the following commands from the installer directory, ~/anaconda-enterprise-<VERSION>, on your intended master and worker nodes:

To perform system checks on the master node, run the following command as sudo or root user:

```
sudo ./gravity check --profile ae-master
```

To perform system checks on a worker node, run the following command as sudo or root user:

```
sudo ./gravity check --profile ae-worker
```

If all of the system checks pass and all requirements are met, the output from the above commands will be empty. If the system checks fail and some requirements are not met, the output will indicate which system checks failed.

### 1.3.14 Pre-installation checklist

Anaconda has created this pre-installation checklist to help you verify that you have properly prepared your environment prior to installation. You can run the system verification checks to automatically verify many of the requirements for you.

**Caution:** System verification checks are not comprehensive, so make sure you manually verify the remaining requirements.

### **Gravity pre-inallation checklist**

All nodes in the cluster meet the minimum or recommended specifications for CPU, RAM, and disk space.

All nodes in the cluster meet the minimum IOPS required for reliable performance.

All cluster nodes are operating the same version of the OS, and that *the OS version is supported* by Workbench.

NTP is being used to synchronize computer system clocks, and all nodes are in sync.

The user account performing the installation has sudo access on all nodes and is not a root user.

All required kernel modules are loaded.

The sysctl settings are configured correctly.

Any GPUs to be used with Workbench have a supported NVIDIA CUDA driver installed.

The system meets *all network port requirements*, whether the specified ports need to be open internally, externally, or during installation only.

The *firewall is configured correctly*, and an rules designed to limit traffic have been *temporarily* disabled until Workbench is installed and verified.

If necessary, the *domains required* for *online* package mirroring have been allowlisted.

The final *TLS/SSL certificates* < grav\_tls\_ssl\_reqs> to be installed with Workbench have been obtained, including the private keys.

The Workbench A or CNAME domain record is fully operational, and points to the IP address of the master node.

The wildcard DNS entry for Workbench is also fully operational, and points to the IP address of the master node. More information about the wildcard DNS requirements can be found *here*.

The /etc/resolv.conf file on all the nodes *does not* include the rotate option.

Any existing installations of Docker (and dockerd), dnsmasq, and 1xd have been removed from all nodes, as they will conflict with Workbench.

	s Workbench are		

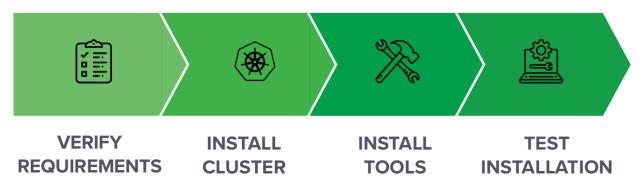
## INSTALLING WORKBENCH

Data Science & AI Workbench can be installed on a variety of Kubernetes clusters, in addition to the Gravity Kubernetes environment we have historically provided.

When installing the Gravity environment, you will initially install it onto a single control plane node. After completing this basic installation, you can *add or remove nodes* on the cluster as needed, including GPUs.

When installing into your own Kubernetes environment, these instructions assume that the cluster itself has already been provisioned. Our instructions provide you with the details you need to verify that the cluster meets the basic requirements dictated by Workbench, and to provision the storage and networking resources required to run this application.

When you've determined an initial topology for your cluster, follow this high-level process to install Workbench:



# 2.1 Gravity Installation

Gravity is an open-source, containerized Kubernetes framework that allows containerized applications like Anaconda Enterprised to be delivered in a self-contained installer that includes a special containerized implementation of Kubernetes. Anaconda has shipped Data Science & AI Workbench in a Gravity package since its first release.

Due to changes in the business model of Teleport, the original developers of Gravity, we are limiting the use of the Gravity-based installer to existing customers. For more information, see Anaconda's *Gravity update policy*.

New customers, and any existing customers who no longer want to use Gravity, are encouraged to explore our *Bring Your Own Kubernetes (BYOK8s)* option.

After you have verified that your *environment is prepared for installation* by completing the *pre-installation checklist*, you're ready to install the cluster!

**Note:** The Gravity installation process only sets up a single control plane node. Once installation is complete, you can add additional nodes as desired, including GPUs.

- 1. Open a terminal on the master node and log in to a service account with sudo access.
- 2. Download the installer by running the following command:

```
# Replace <INSTALLER_LOCATION> with the provided location of the installer
curl -0 <INSTALLER_LOCATION>
```

Note: If you are in an airgapped environment, you must deliver the installer to the master node in an alternate manner.

3. Decompress the installer and enter the top directory by running the following commands:

```
# Replace <INSTALLER> with the installer file you just downloaded
tar xzvf <INSTALLER>
cd <INSTALLER>
```

4. Set up and initialize your conda environment by running the following command:

```
./conda-bootstrap.sh
```

- 5. Enter yes to allow the installer to initialize your ae5-conda environment by running conda init when prompted.
- 6. Verify your environment successfully created by running the following command:

```
conda env list
```

**Note:** This command returns a list of existing conda environments. You should see a base environment present with an asterisk next to the directory to indicate it is currently active.

7. Perform the Anaconda preflight system check on the master node by running the following command:

```
ae-preflight
```

Note: You will receive an overall PASS or WARN result from the script. The output is captured in a results.txt file for later reference, if necessary. If something goes wrong, you can view the results to determine the issue and correct it.

8. Create a file named values.yaml and populate it with the following content:

```
# Replace <HOSTNAME> with the fully qualified domain name of the host server
# Replace <GID> with the group ID with read-write access to the shared network file_
→system (NFS) volume
# Replace <SERVER_FQDN> with the address of the shared NFS volume
# Replace <SERVER_PATH> with the path to the shared NFS volume
apiVersion: v1
```

(continues on next page)

(continued from previous page)

```
kind: ConfigMap
metadata:
   name: anaconda-enterprise-install
data:
   values: |
   hostname: <HOSTNAME>
    persistence:
      groupID: <GID>
      nfs:
      server: <SERVER_FQDN>
      path: <SERVER_PATH>
```

If you are installing a single-node cluster and you want to use the /opt/anaconda/storage volume for managed persistence, the persistence: section of the values.yaml can be entered as:

```
persistence:
pvc: anaconda-storage
```

**Note:** The direct use of /opt/anaconda/storage for managed persistence works *only* for single-node clusters. It must be replaced with a network file system (NFS) configuration *before* adding additional nodes to the cluster. Otherwise, sessions and deployments launched on the additional nodes will fail to start. For more information, see *managed persistence*.

9. Run the gravity installation command:

```
# Replace <ADVERTISE-ADDR> with the IP address you want to be visible to the other...
--nodes. If you have a private network and all nodes can communicate on it, use the...
--private IP address for the advertise address

# Replace <CLUSTERNAME> with the name you're giving to your cluster. Alphanumeric...
--characters and periods only

# Replace <VALUES_PATH> with the filepath to the values.yaml file you just created sudo ./gravity install --advertise-addr=<ADVERTISE-ADDR> --cluster=<CLUSTERNAME> --
---config <VALUES_PATH> --service-uid=$(id -u) --cloud-provider=generic
```

Here is an example of what the installation command looks like for a single-node cluster:

```
sudo ./gravity install --advertise-addr=192.168.1.1 --cluster=MyCluster --config ./

→values.yaml --service-uid=$(id -u) --cloud-provider=generic
```

If you are using an alternate TMPDIR, prepend the command with this new value. For example:

```
sudo TMPDIR=/mytmp ./gravity install --advertise-addr=192.168.1.1 --

→cluster=MyCluster --config=./values.yaml --service-uid=$(id -u) --cloud-

→provider=generic
```

**Caution:** The --service-uid is the service account UID and cannot be a root user. If you are performing a reinstallation of Workbench, you must delete the previous planet user prior to setting the new --service-uid.

10. Monitor the progress from the command line. Your output will look like this:

```
* [0/100] starting installer

* [0/100] preparing for installation... please wait

* [0/100] application: AnacondaEnterprise:5.5.x

* [0/100] starting non-interactive install

* [0/100] initializing the operation

* [20/100] configuring packages

* [50/100] installing software
```

- 1. Allow time for the process to complete (approximately 20-30 minutes).
- 2. Create a local user account and password to log in to the Gravity Operations Center. To do so, enter the Gravity environment from any of the nodes by running the following command:

```
sudo gravity enter
```

3. Create a local user account and password by running the following command:

```
# Replace <EMAIL> with the email address for the account

# Replace <PASSWORD> with a password for the account. The password must be six_
characters long

gravity --insecure user create --type=admin --email=<EMAIL> --password=<PASSWORD> --
ops-url=https://gravity-site.kube-system.svc.cluster.local:3009
```

4. Inform gravity that you have completed the installation by running the following command from within the Gravity environment:

```
gravity site complete
```

5. Proceed to the *post-install configuration steps*.

**Note:** Workbench itself will likely require another 30 minutes or so to fully load. The post-install configuration steps describe how to confirm that this process is fully complete.

# 2.2 Bring Your Own Kubernetes (BYOK8s) Installation

Data Science & AI Workbench can be installed on a wide variety of CNCF-compliant Kubernetes cluster. Installation is performed using the industry standard Helm package manager. Successful deployment *requires the expertise of your in-house Kubernetes administrators* to validate our requirements and provision the necessary compute, storage, and networking resources.

After you have verified that your *environment is prepared for installation* by completing the *pre-installation checklist*, you are ready to install the cluster!

Perform the following actions from your established administration server.

1. Installation is largely performed in the custom namespace. Modify the default kubectl context to point to the new namespace by running the following command:

```
# Replace <NAMESPACE> with the namespace that you reserved for Workbench kubectl config set-context --current --namespace=<NAMESPACE>
```

The remainder of these instructions assume this change has been made.

2. Open and unpack the Helm chart archive provided by Anaconda, and cd into the root directory of this archive.

**Note:** The archive root directory contains the following items of importance:

- The Anaconda-Enterprise/ directory. This contains your helm chart.
- The values.yaml override file. Edit this file to include custom configurations for the application.
- A version of these instructions labeled as INSTALL. {md, pdf}.
- 3. If you need a pull secret to access the Docker registry, create it now, then verify the presence of the pull secret by running the following command:

```
# Replace <PULL_SECRET_NAME> with the name for your pull secret
kubectl describe secret <PULL_SECRET_NAME>
```

4. Using your preferred text editor, open the values.yaml override file. Add or modify necessary values to this file. At minimum, you must provide the following:

### values.yaml override file configurations

**Note:** View the values.yaml override file *here*.

Configuration	Description		
hostname	The fully qualified domain name (FQDN) of the host.		
serviceAccountName	The name of the service account with the necessary permissions to		
	install and run the platform.		
uid	The UID under which the containers will be run.		
storage.pvc	The name of the persistent volume for the anaconda-storage		
	function.		
persistence.pvc	The name of the persistent volume for the anaconda-persistence		
	function.		
image.server	The address of the Docker image registry		
image.pullSecrets	Information about any pull secrets required to authenticate to the		
	Docker registry:		
	An empty value indicates that no pull secret is required		
	A string containing the name of a single secret		
	• A list of strings, with one secret name per entry		
dnsServer	(OpenShift Only) The FQDN of the internal cluster DNS server.		
	Uncomment the line provided in the file for this.		
ingress.className	The name of the IngressClass resource used by your ingress con-		
	troller. If you are installing the Anaconda-supplied ingress, no mod-		
	ification is necessary for this configuration.		
(If necessary) key-	The path to your <i>LDAPS</i> truststore (e.g. /etc/secrets/certs/		
cloak.truststore	ldaps.jks).		
(If necessary) key-	The LDAPS truststore password		
cloak.truststore_password			
(If necessary) key-	The name of the secret you created containing the truststore		
cloak.truststore_secret			

5. Start your installation by running the following command:

helm install --values ./values.yaml anaconda-enterprise ./Anaconda-Enterprise/

**Caution:** The current version of Workbench requires the release name anaconda-enterprise. Do not change this value.

6. Monitor the progress of your installation by running the following command:

watch kubectl get pods

7. Wait for all of the pods to get to the Running or Completed state.

**Tip:** Installation can take several minutes depending on the performance characteristics of your system. If a particular pod is behaving in an unexpected manner, you can investigate the cause using commands such as:

kubectl logs <POD-NAME>
kubectl describe pod <POD-NAME>

8. If you chose to install the Anaconda-supplied ingress, you must now need to determine its assigned IP address, and create your DNS records for the cluster. To determine this address, run the command:

kubectl get svc anaconda-enterprise-nginx.ingress

**Note:** The IP address is provided in the External IP column. Create your DNS records for your FQDN and for its wildcard. For example, anaconda.example.com and \*.anaconda.example.com.

Once these DNS changes have propagated, you can proceed to the next step.

9. Connect to Workbench with your browser.

**Note:** Your browser may initially refuse to connect due to the use of our generated, self-signed SSL certificates. You should temporarily permit your browser to proceed anyway. You will also have to do the same every time you start a new session or deployment, so it is best to comlpete the next step as soon as possible.

10. Update your SSL cerfiticates.

**Note:** If you are installing the Anaconda-supplied ingress, you can expect it to enter a CrashLoopBackoff state until the SSL certificate generation task has completed. This is expected behavior. Once the preliminary certificates are generated, the next automatic restart of the ingress should proceed without incident.

Basic installation of Workbench is now complete! You can now perform any additional *post-installation steps* necessary.

If you need to uninstall Workbench, use the command:

helm uninstall anaconda-enterprise

**Note:** This will remove any of the resources that were installed by the application, but will not remove anything that you had created prior to the helm install command, such as persisent volumes.

# 2.3 Post-install configuration

Once you have successfully completed the browser-based or command-line installation, there are a few additional steps to take before beginning to use Data Science & AI Workbench.

# 2.3.1 Final configuration steps

- 1. The installation includes self-signed SSL certificates for encrypting web traffic. While these certificates can be used on a temporary basis to get started, you should replace them with non-self-signed SSL certificates for production operation. See *Updating TLS/SSL certificates* for information on modifying certificates.
- 2. (Gravity only) To add worker nodes to the cluster, follow the instructions here: Adding and removing nodes (Gravity)
- 3. Wait for the application to fully stabilize. Workbench can take up to 30 minutes to fully finish loading, depending upon the performance of the cluster network and attached disks. You may be able to log in to the main UI immediately, but sessions and deployments will not be ready to start. To confirm that the cluster is fully loaded, run the following command from the master node:

```
watch kubectl get pods
```

The output will look something like this:

Every 2.0s: kubectl get pods		aip2:	_
→Sat Jan 22 22:21:24 2022			
NAME	READY	STATUS	ш
→ RESTARTS AGE			
anaconda-enterprise-ap-auth-5c7d6589f7-4fh9m	1/1	Running	ш
→ 0 8m29s			
anaconda-enterprise-ap-auth-api-cbc8c54df-946pj	1/1	Running	ш
→ 0 8m28s			
anaconda-enterprise-ap-auth-escrow-57c49f66d5-fnjwx	1/1	Running	ш
→ 0 8m29s			
anaconda-enterprise-ap-deploy-5c7dbbd7b-1777x	1/1	Running	ш
→ 0 8m28s			
anaconda-enterprise-ap-docs-5df4f8744c-8qwll	1/1	Running	ш
→ 0 8m28s			
anaconda-enterprise-ap-git-storage-8597677fbf-dffjn	2/2	Running	ш
→ 0 8m29s			
anaconda-enterprise-ap-object-storage-54bf8b568b-f5nz7	1/1	Running	ш
→ 0 8m28s			
anaconda-enterprise-ap-operation-controller-67cdb6cdbb-rfph9	1/1	Running	ш
→ 0 8m28s			
anaconda-enterprise-ap-repository-95cb86f85-chbt9	1/1	Running	ш
→ 0 8m28s			
anaconda-enterprise-ap-storage-57668b8765-5r64r	1/1	Running	ш
0 8m28s		(continues on nex	t page)

(continued from previous page)

		(		10.
anaconda-enterprise-ap-ui-56c798bbcb-lrtsh		1/1	Running	ш
→ 0 8m28s				
anaconda-enterprise-ap-worksp	ace-7ff495c5dd-mg42z	1/1	Running	ш
→ 0 8m28s				
anaconda-enterprise-app-images-qbtvv		0/3	ш	
<pre>→ContainerCreating 0</pre>	8m29s			
anaconda-enterprise-nginx-ing	ress-rc-sj55d	1/1	Running	ш
→ 3 8m29s				
anaconda-enterprise-operation-images-f2bmn		1/1	Running	ш
→ 0 8m28s				
anaconda-enterprise-postgres-7b947455cf-bd667		1/1	Running	ш
→ 0 8m28s				
anaconda-enterprise-redis-77d49d5777-22rk4		1/1	Running	ш
→ 0 8m27s				
generate-certs-dfmsj		0/1	Completed	ш
→ 0 8m28s				

Wait for every pod to indicate a status of either Running or Completed. The app-images pods will be the last to stabilize. You can exit the watch command at any time with ctrl-C.

# 2.3.2 Testing your installation

Once the application has fully loaded, you can do the following to verify that your installation succeeded.

- Access the Workbench console by entering the URL of your Workbench server in a web browser: https://anaconda.example.com, replacing anaconda.example.com with the fully-qualified domain name of the host server.
- 2. Log in using the default username and password anaconda-enterprise / anaconda-enterprise.
- 3. Update the passwords for your anaconda-enterprise and admin users. See *Configuring user access* for information and steps for updating your password.

**Caution:** Do not alter the anaconda-enterprise account user name; it is used for certain cluster administration functions.

If you do not update your passwords, anyone with the default credentials will be able to access your admin console. This poses a security risk, so it is important to change these passwords immediately.

You can verify a successful installation by doing any or all of the following:

- Creating a new project and starting an editing session
- · Deploying a project
- Generating a token from a deployment

**Note:** Some of the sample projects can only be deployed after *mirroring the package repository*. To test your installation without doing this first, you can deploy the "Hello Anaconda Enterprise" sample project.

#### **Next steps:**

Now that you've completed these essential steps, you can do any of the following optional steps:

- Install the Workbench CLI.
- Install conda for package mirroring.
- Install Livy server for Hadoop Spark access.
- Set TLS/SSL certificates for the platform.

# 2.4 Updating TLS/SSL certificates

You can replace the self-signed certificates Data Science & AI Workbench generates as part of the *initial post-install configuration* at any time.

To minimize downtime, make sure that you have completed the *preparation steps* below before scheduling a maintenance interval to update the certificates. When you are ready, you can update the certificates using either *the admin console* or *the command line*. We strongly recommend the use of the admin console to minimize the potential for error.

**Caution:** Instruct users to *save their work and log out* before updating the certificates. While sessions, deployments, and jobs will still continue to function properly, there will be a brief interruption of front-end service during the update process.

## 2.4.1 Preparing for the update

Anaconda recommends that you gather the following information and files before you proceed. If you are using the admin console, have your files present in a location where you can enter their contents into the applicable fields within the UI. If you are updating your certificates using the command line, you'll need to copy the files to your server.

Most installations will need the following items:

- The fully qualified domain name of the server.
- The public SSL certificate for the domain <FQDN>: tls.crt
- The private SSL key for the domain <FQDN>: tls.key.
- If applicable, the intermediate certificate bundle: intermediate.pem.
- If your certificate was issued by a private root CA, the public certificate for that CA: rootca.crt.

*If you are using LetsEncrypt*, the filenames will be different:

- The public SSL certificate: cert.pem
- The private SSL key for the domain: privkey.pem.
- The intermediate certificate bundle: chain.pem.
- No root CA file is needed in this case.

If you are using a different domain and/or SSL certificate for the session/deployment subdomains, you will need this additional information:

- The wildcard subdomain. We will refer to this as <DEPLOYMENT-FQDN> below.
- The public SSL certificate for \*.<DEPLOYMENT-FQDN>: wildcard.crt
- The private SSL key for \*.<DEPLOYMENT-FQDN>: wildcard.key

Note that Workbench assumes that the intermediate certificate and root CA, if applicable, are identical for both certificates.

Finally, if you intend to use the command-line approach to updating the certificates, you will need a copy of the the latest set of publicly-trusted root certificates. A copy of this file, current to when the Workbench installer package was built, can be found at DIY-SSL-CA/CA/pubCA.crt in your unpacked installer assets.

**Note:** Workbench version 5.6+ automatically applies the updated SSL certificates to all running sessions and deployments.

# 2.4.2 Using the admin console

- 1. Log in to Workbench as an administrator.
- 2. Open the **My Account** dropdown menu and select **Admin Console**.
- 3. Select **Web Certificates** from the left-hand menu.
- 4. Enter your certificate and key information from the files you gathered during preparation into the appropriate fields. Make sure to paste the *content* of each file into the appropriate box, not the *filenames*.
  - Domain name: <FQDN>
  - SSL Certificate: tls.crt/cert.pem
  - SSL Private Key: tls.key/privkey.pem
  - Root Certificate: rootca.crt, if applicable
  - Intermediate Certificate: intermediate.pem / chain.pem, if applicable
  - Wildcard Domain: <pre
  - Wildcard Certificate: wildcard.crt if different; tls.crt / cert.pem if identical.
  - Wildcard Private Key: wildcard.key if different; tls.key / privkey.pem if identical.
- 5. Click **Save** to update the platform with your changes.
- 6. If your root CA has changed, you will need to restart the Workbench system pods to make sure that the pods that use this certificate will pick up the new copy.

```
kubectl get pods | grep ap- | cut -d ' ' -f 1 | xargs kubectl delete pods
```

# 2.4.3 Using the command line

1. Log in to the master node and make sure all of the files mentioned above have been transferred there, including the root CA trust file cacert.pem. To download the latest version of the root CA trust file, run the command:

```
curl -OL https://curl.se/ca/cacert.pem
```

2. Make sure you have downloaded a copy of cacert.pem per the instructions above. If you have a private root CA, append it to that file:

```
cat rootca.crt >> cacert.pem
```

3. If you have an intermediate bundle, you will need a file that combines that bundle with your main public certificate:

```
cat tls.crt intermediate.pem > fullchain.pem
cat wildcard.crt intermediate.pem > fullchain_wildcard.pem
```

If you are using LetsEncrypt, you may skip this step, as fullchain.pem has already been provided for you.

4. Create a file named certificates.yaml, then add the following information to it:

```
apiVersion: v1
kind: Secret
metadata:
 name: anaconda-enterprise-certs
type: kubernetes.io/tls
data:
    tls.crt: TLS_CRT
   tls.key: TLS_KEY
   rootca.crt: ROOT CA
apiVersion: v1
kind: Secret
metadata:
 name: anaconda-enterprise-wildcard
type: kubernetes.io/tls
data:
    tls.crt: WILDCARD_CRT
    tls.key: WILDCARD_KEY
```

This file will be used to provide your updated TLS/SSL certificate and key to access the kubernetes cluster. To keep your information secure, use placeholders for the actual values of the certificate and key.

- 5. Save your work and close the file.
- 6. Replace the placeholders in the certificates.yaml file with an encoded version of the actual certificate and key. Run the commands that fit your environment:

If you have a single certificate with an intermediate, run the following commands:

```
TLS_CRT=$(base64 -i --wrap=0 fullchain.pem)
TLS_KEY=$(base64 -i --wrap=0 tls.key)
ROOT_CA=$(base64 -i --wrap=0 cacert.pem)
echo "s@TLS_CRT@$TLS_CRT@;" "s@TLS_KEY@$TLS_KEY@;" "s@WILDCARD_CRT@$TLS_CRT@;"

--"s@WILDCARD_KEY@$TLS_KEY@;" "s@ROOT_CA@$ROOT_CA@" | sed -i.bak -f - certificates.
---yaml
```

If you are using a *LetsEncrypt* certificate, run the following commands:

```
TLS_CRT=$(base64 -i --wrap=0 fullchain.pem)
TLS_KEY=$(base64 -i --wrap=0 privkey.pem)
ROOT_CA=$(base64 -i --wrap=0 cacert.pem)
echo "s@TLS_CRT@$TLS_CRT@;" "s@TLS_KEY@$TLS_KEY@;" "s@WILDCARD_CRT@$TLS_CRT@;"

\( \to \' \s@WILDCARD_KEY@$TLS_KEY@;'' \' \s@ROOT_CA@$ROOT_CA@' \| \sed \( -i \).bak \( -f \) \certificates.
\( \to \)yaml
```

If you have separate certificates for the main and wildcard domains, run the following commands:

7. Verify that your certificates.yaml file now contains the correct certificate and key by running the following command:

```
grep -E 'TLS_(CRT|KEY)' certificates.yaml
```

If the command returns nothing, your certificates.yaml file has been successfully created.

If the command returns a message, re-verify the file name and its contents are correct, then run the command again.

8. Run the following command to create a backup of your existing secret:

```
kubectl get secret anaconda-enterprise-certs anaconda-enterprise-wildcard -o yaml -- \rightarrow export > certificates.orig
```

9. Remove the existing secrets, then recreate them from the new file by running the commands:

```
kubectl delete secret anaconda-enterprise-certs anaconda-enterprise-wildcard kubectl create -f certificates.yaml
```

10. Restart all Workbench system pods by running the command:

```
kubectl get pods | grep ap- | cut -d ' ' -f 1 | xargs kubectl delete pods
```

If you are using a gravity cluster, you will need to perform the following additional steps:

 $1. \ \ Create\ a\ file\ named\ gravity-certificates.yaml,\ then\ add\ the\ following\ information\ to\ it:$ 

```
apiVersion: v1
kind: Secret
metadata:
   name: cluster-tls
type: Opaque
data:
   certificate: TLS_CRT
   privatekey: TLS_KEY
```

This file will be used to provide your updated TLS/SSL certificate and key to access the gravity cluster. To keep your information secure, use placeholders for the actual values of the certificate and key.

- 2. Save your work and close the file.
- 3. Replace the placeholders in the gravity-certificates.yaml file with an encoded version of the actual certificate and key by running the following commands:

```
TLS_CRT=$(base64 -i --wrap=0 fullchain.pem)

TLS_KEY=$(base64 -i --wrap=0 tls.key)

echo "s@TLS_CRT@$TLS_CRT@;" "s@TLS_KEY@$TLS_KEY@" | sed -i.bak -f - gravity-

certificates.yaml (continues on next page)
```

(continued from previous page)

If you have a separate set of certificates for the ops-center domain, the command will look something like this:

```
TLS_CRT=$(base64 -i --wrap=0 fullchain_opscenter.pem)
TLS_KEY=$(base64 -i --wrap=0 opscenter.key)
echo "s@TLS_CRT@$TLS_CRT@;" "s@TLS_KEY@$TLS_KEY@" | sed -i.bak -f - gravity-
--certificates.yaml
```

4. Verify that your gravity-certificates.yaml file now contains the correct certificate and key by running the following command:

```
grep -E 'TLS_(CRT|KEY)' gravity-certificates.yaml
```

If the command returns nothing, your gravity-certificates.yaml file has been successfully created.

If the command returns a message, re-verify the file name and its contents are correct, then run the command again.

5. Run the following command to create a backup of your existing secret:

```
kubectl get secret -n kube-system cluster-tls -o yaml --export > gravity-

--certificates.orig
```

6. Remove the existing secrets by running the command:

```
kubectl delete secret -n kube-system cluster-tls
```

7. Create new secrets using the gravity-certificates.yaml file you created by running the command:

```
kubectl create -n kube-system -f gravity-certificates.yaml
```

# 2.5 Upgrading Workbench

The process of moving from one version of Data Science & AI Workbench to another varies slightly, depending on which version you are migrating to and from, so follow the instructions that correspond to your Workbench implementation. If you're moving from an implementation of Anaconda Enterprise 4 to Workbench, we consider that a *migration*. If you're moving between point releases of the same version, we consider that an *upgrade*.

Migrating between major releases of Workbench requires administrators to *migrate the package repository* and project owners to *migrate their notebooks*.

Upgrading Workbench generally involves exporting or backing up your current package repository and all project data, uninstalling the existing version and *installing* the newer version, then importing or restoring this information on the new platform.



# 2.5.1 Upgrading Workbench on Gravity

Contact the Anaconda implementation team before you begin for assistance upgrading your version of Data Science & AI Workbench. Follow along with these instructions as an Anaconda implementation team member guides you through the upgrade process.

### **Prerequisites:**

- You must have jq installed.
- You must have a service account with sudo access on your Kubernetes (K8s) master node.
- Your K8s master node must be configured with a DNS A record. The A record sets the domain name you use for your Workbench instance.
- If you are upgrading Workbench on a system with multiple nodes, you must verify the clock on each node is in sync with the others. Anaconda recommends using Network Time Protocol (NTP) to synchronize computer system clocks automatically over your network. For more information on installing and using Chrony to manage the NTP, see the instructions provided here.
- Create a backup of your anaconda-enterprise-anaconda-platform.yml file.
- You must have the ae-preflight package installed. For more information, see Workbench pre-flight check.

**Caution:** Anaconda recommends the use of managed persistence to ensure open sessions and deployments are captured by the upgrade process. If you are not using managed persistence, have all users save their work, stop any open sessions and deployments, and log out of the platform during the upgrade process.

## **Upgrade Processes**

After you have verified that your system meets all of the installation requirements, you're ready to upgrade the cluster.

There are two basic types of upgrades for Gravity users: in-place upgrades and fresh reinstallation upgrades. Follow along with the instructions here as your Anaconda implementation team member guides you through upgrading your software.

**Caution:** As a precautionary measure, Anaconda recommends you *create a backup* of your current instance of Workbench before performing an upgrade.

### In-place upgrade

**Warning:** In-place upgrades of Workbench are not supported for versions that are moving from Gravity 6 to Gravity 7. Instead, you must perform a fresh reinstallation to upgrade your software. If you do not, your installation will break. Check the version of Gravity and Workbench before you begin, and choose the upgrade process that best suits your needs.

In-place upgrades are also not supported when upgrading from Workbench 5.6.x to 5.6.2. Please perform a fresh reinstallation to upgrade between these versions.

In-place upgrades are performed while the software is still running. To perform an in-place upgrade:

- 1. Log in to a service account with sudo access on the master node running your Workbench software.
- 2. Download the installer file by running the following command:

# Replace <INSTALLER\_LOCATION> with the provided location of the installer file
curl -0 <INSTALLER\_LOCATION>

3. Decompress the installer file by running the following command:

# Replace <INSTALLER> with the installer file you just downloaded tar xvzf <INSTALLER>

4. Enter the installer directory by running the following command:

# Replace <INSTALLER> with the version of your installer
cd <INSTALLER>

5. Run the following command to verify your environment is properly prepared:

ae-preflight

If the check returns an overall result of WARN, you can view the results of the check by running the following command:

cat results.txt

If necessary, make applicable corrections to properly prepare your environment to meet the *Installation requirements*. Once you've verified that your environment is properly configured, you can begin the upgrade process.

6. To start the upload and upgrade process, run the following commands:

sudo ./upload
sudo ./gravity upgrade

The upgrade process can take up to an hour or more to complete, primarily due to the upload step. You can view the status of the upgrade process at any time by running the following command:

sudo watch ./gravity plan

Once the upgrade process is complete, the pods will start to come up on their own, but this process also takes some time to finish. Monitor the pods status by running the following command:

```
sudo watch kubectl get pods
```

If you encounter errors while doing your in-place upgrade, you can view which phase of the upgrade failed by running the following command:

```
sudo ./gravity plan
```

You can return to any phase of the upgrade process by running the rollback command against the name of the phase as it's listed in the **Phase** column of the ./gravity plan commands' return:

```
# Replace <NAME_OF_PHASE> with the name listed in the Phase column
sudo ./gravity plan rollback --phase=/<NAME_OF_PHASE>
```

After addressing any errors, resume the upgrade by running the following command:

```
sudo ./gravity upgrade --resume --force
```

Once you have resolved your errors, or if no errors have occurred, it's time to verify your installation.

### Fresh reinstallation upgrade

A fresh reinstallation upgrade backs up your current Workbench software configurations and settings, then uninstalls and reinstalls the software. After installation is complete, you can apply your saved configurations and settings to the new software version.

To perform a fresh reinstallation upgrade:

- Back up your configuration
- Uninstall
- · Reinstall and apply your saved settings

### Back up your configurations

- 1. Log in to a service account with sudo access on the master node running your Workbench software.
- 2. Download the installer file by running the following command:

```
# Replace <INSTALLER_LOCATION> with the provided location of the installer file
curl -0 <INSTALLER_LOCATION>
```

3. Decompress the installer file by running the following command:

```
# Replace <INSTALLER> with the installer file you just downloaded
tar xvzf <INSTALLER>
```

4. Enter the installer directory by running the following command:

```
# Replace <INSTALLER> with the installer file you just decompressed
cd <INSTALLER>
```

**Note:** The installer bundle contains the extract\_config.sh script, which retains the following files when run:

• anaconda-enterprise-certs.yaml

- anaconda-enterprise-keycloak.yaml
- helm\_values.yaml
- gravity\_values.yaml
- 5. Create a directory to contain the configuration data extracted by the script and name it "reinstall":

mkdir reinstall

6. Enter the directory you just created:

cd reinstall

7. Run the extract\_config.sh script by running the following command:

sudo bash extract\_config.sh

Once the script has completed, you will need to manually save some additional configurations and secrets to your reinstall directory.

8. Export your configmap to a .yaml file by running the following command:

sudo kubectl get cm -o yaml --export > configmap.yaml

#### Uninstall

Uninstall Workbench on all nodes and reboot your instance.

**Warning:** Do not run the command sudo rm -rf /opt/anaconda/storage on the master node as part of your uninstall process. If you do you, will lose your configuration settings and user data.

### Reinstall and apply your saved settings

1. Run the following command to verify your environment is properly prepared:

ae-preflight

If the check returns an overall result of WARN, you can view the results of the check by running the following command:

cat results.txt

If necessary, make applicable corrections to properly prepare your environment to meet the *Installation requirements*. Once you've verified that your environment is properly configured, you can begin the upgrade process.

- 2. Run the Gravity installation command using the gravity\_values.yaml file stored in the reinstall directory you created earlier instead of creating a new file during installation:
  - # Replace <ADVERTISE-ADDR> with the IP address you want to be visible to the other ondes. If you have a private network and all nodes can communicate on it, use the
  - →private IP address for the advertise address
  - # Replace <CLUSTERNAME> with the name you're giving to your cluster. Alphanumeric\_
  - →characters and periods only

(continues on next page)

(continued from previous page

```
# Replace <VALUES_PATH> with the filepath to the gravity_values.yaml file created_

ofrom running the extract_config.sh script
sudo ./gravity install --advertise-addr=<ADVERTISE-ADDR> --cluster=<CLUSTERNAME> --

oconfig <VALUES_PATH> --service-uid=$(id -u) --cloud-provider=generic
```

Here is an example of what the installation command looks like for a single-node cluster:

```
sudo ./gravity install --advertise-addr=192.168.1.1 --cluster=MyCluster --config ./
--gravity_values.yaml --service-uid=$(id -u) --cloud-provider=generic
```

- 3. Complete the rest of the installation process.
- 4. Replace your SSL certificate by running the following commands:

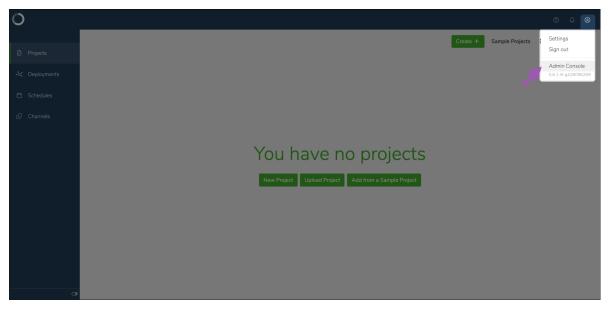
```
sudo kubectl create -f anaconda-enterprise-certs.json
sudo kubectl replace -n kube-system -f cluster-tls.json
```

5. Replace your user secrets by running the following command:

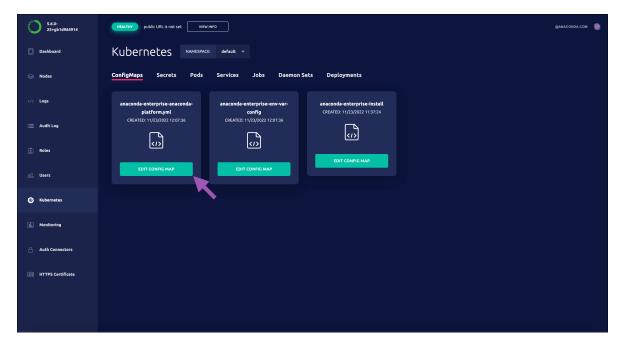
```
sudo kubectl create -f user-secrets.json
```

The configmap.yaml file that you created earlier contains settings that need to be restored to the anaconda-enterprise-anaconda-platform.yml file manually.

- 6. Open a web browser and log in to Workbench as an Administrator.
- 7. Open the **User** dropdown menu and select **Admin Console**.



- 8. Select **Manage Resources** to open your Gravity Ops Center.
- 9. Log in to your Gravity Ops Center. Contact your Gravity systems administrator if you need access.
- 10. Select **Kubernetes** from the left-hand navigation menu.
- 11. Select **Edit config map** under anaconda-enterprise-anaconda-platform.yml to open the file. Leave this browser open for now.



12. Return to your terminal and view the contents of your configmap.yaml file by running the following command:

cat configmap.yaml

- 13. Review the contents of the anaconda-enterprise-anaconda-platform.yml file and verify that your configuration values have been properly restored. If necessary, replace the applicable sections of the anaconda-enterprise-anaconda-platform.yml file with the configurations saved in your configmap. yaml file.
- 14. Restart the Anaconda platform pods by running the following command:

```
kubectl get pods | grep ap- | cut -d ' ' -f 1 | xargs kubectl delete pods
```

15. Add the worker nodes to the cluster.

Once the upgrade process is complete, it's time to verify your installation.

#### Verifying your installation

1. Verify all pods are running by running the following command:

```
sudo kubectl get pods
```

2. Open a web browser and navigate to your Authentication Center.

# Replace <FQDN> with the fully qualified domain name of your Workbench server https://<FQDN>/auth/admin

- 3. Select Users from the Manage menu, then click View all users and verify your users' data is present.
- 4. Open a web browser and navigate to your Workbench URL. Log in using the same credentials you used for your previous installation.
- 5. Review the **Projects** list to verify that all project data has been restored.

### **Additional configurations**

#### TLS/SSL certificates

If you did not configure SSL certificates as part of the post-install configuration, do so now. For more information, see *Updating TLS/SSL certificates*.

### **External git repository**

Workbench uses configurable parameters in the **External git** section of the anaconda-enterprise-anaconda-platform.yml file to connect to external git repositories. Verify your parameters are mapped correctly, *as described here*.

### Spark/Hadoop

After verifying your installation, run the following command on the *master node* of the Workbench server:

# Replace <PATH\_TO\_SECRETS.yaml> with the path to your anaconda secrets .yaml file sudo kubectl replace -f <PATH\_TO\_SECRETS.yaml>

To verify that your configuration upgraded correctly:

- 1. Log in to Workbench.
- 2. If your configuration uses Kerberos authentication, open a Hadoop terminal and authenticate yourself through Kerberos using the same credentials you used previously. For example, kinit <USERNAME>.
- 3. Open a Jupyter Notebook that uses Sparkmagic and verify that it behaves as expected by running the sc command to connect to Sparkmagic and start Spark.

### Cleaning up

As part of the upgrade process, the script you run automatically removes the unused packages and images from the previous installation and repopulates the registry to include only those images required by the current installation. This helps prevent the cluster from running out of disk space on the master node.

# 2.5.2 Bring Your Own Kubernetes (BYOK8s) Upgrade

Contact the Anaconda implementation team before you begin for assistance upgrading your version of Data Science & AI Workbench. It is intended for you to follow along with these instructions as an Anaconda Implementation team member guides you through the upgrade process. They will provide you with a Helm Chart archive that contains all the necessary components needed for upgrading.

**Caution:** Have all users save their work, stop any open sessions, jobs, and deployments, and log out of the platform prior to the upgrade process. Any sessions, deployments, or jobs not stopped prior to the upgrade process will not be using the latest docker images and must be restarted manually post upgrade.

### **Prerequisites:**

- You must have access to the *Administrative Server*.
- You must verify the Service Account used for the upgrade has the *correct permissions*.
- Air-gapped servers must have the Docker images added to their Docker image repository.

### **Upgrade process:**

Upgrade your BYOK8s cluster by performing the following steps:

- 1. Log in to the service account on the Administrative Server running your Workbench software.
- 2. Unpack the Helm Charts provided by Anaconda onto the Administrative Server:

```
# Replace <HELM_CHARTS> with the Helm Charts you received from Anaconda
tar xvzf <HELM_CHARTS>
```

3. Save your current configurations with the extract\_config.sh script delivered with your Helm Chart by running the following command:

```
# Replace <NAMESPACE> with the namespace Workbench is installed in
NAMESPACE=<NAMESPACE> ./extract_config.sh
```

The extract\_config.sh script creates a file called helm\_values.yaml and saves it in the directory where the script was run.

- 4. Verify the information captured in helm\_values.yaml file is correct and contains all of your current cluster configuration settings.
- 5. Begin the upgrade by running the following command:

```
helm upgrade --values ./helm_values.yaml anaconda-enterprise ./Anaconda-Enterprise/
```

If the upgrade is successful, your output will look like this:

```
Release "anaconda-enterprise" has been upgraded. Happy Helming!
```

NAME: anaconda-enterprise

LAST DEPLOYED: Wed Dec 7 21:52:34 2022

NAMESPACE: <YOUR\_NAMESPACE>

STATUS: deployed REVISION: 10 TEST SUITE: None

The duration of the upgrade depends on both the cluster node count and the speed of the new docker images loading into each node. You can view the status of the upgrade process by running the following command:

```
kubectl get pods
```

Once all pods display a **Running** status and each pod is running all their containers, you may return to using the cluster as normal.

**Caution:** If you are upgrading from Workbench version 5.6.2 or earlier *and* have RStudio installed, *you must reinstall RStudio*. This is due to the upgrade of the Workbench base image from UBI8 to UBI9 in version 5.7. UBI9 requires a different binary for RStudio.

# 2.5.3 Backing up and restoring Workbench

Backing up Data Science & AI Workbench protects your data in case of accidents (deletion of important data) or technical issues (failed hard drive). You can back up at any time, but refer to your company's Disaster Recovery policy for best practices.

**Warning:** Do not attempt to restore backup files created from a different version of Workbench. To upgrade your version of Workbench, reference Upgrading between versions of Workbench.

**Caution:** Anaconda recommends the use of managed persistence to ensure open sessions and deployments are captured by the backup process.

If you are not using managed persistence, have all users save their work, stop any open sessions and deployments, and log out of the platform during the backup process.

**Note:** The backup/restore script supports synchronizing your production cluster to a "hot" backup cluster at periodic intervals. This is commonly used for Disaster Recovery. To learn more about this process, please speak with our Integration Team.

### **Prerequisites**

- You have sudo access.
- The jq conda package is installed in your base environment.

or

Optionally, you can install this ae5-conda environment, which already contains the necessary packages.

## Install the backup script

**Note:** The ae5-conda environment mentioned in the prerequisites already contains the backup script. If you choose to install the environment, skip ahead to *verify your installation*.

### Standard environment backup script

Install the ae5\_backup\_restore package into your base conda environment:

conda install -c ae5-admin ae5\_backup\_restore

Once complete, verify the installation.

### Air-gapped environment backup script

Download the latest ae5\_conda installer file and move it to your master node.

Set the installer file to be executable, then run the script to install the ae5\_conda environment:

```
chmod +x ae5-conda-latest-Linux-x86_64.sh
./ae5-conda-latest-Linux-x86_64.sh
```

Once complete, verify the installation.

### Verify your installation

Verify your installation by testing a basic package command. Let's try the help command:

```
ae_backup.sh -h
```

If your terminal returns the usage help text, then your installation of the backup/restore script was successful! You are now ready to run the backup script.

### Run the backup script

Run the ae\_backup.sh script to create backup files of your cluster in the current directory:

```
bash ae_backup.sh
```

Or specify a destination for your backup files:

```
bash ae_backup.sh /your/file/path/here
```

The backup script creates two tarball files:

```
ae5_config_db_YYYYMMDDHHMM.tar.gz
ae5_data_YYYMMDDHHMM.tar.gz
```

**Note:** YYYYMMDDHHMM is the format for the timestamp of your backup data.

The ae5\_config\_db file stores your Kubernetes resources and Postgres data. The ae5\_data file stores your /opt/anaconda/storage data.

Note: The backup script does not back up the package repository.

### **Backup command line options**

- -h, --help: Prints help and exits.
- -d <DIR>, --ae-data <DIR>: Changes the location of the Workbench storage. The default location is /opt/anaconda/storage, and should not be changed when used on a Gravity cluster.
- -b <DIR>, --backup-dir <DIR>: Changes the location where the backup files are saved. The default location is the current directory. Use when the space available in the current directory is insufficient to hold the backup.
- -s, --skip-clean: If supplied, the script will not remove the intermediate files it generates during the backup process. This is useful for informational or debugging purposes.
- -c, --config-db: If supplied, the script will not create a data tarball, only the config/postgres tarball. This is useful if the script is combined with an alternate mechanism for taking snapshots/backups of the data.
- -r, --repository: If supplied, the script will include the full package repository in the data tarball. It doesn't do this by default because the repository is likely to be large and incompressible.

### Restore from backup data

**Warning:** The restore script requires backup files to be created from the same output of the backup script. Do not attempt to load files that were created from different backups.

Run the restore script to restore your cluster from previously-created backup data:

bash ae\_restore.sh ae5\_config\_db\_YYYYMMDDHHMM.tar.gz ae5\_data\_YYYYMMDDHHMM.tar.gz

#### **Restoration modes**

The restore script has three different modes for data restoration that can be used to customize how Workbench is restored.

#### Restoring to the original host

In this mode, all resources are restored from backup, except for the base ingress specification.

This mode is used when a clean reinstall of an existing cluster has been performed and you wish to perform a full restoration from backup. User workload will be restored (deployments, sessions, jobs), except they will be placed in a paused state. The script provides instructions on how to unpause user workload once the administrator is satisfied that the restoration has completed successfully.

### Restoring to a different host without a hostname change

In this mode, only some resources are restored, as described below.

Restored data:

- Kubernetes secrets (non-ssl)
- User/Project Data
- · Postgres

Non-restored data:

- Hostname
- · SSL certificates
- · Configmaps
- Ingress
- · Kubernetes resources for user workload

This mode is used if you wish to restore the backup to a separate existing cluster for inspection. By preserving the cluster's native configuration, the operation of the cluster is preserved but disconnected from the source.

### Restoring to a different host, but with a hostname change

This mode fully restores all resources, including the deployments and scheduled jobs. *The ingress is also updated in this case to reflect the new hostname*. This is used if you need to replace a faulty master node with a hot backup that was already running *under a different hostname*.

### **Restoration command line options**

- -h, --help: Prints help and exits.
- -d <DIR>, --ae-data <DIR>: Changes the location of the Workbench storage. The default location is /opt/anaconda/storage, and should not be changed when used on a Gravity cluster.
- -b <DIR>, --backup-dir <DIR>: Changes the location where the backup files are found. The default location is the current directory. Use when the space available in the current directory is insufficient to hold the backup.
- -s, --skip-clean: If supplied, the script will not remove the intermediate files it generates during the backup process. This is useful for informational or debugging purposes.
- -u, --update-hostname: If supplied and necessary, the script modifies the local hostname to match the backup content. Otherwise, the script preserves the local hostname.
- -c, --config-only: If supplied, the script only restores the configuration data (SSL, secrets, configmaps, etc.) It does not modify the Postgres database and the data.
- -w, --wait: When the system pods are restarted, wait for them to stabilize for exiting the script.
- -p, --pause: Leaves the cluster in a paused state upon completion of the restore process.
- -y, --yes: Restore function will not ask for confirmation before proceeding. Should be used with care.

## **Bring your own Kubernetes**

Customer supplied Kubernetes clusters (non-gravity) can take advantage of this backup/restore script. However the backup/restore process will be slightly different.

When taking a backup, you will need to supply the *-c, -config-db* command line argument, as the backup script will only be able to capture your Workbench configuration data. This will not capture user/project data, and you will need to ensure you are taking regular backups of your provided storage solution. This includes the Persistent Volume used for both anaconda-storage and anaconda-persistence that were configured at time of install.

When restoring from a backup, you will need to supply the -c, -config-only command line option, as the restore script will only be able to restore your Workbench configuration data. This will not restore user/project data, and you will need to ensure you have also restored a backup of your provided storage solution.

## 2.5.4 Migrating from Gravity to Bring Your Own Kubernetes (BYOK8s)

Contact the Anaconda implementation team before you begin for assistance migrating your version of Data Science & AI Workbench from Gravity to BYOK8s. Follow along with these instructions as an Anaconda implementation team member guides you through the migration process.

**Note:** If you are using network file storage (NFS) or dynamic storage for both your anaconda-storage and anaconda-persistence volumes within your Gravity cluster, it is not necessary to perform these steps, as you can unmount these volumes from your Gravity cluster and remount them on your BYOK8s cluster. An Anaconda implementation team member will guide you through this process, if necessary.

**Caution:** Have all users save their work, stop any open sessions and deployments, and log out of the platform so no data is lost during the migration process.

### **Prerequisites**

The following conditions must be met to complete this task successfully:

- You must have sudo access.
- The jq conda package must be installed in your base environment.

#### OR

• Optionally, you can install this ae5-conda environment, which already contains the necessary packages.

### Install the backup and restore tool

Migrating your data from Gravity to BYOK8s involves using the Workbench backup and restore tool, which you can obtain using conda.

**Note:** If you chose to install the ae5-conda environment listed in the prerequisites, skip ahead to *verify your installation*.

### Standard environment

For a standard environment, you can install the ae5\_backup\_restore package into your base conda environment by opening a terminal and running the following command:

conda install -c ae5-admin ae5\_backup\_restore

Once complete, verify the installation.

### Air-gapped environment

For Airgapped networks, download the latest ae5\_conda installer file and move it to your master node.

Set the installer file to be executable, then install the ae5\_conda environment by running the following commands:

```
chmod +x ae5-conda-latest-Linux-x86_64.sh
./ae5-conda-latest-Linux-x86_64.sh
```

Once complete, verify the installation.

### Verify your installation

Verify you've successfully installed the backup and restore tool by running the following command:

```
ae_backup.sh -h
```

If your terminal returns the usage help text for the backup script, your installation was successful.

### Run the backup script

When running the backup script, include the --config-db or -c command line argument to instruct the backup script to only capture your Workbench configuration data. User and project data is migrated separately later in this process.

To create a backup file of your cluster in your current working directory, run the command:

```
bash ae_backup.sh -c
```

To create a backup file of your cluster and store it in a specific directory, run the command:

```
# Replace <DIRECTORY_PATH> with the path to the directory you want to contain your backup
bash ae_backup.sh -c <DIRECTORY_PATH>
```

The backup script creates a tarball file that stores your Kubernetes resources and Postgres data with the following naming scheme:

```
ae5_config_db_YYYYMMDDHHMM.tar.gz
```

**Note:** YYYYMMDDHHMM is the format for the date timestamp of your backup data.

### Migrate user data

Before you can migrate user data, you must prevent users from making edits to the disk.

- 1. Open a terminal.
- 2. SSH into your Workbench master node.
- 3. Enter Gravity by running the following command:

```
gravity enter
```

4. Scale down the Anaconda Platform pods, identified by their ap- prefix, by running the following command:

```
kubectl get deploy | grep ap- | cut -d' ' -f1 | xargs kubectl scale deploy --

⊶replicas=0
```

5. Verify that the ap- prefixed pods have stopped by running the following command:

```
watch kubectl get pods
```

6. Once the ap- prefixed pods have stopped, migrate your user data by moving the following directories into the new cluster:

```
/opt/anaconda/storage/git/repositories/anaconda
```

/opt/anaconda/storage/projects

This can be done by either directly mounting the /opt/anaconda/storage volume onto a workstation with access to the BYOK8s cluster, or by compressing the directories separately, then copying them into the storage pod.

**Caution:** Migrating user data requires that both the Gravity and BYOK8s clusters have identical UID/GID permissions set. If permissions are not correctly aligned, your user data will not migrate.

To directly mount the /opt/anaconda/storage volume onto a workstation with access to the BYOK8s cluster, move both directories directly into place on top of the pre-existing directories on the BYOK8s cluster.

If you are unable to directly mount the /opt/anaconda/storage volume, you will instead need to copy both tarballs you have taken to the /tmp directory by running the following commands:

```
# Replace <PATH_TO_GIT_TARBALL> with the path to your /opt/anaconda/storage/git/
    repositories/anaconda tarball
# Replace <PATH_TO_PROJECTS_TARBALL> with the path to your /opt/anaconda/storage/
    projects tarball
# Replace <WORKBENCH_GIT_POD> with the ID of the Workbench git storage pod (the pod_
    prefix is "anaconda-enterprise-ap-git-storage-")
# Replace <WORKBENCH_PROJECTS_POD> with the ID of the Workbench project storage pod (the_
    pod prefix is "anaconda-enterprise-ap-storage")
kubectl cp <PATH_TO_GIT_TARBALL> <WORKBENCH_PROJECTS_POD>:/tmp
kubectl cp <PATH_TO_PROJECTS_TARBALL> <WORKBENCH_PROJECTS_POD>:/tmp
```

Once you have copied the tarballs to /tmp, exec into the pods, then move them into place by running the following commands:

### Run the restore script

### Restoring to a different host without a hostname change

Use this method if you are migrating to a cluster with a URL that differs from the URL of your Gravity cluster.

Restored data:

- Kubernetes secrets (non-ssl)
- · Postgres
- · ConfigMaps
- SSL certs
- · Secrets

Non-restored data:

- Hostname
- Ingress

Restore your data by running the following command:

```
bash ae_restore.sh ae5_config_db_YYYYMMDDHHMM
```

### Restoring to a different host with a hostname change

Use this method if you are migrating to a new cluster but you want to retain the Gravity cluster's URL.

Restore your data by running the following command:

```
bash ae_restore.sh -u ae5_config_db_YYYYMMDDHHMM
```

After you have moved user data over, and have completed running the Restore script, the migration is complete and you can confirm that your BYOK8s cluster contains all data from the old Gravity cluster.

# 2.5.5 Uninstalling Workbench

Before using the following instructions to uninstall Data Science & AI Workbench, be sure to follow the steps to *backup your current installation* so you'll be able to restore your data from the backup after installing Workbench 5.2.

To uninstall Workbench on a healthy cluster worker node, run:

```
sudo gravity leave
sudo killall gravity
sudo killall planet
```

To uninstall Workbench on a healthy cluster master node, run:

```
sudo gravity system uninstall
sudo killall gravity
sudo killall planet
sudo rm -rf /var/lib/gravity /opt/anaconda
```

To uninstall a failed or faulty cluster node, run:

```
sudo gravity remove --force
```

To remove an offline node that cannot be reached from the cluster, run:

```
sudo gravity remove <node>
```

Where <node> specifies the node to be removed. This value can be the node's assigned hostname, its IP address (the one that was used as an "advertise address" or "peer address" during install), or its Kubernetes name (which you can obtain by running kubectl get nodes).

## 2.5.6 Keycloak Upgrade

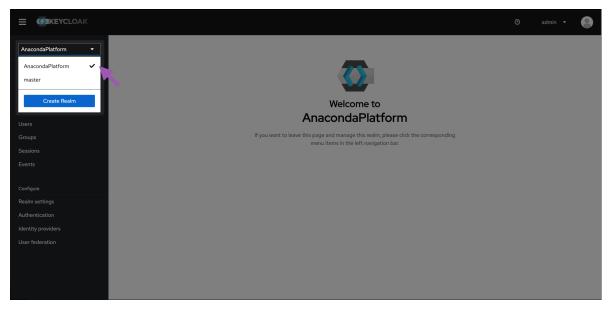
With the release of Data Science & AI Workbench 5.6, significant improvements have been made to our Keycloak implementation. For details, please see the release notes.

Upgrading to Workbench 5.6 requires Keycloak configuration changes to access your instance. You'll need to add a service account with correct permissions to the anaconda-platform client, then add a protocol mapper to the roles client scope.

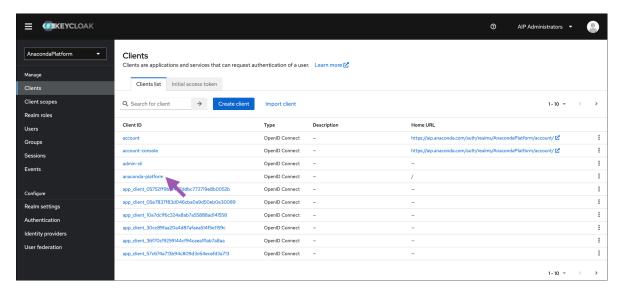
### **Enabling the service account**

After your upgrade to Workbench 5.6+ completes:

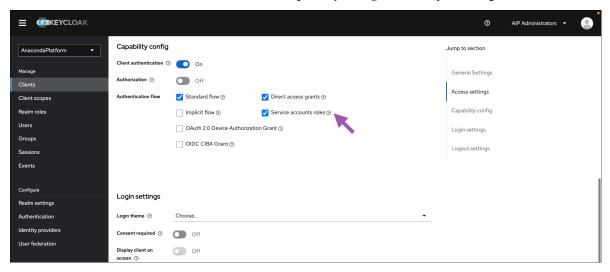
- 1. Open a browser and log in to your Keycloak admin panel using your existing Keycloak credentials. Your Keycloak admin panel can be found at https://<FQDN>auth/admin where <FQDN> is your Workbench fully qualified domain name.
- 2. Verify you are on the **anaconda-platform** realm.



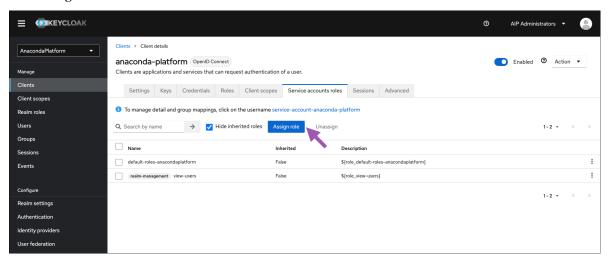
3. Select **Clients** from the left-hand navigation, then select **anaconda-platform** from the list of available clients.



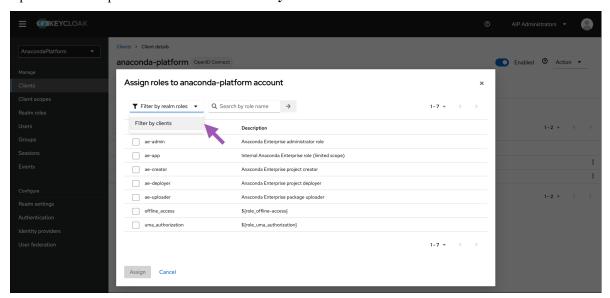
4. Select the Service accounts roles checkbox under Capability config, then save your changes.



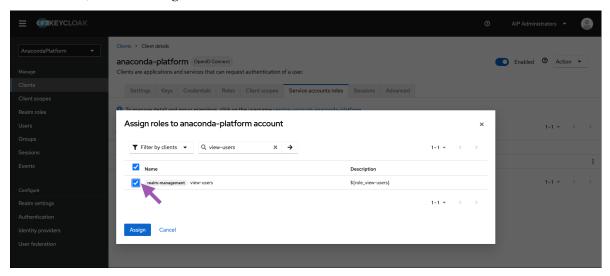
- 5. Select the new **Service accounts roles** tab that appears at the top of the page.
- 6. Click **Assign role**.



7. Open the filter dropdown menu and select **Filter by clients**.

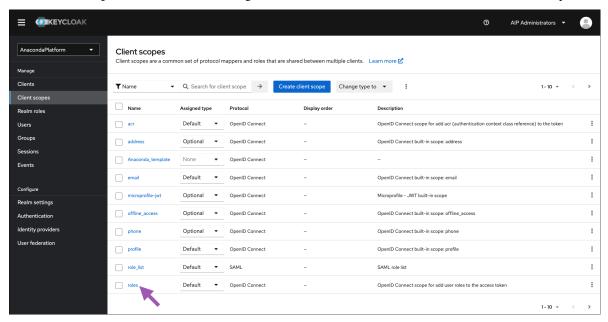


- 8. Search for the view-users role.
- 9. Select the role, then click **Assign**.

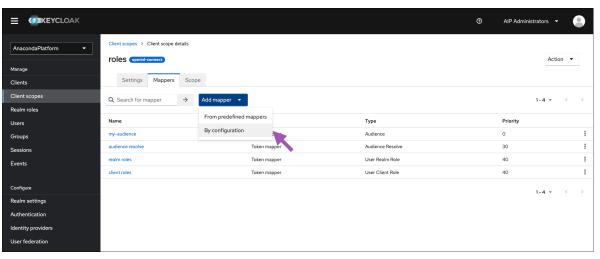


# Adding the protocol mapper

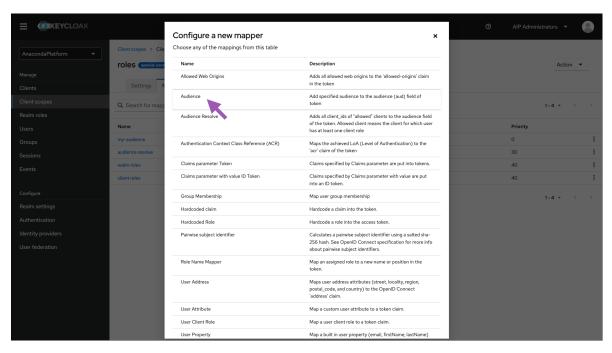
1. Select **Client scopes** from the left-hand navigation, then select **roles** from the list of available client scopes.



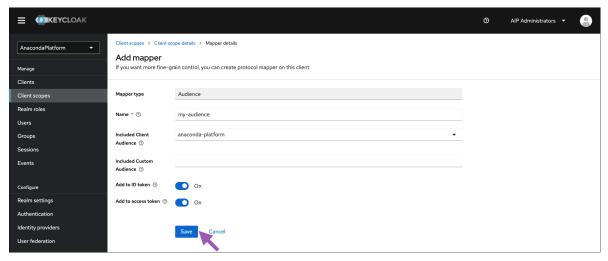
- 2. Select the **Mappers** tab.
- 3. Open the Add mapper dropdown menu and select By configuration.



4. Select Audience.



- 5. Complete the fields and set the toggle switches as indicated:
  - Name my-audience
  - Included Client Audience anaconda-platform
  - Add to ID token ON
  - · Add to access token ON
- 6. Click Save.



Success! You can now log in to your instance from an existing account and use Workbench normally.

# ADMINISTERING WORKBENCH

There are several aspects of Workbench that can be configured to meet your organization's specific requirements, including the following:

- Configuring and monitoring the utilization of *cluster resources*.
- Configuring user access to the platform and its resources.
- Configuring channels of packages, plus environments and custom installers to distribute software.
- Configuring advanced platform settings, including configuring Livy server for Hadoop Spark access, configuring external version control, and mounting NFS shares.

Administrators use different consoles to perform tasks in each of these areas, with credentials required to access each console. This gives enterprises the flexibility they need to choose whether to grant the permissions required to access a particular console to a single Admin, or different individuals, based on their area(s) of expertise within the organization.

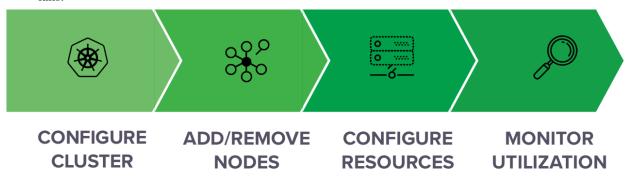
Some configuration options fall outside of these general categories—and you may not necessarily follow this linear process—however, the following offers a high-level overview of the configuration workflow you're likely to follow:



# 3.1 Managing cluster resources

After you've *installed an Data Science & AI Workbench cluster*, you'll need to continue managing and monitoring the cluster to ensure that it scales with your organization. These on-going tasks include:

- Adding nodes: If you outgrow your initial Workbench cluster installation, you can always add nodes to your cluster—including GPUs. Once added, make these nodes available to platform users by configuring resource profiles.
- Monitoring and managing resources: To help you manage your organization's cluster resources more efficiently, Workbench enables you to *monitor which sessions and deployments are running* on specific nodes or by specific users. You can also *monitor cluster resource usage* in terms of CPU, memory, disk space, network, and GPU utilization.
- **Troubleshooting and debugging:** To help you gain insights into user services and troubleshoot issues, Workbench provides *detailed logs* and debugging information related to the Kubernetes services it uses, as well as all *activity performed by users*. See *fault tolerance in Workbench* for information about what to do if a master node fails.



# 3.1.1 Adding and removing nodes (Gravity)

You can view, add, edit and delete server nodes from Data Science & AI Workbench using the Admin Console's Operations Center. If you would prefer to use a command line to join additional nodes to the Workbench master, follow the instructions provided below.

#### **NOTES:**

- Workbench does not support running heterogeneous versions in the same cluster. Before adding a new node, verify that the node is operating the same version of the OS as the rest of the cluster.
- If you're adding a GPU node, make sure it meets the GPU requirements.
- Each installation can only support a single Workbench master node, as this node includes storage for the platform. **DO NOT add an additional Workbench master node to your installation**.

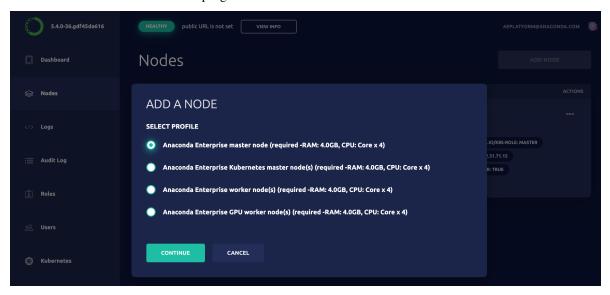
### To manage the servers on your system:

1. Log in to Workbench, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window. You must be logged in with a user assigned to the ae-admin role.

- 2. Click Manage Resources.
- 3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
- 4. Select **Nodes** from the menu on the left to display the configured nodes in your cluster, their IP address, hostname and profile.

# To add an existing server to Workbench:

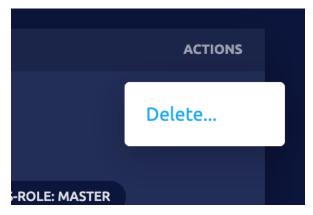
1. Click the Add Node button at the top right.



- 2. Select an appropriate profile for the server and click **Continue**.
- 3. Copy and paste the command provided into a terminal window to add the server. When you refresh the page, your server will appear in the list.

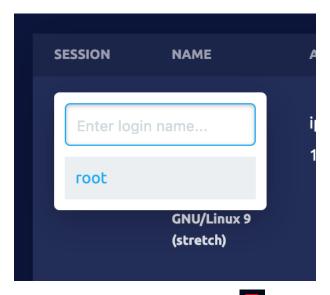
#### To remove a server node:

Click the Actions menu at the far right of the node you want to remove and select **Delete...**.



#### To log on to a server:

Click on the terminal icon of the server you want to work with, and select **root** to open a terminal window. It will open a new tab in your browser.



When you are finished, simply close the console window by clicking the icon.

# Using the command line to add nodes

- 1. Download the gravity binary that corresponds to your version of Workbench from the S3 location provided to you by Anaconda onto the server you're adding to the cluster.
- 2. Rename the file to something simpler, then make it executable. For example:

```
mv gravity-binary-6.1.9 gravity chmod +x gravity
```

3. *On the Workbench master*, run the following command to obtain the join token and IP address for the Workbench master node:

```
gravity status
```

The results should look similar to the following:

```
Cluster status:
                        active
Application:
                        AnacondaEnterprise, version 5.4.0-36.gdf45da616
Join token:
                        9954bf9f357b0eff8d2d2a4a48c8d9e6
Periodic updates:
                        Not Configured
                                                    Join Token
Remote support:
                        Not Configured
Last completed operation:
    * operation_install (9c1bc3ec-4ff4-4069-8096-9be6ac28960c)
                        Tue Oct 29 14:25 UTC (45 minutes ago)
      started:
                        Tue Oct 29 14:39 UTC (32 minutes ago)
      completed:
Cluster endpoints:
    * Authentication gateway:
        - 172.31.67.113:32009
                                             Master IP Address
    * Cluster management URL:
        - https://172.31.67.113:32009
Cluster nodes: test-install.training.anaconda.com
   Masters:
        * ip-172-31-67-113.ec2.internal / 172.31.67.113 / ae-master
            Status:
                        healthy
```

- 4. Copy and paste the join token for the cluster and the IP address for the Workbench master somewhere accessible. You'll need to provide this information when you add a new worker node. You'll also need the IP address of the server node you're adding.
- 5. On the worker node, run the following command to add the node to the cluster:

```
./gravity join --token JOIN-TOKEN --advertise-addr=NODE-IP --role=NODE-ROLE --cloud-
→provider=CLOUD-PROVIDER MASTER-IP-ADDR
```

## Where:

JOIN-TOKEN = The join token that you obtained in Step 3.

NODE-IP = The IP address of the worker node. This can be a private IP address, as long as the network it's on can access the Workbench master.

NODE-ROLE = The type of node you're adding: ae-worker, gpu-worker, or k8s-master.

CLOUD-PROVIDER = This is auto-detected, and can therefore be excluded unless you don't have Internet access. In this case, use generic.

MASTER-IP-ADDR = The IP address of the Workbench master that you obtained in Step 3.

**Warning:** The --role flag must be provided and assigned to either ae-worker, gpu-worker, or k8s-master. Without it, the node will be added with the role ae-master and may cause your cluster to crash.

The progress of the join operation is displayed:

```
[root@dtk-worker ~]# ./gravity join --token 24f51901025ebaa3e3e38db5767d3809751d9e40a6b4013089fe064658f56970 --advertise-addr-10.200.30.180 --role=ae-worker 10.200.30.165
Tue Nov 27 17:272:24 UTC Joining cluster
Tue Nov 27 17:272:24 UTC Connecting to cluster
Tue Nov 27 17:272:24 UTC Connected to existing cluster at 10.200.30.165
Tue Nov 27 17:27:26 UTC Operation has been created
Tue Nov 27 17:27:28 UTC Connecting the cecks
Tue Nov 27 17:27:28 UTC Configuring system packages
Tue Nov 27 17:27:28 UTC Configuring system packages
Tue Nov 27 17:42:55 UTC Registering with Kubernetes
Tue Nov 27 17:42:55 UTC Registering with Kubernetes
Tue Nov 27 17:42:57 UTC Updating Kubernetes node labels
[root@dtk-worker -]#
```

6. To monitor the impact of the join operation on the cluster, run the gravity status command *on the Workbench* master.

The output will look similar to the following:

```
Cluster status:
                        expanding
                        AnacondaEnterprise, version 5.4.0-36.gdf45da616
Application:
                        9954bf9f357b0eff8d2d2a4a48c8d9e6
Join token:
Periodic updates:
                        Not Configured
Remote support:
                        Not Configured
Active operations:
    * operation_expand (27542137-ee9d-492c-8749-0f06fd878fd6)
      started: Tue Oct 29 16:09 UTC (7 seconds ago)
      Pull packages on the joining node, 20% complete
Last completed operation:
    * operation_install (9c1bc3ec-4ff4-4069-8096-9be6ac28960c)
                        Tue Oct 29 14:25 UTC (1 hour ago)
     started:
                        Tue Oct 29 14:39 UTC (1 hour ago)
      completed:
Cluster endpoints:
    * Authentication gateway:
        - 172.31.67.113:32009
    * Cluster management URL:
        - https://172.31.67.113:32009
Cluster nodes: test-install.training.anaconda.com
   Masters:
        * ip-172-31-67-113.ec2.internal / 172.31.67.113 / ae-master
            Status:
                        healthy
   Nodes:
        * ip-172-31-72-9.ec2.internal / 172.31.72.9
                    offline
            Status:
```

Note that the size of the cluster is expanding and the status of the new node being added is offline. When the node has successfully joined, the cluster returns to an active state, and the status of the new node changes to healthy:

```
Cluster status:
                        active
Application:
                        AnacondaEnterprise, version 5.4.0-36.gdf45da616
Join token:
                        9954bf9f357b0eff8d2d2a4a48c8d9e6
Periodic updates:
                        Not Configured
                        Not Configured
Remote support:
Last completed operation:
    * operation_expand (27542137-ee9d-492c-8749-0f06fd878fd6)
      started:
                        Tue Oct 29 16:09 UTC (4 minutes ago)
                        Tue Oct 29 16:11 UTC (2 minutes ago)
      completed:
Cluster endpoints:
    * Authentication gateway:
        - 172.31.67.113:32009
    * Cluster management URL:
        - https://172.31.67.113:32009
Cluster nodes: test-install.training.anaconda.com
    Masters:
        * ip-172-31-67-113.ec2.internal / 172.31.67.113 / ae-master
            Status:
                        healthy
    Nodes:
        * ip-172-31-72-9.ec2.internal / 172.31.72.9 / ae-worker
            Status:
                        healthy
```

# 3.1.2 Configuring workload resource profiles

Each project editor session and deployment consumes compute resources within the Workbench cluster. If you need to run applications that require more memory or compute power than Anaconda provides by default, you can create customized resource profiles and configure the number of cores and amount of memory/RAM available for them. You can then allow users to access your customized resource profiles either globally or based on their role, assigned group, or as individual users.

For example, if your installation includes nodes with GPUs, you can add a GPU resource profile so users can access the GPUs to accelerate computation within their projects, which is essential for AI/ML model training.

Resource profiles that you create are listed for users when they *create* or *deploy* a project. You can create as many resource profiles as needed.

### Configuring resource profiles via ConfigMap

- 1. Connect to your instance of Workbench.
- 2. View a list of your configmaps by running the following command:

```
kubectl get cm
```

3. Edit the anaconda-enterprise-anaconda-platform.yml file.

```
kubectl edit cm anaconda-enterprise-anaconda-platform
```

**Caution:** Anaconda recommends making a backup copy of this file before you edit it. Any changes you make will impact how Workbench functions.

- 4. Find the resource-profiles: section of the file.
- 5. Add any additional resources using the following examples as a template for your resource profiles, then customize them for your environment:

# Resource profile examples

```
resource-profiles:
 resource-profile:
    description: Custom resource profile (global)
    resources:
     limits:
        cpu: "2"
       memory: 4096Mi
    user_visible: true
 roles_profile:
    description: Custom resource profile (roles)
    resources:
     limits:
        cpu: "3"
        memory: 4096Mi
   user_visible: true
    acl:
     roles:
        - ae-creator
 groups_profile:
    description: Custom resource profile (group)
    resources:
     limits:
        cpu: "4"
        memory: 4096Mi
    user_visible: true
    acl:
     groups:
        - managers
 users_profile:
    description: Custom resource profile (users)
    resources:
     limits:
        cpu: "1"
       memory: 4096Mi
    user_visible: true
    acl:
     users:
        - user2
```

(continues on next page)

(continued from previous page)

```
not_visible_to_anyone:
  description: Custom resource profile (nv)
  resources:
    limits:
      cpu: "3"
      memory: 4096Mi
  user_visible: false
gpu-profile:
  description: GPU resource profile
  resources:
    limits:
      cpu: "4"
      memory: 8Gi
      nvidia.com/gpu: 1
    requests:
      cpu: "1"
      memory: 2048Mi
      nvidia.com/gpu: 1
  user_visible: true
```

**Note:** Resource profiles display their description: as their name. Profiles are listed in alphabetical order, after the default profile.

6. (Optional) By default, CPU sessions and deployments are allowed to run on GPU nodes. To reserve your GPU nodes for sessions and deployments that require them, comment out the affinity: specification in the file as shown:

```
220
       221
222
       #affinity:
223
224
       # nodeAffinity:
225
          preferredDuringSchedulingIgnoredDuringExecution:
226
           - preference:
227
       #
             matchExpressions:
228
       #
              - key: gpu
229
       #
               operator: NotIn
230
       #
               values:
231
       #
               - "true"
232
            weight: 100
```

7. (Optional) If you need to schedule user workloads on a specific node, add a node\_selector to your resource profile. Use node selectors when running different CPU types, such as Intel and AMD; or different GPU types, such as Tesla v100 and p100. To enable a node selector, add node\_selector to the bottom of your resource profile, with the model: value matching the label you have applied to your worker node.

# **GPU** node selector example

```
gpu-profile:
    description: GPU resource profile
    resources:
        limits:
            cpu: "4"
            memory: 8Gi
            nvidia.com/gpu: 1
        requests:
            cpu: "1"
            memory: 2048Mi
            nvidia.com/gpu: 1
    user_visible: true
    node_selector:
        model: v100
```

- 8. Save your changes to the file.
- 9. Restart the workspace and deploy services by running the following command:

```
kubectl delete pods -l 'app in (ap-workspace, ap-deploy)'
```

### Configuring resource profiles via Helm chart

Anaconda's Kubernetes Helm chart encapsulates application definitions, dependencies, and configurations into a single values.yaml file. For more information about the Helm chart, see *Helm values template*.

- 1. Connect to your instance of Workbench.
- 2. Save your current configurations using the extract\_config.sh script by running the following command:

```
# Replace <NAMESPACE> with the namespace Workbench is installed in
NAMESPACE=<NAMESPACE> ./extract_config.sh
```

**Note:** The extract\_config.sh script creates a file called helm\_values.yaml and saves it in the directory where the script was run.

- 3. Verify that the information captured in helm\_values.yaml file contains your current cluster configuration settings.
- 4. Create a new section at the bottom of the helm\_values.yaml file and add any additional resources using the following examples as a template for your resource profiles, then customize them for your environment:

# Resource profile examples

```
resource-profiles:
 resource-profile:
   description: Custom resource profile (global)
   resources:
     limits:
        cpu: "2"
       memory: 4096Mi
   user_visible: true
 roles_profile:
   description: Custom resource profile (roles)
   resources:
     limits:
        cpu: "3"
       memory: 4096Mi
   user_visible: true
   acl:
     roles:
        - ae-creator
 groups_profile:
   description: Custom resource profile (group)
   resources:
     limits:
       cpu: "4"
       memory: 4096Mi
   user_visible: true
   acl:
     groups:
        - managers
 users_profile:
   description: Custom resource profile (users)
   resources:
     limits:
        cpu: "1"
       memory: 4096Mi
   user_visible: true
   acl:
     users:
       - user2
 not_visible_to_anyone:
   description: Custom resource profile (nv)
   resources:
     limits:
        cpu: "3"
       memory: 4096Mi
   user_visible: false
 gpu-profile:
```

(continues on next page)

(continued from previous page)

```
description: GPU resource profile
  resources:
    limits:
        cpu: "4"
        memory: 8Gi
        nvidia.com/gpu: 1
    requests:
        cpu: "1"
        memory: 2048Mi
        nvidia.com/gpu: 1
    user_visible: true
```

**Note:** Resource profiles display their description: as their name. Profiles are listed in alphabetical order, after the default profile.

5. (Optional) By default, CPU sessions and deployments are allowed to run on GPU nodes. To reserve your GPU nodes for sessions and deployments that require them, comment out the affinity: configuration in the file as shown:

```
COMPUTED VALUES:

#affinity:

#nodeAffinity:

#nodeAffinity:
```

6. (Optional) If you need to schedule user workloads on a specific node, add a node\_selector to your resource profile. Use node selectors when running different CPU types, such as Intel and AMD; or different GPU types, such as Tesla v100 and p100. To enable a node selector, add node\_selector to the bottom of your resource profile, with the model: value matching the label you have applied to your worker node.

#### **GPU** node selector example

```
gpu-profile:
  description: GPU resource profile
  resources:
    limits:
       cpu: "4"
       memory: 8Gi
       nvidia.com/gpu: 1
```

(continues on next page)

(continued from previous page)

```
requests:
    cpu: "1"
    memory: 2048Mi
    nvidia.com/gpu: 1
user_visible: true
node_selector:
    model: v100
```

7. Perform a Helm upgrade by running the following command:

```
helm upgrade --values ./helm_values.yaml anaconda-enterprise ./Anaconda-

⊸Enterprise/
```

Open a project and view its settings to verify that the resource profiles you added appear in the **Resource Profile** dropdown.

# 3.1.3 Monitoring cluster utilization

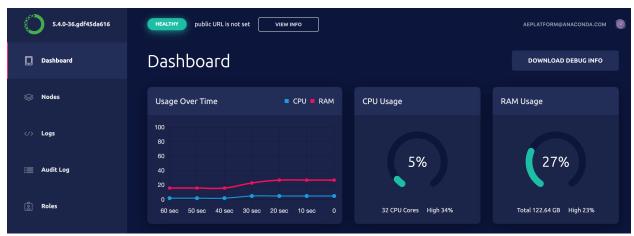
Data Science & AI Workbench enables you to monitor cluster resource usage in terms of CPU, memory, disk space, network and GPU utilization.

# To access the Operations Center:

- 1. Log in to Workbench, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Login to the Operations Center using the Administrator credentials configured after installation

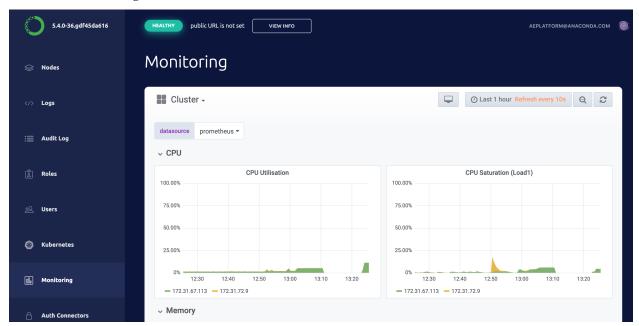
## Total cluster resource utilization

The **Dashboard** tab in the Operations Center displays the total CPU and Memory utilize aggregated across all nodes (master and worker) nodes in the Workbench cluster.



# Monitoring dashboard

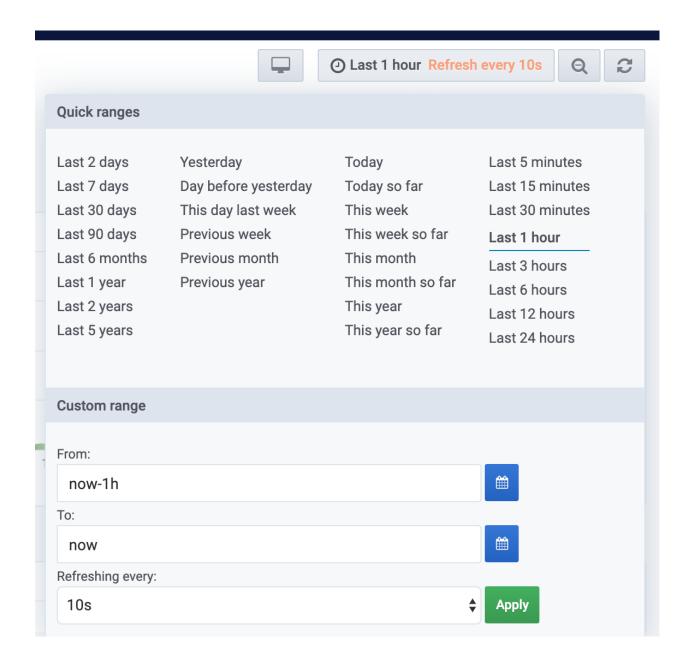
4. Click **Monitoring** in the menu on the left.



The graphs displayed include the following:

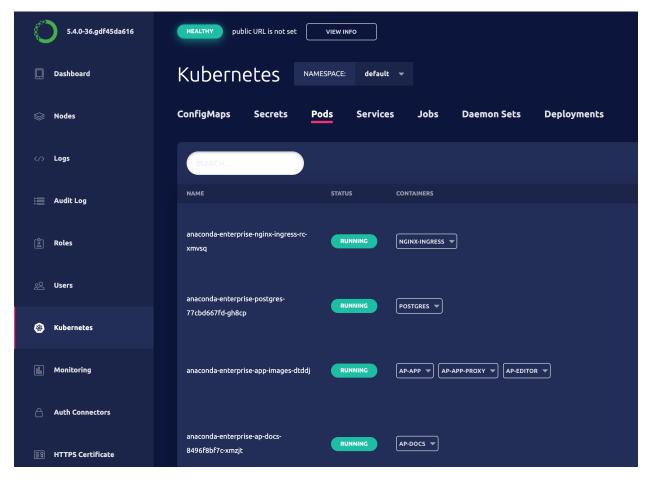
- Overall Cluster CPU Usage
- CPU Usage by Node
- Individual CPU Usage
- Overall Cluster Memory Usage
- Memory Usage by Node
- Individual Node Memory Usage
- Overall Cluster Network Usage
- Network Usage by Node
- Individual Node Network Usage
- Overall Cluster Filesystem Usage
- Filesystem Usage by Node
- Individual Filesystem Usage

Use the control in the upper right corner to specify the range of time for which you want to view usage information, and how often you want to refresh the results.



# **Monitoring Kubernetes**

To view the status of your Kubernetes nodes, pods, services, jobs, daemon sets and deployments from the Operations Center, click **Kubernetes** in the menu on the left and select **Pods**.



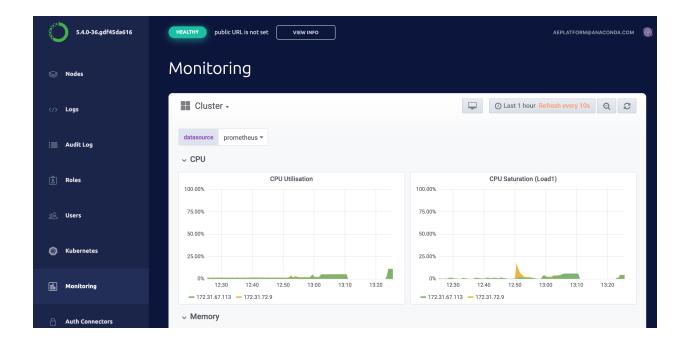
See Monitoring sessions and deployments for more information.

To view the status or progress of a cluster installation, click **Operations** in the menu on the left, and select an operation in the list. Clicking on a specific operation switches to the **Logs** view, where you can also view logs based on container or pod.

# 3.1.4 Monitoring sessions and deployments

Data Science & AI Workbench enables you to see which sessions and deployments are running on specific nodes or by specific users, so you can monitor cluster resource usage. You can also view session details *for a specific user* in the Authorization Center. See *Managing users* for more information.

- 1. Log in to Workbench, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
- 4. Select **Monitoring** from the menu on the left to display the monitoring dashboards.

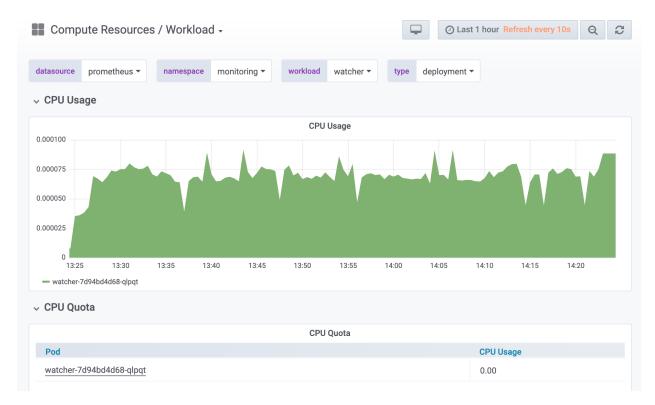


# Individual pod

To display the monitoring graph for a user session or deployment you'll need to identify the appropriate Kubernetes pod name.

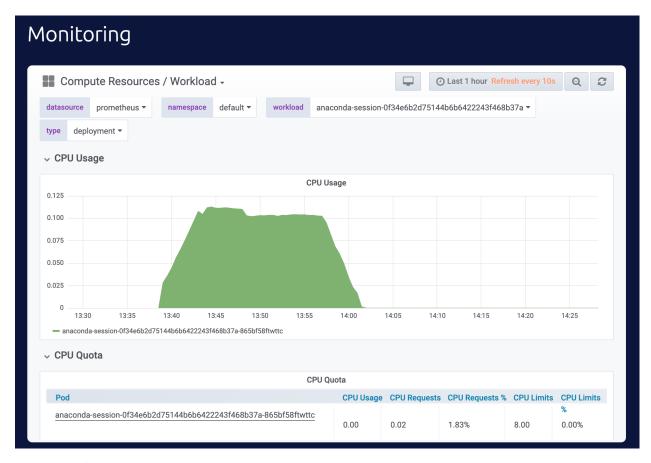
For an editor session the Kubernetes pod name corresponds to the hostname of the session container. Run hostname in a terminal window. For deployments the pod name is available from the logs tab of the deployment under the heading **name**.

- 1. Click the **Monitoring** tab from the menu on the left
- 2. Click Cluster at the top left of the dashboard
- 3. Select Compute Resource / Workload

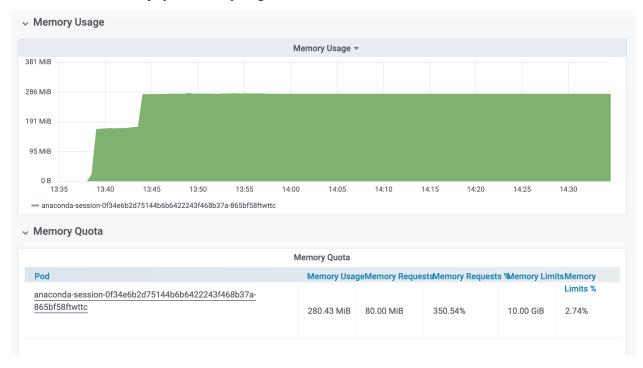


To display the monitoring graph for an individual pod

- 1. Select default from the namespace menu
- 2. Select the desired pod from the the workload menu



Scroll down further to display the memory usage.



**Using the CLI:** 

**Note:** For more expanded monitoring, see Workbench Tools.

- 1. Open an SSH session on the master node in a terminal by logging into the Operations Center and selecting **Servers** from the menu on the left.
- 2. Click on the IP address for the Workbench master node and select SSH login as root.
- 3. In the terminal window, run sudo gravity enter.

To view total node CPU and memory utilization run

```
kubectl top nodes --heapster-namespace=monitoring
```

To view CPU and memory utilization per pod run

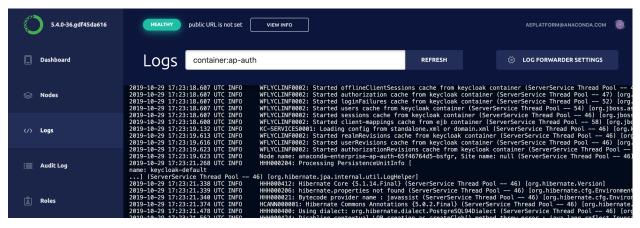
kubectl top pods --heapster-namespace=monitoring

# 3.1.5 Viewing system logs

To help you gain insights into user services and troubleshoot issues, Data Science & AI Workbench provides detailed logs and debugging information related to the Kubernetes services and containers it uses.

## To access these logs from the Operations Center:

- 1. Log in to Workbench, select the **Menu** icon in the top right corner, and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
- 4. Click **Logs** in the left menu to display the complete system log.
- 5. Use the **Filter** drop-down to view logs based on a container.



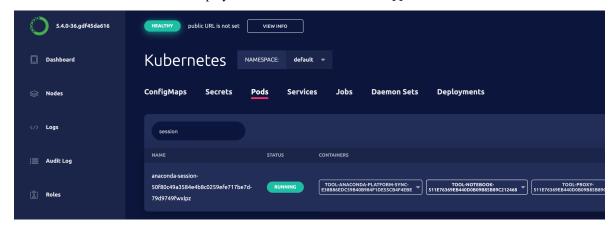
**Note:** You can also access the logs for a specific pod by clicking **Kubernetes** in the left menu, clicking the **Pods** tab, clicking the name of a pod, and selecting **Logs**.

# **Individual pods**

To display the logs for a user session or deployment you'll need to identify the appropriate Kubernetes pod name.

For an editor session the Kubernetes pod name corresponds to the hostname of the session container. Run hostname in a terminal window. For deployments the pod name is available from the logs tab of the deployment under the heading name.

- 1. Click the **Kubernetes** tab from the menu on the left.
- 2. Click the **Pods** tab to display a list of all pods and containers. Editor sessions are named anaconda-session-XXXXX and deployments are named anaconda-app-XXXX.



3. For the chosen pod click the pull-down button on an individual container to view the Logs or to gain SSH access.



# To use the CLI:

- 1. Open an SSH session on the master node in a terminal by logging into the Operations Center and selecting **Servers** from the menu on the left.
- 2. Click on the IP address for the Workbench master node and select SSH login as root.
- 3. In the terminal window, run sudo gravity enter.
- 4. Run kubectl get pods to view a list of all running session pods.
- 5. Run kubectl logs <POD-NAME> to display the logs for the pod specified.

# 3.1.6 Viewing activity logs

Data Science & AI Workbench logs all activity performed by users, including the following:

- · Each system login.
- · All Admin actions.
- Each time a project is created and updated.
- Each time a project is deployed.

In each case, the user who performed the action and when it occurred are tracked, along with any other important details.

As an Administrator, you can log in to the Administrative Console's Authentication Center to view the log of all login and Admin events:

- 1. Log in to Workbench, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Users.
- 3. Log in to the Authentication Center using the Administrator credentials required to access it.
- 4. Click **Events** in the left menu to display a log of all **Login Events**.
- 5. Click the Admin Events tab to view a summary of all actions performed by Admin users.

#### To filter events:

Event data can become difficult to manage as it accumulates, so Workbench provides a few options to make it more manageable:

- 1. Click the **Config** tab to configure the type of events you want Workbench to log, clear events, and schedule if you want to periodically delete event logs from the database.
- 2. Use the **Filter** options available on both the **Login Events** and **Admin Events** windows to control the results displayed based on variables such as event or operation, user or resource, and a range of dates.



- Click **Update** to refresh the results based on the filter you configured, and **Reset** to return to the original log results.
- Select the maximum number of results you want displayed: 5, 10, 50 or 100.

### To view activity at the project level:

- 1. Switch to the User Console and click **Projects** in the top menu.
- 2. Select the project you want to view information about to display a list of all actions performed on the project in the **Activity** window.

# 3.1.7 Fault tolerance in Workbench

Data Science & AI Workbench employs automatic service restarts and health monitoring to remain operational if a process halts or a worker node becomes unavailable. Additional levels of fault tolerance, such as service migration, are provided if there are *at least three nodes* in the deployment. However, the master node cannot currently be configured for automatic failover and does present a single point of failure.

When Workbench is deployed to a cluster with *three or more nodes*, the core services are automatically configured into a fault tolerant mode—whether Workbench is initially configured this way or changed later. As soon as there are three or more nodes available, the service fault tolerance features come into effect.

This means that in the event of any service failure:

- Workbench core services will automatically be restarted or, if possible, migrated.
- User-initiated project deployments will automatically be restarted or, if possible, migrated.

If a *worker node* becomes unresponsive or unavailable, it will be flagged while the core services and backend continue to run without interruption. If additional worker nodes are available the services that had been running on the failed worker node will be migrated or restarted on other still-live worker nodes. This migration may take a few minutes.

The process for adding new worker nodes to the Workbench cluster is described in *Adding and removing nodes (Gravity)*.

# Storage and persistency layer

Workbench does not automatically configure storage or persistency layer fault tolerance when using the default storage and persistency services. This includes the database, Git server, and object storage. If you have configured Workbench to use external storage and persistency services then you will need to configure these for fault tolerance.

# Recovering after node failure

Other than storage-related services (database, Git server, and object storage), all core Workbench services are resilient to master node failure.

To maintain operation of Workbench in the event of a master node failure, /opt/anaconda/ on the master node should be located on a redundant disk array or backed up frequently to avoid data loss. See *Backing up and restoring Workbench* for more information.

#### To restore Workbench operations in the event of a master node failure:

1. Create a new master node. Follow the installation process for adding a new cluster node, described in *command-line installations*.

Note: To create the new master node, select --role=ae-master instead of --role=ae-worker.

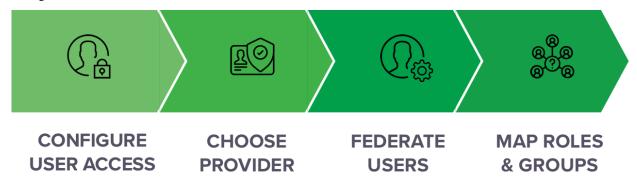
2. Restore data from a backup. After the installation of the new master node is complete, follow the instructions in *Backing up and restoring Workbench*.

# 3.2 Configuring user access

As an Administrator, you'll need to authorize users so they can use Workbench. This involves *adding users to the system*, setting their credentials, *mapping them to roles*, and optionally *assigning them to one or more groups*.

To help expedite the process of authorizing large groups of users, you can *connect to an external identity provider* such as LDAP or Active Directory and federate those users.

You'll need access to the Administrative Console's Authentication Center to be able to use it to configure identity and access management for Data Science & AI Workbench. Follow *these instructions* to grant Admins permission to manage Workbench users.



# 3.2.1 Connecting to external identity providers

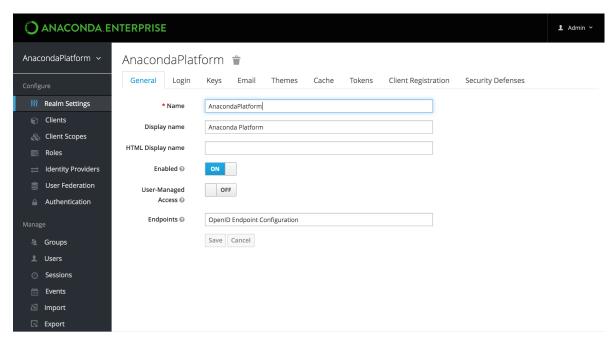
Data Science & AI Workbench comes with out-of-the-box support for LDAP, Active Directory, SAML and Kerberos. As each enterprise configuration is different, coordinate with your LDAP/AD Administrator to obtain the provider-specific information you need to proceed. We've also provided an *example of an LDAP setup* to help guide you through the process.

**Note:** You must have pagination turned off before starting.

# Adding a provider

You'll use the Administrative Console's Authentication Center to add an identity provider:

- 1. Login to Workbench, click the **Menu** icon in the top right corner, then click the **Administrative Console** link in the bottom of the slideout menu.
- 2. Click Manage Users.
- 3. Login to the Authentication Center using the Administrator credentials required to access it.



- 4. In the Configure menu on the left, select **User Federation**.
- 5. Select 1dap from the Add provider selector to display the initial **Required Settings** screen.

Multiple fields are required. The most important is the **Vendor** drop-down list, which will prefill default settings based on the LDAP provider you select. Make sure you select the correct one: Active Directory, Red Hat Directory Server, Tivoli, or Novell eDirectory. If none of these matches, select Other and coordinate with your LDAP Administrator to provide values for the required fields:

#### **Username LDAP attribute**

Name of the LDAP attribute that will be mapped to the username. Active Directory installations may use cn or sAMAccountName. Others often use uid.

#### **RDN LDAP attribute**

Name of the LDAP attribute that will be used as the RDN for a typical user DN lookup. This is often the same as the above "Username LDAP attribute", but does not have to be. For example, Active Directory installations may use cn for this attribute while using samacountName for the "Username LDAP attribute".

### **UUID LDAP attribute**

Name of an LDAP attribute that will be unique across all users in the tree. For example, Active Directory installations should use objectGUID. Other LDAP vendors typically define a UUID attribute, but if your implementation does not have one, any other unique attribute (such as uid or entryDN) may be used.

#### **User Object Classes**

Values of the LDAP objectClass attribute for users, separated by a comma. This is used in the search term for looking up existing LDAP users, and if read-write sync is enabled, new users will be added to LDAP with these objectClass values as well.

#### **Connection URL**

The URL used to connect to your LDAP server. Click **Test connection** to make sure your connection to the LDAP server is configured correctly.

#### Users DN

The full DN of the LDAP tree-the parent of LDAP users. For example, 'ou=users, dc=example, dc=com'.

#### **Authentication Type**

The LDAP authentication mechanism to use. The default is simple, which requires the Bind DN and password of the LDAP Admin.

#### **Bind DN**

The DN of the LDAP Admin, required to access the LDAP server.

#### **Bind Credential**

The password of the LDAP Admin, required to access the LDAP server. After supplying the DN and password, click **Test authentication** to confirm that your connection to the LDAP server can be authenticated.

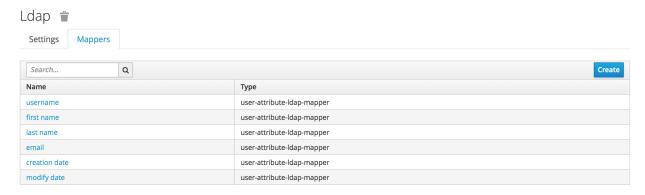
# **Configuring sync settings**

By default, users will not be synced from the LDAP / Active Directory store until they log in. If you have a large number of users to import, it can be helpful to set up batch syncing and periodic updates.



# **Configuring mappers**

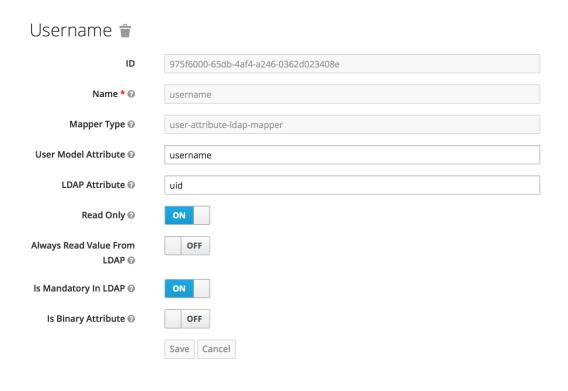
After you complete the initial setup, the auth system generates a set of "mappers" for your configuration. Each mapper takes a value from LDAP and maps it to a value in the internal auth database.



Go through each mapper and make sure it is set up appropriately.

- Check that each mapper reads the correct "LDAP attribute" and maps it to the right "User Model Attribute".
- Check that the attribute's "read-only" setting is correct.
- Check whether the attribute should always be read from the LDAP store and not from the internal database.

For example, the username mapper sets the Workbench username from the LDAP attribute configured.



## **Configuring advanced mappers**

Instead of manually configuring each user, you can automatically import user data from LDAP using additional mappers. The following mappers are available:

### User Attribute Mapper (user-attribute-ldap-mapper)

Maps LDAP attributes to attributes on the Workbench user. These are the default mappers set up in the initial configuration.

# FullName Mapper (full-name-ldap-mapper)

Maps the full name of the user from LDAP into the internal database.

### Role Mapper (role-ldap-mapper)

Sets role mappings from LDAP into realm role mappings. One role mapper can be used to map LDAP roles (usually groups from a particular branch of an LDAP tree) into realm roles with corresponding names.

Multiple role mappers can be configured for the same provider. It's possible to map roles to a particular client (such as the anaconda-deploy service), but it's usually best to map in realm-wide roles.

# Hardcoded Role Mapper (hardcoded-ldap-role-mapper)

Grants a specified role to each user linked with LDAP.

#### Hardcoded Attribute Mapper (hardcoded-ldap-attribute-mapper)

Sets a specified attribute to each user linked with LDAP.

# Group Mapper (group-ldap-mapper)

Sets group mappings from LDAP. Can map LDAP groups from a branch of an LDAP tree into groups in the Anaconda Platform realm. It will also propagate user-group membership from LDAP. We generally recommend using roles and not groups, so the role mapper may be more useful.

**Warning:** The group mapper provides a setting Drop non-existing groups during sync. If this setting is turned on, existing groups in Workbench Authentication Center will be erased.

## MSAD User Account Mapper (msad-user-account-control-mapper)

Microsoft Active Directory (MSAD) specific mapper. Can tightly integrate the MSAD user account state into the platform account state, including whether the account is enabled, whether the password is expired, and so on. Uses the userAccountControl and pwdLastSet LDAP attributes.

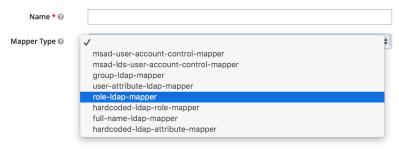
For example if pwdLastSet is 0, the user is required to update their password and there will be an UPDATE\_PASSWORD required action added to the user. If userAccountControl is 514 (disabled account), the platform user is also disabled.

# **Mapper configuration example**

To map LDAP group membership to Anaconda Platform roles, use a role mapper.

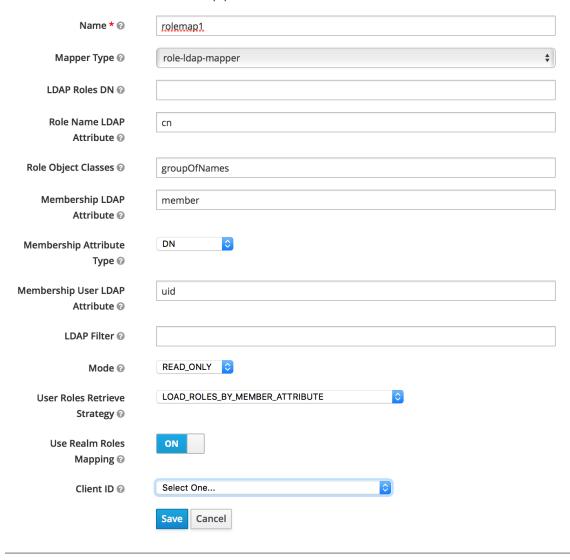
Add a mapper of the role-ldap-mapper type:

Add user federation mapper



In consultation with your LDAP administrator and internal LDAP documentation, define which LDAP group tree will be mapped into roles in the Anaconda Platform realm. The roles are mapped directly by name, so an LDAP membership of ae-deployer will map to the role of the same name in Anaconda Platform.

# Add user federation mapper



# **Authorizing LDAP groups and roles**

To authorize LDAP group members or roles synced from LDAP to perform various functions, add them to the anaconda-enterprise-anaconda-platform.yml configmap.

EXAMPLE: To give users in the LDAP group "Workbench", and users with the LDAP-synced role "Publisher", permission to deploy apps, the deploy section would look like this:

```
deploy:
    port: 8081
    prefix: '/deploy'
    url: https://abc.demo.anaconda.com/deploy
    https:
        key: /etc/secrets/certs/privkey.pem
        certificate: /etc/secrets/certs/cert.pem
```

(continues on next page)

(continued from previous page)

```
hosts:
    - abc.demo.anaconda.com
db:
    database: anaconda_deploy
users: '*'
deployers:
    users: []
    groups:
    - developers
    - Workbench
    roles:
    - Publisher
```

After editing the configmap, restart all pods for your changes to take effect:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

# **Configuring LDAPS (Outbound SSL)**

To make correct requests to secure internal resources such as internal enterprise LDAP servers using corporate SSL certificates, you must configure a "trust store". *This is optional*. If your internal servers instead use certificates issued by a public root CA, then the default trust store is sufficient.

To create a trust store, you must have the public certificates you wish to trust available.

Note: These are certificates for your trusted server such as Secure LDAP, not for Workbench.

# Option 1

If the CA certificates are directly available to you, run the following command, replacing CAFILE.cert with your CA certificate file:

```
keytool -import -file CAFILE.cert -alias auth -keystore LDAPS.jks
```

**Note:** If you want to add an intermediate certificate, run this command again **with a unique alias**, to include it in the LDAPS.jks file.

## Option 2

Alternatively, if you also have the server certificate and key, you can construct a full trust chain in the store.

1. Convert the certificate and key files to PKCS12 format—if they are not already—by running the following command:

```
openssl pkcs12 -export -chain -in CERT.pem -inkey CERT-KEY.pem -out PKCS-CHAIN.p12 - \rightarrowname auth -CAfile CA-CHAIN.pem
```

In this example, replace CERT.pem with the server's certificate, CERT-KEY.pem with the server's key, PKCS-CHAIN.p12 with a temporary file name, and CA-CHAIN.pem with the trust chain file (up to the root certificate of your internal CA).

2. Create a Java keystore to store the trusted certs:

```
keytool -importkeystore -destkeystore LDAPS.jks -srckeystore PKCS-CHAIN.p12 -alias_auth
```

3. You will be prompted to set a password. Record the password.

# **Final steps**

For both options, you'll need to follow the steps below to expose the certificates to the Workbench Auth service:

1. Export the existing SSL certificates for your system by running the following commands:

```
sudo gravity enter
kubectl get secrets anaconda-enterprise-certs --export -o yaml > /opt/anaconda/

secrets-exported.yml
```

2. Exit the gravity environment, and back up the secrets file before you edit it:

```
cp secrets-exported.yml secrets-exported-orig.yml
```

3. Run the following command to encode the newly created truststore as base64:

```
base64 -i --wrap=0 LDAPS.jks >> OUTPUT.jks
```

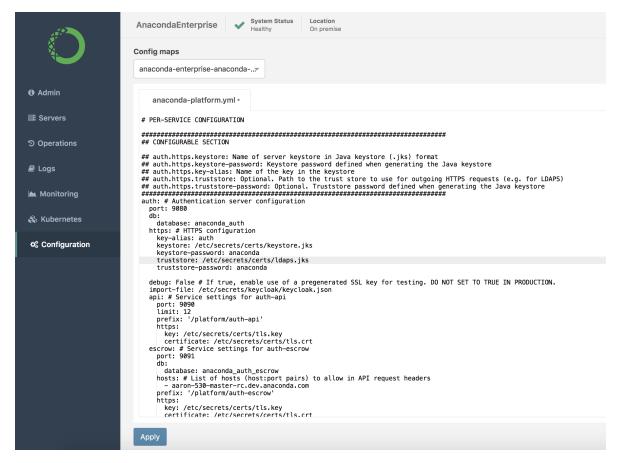
- 4. Copy the output of this command, and paste it into the data section of the secrets-exported.yml file.
- 5. Run the following commands to update Workbench with the secrets certificate:

```
sudo gravity enter
kubectl replace -f /opt/anaconda/secrets-exported.yml
```

6. Verify that the LDAPS. jks entry has been added to the secret:

```
kubectl describe secret anaconda-enterprise-certs
```

7. *Edit the platform configuration* by setting the auth.https.truststore configuration key to /etc/secrets/certs/ldaps.jks, and auth.https.truststore-password to the matching password. For example, after editing, it should resemble the following:



8. Run the following commands to restart the auth service:

```
sudo gravity enter kubectl get pods | grep ap-auth | cut -d' ' -f1 | xargs kubectl delete pods
```

**Caution:** If you are using Workbench version 5.6.0, you must also complete the following steps to finish your LDAPS configuration. If you do not, your LDAPS configuration will not connect successfully.

9. Create a backup of your current auth configuration:

```
kubectl get deployment anaconda-enterprise-ap-auth -o yaml > auth.yaml
```

10. Edit the auth pod deployment:

```
kubectl edit deployment anaconda-enterprise-ap-auth
```

11. Add the below to the JAVA\_OPTS key/value:

```
-Djavax.net.ssl.trustStore=/etc/secrets/certs/ldaps.jks
-Djavax.net.ssl.trustStorePassword=anaconda
```

12. Save your changes

# 3.2.2 Managing users

Managing access to Data Science & AI Workbench involves adding and removing users, setting passwords, *mapping users to roles, and optionally assigning them to groups*. To help expedite the process of authorizing large groups of users at once, you can *connect to an external identity provider* using LDAP, Active Directory, SAML, or Kerberos to federate those users.

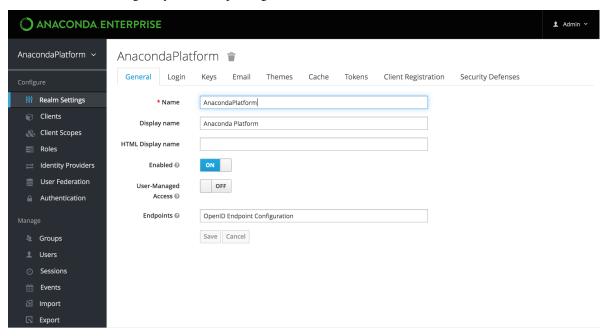
**Note:** To be able to perform these actions, you'll need the appropriate login credentials required to access the Administrative Console's Authentication Center.

The process of authorizing Operations Center Admins is slightly different. See *Managing System Administrators* for more information.

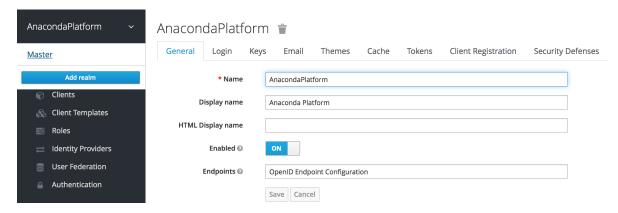
# Creating and configuring users

## To access the Authentication Center:

- 1. Login to Workbench, click the **Menu** icon in the top right corner, then click the **Administrative Console** link in the bottom of the slideout menu.
- 2. Click Manage Users.
- 3. If this is the first time accessing the Authentication Center, log in using the default admin credentials. Otherwise, use the credentials that grant you Admin privileges in the Authentication Center.



**Note:** To create and manage other Authentication Center Admins, use the realm selector in the upper left corner to switch to the **Master** realm before proceeding.



- 4. In the Manage menu on the left, click Users.
- 5. On the **Lookup** tab, click **View all users** to list every user in the system, or search the user database for all users that match the criteria you enter, based on their first name, last name, or email address.

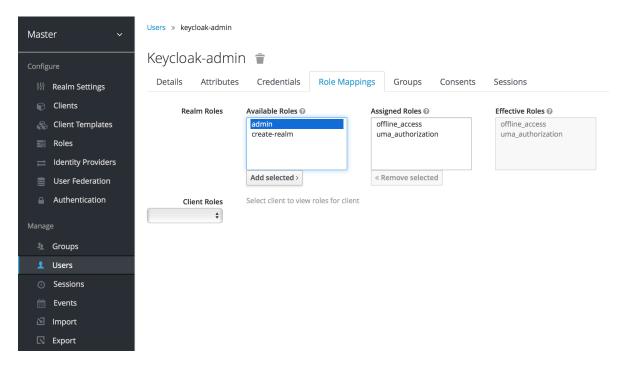
**Note:** This will search the local user database and not the federated database (such as LDAP) because not all external identity provider systems include a way to page through users. If you want users from a federated database to be synced into the local database, select **User Federation** in the Configure menu on the left, and adjust the **Sync Settings** for your user federation provider.

6. **To create a new Workbench user**, click **Add user** and specify a user name—and optionally provide values for the other fields—before clicking **Save**.

**Note:** Usernames containing unicode characters—special characters, punctuation, symbols, spaces—are not permitted.

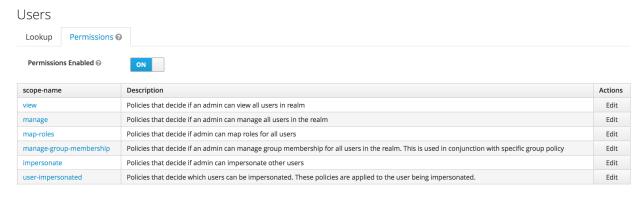
- 7. **To configure a user**, click the user's ID in the list and use the available tabs as follows:
  - Use the **Details** tab to specify information for the user, optionally enable user registration and required
    actions and impersonate the user. If you include an email address, an invitation to join Workbench will be
    sent to the email address specified.
  - Use the **Credentials** tab to manage the user's password. If the **Temporary** switch is on, this new password can only be used once—the user will be asked to change their password after they use it to log in to Workbench.
  - Use the **Role Mappings** tab to assign the user one or more roles, and the **Groups** tab to add them to one or more groups. See *managing roles and groups* for more information.

**Note:** To grant **Authentication Center Administrators** sufficient authority to manage Workbench users, you'll need to assign them the admin role.



• Use the **Sessions** tab to view a summary of all sessions the user has started, and log them out of all sessions in a single click. This is handy if a user goes on vacation without logging out of their sessions. You can use the Operations Center to view a summary of all sessions running on specific nodes or by specific users. See *monitoring sessions and deployments* for more information.

To view and edit a set of fine grain permissions that you can enable and use to define policies for allowing other users to manage users in the selected realm, return to the Users list and select the **Permissions** tab:



## Add a new master realm admin user/Reset password

Follow these steps from the command line to add a new admin user to the master realm *or* to reset your admin password if you're locked out or have forgotten your password.

1. Exec into the Keycloak container:

```
# Replace <KEYCLOAK_CONTAINER_ID> with your keycloak container ID
docker exec -it <KEYCLOAK_CONTAINER_ID> /bin/bash
```

2. Create a user:

3. Restart the server. Restarting the server will delete the container and any current state:

```
/ \texttt{opt/jboss/keycloak/bin/standalone.sh} \ - \texttt{Djboss.socket.binding.port-offset} = 100 \\
```

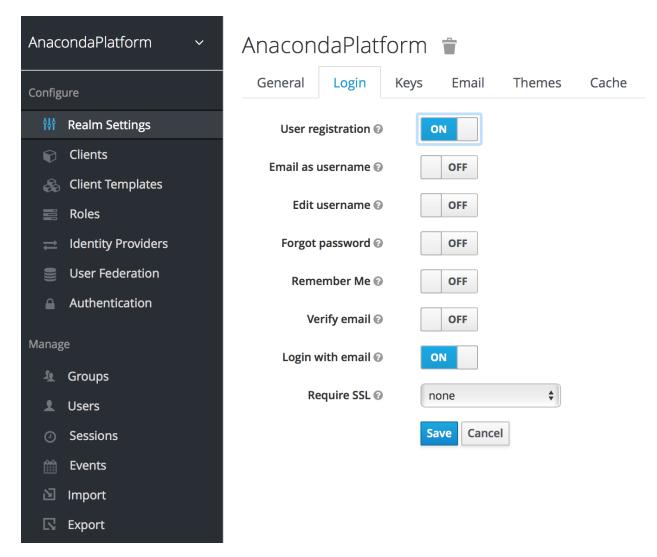
4. Log in to Keycloak from the following URL:

```
# Replace <FQDN> with your fully qualified domain name
<https://<FQDN>/auth/admin/master/console>.
```

# **Enabling user registration**

You can use the Authentication Center to enable users to self register and create their own account. When enabled, the login page will have a **Register** link users can click to open the registration page where they can enter the user profile information and password required to create their new account.

- 1. Click **Realm Settings** under Configure in the menu on the left menu.
- 2. Click the **Login** tab, and enable the **User registration** switch.



You can change the look and feel of the registration form as well as removing or adding additional fields that must be entered. See the Server Developer Guide for more information.

# **Enabling required actions**

You can use the **Required User Actions** drop-down list—on the **Details** tab for each user—to select the tasks that a user must complete (after providing their credentials) before they are allowed to log in to Workbench:

## **Update Profile**

This requires the user to update their profile information, such as their name, address, email, and phone number.

## **Update Password**

When set, a user must change their password.

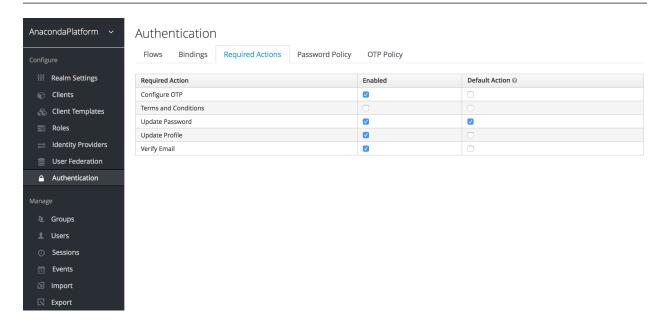
#### **Configure OTP**

When set, a user must configure a one-time password generator on their mobile device using either the Free OTP or Google Authenticator application.

# Setting default required actions

You can specify default required actions that will be added to all new user accounts. Select **Authentication** from the Configure menu on the left and use the **Required Actions** tab to specify whether you want each required action to be enabled—available for selection—or also pre-populated as a default for all new users.

**Note:** A required action must be enabled to be specified as a default.



## Using terms and conditions

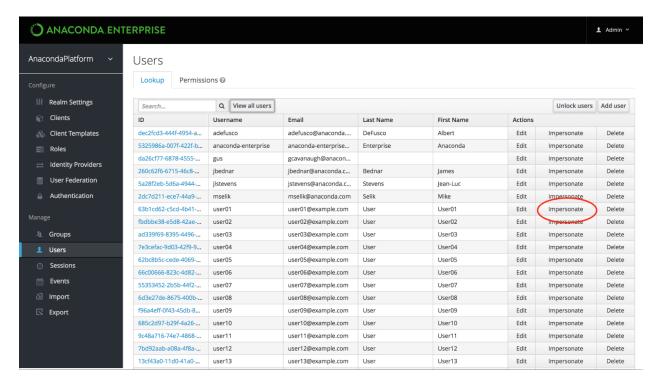
Many organizations have a requirement that when a new user logs in for the first time, they need to agree to the terms and conditions of the website. This functionality can be implemented as a required action, but it requires some configuration. In addition to enabling **Terms and Conditions** as a required action, you must also edit the terms.ftl file in the *base* login theme. See the Server Developer Guide for more information on extending and creating themes.

### Impersonating users

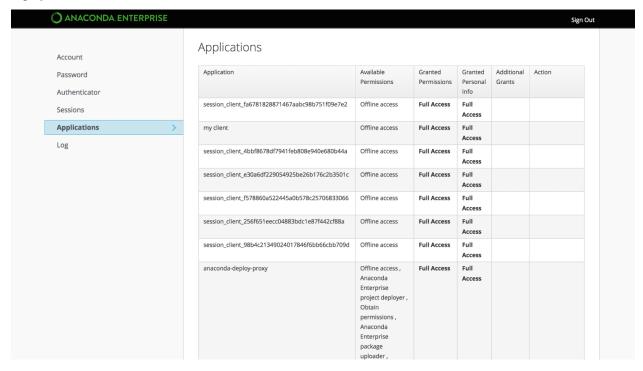
It is often useful for an Administrator to impersonate a user. For example, a user may be experiencing an issue using an application and an Admin may want to impersonate the user to see if they can duplicate the problem.

**Note:** Any user with the realm's impersonation role can impersonate a user.

The Impersonate command is available from both the Users list and the Details tab for a user.



Click **Impersonate** to display a list of applications the user has accessed on the platform, including editor sessions and deployments.



Click the Anaconda Platform link to interact with Workbench as the impersonated user.

	profile in  Account				
Anaconda Platform	Offline access , Anaconda Enterprise project deployer , Obtain permissions , Anaconda Enterprise package uploader , Anaconda Enterprise project creator , Manage account in <b>Account</b> , View profile in <b>Account</b>	Full Access	Full Access	Offline Token	Revoke Grant
session_client_68b7eda764c1465bb6dca2c72564160c	Offline access	Full Access	Full Access		
session_client_a1d00839825b44e592a2b3351841f00e	Offline access	Full Access	Full Access		
session_client_a5b7e3185bd3417fa8d15fdddb555760	Offline access	Full Access	Full Access		
Anaconda Enterprise Notebooks	Offline access , Anaconda Enterprise project deployer , Obtain	Full Access	Full Access		

**Note:** If the Admin and the user are in the same realm, the Admin will be logged out and automatically logged in as the user being impersonated. If the Admin and user are not in the same realm, the Admin will remain logged in and be logged in as the user in that user's realm.

# 3.2.3 Roles and groups

Assigning access and permissions for a user is a time consuming process. For an enterprise of hundreds or thousands of users, this would be too time consuming for an IT administrator to realistically perform.

Don't worry! Data Science & AI Workbench lets you utilize roles and groups within Keycloak to take this impossible task and break it down into a manageable workflow. But before you begin creating groups and assigning roles to them, you should understand exactly what each provides you with as an IT administrator.

# Roles

Roles determine the level of access a user has within Anaconda. Some custom roles have been embedded into Keycloak to provide users with varying levels of access to the software's available features.

- ae-admin provides the user with full access to the platform and the administrative console
- ae-creator allows the user to create new projects
- ae-deployer allows the user to create new deployments from projects
- ae-uploader allows the user to upload packages

If the default Anaconda roles do not suit your use case, you can create a custom role for your users.

# **Groups**

The most important idea to understand about groups is that any permissions that can be granted to an individual by assigning them a role can also be granted to multiple people by assigning the role to a group.

To get you started, Anaconda provides a set of default groups with different roles mapped for each. You can use these defaults as is, or as an example for creating your own groups.

The groups provided by anaconda are structured as follows:

- admins
  - Roles:
    - \* ae-admin
- developers
  - Roles:
    - \* ae-deployer
    - \* ae-uploader
    - \* ae-creator
- everyone
  - Roles:
    - \* none
- managers
  - Roles:
    - \* none
- · product managers
  - Roles:
    - \* ae-deployer
    - \* ae-uploader
    - \* ae-creator

# Managing roles and groups

For Workbench customers, Anaconda has several possible user personas. IT administrators, Business Analysts, Data Scientists, Dev Ops Engineers, and Data Engineers all benefit from using Workbench, but each persona also has different needs based on their work.

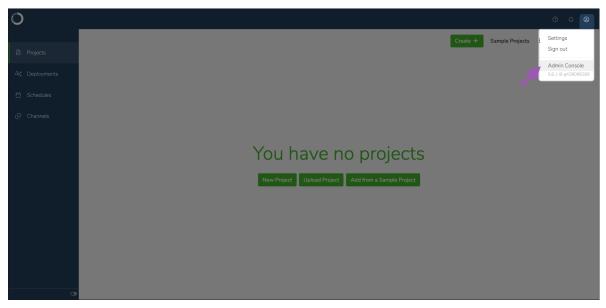
You can give every one of your users exactly what they need by using roles and groups!

As stated before, Anaconda utilizes Keycloak to manage roles and groups. This includes:

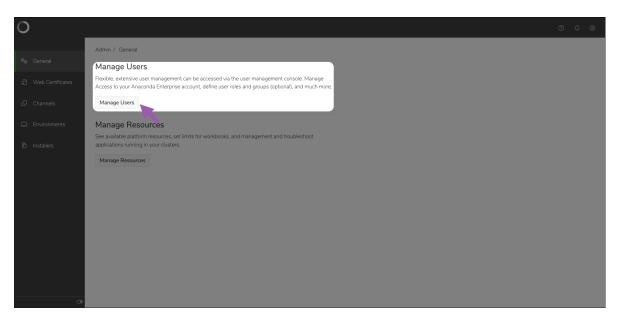
- Viewing available roles and groups
- · Creating new roles
- · Creating new groups
- Assigning roles to groups
- Configuring the default groups
- Configuring the default roles

# To access Keycloak:

- 1. Log in to Workbench.
- 2. Navigate to the **Admin Console**.

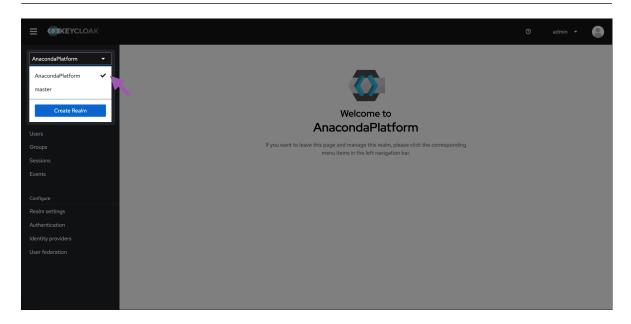


3. Select Manage Users to access the Keycloak user interface (UI).



4. Log in using your Keycloak admin account credentials.

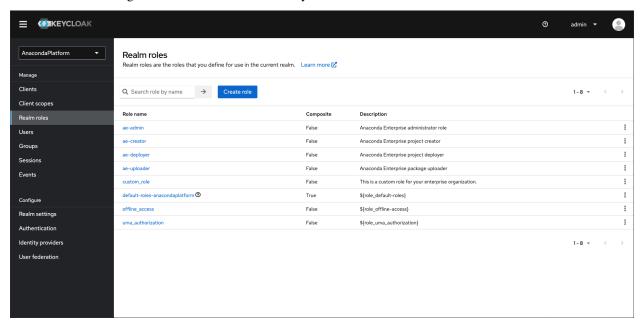
**Note:** Verify that you are working on the **AnacondaPlatform** realm when working with roles and groups for Workbench.



# Viewing roles and groups

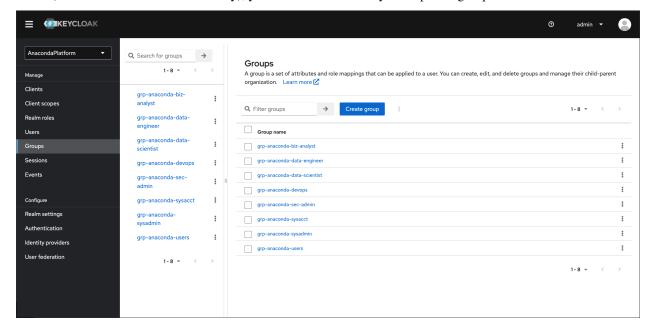
### **Roles**

Use the left-hand navigation to view Realm roles at any time.



## Groups

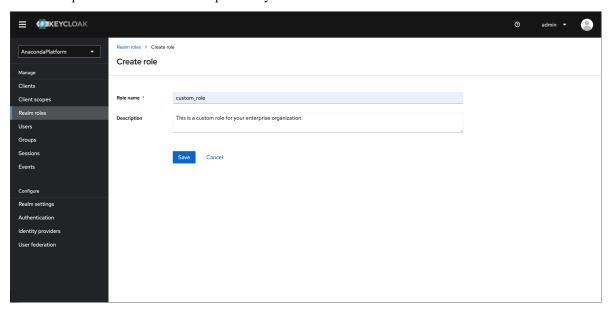
Likewise, use the left-hand navigation to view Groups at any time. If you are importing groups from an external identity server (such as *LDAP* or Active Directory), you will be able to view your imported groups here.



## **Creating custom roles**

To create a custom role for use in Anaconda:

- 1. From the **Realm roles** page, select **Create role**.
- 2. Enter a unique name and a brief description for your role.



3. Click Save. A notification will appear to inform you that your role has been created successfully.

# Assigning permissions for custom roles

Custom roles do not grant permissions upon creation. You must define the permissions for your custom role in your anaconda-enterprise-anaconda-platform.yml config map file.

Assign permissions to your custom role by completing the following steps:

- 1. Navigate to your Kubernetes cluster admin UI, or access your cluster using kubectl commands at the CLI.
- 2. Find your anaconda-enterprise-anaconda-platform.yml config map file.
- 3. Create and save a copy of this file before you begin assigning permissions for your custom role.

**Caution:** The changes you're about to make will impact how Workbench functions. The backup you've just created is a failsafe, in case you need to restore the previous configurations.

- 4. Open your anaconda-enterprise-anaconda-platform.yml config map file.
- 5. Locate the following sections of the file and add your role as needed to assign the correct level of permissions:
  - admin: users: Add your role here to provide it with full access to the platform and the administrative console.
  - deploy: deployers: Add your role here to allow it to create deployments from projects.
  - workspace: users: Add your role here to allow it to open project sessions.
  - storage: creators: Add your role here to allow it to create projects.

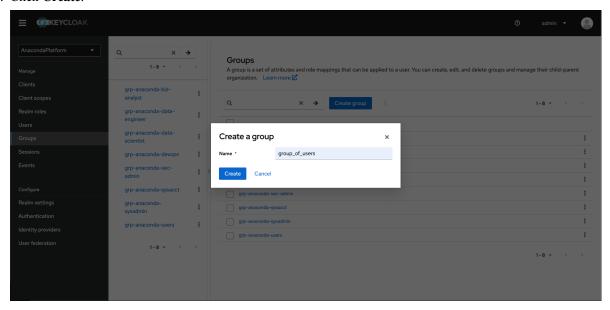
• repository: uploaders: - Add your role here to allow it to upload packages to the Workbench repository.

**Note:** Roles listed in the admin: users: section of the anaconda-enterprise-anaconda-platform. yml file must also be listed in each instance of superusers: and in ui: admin-links: admin: acl: to grant administrator level permissions.

# **Creating groups**

You can create as many groups as your organization needs. To create a new group, complete the following steps:

- 1. From the **Groups** page, select **Create group**.
- 2. Enter a unique name for your group.
- 3. Click Create.

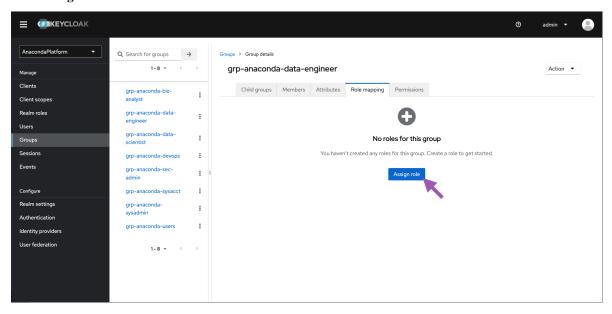


**Note:** Creating a group here will not add it to an externally connected server (such as LDAP).

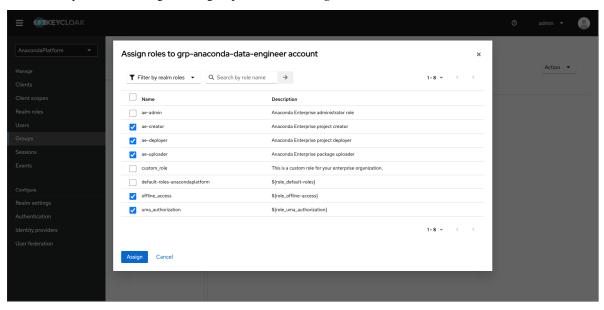
# Assigning roles to groups

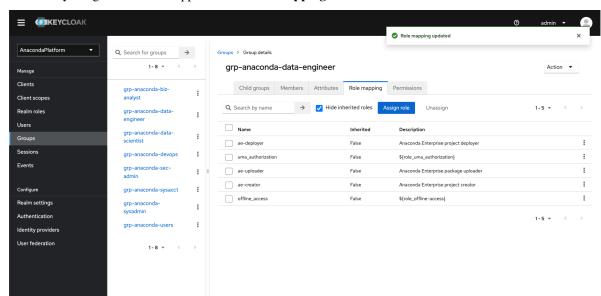
Assigning a role to a group provides all group members with the permissions granted by the role. To assign a role to a group, complete the following steps:

- 1. Navigate to the groups page and select a group to assign roles.
- 2. Select the Role mapping tab.
- 3. Click Assign role.



4. Select roles you want to assign to the group, then click Assign.





5. Your newly assigned roles will appear in the **Role mapping** tab.

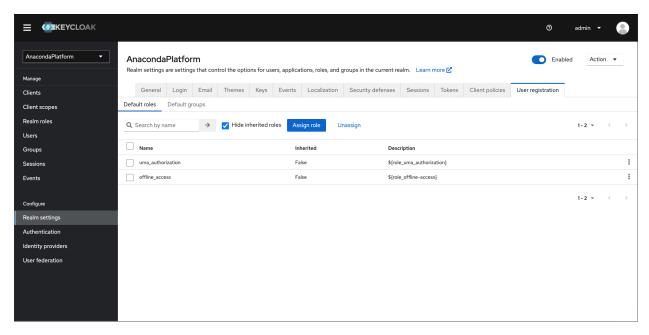
# **Default roles and groups**

## **Roles**

Default roles allow you to automatically assign user role mappings whenever a user is newly created or imported. By default, Workbench assigns the following roles to newly created and imported users:

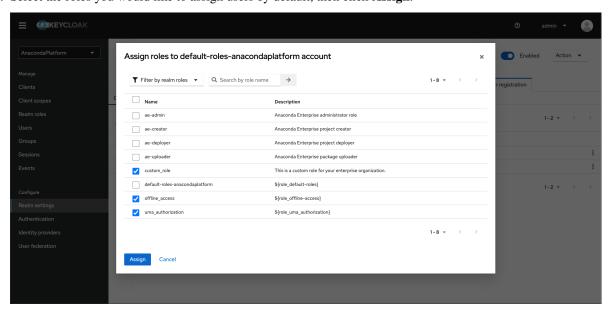
- · ae-deployer
- ae-uploader
- · ae-creator
- · offline\_access
- · uma\_authorization

You can disable these roles, if necessary, with the exception of the offline\_access and uma\_authorization roles. These are system configuration requirements.



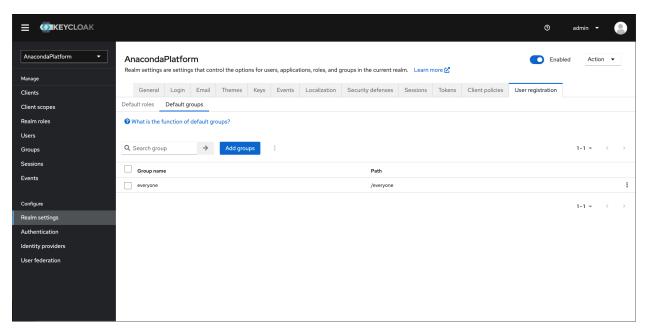
To assign default roles, complete the following steps:

- 1. Select **Realm settings** from the left-hand navigation.
- 2. Select the **User registration** tab.
- 3. Select the **Default roles** tab.
- 4. Click Assign role.
- 5. Select the roles you would like to assign users by default, then click **Assign**.



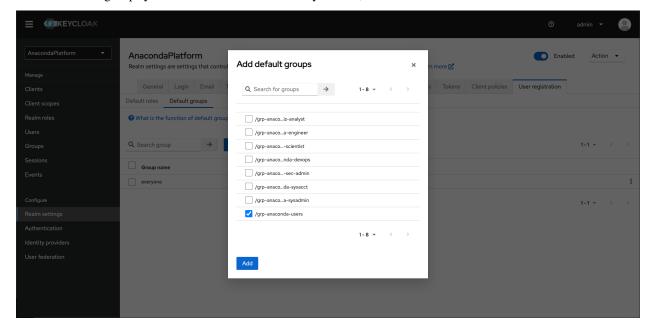
# Groups

Default groups allow you to automatically assign group membership whenever a new user is created or imported. By default, Anaconda adds all new users to the **everyone** group.



To assign default groups, complete the following steps:

- 1. Select **Realm settings** from the left-hand navigation.
- 2. Select the **User registration** tab.
- 3. Select the **Default groups** tab.
- 4. Click Add groups.
- 5. Select the groups you would like to add users to by default, then click Add.



# 3.2.4 Managing System Administrators

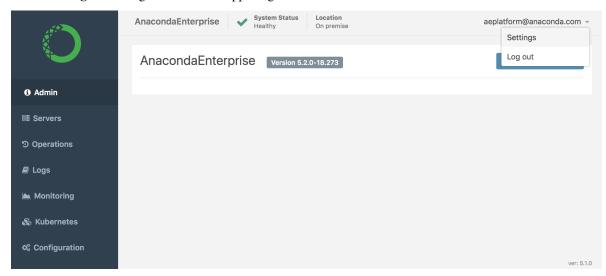
Data Science & AI Workbench distinguishes between System Administrators responsible for authorizing Workbench platform *users*, and System Administrators responsible for managing Workbench *resources*. This enables enterprises to grant the permissions required for configuring each to different individuals, based on their area of responsibility within the organization.

- Sys Admins who are granted permission to access the **Authentication Center** can *configure authentication for all platform users*, including platform Admins. See *managing users* for information on how to create and manage Authentication Center Admins.
- Sys Admins who are granted permission to access the **Operations Center** can *manage Workbench resources* and *configure advanced platform settings*.

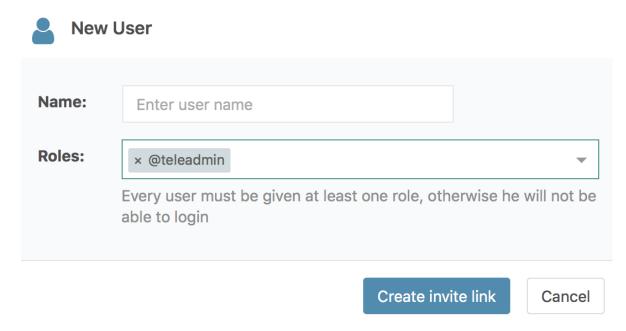
**Note:** The login credentials for the Operations Center are initially set as part of the *post-install configuration process*. Follow the steps outlined below to authorize additional Admin users to manage cluster resources, *using the Operations Center UI* or *using a command line*. If you prefer to use OpenID Connect (OIDC), see oidc.

# **Managing Operations Center Admins using the UI**

- 1. Log in to Workbench, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Login to the Operations Center using the Administrator credentials *configured after installation*.
- 4. Select **Settings** in the login menu in the upper-right corner.

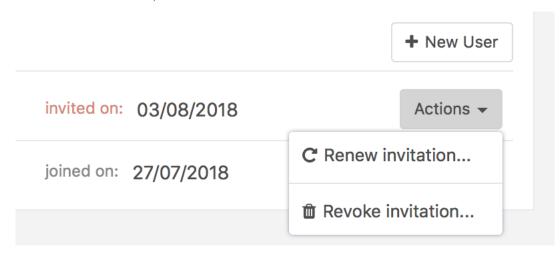


- 5. In the left menu, select **Users**, then click + **New User** in the upper-right corner.
- 6. Select @teleadmin from the Roles drop-down list, and click Create invite link.



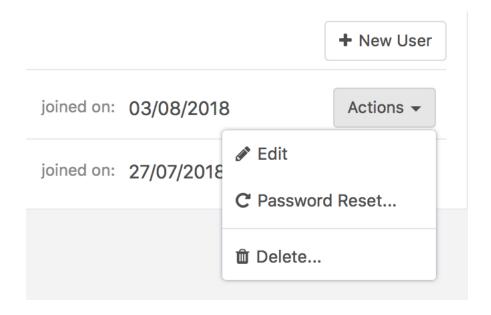
7. Copy the invitation URL that is generated, replace the private IP address with the fully-qualified domain name of the host, if necessary, and send it to the individual using your preferred method of secure communication. They'll use it to set their password, and will be automatically logged in to the Operations Center when they click **Continue**.

To generate a new invitation URL, select **Renew invitation** in the **Actions** menu for the user.



Select **Revoke invitation** to prevent them from being able to use the invitation to create a password and access the Operations Center. This effectively deletes the user before they have a chance to set their credentials.

To delete—or otherwise manage—an Operations Center user after they have set their credentials and completed the authorization process, select the appropriate option from the **Actions** menu.



# Managing Operations Center Admins using a command line

### To create a new Admin:

Run the following commands *on the Workbench master node*, replacing <email> and <yourpass> with the email address and password for the user:

```
sudo gravity enter gravity --insecure user create --type=admin --email=<email> --password=<yourpass> --ops-
--url=https://gravity-site.kube-system.svc.cluster.local:3009
```

To verify that the user was created, run the following command:

```
sudo gravity resource get users
```

## To update an Admin user's password:

To update an Admin user's password, you'll need to delete the user account, then re-create it, replacing <email> and <yourpass> with the email address and new password:

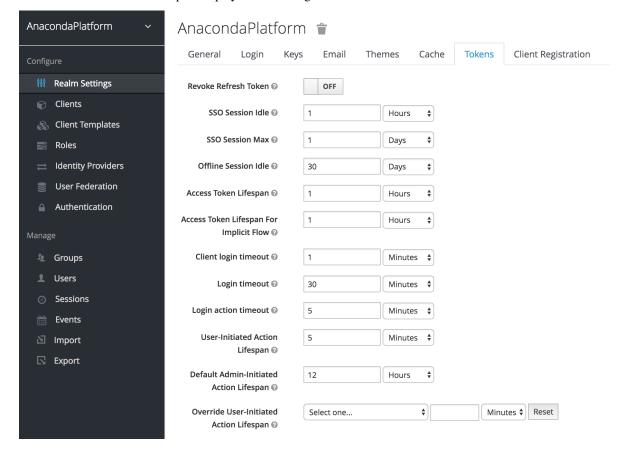
```
sudo gravity enter
gravity --insecure user delete --email=<email> --ops-url=https://gravity-site.kube-
system.svc.cluster.local:3009
gravity --insecure user create --type=admin --email=<email> --password=<yourpass> --ops-
surl=https://gravity-site.kube-system.svc.cluster.local:3009
```

# 3.2.5 Configuring session timeouts

As an Administrator, you can configure session timeouts for Data Science & AI Workbench platform users, to help you adhere to your organization's security standards or enforce policies.

You'll use the Administrative Console's Authentication Center to set the various parameters related to session timeouts:

- 1. Login to Workbench, click the **Menu** icon in the top right corner, then click the **Administrative Console** link in the bottom of the slideout menu.
- 2. Click Manage Users.
- 3. Login to the Authentication Center using the Administrator credentials required to be able to access it.
- 4. In the Configure menu on the left, select **Realm Setting**.
- 5. Click the **Tokens** tab at the top to display the following:



- 6. Use the available configuration options to specify maximum thresholds for each aspect of user sessions, including the following:
  - Time limits for idle browser sessions and single sign on (SSO) tokens
  - Lifespans for OpenID access tokens
  - · Time limits for login-related actions, such as resetting a forgotten password

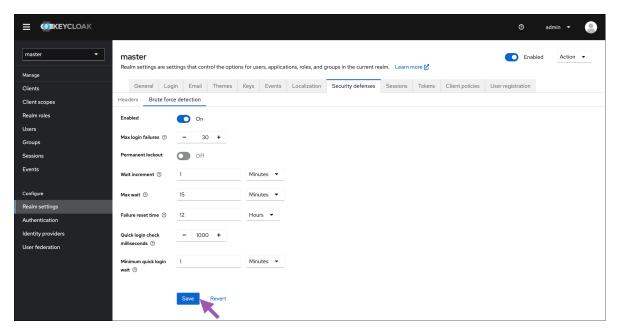
Configuration option	Description			
Revoke Refresh Token	If enabled, limits refresh tokens to one-time use			
SSO Session Idle	User will be logged out of session if inactive for this length of time			
SSO Session Max	Maximum time a user session can remain active, regardless of activity			
Offline Session Idle	Amount of time an offline session can be idle before the access token			
	is revoked			
Access Token Lifespan	Amount of time an access token will remain valid, before expiring			
Access Token Lifespan For Implicit	Timeout for access tokens created with Implicit Flow–no refresh token			
Flow	is provided			
Client login timeout	Maximum time a client can take to complete the authorization process			
Login timeout	Maximum time a user can take to authenticate before the process			
	restarts			
Login action timeout	Maximum time a user can spend on any one page in the authentication			
	process			
User-Initiated Action Lifespan	Maximum time before a user-initiated action (e.g., forgot password			
	email) expires			
Default Admin-Initiated Action	Maximum time before an admin-initiated action (e.g., issue token to			
Lifespan	user) expires			
Override User-Initiated Action	Use to optionally configure different timeouts for each user-initiated			
Lifespan	action			

7. Click **Save** to save your changes to the Workbench platform.

# 3.2.6 Enabling brute force protection

Keycloak provides a number of mechanisms to help secure your Workbench from identity-based attacks. A brute force attack is a method in which an attacker guesses your password by repeated guessing. To protect your installation against such attacks, follow these steps:

- 1. Log in to your Keycloak administrative console.
- 2. Select **Realm Settings** from the left-hand navigation menu.
- 3. Select the **Security Defenses** tab.
- 4. Select the **Brute Force Detection** tab.
- 5. Toggle Enabled to ON.
- 6. Set the parameters for your organization's brute force defenses. Hover your mouse over the question mark icon to see what each parameter manages.
- 7. Click Save.



To disable these settings at any time, return to the **Brute Force Detection** tab and toggle **Enabled** to OFF.

# Other security mitigations

For more information about brute force protection and using Keycloak to mitigate other security threats, please see Keycloak's official documentation.

# 3.2.7 Establishing an LDAP connection

The Lightweight Directory Access Protocol (LDAP) is a standardized set of rules that governs how information is transmitted and handled between a directory server and other servers on a network. Directory servers are external storage that contains information about users' identities. This could include their first and last name, email address, and employee ID number. Directory servers also often contain sensitive information, such as account passwords. For more information about LDAP, see the official LDAP documentation.

## Before you begin

Anaconda highly recommends you have knowledge of your LDAP server and organizational structure to complete this procedure. Configuring identity and access management is complex, and each enterprise has a unique LDAP directory structure.

While your implementation will be based on the specific structure and needs of your organization, the principals and processes described here will enable you to:

- Gather directory information about your LDAP server.
- Establish a connection to your LDAP server in Keycloak.
- Reduce the number of users that need to be mapped into Anaconda through the use of groups and roles.
- Reduce the number of groups that need to be mapped into Anaconda by filtering groups.
- Automate importing new groups for team memberships based on filters.
- Automate the provisioning of permissions to users based on group membership.

# **Prerequisites**

You must have credentials for the "bind user" service account to perform this task. If you do not have the proper credentials, get them from your LDAP Server Administrator.

## **Gathering directory structure information**

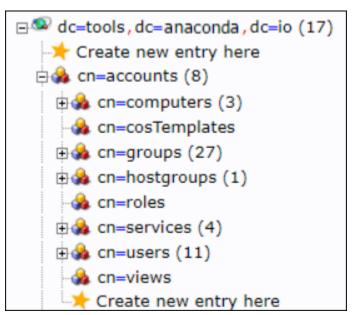
In order to configure Keycloak to validate credentials from your external LDAP server, you need to gather some information about your LDAP directory structure. You'll use this information to link user attributes (object classes, email, user ID, password, etc.) for use in Anaconda.

While you cannot discern a complete picture of your LDAP directory structure from the bind user credentials, you can make some general assumptions about it based on them. For example, if the bind user distinguished name (DN) is:

```
uid=binduser,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
```

We can see the "root" or "base" of the directory tree is dc=tools,dc=anaconda,dc=io. From there we can discern the rest of the tree structure. In this example, we can see that the uid attribute is stored in the *users* folder, which is stored in the *accounts* folder.

If you prefer, tools are available to aid in the visualization, navigation, and updating of your organization's LDAP directory server, such as phpldapadmin, which was used to generate the following view. This provides additional information about the LDAP structure that you can't discern from just looking at the bind credentials, such as the location of *groups*, which is also stored in the *accounts* folder.



Now that you better understand how to discern your LDAP directory structure, you are ready to use the ldapsearch tool, along with the bind user credentials, to learn details about an individual user based on their User ID. For more information about the ldapsearch tool, see the official documentation.

Gather the information you'll need to configure user federation within Keycloak by running the following command against a known user ID:

```
# Replace <BIND_USER> with your bind user address
# Replace <LDAP_ADDRESS> with the address of your LDAP server
# Replace <DIRECTORY_ROOT> with the root of your LDAP server
```

(continues on next page)

(continued from previous page)

```
# Replace <USER_ID> with the ID of a user on your LDAP sever
ldapsearch -D '<BIND_USER>' -W -H <LDAP_ADDRESS> -b <DIRECTORY_ROOT> "(<USER_ID>)"
```

Following the example for the bind user DN, to find information about user User1 the command would look like this:

```
ldapsearch -D 'uid=binduser,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io' -W -H ldap:/

→/ipa.tools.anaconda.io -b dc=tools,dc=anaconda,dc=io "(uid=User1)"
```

The return from the command will look like this:

```
# User1, users, compat, tools.anaconda.io
dn: uid=User1,cn=users,cn=compat,dc=tools,dc=anaconda,dc=io
objectClass: posixAccount
objectClass: ipaOverrideTarget
objectClass: top
gecos: User1
cn: User1
uidNumber: 1666600031
gidNumber: 1666600031
loginShell: /bin/sh
homeDirectory: /home/User1
ipaAnchorUUID::_
→OklQQTp0b29scy5jb250aW51dW0uaW860TEyYTMwNjgtZDhmYy0xMWU4LTgzYTUtMTIyYTE3YWN1MzJh
uid: User1
# User1, users, accounts, tools.anaconda.io
dn: uid=User1, cn=users, cn=accounts, dc=tools, dc=anaconda, dc=io
displayName: User1
uid: User1
krbCanonicalName: User1@TOOLS.anaconda.IO
objectClass: top
objectClass: person
objectClass: organizationalperson
objectClass: inetorgperson
objectClass: inetuser
objectClass: posixaccount
objectClass: krbprincipalaux
objectClass: krbticketpolicyaux
objectClass: ipaobject
objectClass: ipasshuser
objectClass: ipaSshGroupOfPubKeys
objectClass: mepOriginEntry
loginShell: /bin/sh
initials: U1
gecos: User1
sn: LastName
homeDirectory: /home/User1
mail: User1@tools.anaconda.io
krbPrincipalName: User1@TOOLS.anaconda.IO
givenName: User1
cn: User1
ipaUniqueID: 912a3068-d8fc-11e8-83a5-122a17ace32a
```

(continues on next page)

(continued from previous page)

```
uidNumber: 1666600031
gidNumber: 1666600031
krbPasswordExpiration: 20181026085310Z
krbLastPwdChange: 20181026085310Z
memberOf: cn=ipausers,cn=groups,cn=accounts,dc=tools,dc=anaconda,dc=io
memberOf: cn=grp-anaconda-data-scientist,cn=groups,cn=accounts,dc=tools,dc=anaconda,dc=io
memberOf: cn=grp-anaconda-users,cn=groups,cn=accounts,dc=tools,dc=anaconda,dc=io
memberOf: cn=grp-finance,cn=groups,cn=accounts,dc=tools,dc=anaconda,dc=io

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

Remember, the information you need to configure Keycloak to authenticate your users can be found in these results. Save this information and keep it somewhere accessible for reference.

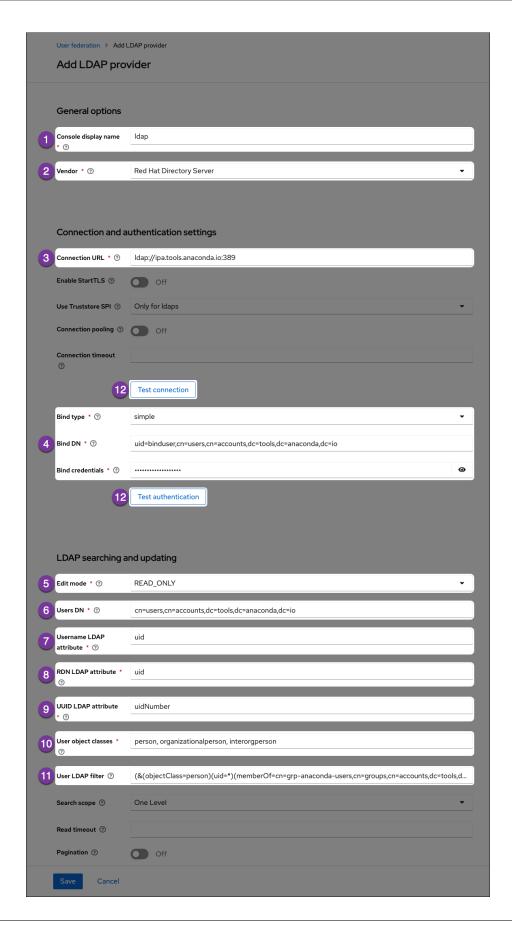
## Setting up LDAP user federation

Now that you have gathered information about your directory, you need to tell Keycloak how to interpret that information, so it can use it with Anaconda software.

**Note:** These attributes can be remapped later, if necessary.

- 1. Log in to your Keycloak console using an account that has administrator privileges.
- 2. Select **User federation** from the left-hand navigation menu.
- 3. Select **Idap** from the **Add provider...** dropdown menu.

Let's take a look at the Add LDAP provider page. Keep in mind that entries shown here are in alignment with the example provided above.



## 1. Console Display Name

Enter a name for Keycloak to display for the LDAP server.

#### 2. Vendor

Select your LDAP server vendor from the dropdown menu. If you do not know your vendor, ask your LDAP server administrator.

#### 3. Connection URL

Enter the LDAP server's URL here.

### 4. Bind User information

Enter the bind user information here.

### 5. Edit Mode

Set the edit mode to READ\_ONLY so you can view and import user information but don't have to worry about making unwanted changes to your LDAP server.

### 6. Users DN

Enter the LDAP directory location for your users here.

### 7. Username LDAP attribute

Get this information from your ldapsearch return. This attribute determines what is displayed as your user's name when they sign into Anaconda. In this example, the username attribute is uid.

**Caution:** Usernames cannot be all numeric values. All numeric usernames are indistinguishable from UUID's, and will break managed persistence if implemented.

## 8. RDN LDAP attribute

Get this information from your ldapsearch return. Usually, the relative distinguished name (RDN) attribute is the same as the username attribute, but this field may default to something else depending on your vendor.

# 9. UUID LDAP attribute

Get this information from your ldapsearch return. Your users' unique identifiers (UUID).

# 10. User object classes

Get this information from your ldapsearch return. Generally, the user object classes field will have more than one entry, separated by a comma.

## 11. User LDAP filter

The user LDAP filter restricts which users are returned from your LDAP directory. In the example, we only want users with the attribute objectClass=person that also have a uid and are in the group cn=grp-anaconda-users.

Because users must explicitly be added to the group, unauthorized access is prevented, and license management is simplified.

Filters also limit the need to synchronize a large number of objects from LDAP, which will help prevent out-of-memory errors in the auth pod.

**Caution:** Avoid the temptation to add new groups into the Custom User LDAP Filter! 1dapsearch utilizes regular expressions and is notorious for its complexity. If implemented incorrectly, a custom filter could cause all users to have their access suspended or be functionally disabled.

#### 12. Test buttons

Use the **Test connection** and **Test authentication** buttons to verify that Anaconda can connect to the provider with the credentials provided. You'll need to resolve any errors before continuing.

**Tip:** This is a good method for making sure your certifications are in the correct place if you are using LDAPS.

By default, users will not be synced from LDAP until they log in. To test whether the custom user LDAP Filter is working correctly, add or remove users in LDAP, then enable the sync settings to see if your changes are picked up and user authentication works as expected.

After you save your configurations, your LDAP server can be viewed on the User Federation page.

```
Caution: If you are using LDAPS, you must also complete the steps provided here.
```

Once you've saved your changes, you can navigate to **Users** in the left-hand navigation, then select **View all users** to verify that your users' information was imported correctly.

## **Configuring Group Mappers**

Once user federation is established, you can set up a *group mapper* for Keycloak to import your LDAP server's groups automatically for you. Once again, use the ldapsearch tool to gather information about your LDAP directory, only this time, look for information pertaining to your organizations groups.

To gather information about groups in your LDAP directory, run the following command against a known group DN:

```
# Replace <BIND_USER> with your bind user address
# Replace <LDAP_ADDRESS> with the address of your LDAP server
# Replace <DIRECTORY_ROOT> with the root of your LDAP server
# Replace <GROUP_DN> with the group's distinguished name
ldapsearch -D '<BIND_USER>' -W -H <LDAP_ADDRESS> -b <DIRECTORY_ROOT> "(<GROUP_DN>)"
```

Following with the earlier example, the command looks like this:

```
ldapsearch -D 'uid=binduser,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io' -W -H ldap:/

→/ipa.tools.anaconda.io -b dc=tools,dc=anaconda,dc=io "(cn=grp-anaconda-users)"
```

The return from the command will look like this:

```
# grp-anaconda-users, groups, compat, tools.anaconda.io
dn: cn=grp-anaconda-users,cn=groups,cn=compat,dc=tools,dc=anaconda,dc=io
objectClass: posixGroup
objectClass: ipaOverrideTarget
objectClass: ipaexternalgroup
objectClass: top
gidNumber: 1666600026
memberUid: User1
memberUid: User2
memberUid: User3
memberUid: User4
memberUid: User5
memberUid: User6
memberUid: User7
memberUid: User8
memberUid: User9
```

(continues on next page)

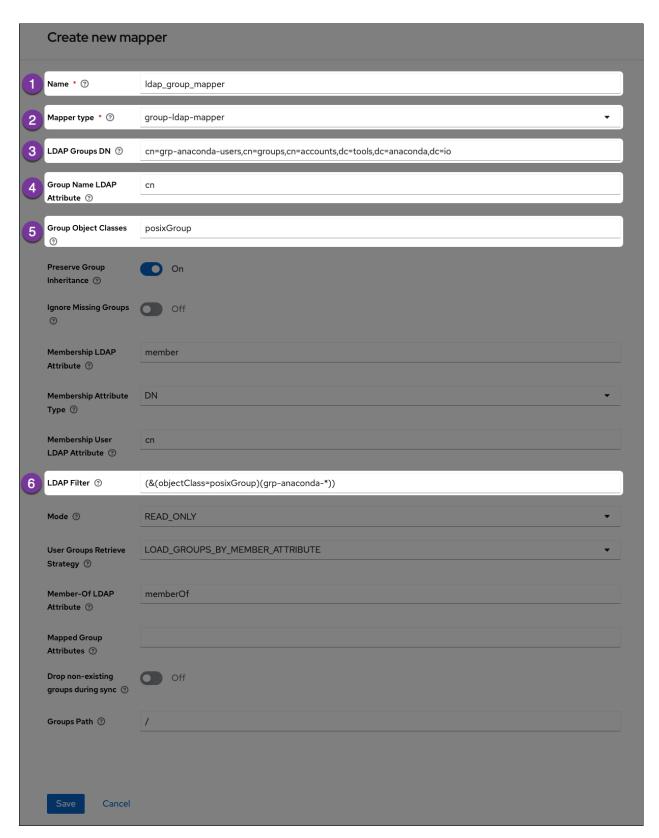
(continued from previous page)

```
ipaAnchorUUID:: 0klQQTp0b29scy5jb250aW51dW0uaW86NGFh0TQ4NzYtZDq4YS0xMWU4LWE2ZD
ctMTIyYTE3YWN1MzJh
cn: grp-anaconda-users
# grp-anaconda-users, groups, accounts, tools.anaconda.io
dn: cn=grp-anaconda-users,cn=groups,cn=accounts,dc=tools,dc=anaconda,dc=io
objectClass: top
objectClass: groupofnames
objectClass: nestedgroup
objectClass: ipausergroup
objectClass: ipaobject
objectClass: posixgroup
cn: grp-anaconda-users
ipaUniqueID: 4aa94876-d88a-11e8-a6d7-122a17ace32a
gidNumber: 1666600026
member: uid=User1,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
member: uid=User2,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
member: uid=User3,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
member: uid=User4,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
member: uid=User5,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
member: uid=User6,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
member: uid=User7,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
member: uid=User8,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
member: uid=User9,cn=users,cn=accounts,dc=tools,dc=anaconda,dc=io
# search result
search: 2
result: 0 Success
# numResponses: 3
# numEntries: 2
```

Remember, the information you need to create a group mapper can be found in these results. Save this information and keep it somewhere accessible for reference. Return to Keycloak and complete the following steps:

- 1. Select **User Federation** from the left-hand navigation.
- 2. Select your LDAP server.
- 3. Select the **Mappers** tab.
- 4. Select **Add mapper**.
- 5. Enter a name for your group mapper, such as *ldap\_group\_mapper*.
- 6. Select **group-ldap-mapper** from the **Mapper Type** dropdown menu. More options will appear based on your dropdown selection.

Let's take a look at the Create new mapper page. Keep in mind that entries shown here are in alignment with the example provided above.



## 1. Name

Enter a name for Keycloak to display for the LDAP group mapper.

## 2. Mapper Type

Select **group-ldap-mapper** from the dropdown menu.

## 3. LDAP Groups DN

Get this information from your ldapsearch return. Provide the distinguished name of the group you would like to map.

### 4. Group Name LDAP Attribute

Get this information from your ldapsearch return. Enter the attribute that is associated with groups. In this example, the attribute is cn.

## 5. Group Object Classes

Get this information from your ldapsearch return. This field will often have multiple entries, separated by a comma.

### 6. LDAP Filter

If you have established groups of users in your LDAP server that you plan to provide with access to Anaconda, you can import those specific groups by providing search filter criteria here. However, depending on how your LDAP server is structured, filtering to the correct groups can be complicated.

The search filter utilizes regular expressions (i.e. supports the use of wildcard characters). This example shows the search filter as grp-anaconda-\*, which will reach out to the LDAP server and import all groups that begin with grp-anaconda-.

**Caution:** If implemented incorrectly, a custom filter could cause all users to have their access suspended or be functionally disabled.

Select Save to create your ldap server mapper. An ID field appears at the top of the Mapper details page once it's established.

# Mapping group roles

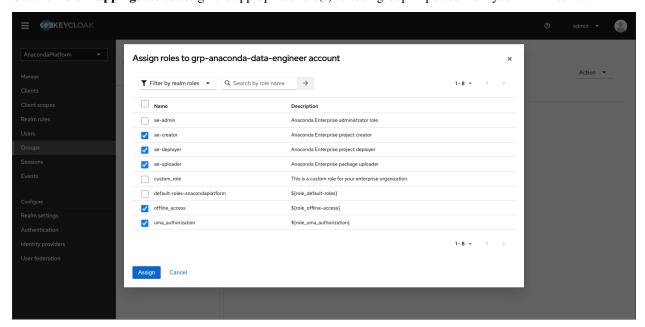
If you have groups of users that you plan to provide with access to Anaconda, you can import them using the LDAP filter as described above. Once complete, you need to provide your groups of users with permissions to work within Anaconda.

Roles determine a user's permissions within Anaconda, and a set of "realm" roles are established for you by default when you install. As a final step in establishing a connection to your LDAP server, you can map these provided roles to groups within your LDAP server to provide your team members with everything they need to work within Anaconda. Once the group roles are established, if a new team member arrives, adding them to the correct LDAP group provides them with all the correct permissions they need to use Anaconda right away. For more information about roles, see *Roles and groups*.

In the example below, groups have been established within the LDAP Server specifically for Anaconda users. Roles have been assigned to these groups to provide their members with the "proper" level of functionality within Anaconda for them to perform their jobs. A brief explanation on what their set role permissions will allow them to do has been provided in the right-hand column.

LDAP	ae-	ae-	ae-	ae-	of-	uma_a	a LED ressriz i i poti i pom
Group	admi	ncreat	odepl	oyuepoloa	a <b>dën</b> e_	access	
grp-					X		Business Analysts can access the system. They cannot create
anaconda	-						projects or grant others access to the system.
biz-							
analyst							
grp-		X	X	X	X	X	Data Scientists can create and share projects, but cannot de-
anaconda	-						ploy them.
data-							
scientist							
grp-		X		X	X	X	Data Engineers can additionally deploy projects, as well as
anaconda	-						grant access to others.
data-							
engineer							
grp-			X	X	X		DevOps can deploy projects and upload packages, but cannot
anaconda	-						create projects.
devops							
grp-							This group should be used to administer user access within
anaconda	-						the system. Therefore, no roles should be defined in the
sec-							AnacondaPlatform realm. If required, roles can be defined
admin							and access granted in the Auth Center Master realm.
grp-	X						By default, the ae-admin role is a superuser for all other
anaconda	-						roles.
sysadmin							
grp-							The roles for system accounts are yet to be defined. These
anaconda	-						could be used for automated CI/CD tasks.
sysacct							
grp-							This is used as a coarse-grained control for access to Work-
anaconda	-						bench, so no roles are defined.
users							

Use the **Role Mappings** tab to assign the appropriate role(s) to each group imported from your LDAP server.



# **Configuring LDAPS**

LDAP over SSL, or LDAPS, allows you to encrypt your LDAP server data while it travels during communications, in order to protect it from attacks like certificate theft. For more information, see the official Keycloak documentation on LDAPS.

## **Prerequisites**

You must have the Java jre package installed to complete this procedure. You must have SSL/TLS certificates for your LDAP server.

# **Establishing LDAPS**

# Gravity

If you are using a custom certificate authority (CA), you must create a truststore on your host. If you are using a public CA, you can skip the truststore installation and go straight to exporting your existing SSL certificates to the Data Science & AI Workbench auth service.

1. If the CA certificates are directly available to you, generate your truststore by running the following command:

```
# Replace <CA_CERT> with your with your CA .pem file
keytool -import -file <CA_CERT> -alias auth -keystore LDAPS.jks
```

**Note:** If you need to add an intermediate certificate, run this command again *with a unique alias*, to include it in the LDAPS. jks file.

2. Export the existing SSL certificates for your system by running the following commands:

```
# Replace <PATH_TO> with the filepath for your secrets-exported.yml file kubectl get secrets anaconda-enterprise-certs --export -o yaml > /<PATH_TO>/

--secrets-exported.yml
```

3. As a precautionary measure, create a backup of the secrets-exported.yml file prior to encoding it by running the following command:

```
cp secrets-exported.yml secrets-exported-orig.yml
```

4. Run the following command to encode the newly created truststore as base64:

```
base64 -i --wrap=0 LDAPS.jks >> OUTPUT.jks
```

- 5. Copy the output of this command, and paste it into the data section of the secrets-exported.yml file.
- 6. Run the following command to update Workbench with the secrets certificate:

```
# Replace <PATH_TO> with the filepath for your secrets-exported.yml file
kubectl replace -f /<PATH_TO>/secrets-exported.yml
```

7. Verify that the LDAPS. jks entry has been added to the secret:

kubectl describe secret anaconda-enterprise-certs

8. Run the following command to restart the auth service:

```
kubectl get pods | grep ap-auth | cut -d' ' -f1 | xargs kubectl delete pods
```

**Caution:** If you are using Workbench version 5.6.0 or later, you must also complete the following steps to finish your LDAPS configuration. If you do not, your LDAPS configuration will not connect successfully.

9. Create a backup of your current auth configuration:

```
kubectl get deployment anaconda-enterprise-ap-auth -o yaml > auth.yaml
```

10. Edit the auth pod deployment:

```
kubectl edit deployment anaconda-enterprise-ap-auth
```

11. Add the below to the JAVA\_OPTS key/value:

```
-Djavax.net.ssl.trustStore=/etc/secrets/certs/ldaps.jks
-Djavax.net.ssl.trustStorePassword=anaconda
```

Here is an example of what your auth.yaml file's JAVA\_OPTS might look like when complete.

```
- name: JAVA_OPTS
value: >-
    -Xms64m -Xmx2048m -XX:MetaspaceSize=96M
    -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true
    -Djboss.modules.system.pkgs=org.jboss.byteman
    -Djava.awt.headless=true
    -Djavax.net.ssl.trustStore=/etc/secrets/certs/ldaps.jks
    -Djavax.net.ssl.trustStorePassword=anaconda
```

12. Save your changes.

## BYOK8s

1. If the CA certificates are directly available to you, generate your truststore by running the following command:

```
# Replace <CA_CERT> with your with your CA .pem file keytool -import -file <CA_CERT> -alias auth -keystore LDAPS.jks
```

**Note:** If you need to add an intermediate certificate, run this command again *with a unique alias*, to include it in the LDAPS.jks file.

2. Encode the newly created truststore as base64 by running the following command:

```
base64 -i --wrap=0 LDAPS.jks >> <OUTPUT.jks>
```

3. Create the secret template for <0UTPUT.jks>

```
# Replace <OUTPUT.jks> with the encoded truststore file you just created
# Replace <TRUSTSTORE_SECRET> with the name of your LDAPS truststore secret
# Replace <NAMESPACE> with your Anaconda Enerprise namespace
kind: Secret
apiVersion: v1
metadata:
    name: <TRUSTSTORE_SECRET>
    namespace: <NAMESPACE>
data:
    <OUTPUT.jks>
type: kubernetes.io/tls
```

4. Create the new kubernetes secret by running the following command:

```
# Replace <SECRET_NAME> with a name for your truststore secret file
kubectl create -f <SECRET_NAME>
```

5. Save your configuration changes using the extract\_config.sh script by running the following command:

```
# Replace <NAMESPACE> with your Workbench namespace
NAMESPACE=<NAMESPACE> ./extract_config.sh
```

The extract\_config.sh script creates a file called helm\_values.yaml and saves it in the directory the script was run from.

- 6. Verify the information captured in the helm\_values.yaml file is correct and contains all of your current cluster configuration settings.
- 7. Open the helm\_values.yaml file and include the following lines:

```
# Replace <OUTPUT.jks> with your encoded truststore file
# Replace <TRUSTSTORE_PASS> with the password of thevtruststore
# Replace <TRUSTSTORE_SECRET> with the name of your truststore secret
keycloak:
    truststore: /etc/secrets/certs/<OUTPUT.jks>
    truststore_password: "<TRUSTSTORE_PASS>"
    truststore_secret: <TRUSTSTORE_SECRET>
```

#### **Example config variables**

```
keycloak:
    truststore: /etc/secrets/certs/ldaps.jks
    truststore_password: anaconda
    truststore_secret: ldaps_secret
```

8. Update your Helm Chart by running the following command:

```
helm upgrade --values ./helm_values.yaml anaconda-enterprise ./Anaconda-

Genterprise/
```

# 3.2.8 Google IAM setup example

In addition to providing out-of-the-box support for LDAP, Active Directory, SAML and Kerberos, Data Science & AI Workbench also enables you to configure the platform to use other external identity providers to authenticate users. If your enterprise uses Google's Cloud IAM (Identity and Access Management) to manage access to Google Cloud Platform (GCP) resources, for example, you can use the following process to configure the platform to use Cloud IAM as your identity provider. This will allow users to log in to the platform using their Google (or G-Suite) credentials.

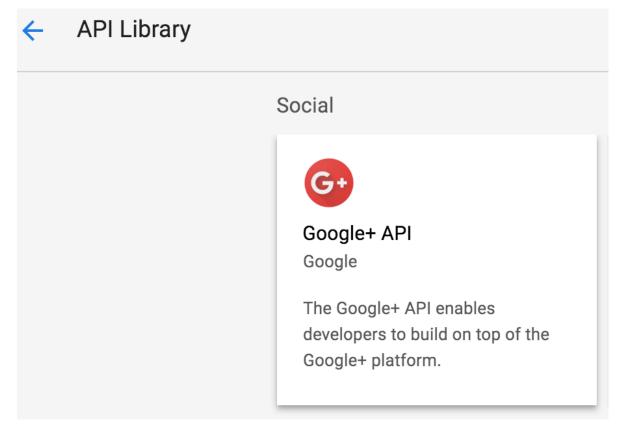
#### Before you begin:

- You'll need to configure a Google Cloud project on GCP.
- You'll need to enable the Google+ API for the project.
- You'll need to create the credentials to use to authorize the platform to connect to Google IAM.

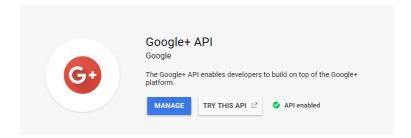
#### **Enabling the Google+ API**

With your project selected in Google Cloud Platform:

- 1. Select **APIs & Services** from the menu on the left.
- 2. Select **ENABLE APIs AND SERVICES**, then locate and select the Google+ API card in the API library.



3. Click ENABLE.



Now you can create credentials for the platform to access your Google Cloud project.

# **Creating Google+ credentials**

With your project selected in Google Cloud Platform:

- 1. Select **APIs & Services > Credentials** from the menu on the left.
- 2. Click **Create credentials** and select **Help me choose** from the drop-down menu.

Note: If you haven't already, be sure to enable the Google+ API before proceeding.

#### APIs

# Credentials

You need credentials to access APIs. Enable the APIs you plan to use and then create the credentials they require. Depending on the API, you need an API key, a service account, or an OAuth 2.0 client ID. Refer to the API documentation for details.

# Create credentials -

# API key

Identifies your project using a simple API key to check quota and access

#### OAuth client ID

Requests user consent so your app can access the user's data

# Service account key

Enables server-to-server, app-level authentication using robot accounts

# Help me choose

Asks a few questions to help you decide which type of credential to use

3. Select **Google+ API** from the API drop-down list, **Web server** from the next drop-down, and **User data** for the last question.

# Credentials

# Add credentials to your project

# 1 Find out what kind of credentials you need

We'll help you set up the correct credentials

If you wish you can skip this step and create an API key, client ID, or service account

# Which API are you using?

Different APIs use different auth platforms and some credentials can be restricted to only call certain APIs.



# Where will you be calling the API from?

Credentials can be restricted using details of the context from which they're called. Some credentials are unsafe to use in certain contexts.



#### What data will you be accessing?

Different credentials are required to authorize access depending on the type of data that you request.

User data

Access data belonging to a Google user, with their permission

Application data

Access data belonging to your own application

What credentials do I need?

2 Get your credentials

Cancel

4. Click **What credentials do I need?** to create the appropriate credentials for the platform.

# Credentials

# Add credentials to your project

Find out what kind of credentials you need Calling Google+ API from a web server

2 Create an OAuth 2.0 client ID



Web client 1

#### Restrictions

Enter JavaScript origins, redirect URIs, or both Learn More

Origins and redirect domains must be added to the list of Authorized Domains in the OAuth consent settings.

#### Authorized JavaScript origins

For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (https://\*.example.com) or a path

(https://example.com/subdir). If you're using a nonstandard port, you must include it in the origin URI.

https://www.example.com

#### Authorized redirect URIs

For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.

https://www.example.com

**Create OAuth client ID** 

- 5. Enter a meaningful name, such as Data Science & AI Workbench, to identify the platform (and help differentiate it from any other web applications you may have configured to use Google IAM).
- 6. In the **Authorized JavaScript origins** field, provide the FQDN of the Workbench server instance.
- 7. Open the Workbench Auth Center (see instructions below), and copy and paste the value from the **Redirect URI** field into the Authorized redirect URIs field here.

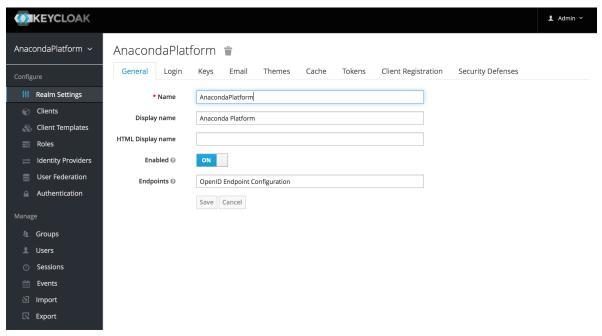
**Note:** If the domain is not an authorized domain, you'll see an **Invalid** Redirect error, and be prompted to add it to the authorized domains list before proceeding.

- 8. Click Create OAuth client ID.
- 9. On the **OAuth consent screen** tab:
  - Set the **Application type** to **Public**.
  - Set the **Application name** to Data Science & AI Workbench (or something else meaningful to platform users).
  - Optionally, upload a logo to help users recognize Workbench.
  - Provide a **Support email** address for users to reach out for help.
  - Provide the full path to the authorized homepage where users will access Anaconda Enteprise.
  - Optionally provide authorized links to a your organization's privacy policy and terms of service.
- 10. Click **Create** to display the OAuth client credentials that you'll need to copy and paste into Workbench, to enable the platform to authenticate with Google. (See Step 5 below.)

#### Configuring Google to be your identity provider

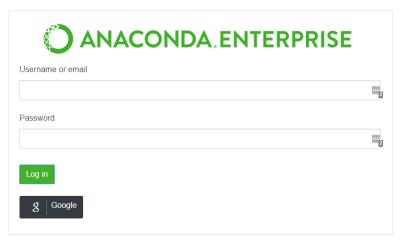
Now that you've configured your GCP project to work with Workbench, you need to use the Workbench Administrative Console's Authentication Center to configure Google as your external identity provider:

- 1. Login to Workbench, click the **Menu** icon in the top right corner, then click the **Administrative Console** link in the bottom of the slideout menu.
- 2. Click **Manage Users** and login to the Authentication Center using the Administrator credentials *configured after installation*.



- 3. In the Configure menu on the left, select **Identity Providers** and select **Google** from the Add provider drop-down list.
- 4. The **Settings** tab displays the **Redirect URI** you need to copy to the Google Cloud project's configuration. The Redirect URI will looking similar to this: https://<full-qualified-domain-name>/auth/realms/AnacondaPlatform/broker/google/endpoint.
- Copy and paste the credentials from GCP (Step 9 above) into the Client ID and Client Secret fields, and click Save.

Now that you've completed the configuration, the Anaconda Enteprise login screen will include a Google login option.



**Note:** When users choose this option and log in to the platform, they'll be automatically added as new Workbench users. As an Administrator, you can then configure their group assignments and role mappings. For more information, see *Roles and groups*.

# 3.3 Configuring channels and packages

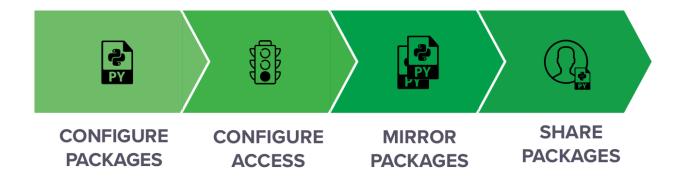
Data Science & AI Workbench enables you to distribute software through the use of *channels* and *packages*. Channels represent locations of repositories where Workbench looks for packages. Packages are used to bundle software files and information about the software—such as its name, specific version and description—into a single file that can be easily installed and managed.

**NOTE:** Workbench supports the use of both conda and pip packages in its repository.

The process for distributing packages within an organization resembles the following:

- 1. Configure access to a cloud-based repository or a private location on a remote or local repository that you or your organization created. See *Accessing remote package repositories* for more information.
- 2. Mirror the entire Anaconda repository or specific packages. You can also mirror packages in a repository in an airgapped environment without internet access.
- 3. Share channels with specific users or groups to give them access to the packages within the channel. You can copy packages from one channel into another, customize each channel by including different versions of packages, and delete channels when they are no longer needed. See *Managing channels and packages* for more information.

Your organization can also optionally configure Workbench to *point conda to an on-premises repository*, or *use a proxy for conda packages*.



# 3.3.1 Accessing remote package repositories

As an Administrator, you can configure Data Science & AI Workbench to use packages from an online package repository such as anaconda and r.

You can then mirror channels & packages into your organization's internal Workbench repository

so users can access the packages from a centralized, on-premises location.

If users are permitted to install packages from off-site package repositories, you can make it easier for users to access them from within their editing sessions by configuring them as default channels.

To do so, *edit your Workbench configuration*—anaconda-enterprise-anaconda-platform.yml—to include the appropriate channels, as follows:

```
conda:
   channels:
   - defaults
   default_channels:
   - https://repo.anaconda.com/pkgs/main
   - https://repo.anaconda.com/pkgs/free
   - https://repo.anaconda.com/pkgs/r
   channel_alias: https://<ANACONDA_ENTERPRISE_FODN>/repository/conda
```

To update Workbench with your changes to the configuration, restart its services:

```
sudo gravity enter
kubectl get pods | grep 'ap-' | cut -d' ' -f1 | xargs kubectl delete pods
```

# 3.3.2 Mirroring channels and packages

Workbench enables you to create a local copy of a repository so users can access the packages from a centralized, on-premises location.

The mirror can be complete, partial, or include specific packages or types of packages. You can also create a mirror in an air gapped environment to help improve performance and security.

**Note:** It can take hours to mirror the full repository.

Before you can use Workbench's convenient syncing tools to configure local mirrors for channels and packages, you'll need to *configure access to the source of the packages* to be mirrored, whether an online repository or a tarball (if an airgapped installation).

#### **Prerequisites:**

- Install and configure Workbench
- · Install conda
- Install the Workbench CLI

#### Types of mirroring:

- To create a complete mirror, see Mirroring the Anaconda repository or Mirroring a PYPI repository.
- To create partial mirror, see Mirroring specific packages.
- To mirror a repository in a system without internet access, see Mirroring in an air-gapped environment.
- To share mirrors, see Configuring Workbench and Sharing channels.

#### **Configuration options:**

- Configuring SSL verification
- Mirroring in a proxied environment
- Mirroring specific platforms and versions
- Using blocklisting to exclude packages
- Using allowlisting to include packages
- Combining multiple configuration arguments

Log in to Workbench as an existing user using the following command:

```
$ anaconda-enterprise-cli login
Username: anaconda-enterprise
Password:
Logged anaconda-enterprise in!
```

**Note:** If Workbench is installed in a proxied environment, see *Mirroring in a proxied environment* for information on setting the NO\_PROXY variable.

# Mirroring the Anaconda repository

Anaconda recommends the following process as a best practice for mirroring the Workbench Repository.

1. Instead of using the default anaconda.yaml file included in the mirror tool installation, create two yaml files, one for mirroring the main channel, and another for mirroring the free channel.

Example main.yaml file:

```
dest_channel: main
channels:
    - https://repo.anaconda.com/pkgs/main
platforms:
```

(continues on next page)

(continued from previous page)

```
- linux-64
- noarch
```

Example free.yaml file:

```
dest_channel: free
channels:
    - https://repo.anaconda.com/pkgs/free
platforms:
    - linux-64
    - noarch
```

2. If you saved both of these files to the home directory, you can use the following commands to mirror these channels. Otherwise, amend the path so that it corresponds to where you saved the files:

```
cas-sync-api-v5 --file ~/main.yaml
cas-sync-api-v5 --file ~/free.yaml
```

This mirrors all of the packages from these channels in the Anaconda repository. If the channel doesn't already exist, it will be automatically created and shared with all authenticated users. You can customize the permissions on the mirrored packages by *sharing the channel*.

**Tip:** If you plan to mirror these channels on a regular basis, consider adding the -c flag to get a clean mirror each time. This will automatically remove any packages that have been removed from the Anaconda repository between mirrors from your internal repository—excluding any packages your organization has blocklisted.

3. Verify that the mirror was successful by logging into your account and navigating to the **Packages** tab. You should see a list of the mirrored packages.

# Mirroring a PyPI repository

The full PyPI mirror size is currently close to 4TB, so ensure that your file storage location has sufficient disk space before proceeding. Rather than mirror the entire PyPI repository, you can use a configuration file such as \$PREFIX/etc/anaconda-platform/mirrors/pypi.yaml to customize the mirror behavior and specify the subset of packages you want to mirror.

#### To create a PyPI mirror:

```
anaconda-enterprise-cli mirror pypi --config pypi.yaml
```

This command loads the packages on https://pypi.org into the pypi user account. Mirrored packages can be viewed at <https://anaconda.example.com>/repository/pypi/pypi/simple/, replacing <https://anaconda.example.com> with the actual URL to your installation of Workbench. (The second pypi in the url should match the user configuration value described below.)

The following configuration options are available for you to customize your configuration file:

Name	Description
user	The local user under which the PyPI packages are imported. Default: pypi.
pkg_list A list of packages to mirror. Only packages listed are mirrored. If this is set, blocklist and allowlist	
	settings are ignored. <b>Default:</b> [].
allowli\$tA list of packages to mirror. Only packages listed are mirrored. If the list is empty, all packages are	
	checked. Default: [].
blocklis	tA list of packages to skip. The packages listed are ignored. Default: [].
latest_or Oyly download the latest versions of the packages. Default: false.	
remote_u	arThe URL of the PyPI mirror. /pypi is appended to build the XML RPC API URL, /simple for the
	simple index and /pypi/{package}/{version}/json for the JSON API. Default: https://pypi.
	python.org/.
xml_rpc_	apicustom value for XML RPC URL. If this value is present, it takes precedence over the URL built using
	remote_url. Default: null.
simple_indexsturn value for the simple index URL. If this value is present, it takes precedence over the URL built	
	using remote_url. Default: null.
use_xml_	righther to use the XML RPC API as specified by PEP381. If this is set to true, the XML RPC API is
	used to determine which packages to check. Otherwise the scripts falls back to the simple index. If the
	XML RPC fails, the simple index is used. Default: true.
use_seriaWhether to use the serial number provided by the XML RPC API. Only packages updated since the last	
	serial saved are checked. If this is set to false, all PyPI packages are checked for updates. Default: true.
create_or@reate the mirror user as an organization instead of a regular user account. All superusers are added to	
	the "Owners" group of the organization. Default: false.

Note that all mirrored PyPI-like channels are publicly available to pull packages from both inside and outside the cluster (i.e. no auth token required).

#### **EXAMPLE**:

# allowlist:

- requests

- six

- numpy

- simplejson
latest\_only: true

remote\_url: https://pypi.org/

use\_xml\_rpc: true

### **Configuring pip**

To configure pip to use this new mirror, create pip.conf as follows:

```
[global] index-url=<https://anaconda.example.com>/repository/pypi/pypi/simple/
```

replacing <a href="https://anaconda.example.com">https://anaconda.example.com</a> with the actual URL to your Workbench.

To configure Workbench sessions and deployments to automatically use the pip.conf, run the following command.

```
anaconda-enterprise-cli spark-config --config /etc/pip.conf pip.conf
```

Alternately, if you can use the --index-url flag directly when invoking pip. For example,

replacing <https://anaconda.example.com> with the actual URL to your Workbench installation, and <package\_name> with the name of a package that is in your local mirror. In the example URL, the second pypi should match the user configuration value described above.

For more specific information on configuring pip, refer to the official documentation at https://pip.pypa.io/en/stable/user\_guide/#config-file.

# Mirroring specific packages

Alternately, you may not wish to mirror all packages. In this case, you can specify which platforms or specific packages you want to mirror — or — use the allowlist, blocklist or license\_blocklist functionality to control which packages are mirrored, by editing the provided mirror files. *You cannot combine these methods*. For more information, see *Mirror configuration options*.

```
cas-sync-api-v5 --file ~/my-custom-anaconda.yaml
```

#### Mirroring R packages

An example configuration file for mirroring R packages is also provided:

```
# This is destination channel of mirrored packages on your local repository.
dest_channel: r

# conda packages from these channels are mirrored to dest_channel on your local_
_____repository.
channels:
    - https://repo.anaconda.com/pkgs/r/

# if doing a mirror from an airgap tarball, the channels should point to the tarball:
# channels:
# - file:///path-to-expanded-tarball/repo-mirrors-<date>/r/pkgs/

# Only conda packages of these platforms are mirrored.
# Omitting this will mirror packages for all platforms available on specified channels.
# If the repository will only be used to install packages on the v5 system, it only_
_____needs linux-64 packages.
platforms:
    - linux-64
```

```
cas-sync-api-v5 --file ~/cas-mirror/etc/anaconda-platform/mirrors/r.yaml
```

#### Mirroring in an air-gapped environment

To mirror the repository in a system with no internet access, create a local copy of the repository by extracting the airgapped tarball and point cas-sync-api-v5 to the extracted tarball.

In this example we will extract to /tmp:

```
# Replace ``<path to>`` with the actual path to the mirror file.
cd /tmp
tar xvf <path to>/mirror.tar
```

Now you have a local file-system repository located at /tmp/mirror/pkgs. You can mirror this repository by editing cpath to cas-mirror>/etc/anaconda-platform/mirrors/anaconda.yaml to contain:

#### channels:

- /tmp/mirror/pkgs

And then run the command:

```
cas-sync-api-v5 --file etc/anaconda-platform/mirrors/conda.yaml
```

This mirrors the contents of the local file-system repository to your Workbench installation under the username anaconda.

#### **Configuring Workbench**

After creating the mirror, *edit your Workbench configuration* to add this new mirrored channel to the default Workbench channels and make the packages available to users.

```
conda:
   channels:
   - defaults
   default_channels:
   - main
   - free
   - r
   channel_alias: https://<anaconda.example.com>/repository/conda
```

Replacing <anaconda.example.com> with the actual URL to your installation of Workbench.

**Note:** The ap-workspace pod must be restarted for the configuration change to take effect on new project editor sessions.

To update the Workbench server with your changes, you'll need to do the following:

1. Run the following command in an interactive shell to identify the pod associated with the workspace services:

```
kubectl get pods
```

2. Restart the workspace services by running the following command:

```
kubectl delete pod anaconda-enterprise-ap-workspace-<pod ID>
```

#### **Sharing channels**

To make your new channels visible to your users in their **Channels** list, you need to share the channels with them.

EXAMPLE: To share new channels main, free, and r with group everyone for read access:

```
anaconda-enterprise-cli channels share --group everyone --level r main anaconda-enterprise-cli channels share --group everyone --level r free anaconda-enterprise-cli channels share --group everyone --level r r
```

After running the share command, verify by logging onto the user interface and viewing the Channels list.

For more information, see Sharing channels and packages

#### Mirror configuration options

You can use the following options to configure your mirror:

- Configuring SSL verification.
- *Mirroring in a proxied environment*.
- Mirroring specific platforms and versions.
- Using blocklisting to exclude packages.
- Using allowlisting to include package.
- Combining multiple configuration arguments.

#### remote url

Specifies the remote URL from which the conda packages and the Anaconda and Miniconda installers are downloaded. The default value is: https://repo.continuum.io/.

#### channels

Specifies the remote channels from which conda packages are downloaded. The default is a list of the channels <remote\_url>/pkgs/free/ and <remote\_url>/pkgs/pro/

All specification information should be included in the same file, and can be passed to the cas-sync-api-v5 command via the --file argument:

```
cas-sync-api-v5 --file ~/cas-mirror/etc/anaconda-platform/mirrors/anaconda.yaml
```

#### destination channel

The configuration option dest\_channel specifies where files will be uploaded. The default value is: anaconda.

#### SSL verification

The mirroring tool uses two different settings for configuring SSL verification. When the mirroring tool connects to its destination, it uses the ssl\_verify setting from anaconda-enterprise-cli to determine how to validate certificates. For example, to use a custom certificate authority:

```
anaconda-enterprise-cli config set sites.master.ssl_verify /etc/ssl/certs/ca-

→certificates.crt
```

The mirroring tool uses conda's configuration to determine how to validate certificates when connecting to the source that it is pulling packages from. For example, to disable certificate validation when connecting to the source:

conda config --set ssl\_verify false

# Mirroring in a proxied environment

If Workbench is installed in a proxied environment, set the NO\_PROXY variable. This ensures the mirroring tool does not use the proxy when communicating with the repository service, and prevents errors such as Max retries exceeded, Cannot connect to proxy, and Tunnel connection failed: 503 Service Unavailable.

export NO\_PROXY=<master-node-domain-name>

#### Platform-specific mirroring

By default, the cas-sync-api-v5 tool mirrors all platforms. If you do not need all platforms, edit the YAML file to specify the platform(s) you want mirrored:

#### platforms:

- linux-64
- osx-64
- win-64

**Note:** The platform argument is evaluated before any other argument.

#### Package-specific mirroring

In some cases you may want to mirror only a small subset of the repository. Rather than blocklisting a long list of packages you do not want mirrored, you can instead simply enumerate the list of packages you DO want mirrored.

**Note:** This argument cannot be used with the blocklist, allowlist or license\_blocklist arguments—it can only be combined with platform-specific and version-specific mirroring.

EXAMPLE:

#### pkg\_list:

- accelerate
- pyqt
- zope

This example mirrors only the three packages: Accelerate, PyQt & Zope. All other packages will be completely ignored.

# Python version-specific mirroring

Mirror the repository with a Python version or versions specified.

**EXAMPLE:** 

Mirrors only Anaconda packages built for Python 3.3.

#### License blocklist mirroring

The mirroring script supports license blocklisting for the following license families:

AGPL
GPL2
GPL3
LGPL
BSD
MIT
Apache
PSF
Public-Domain
Proprietary
Other

#### EXAMPLE:

```
license_blocklist:
- GPL2
- GPL3
- BSD
```

This example mirrors all the packages in the repository EXCEPT those that are GPL2-, GPL3-, or BSD-licensed, because those three licenses have been blocklisted.

#### **Blocklist mirroring**

The blocklist allows access to all packages EXCEPT those explicitly listed. If the license\_blocklist and blocklist arguments are combined, license\_blocklist is evaluated first, and blocklist is a supplemental modifier.

**EXAMPLE:** 

blocklist:
- bzip2
- tk
- openssl

This example mirrors the entire repository EXCEPT the bzip2, Tk, and OpenSSL packages.

# **Allowlist mirroring**

The allowlist argument adds or includes packages that would be otherwise excluded by the blocklist and/or license\_blocklist functions.

**EXAMPLE:** 

license\_blocklist:
- GPL2
- GPL3
allowlist:
- readline

This example mirrors the entire repository EXCEPT any GPL2- or GPL3-licenses packages, but includes readline, despite the fact that it is GPL3-licensed.

#### Combining multiple mirror configurations

You may find that combining two or more of the arguments above is the easiest way to get the exact combination of packages that you want.

**Note:** The platform argument is evaluated before any other argument.

EXAMPLE: This example mirrors only Linux-64 distributions of the dnspython, Shapely and GDAL packages:

platforms:
 - linux-64
pkg\_list:
 - dnspython
 - shapely
 - gdal

If the license\_blocklist and blocklist arguments are combined, license\_blocklist is evaluated first, and blocklist is a supplemental modifier.

EXAMPLE: In this example, the mirror configuration does not mirror GPL2-licensed packages. It does not mirror the GPL3 licensed package pyqt because it has been blocklisted. It does mirror all other packages in the repository:

license\_blocklist:
 - GPL2
blocklist:
 - pyqt

If the blocklist and allowlist arguments are both employed, the blocklist is evaluated first, with the allowlist functioning as a modifier.

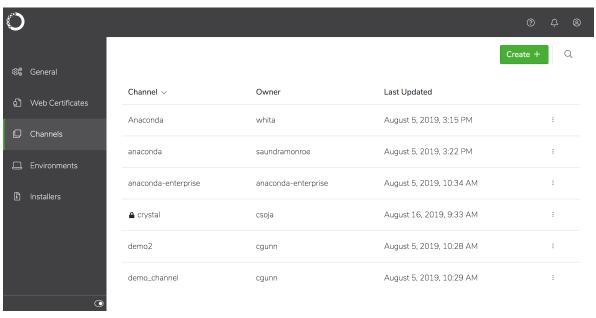
EXAMPLE: This example mirrors all packages in the repository except astropy and pygments. Despite being listed on the blocklist, accelerate is mirrored because it is listed on the allowlist.

blocklist:
- accelerate
- astropy
- pygments
allowlist:
- accelerate

# 3.3.3 Managing channels and packages

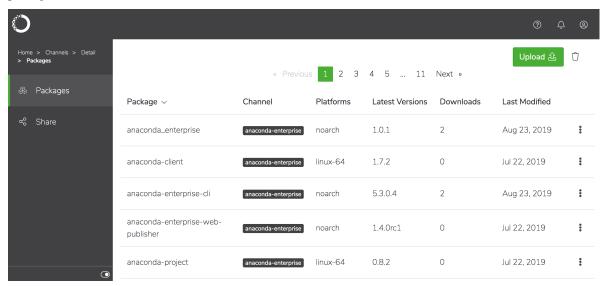
Data Science & AI Workbench makes it easy for you to manage the various channels and packages used by your organization—whether you prefer using the UI or the CLI.

- 1. Log in to the console using the Administrator credentials required to access the Administrative Console.
- 2. Select **Channels** in the left menu to view the list of existing channels, each channel's owner and when the channel was last updated.



**Note:** Private channels are displayed with a lock next to their name in the list, to indicate their secure status.

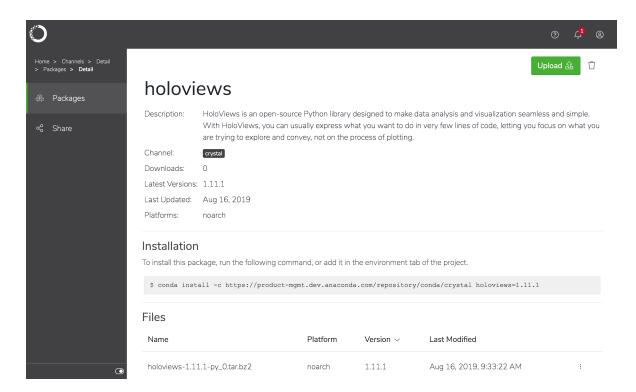
3. Click on a *channel* name to view details about the packages in the channel, including the supported platforms, versions and when each package in the channel was last modified. You can also see the number of times each package has been downloaded.



4. To add a package to an existing channel, click Upload and browse for the package.

**Note:** There is a 1GB file size limit for package files you upload.

5. Click on a *package* name to view the list of files that comprise the package, and the command used to install the package.



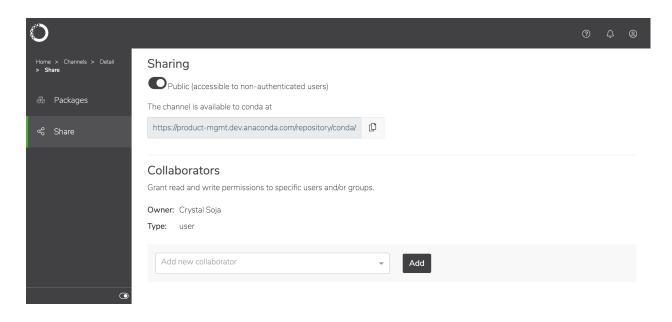
6. To remove a package from a channel, select **Delete** from the command menu for the package in the list.

**Warning:** Do not modify the anaconda-enterprise channel. Modifications to this channel can cause serious problems for the platform.

#### **Sharing channels**

*To share a public channel*, click **Share**, copy the URL location of the channel, and distribute it to the people with whom you want to share the channel.

To give other platform users read-write access to the channel, click **Share** and add them as a collaborator. You can share a channel with individual users or groups of users—the easiest way to control access to a channel. See *Roles and groups* for more information.



**Note:** The default is to grant all collaborators read-write access, so if you want to prevent them from adding and removing packages from the channel, be sure they have read-only access. You'll need to *use the CLI* to grant read-only access to specific users or groups (see below).

#### To create a new channel and add packages to the channel for others to access:

1. Click Create in the top right corner, enter a meaningful name for the channel and click Create.

**Note:** Channels are Public—accessible by non-authenticated users—by default. To make the channel Private, and therefore available to authenticated users only, disable the toggle to switch the channel setting from Public to Private.

2. Click **Upload** to select the packages you want to add to the channel.

#### **Using the CLI:**

Get a list of all the channels on the platform with the channels list command:

```
anaconda-enterprise-cli channels list
```

Share a channel with a specific user using the share command:

```
anaconda-enterprise-cli channels share --user username --level r <channelname>
```

You can also share a channel with an existing group:

```
anaconda-enterprise-cli channels share --group GROUPNAME --level r <channelname>
```

Replacing GROUPNAME with the actual name of the group.

**Note:** Adding --level r grants this group read-only access to the channel.

You can "unshare" a channel using the following command:

```
anaconda-enterprise-cli channels share --user <username> --remove <channelname>
```

Run anaconda-enterprise-cli channels --help to see more information about what you can do with channels.

For help with a specific command, enter that command followed by --help:

```
anaconda-enterprise-cli channels share --help
```

# 3.3.4 Pointing conda to an on-premises repository

Data Science & AI Workbench users who are familiar with conda may use it to install the packages they need, rather than rely on you to make them available for download via shared channels.

If your organization wants to limit platform users to only access packages in your on-premises repository, you can configure conda accordingly. When you do this at the system level, it overrides any user-level configuration files installed by the user, or on individual machines.

Listing channel locations in the .condarc file overrides conda defaults, causing conda to search only the channels listed, in the order specified.

To configure conda, create or update the ~.condarc system configuration file in the root directory of the environment to add the repository channel:

```
channel_alias: https://<your-server.domain.com>/repository/conda/
```

Replacing <your-server.domain.com> with the fully-qualified domain name (FQDN) of your installation of Workbench.

See this section of the conda docs for more information.

# 3.3.5 Using a proxy for conda packages

You can configure Data Science & AI Workbench to use a proxy for conda packages, if your organization's network security policy requires it.

#### Obtain your proxy variables

In order to complete the following sections, you need an appropriate configuration for your proxy. This usually takes the form of environment variables:

- http\_proxy/HTTP\_PROXY
- https://proxy/HTTPS PROXY
- · no proxy/NO PROXY

Ask your system administrator to confirm which variables to use.

#### Verify the proxy works

After you've configured the platform, you can test your changes to verify that it's using the proxy.

- 1. Log in to Workbench.
- 2. Click **Projects**, and open the project you want to use to test the proxy.

Note: If the project already has an open session, you'll need to stop the current session and start a new session.

- 3. Open a terminal window within JupyterLab.
- 4. Set and export your proxy variables.
- 5. Verify the proxy works by running the following command:

```
conda create -n testenv python
```

#### **Configure Workbench**

Once you've confirmed your proxy works, go to Setting global config variables and follow the steps there to apply those variables to all future sessions, deployments, and jobs.

The lines you add to the global config should look something like the following, with your specific proxy URLs and domains substituted in:

```
https_proxy: http://proxy.example.com:1245/
https_proxy: https://proxy.example.com:1245/
no_proxy: *.example.com
HTTP_PROXY: http://proxy.example.com:1245/
HTTPS_PROXY: https://proxy.example.com:1245/
NO_PROXY: *.example.com
```

# 3.4 Configuring environments and sample projects

When you create a new project in Data Science & AI Workbench, you must select a base environment for the project. Anaconda provides several base Anaconda environments, along with Python, R, or Hadoop-Spark environments for you to choose from.

If these environments do not suit your needs, you can create a custom environment, then include it in the list of environments that are available upon project creation.

# 3.4.1 Creating your environment

1. Log in to Workbench as a user with administrator permissions.

```
Tip: The anaconda-enterprise user has the proper permissions.
```

- 2. Click **Create** and select *New Project*.
- 3. Select any **Environment** and **Resource Profile**, then click **Create**.

- 4. Open a session for the project.
- 5. Open a terminal in the project session.
- 6. Configure conda to create new environments in the environments directory by running the following command:

```
export CONDA_ENVS_DIRS=/opt/continuum/envs
export CONDA_PKGS_DIRS=/opt/continuum/envs/.pkgs
```

7. Use conda to create or clone your custom environment.

#### Create an environment

Here is an example command for creating a conda environment:

```
# Replace <ENV> with a name for your new environment
conda create -y -n <ENV> python=3.8 ipykernel
```

#### Clone an environment

Here is an example command for cloning a conda environment:

```
# Replace <CLONE_ENV> with the name of the environment you want to clone
# Replace <ENV> with a name for your new environment
conda create -y --clone <CLONE_ENV> --name <ENV>
```

**Caution:** python and ipykernel *must* be included in every environment you create for Workbench.

# 3.4.2 Providing your environment to users

There are two methods for providing your environment to your users:

- Place the environment in the sample projects gallery and allow other users to clone the environment as a project.
- Include the environment as an available option in the Environment dropdown when creating a new project.

Both of these methods are accomplished by creating a project template archive (.tar.bz2) file that uses the environment you just created, and then placing the file in the correct directory.

The project template archive file must contain, at a minimum, the anaconda-project.yml file. For more information and help building projects, see the official Anaconda Project documentation.

1. In your terminal, navigate to the project directory by running the following command:

```
cd /opt/continuum/project
```

2. Create a directory to store an anaconda-project.yml file for your custom environment by running the following command:

```
# Replace <PROJECT> with the name of your project
mkdir <PROJECT>
```

3. Create a copy of your current project's anaconda-project.yml file in the directory you just created by running the following command:

```
# Replace <PROJECT> with the name of your project
cp anaconda-project.yml <PROJECT>
```

4. Enter the project directory you just created by running the following command:

```
# Replace <PROJECT> with the name of your project
cd /opt/continuum/project/<PROJECT>
```

5. Using your preferred file editor, edit the anaconda-project.yml file in your project directory to represent your custom environment. This involves updating the name:, description:, packages:, and env\_specs: sections of the file.

**Tip:** Delete the commented section below env\_specs: showing available packages. It is likely that these do not align with your custom environment.

**Note:** Your project's name: will display on the **Sample Projects** grid *or* the **Environments** dropdown list after the project is created.

### Example anaconda-project.yml

6. Create an archive file for the project, then set permissions for it by running the following commands:

```
# Replace <PROJECT> with your project name
cd /opt/continuum/project
tar cfj <PROJECT>.tar.bz2 <PROJECT>
chmod 644 <PROJECT>.tar.bz2
```

**Note:** This archive file is the template that other projects will use as a starting point for their own

projects.

7. Move your project archive file to the sample project gallery by running the following command:

```
# Replace <PROJECT> with your project name
mv /opt/continuum/project/<PROJECT>.tar.bz2 /gallery
```

8. Enter the sample gallery directory by running the following command:

```
cd /gallery
```

- 9. (Optional) If you want to include the environment as an available option in the **Environment** dropdown when creating a new project, add the project archive file name (<PROJECT>.tar.bz2) to the TEMPLATES file. Save and close the file when complete. Otherwise, skip this step.
- 10. Update the sample projects gallery to include your project by running the following command:

```
python reindex.py
```

11. Verify that you can either find and select your sample project on the **Sample Projects** page, or select your environment from the **Environment** dropdown when creating a new project.

# 3.5 Generating custom Anaconda installers

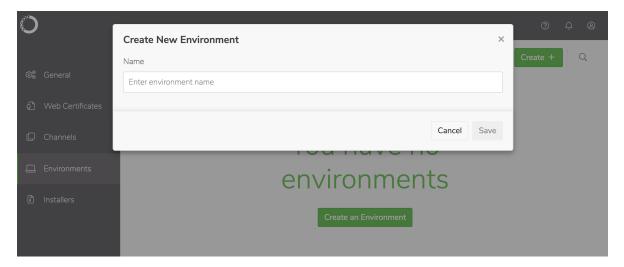
As an Data Science & AI Workbench Administrator, you can *create custom environments*. These environments include specific packages and their dependencies. You can then *create a custom installer* for the environment, that can be shipped to HDFS and used in Spark jobs.

Custom installers enable IT and Hadoop administrators to maintain close control of a Hadoop cluster while also making these tools available to data scientists who need Python and R libraries. They provide an easy way to ship multiple custom Anaconda distributions to multiple Hadoop clusters.

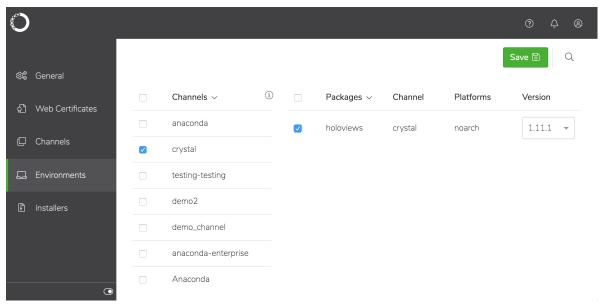
# 3.5.1 Creating an environment

- 1. Log in to the console using the Administrator credentials *configured after installation*.
- 2. Select **Environments** in the left menu.
- 3. Click **Create** in the upper right corner, give the environment a unique name and click **Save**.

**Note:** Environment names can contain alphanumeric characters and underscores only.

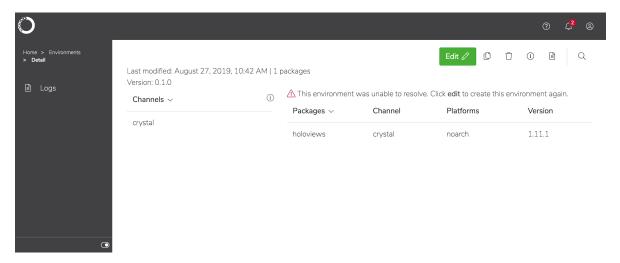


4. Check the channel you want to choose packages from, then select the specific packages—and version of each—you want to include in the installer.



5. Click **Save** in the window banner to create the environment.

Workbench resolves all the package dependencies and displays the environment in the list. If there is an issue resolving the dependencies, you'll be notified and prompted to edit the environment.



You can now use the environment as a basis for creating additional versions of the environment or other environments.

#### To edit an existing environment

- 1. Click on an environment name to view details about the packages included in the environment, then click Edit.
- 2. Change the channels and/or packages included in the environment, and enter a version number for the updated package before clicking Save. The new version is displayed in the list of environments.

#### To copy an environment

1. Select the environment in the list and click the **Duplicate Environment** icon

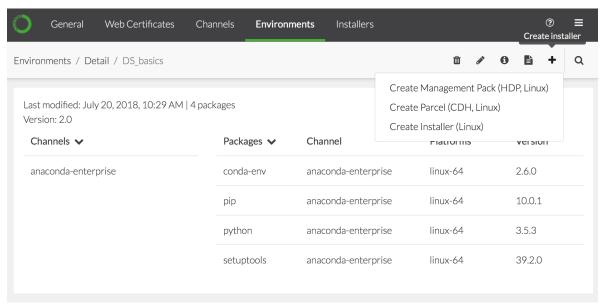


2. Enter a unique name for the environment and click Save. The new environment is displayed in the list of environments.

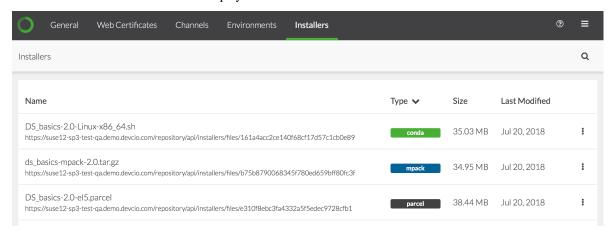
Now that you've created an environment, you can create an installer for it.

# 3.5.2 Creating a custom installer for an environment

1. Select the environment in the list, click the **Create installer** icon , and select the type of installer you want to create:



Workbench creates the installer and displays it in the **Installers** list:



2. To view the relevant logs, download or delete the installer, click the icon and choose the appropriate command

**If you created a management pack**, you'll need to install it on your Hortonworks HDP cluster and add it to your local Ambari server to make it available to users. For more information, see this blog post about generating custom management packs.

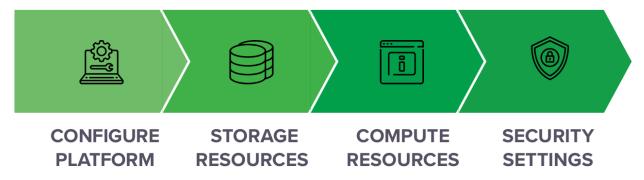
If you created a parcel, you'll need to install it on your CDP cluster to make it available to users.

Alternatively, you can directly download the Parcel onto your CDP cluster as described here.

# 3.6 Advanced platform settings

After installing Data Science & AI Workbench, there are default settings that you may want to update with information specific to your installation, including the *password for the database* and the *redirect URLs* for the Workbench platform.

- If you've *installed Livy server*, you'll need to *configure it to work with the platform* so users can access your Hadoop Spark cluster.
- If your organization already uses a repository such as GitHub, Bitbucket, or GitLab for version control, you can *configure Workbench to use that repository* instead of the internal Git server.
- You can also *add one or more NFS shares* to your organization's configuration, for platform users to store data and source code that they can access within their sessions and deployments.
- You may want to *replace the self-signed certificates generated during installation* with your organization's own certificates—or change other default security settings—after initial installation.



# 3.6.1 Editing platform settings

You configure the Data Science & AI Workbench platform settings using a configuration file or *Config map*. The configuration file, anaconda-enterprise-anaconda-platform.yml, contains both global and per-service configuration settings for your Workbench installation.

You can modify the default configuration using the Operations Center UI, or a command shell. Any changes you make will impact how Workbench functions, so **we strongly recommend that you save a copy of the original file** and familiarize yourself with the configuration options before making any changes.

#### To modify the platform configuration using the UI:

**Note:** Bring Your Own Kubernetes (BYOK8s) installations will need to make changes via a control panel for managing your cluster, or through editing the anaconda-enterprise-anaconda-platform.yml file directly.

- 1. Log in to Workbench, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Log in to the Operations Center using the Administrator credentials *configured after installation*.

- 4. Select **Configuration** from the menu on the left.
- 5. Use the **Config map** drop-down menu to select the anaconda-enterprise-anaconda-platform.yml configuration file.

Warning: Unless you are *configuring global environment variables*, do not change the other entries in the Config maps and Namespace drop-downs. They impact the underlying Kubernetes system, so making changes to them can have unintended consequences or cause the platform to behave unpredictably.

You'll notice that it contains GLOBAL CONFIGURATION specifications related to the following:

- The Authentication Center client URL
- The internal database
- Optional NFS server volume mounts
- · HTTPS certificate settings
- Resource profiles
- The Kubernetes cluster
- Any users, groups or roles with Admin authorization
- The git commit file size limit (The default limit is 50MB, though this limit is configurable. Anaconda recommends keeping files under 100MB.)

It also contains PER-SERVICE CONFIGURATION settings, related to these services:

- The authentication server used to secure access
- The deployment server used to deploy apps
- The workspace server used to run sessions
- The storage server used to store and version projects
- · The local repository server used for channels and packages
- The S3 endpoint and Git server used to store object and data
- The local documentation server URL and platform UI configuration
- 6. Edit the specification in the section that corresponds to the setting you want to update, and click **Apply** to save your changes.

**Note:** If you navigate away from the Config map without saving your edits, you will be warned that you have unsaved changes. You can abandon your edits by clicking **Disregard and continue**, or return to editing by clicking **Close**.

#### To edit the platform configuration using a command line:

1. Enter the following commands in an interactive shell *on the master node*:

sudo gravity enter
kubectl edit cm anaconda-enterprise-anaconda-platform.yml

- 2. Make your changes to the file, and save it.
- 3. Restart all pods using the following command:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

# 3.6.2 Changing the database password

You can change the password for the Data Science & AI Workbench database as needed, to adhere to your organization's policies. To do so, you'll need to connect to the associated pod, make the change, and update the platform with the new password.

1. Run the following command to determine the id of the postgres pod:

```
kubectl get pod | grep postgres
```

2. Run the following command to connect to the postgres pod, where <id> represents the id of the pod:

```
kubectl exec -it anaconda-enterprise-postgres-<id> /bin/sh
```

3. Run this psql command to connect to the database:

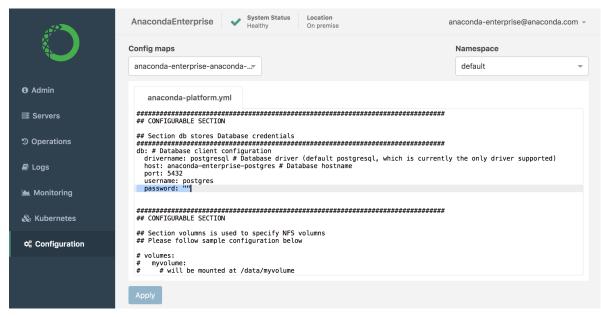
```
psql -h localhost -U postgres
```

4. Set the password by running the following command:

```
ALTER USER postgres WITH PASSWORD 'new_password';
```

#### To update the platform settings with the database password of the host server:

- 1. Access the Workbench Operations Center by entering this URL in your browser: https://anaconda.example.com:32009, replacing anaconda.example.com with the FQDN of the host server.
- 2. Login with the default username and password: aeplatform@yourcompany.com / aeplatform. You'll be asked to change the default password when you log in.
- 3. Click **Configuration** in the left menu to display the Workbench Config map.
- 4. In the GLOBAL CONFIGURATION section of the configuration file, locate the db section and enter the password you just set:



- 5. Click **Apply** to update the platform with your changes.
- 6. Restart all the service pods using the following command:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

# 3.6.3 Installing conda for packaging mirroring

To help improve performance and security, Data Science & AI Workbench enables you to create a local copy of an online package repository so users can access the packages from a centralized, on-premises location. This copy is called a *mirror*. A mirror can be complete, partial, or include specific packages or types of packages.

The Workbench installer contains a bootstrap executable that you can run to install conda.

#### **Prerequisites:**

- Install Workbench
- Complete the post-install configuration
- Update TLS/SSL certs

#### Installing conda

1. Use one of the following methods to obtain conda for your setup:

### Gravity

Open a terminal window and navigate to the directory where you downloaded and extracted the Workbench installer:

```
# Replace <VERSION> with your specific version number
cd anaconda-enterprise-<VERSION>
```

#### BYOK8s

Open a terminal window and download the conda environment bundle by running the following command:

```
curl -0 https://airgap.svc.anaconda.com/misc/ae5-conda-latest-Linux-x86_64. _{\rightarrow} sh
```

2. Run the following command to verify the bzip2 package is installed:

```
which bunzip2
```

If the command returns a valid package, you can run the bootstrap executable. If not, install the binary by running either yum install bzip2 or apt-get install bzip2.

3. Make the bootstrap executable by running the following command:

#### Gravity

./conda-bootstrap.sh

#### **BYOK8s**

./ae5-conda-latest-Linux-x86\_64.sh

- 4. Type yes when prompted to accept the end user license agreement (EULA).
- 5. Accept the default path, or enter an alternate path when prompted.
- 6. When prompted, type yes to activate the conda command at shell initialization.
- 7. Re-initialize your terminal for the previous steps to take effect:

source ~/.bashrc

Now that you've installed conda, you can *configure access* to the source of the packages to be mirrored, whether an online repository or a tarball (if an air-gapped installation). Then you'll be ready to begin *mirroring channels and packages*.

#### 3.6.4 Workbench CLI

The Data Science & AI Workbench CLI is a collection of tools that you can use to interact with your instance of Workbench at the operating system (OS) level. Everything you need to use the CLI is contained in the ae5-conda-latest-Linux-x86\_64.sh script, which is included with the Gravity installation tarball, but is also made available for air-gapped and BYOK8s implementations. If necessary, follow the steps to *install the CLI tools*. Otherwise, start at *Configuring the Workbench CLI*.

#### Installing the Workbench CLI tools

1. Open a terminal and download the script by running the following command (skip this step if you've downloaded the Gravity tarball):

```
curl -0 https://airgap.svc.anaconda.com/misc/ae5-conda-latest-Linux-x86_64.sh
```

2. Make the script you just downloaded executable by running the following command:

```
chmod +x ae5-conda-latest-Linux-x86_64.sh
```

3. Execute the script by running the following command:

bash ae5-conda-latest-Linux-x86\_64.sh

#### Configuring the Workbench CLI

Before you can log in to your instance of Workbench, you'll need to tell the CLI which site to reach out to and interact with. To configure your CLI, complete the following steps:

1. Add your Workbench Repository URL to the set of available sites by running the following command:

```
# Replace <FQDN> with the fully qualified domain name of your Workbench instance
anaconda-enterprise-cli config set sites.main.url https://<FQDN>/repository/api
```

2. Run the following command to configure your instance of Workbench as the default site:

```
anaconda-enterprise-cli config set default_site main
```

3. Verify your configuration was successful by running the following command:

```
anaconda-enterprise-cli config view
```

#### Updating SSL certificates using the CLI

If you *updated the platform's TLS/SSL certificate*, you can use the Workbench CLI to configure the certificates for the repository using the following commands:

```
$ anaconda-enterprise-cli config set ssl_verify true

# On Ubuntu
$ anaconda-enterprise-cli config set sites.master.ssl_verify /etc/ssl/certs/ca-
certificates.crt

# On RHEL/CentOS
$ anaconda-enterprise-cli config set sites.master.ssl_verify /etc/pki/tls/certs/ca-
bundle.crt
```

#### Logging in to the Workbench CLI

1. Access the CLI by running the following command:

```
anaconda-enterprise-cli login
```

2. Use your Workbench username and password to log in when prompted:

```
Username: <YOUR_USERNAME>
Password: <YOUR_PASSWORD>
```

#### **Next Steps:**

You can now configure access to your packages' source mirror.

Once access is configured, you can begin *mirroring channels and packages*.

## **Configuration file locations**

The Workbench CLI reads configuration information from the following places:

System-level configuration: /etc/anaconda-platform/cli.yml

User-level configuration: \$INSTALL\_PREFIX/anaconda-platform/cli.yml and \$HOME/.anaconda/
anaconda-platform/cli.yml

To change how the CLI is configured, modify the appropriate cli.yml files based on your needs.

Note: Changing configuration settings at the user level overrides any system-level configuration.

### **Included CLI tools**

As stated before, the Workbench CLI uses multiple tools to interact with your instance and your repositories, and these tools are provided via the ae5-conda-latest-Linux-x86\_64.sh script.

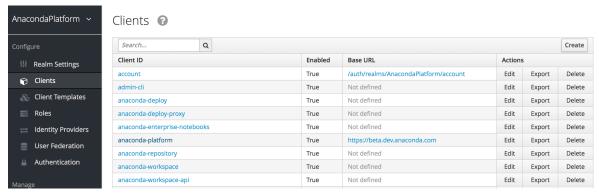
The script contains the following tools:

Package	Version	Build	Channel	Documentation
ae5-tools	0.4.1	pypi_0	ae5-admin	ae5-tools github
ae5_backup_restore	0.3.2	0	ae5-admin	ae5_backup_restore github
ae_preflight	0.1.9	py_0	ae5-admin	ae_preflight github
anaconda-enterprise-cli	5.5.2	pypi_0	ae5-admin	
cas-mirror	5.4.0	pypi_0	pypi	Mirroring channels and packages
python	3.11.2	h7a1cb2a_0		

# 3.6.5 Changing the platform redirect URLs

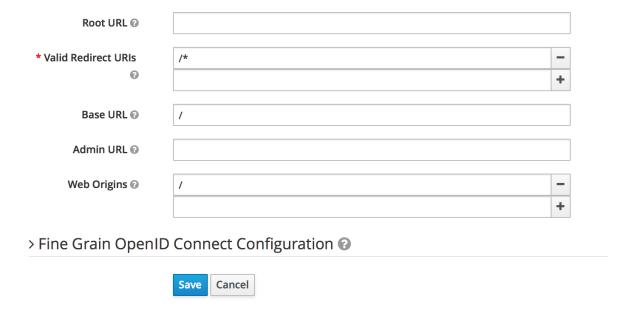
Use the Data Science & AI Workbench Authentication Center to update the redirect URLs for the platform.

- 1. Enter the following URL in your browser, https://<server-name.domain.com>/auth/, replacing server-name.domain.com with the fully-qualified domain name of the host server.
- 2. Login with username and password configured to authorize access to the platform. See *Managing System Administrators* for instructions on setting these credentials, if you haven't already done so.
- 3. Verify that AnacondaPlatform is displayed as the current realm, then select **Clients** from the **Configure** menu on the left.



4. In the **Clients** list, click anaconda-platform to display the platform settings.

5. On the **Settings** tab, update all URLS in the following fields with the FQDN of the Workbench server, or the following symbols:



**Note:** If you choose to provide the FQDN of your Workbench server, be sure each field also ends with the symbols shown. For example, the **Valid Redirect URIs** would look something like this: https://server-name.domain.com/\*.

6. Click **Save** to update the server with your changes.

# 3.6.6 Apache Livy and Workbench

To support your organization's data analysis operations, Data Science & AI Workbench enables platform users to connect to remote Apache Hadoop or Spark clusters. Workbench uses Apache Livy to handle session management and communication to Spark clusters, including different versions of Spark, independent clusters, and even different types of Hadoop distributions, such as those installed by different Cloudera Data Platform (CDP) Parcel versions.

Livy provides all the authentication layers that Hadoop administrators are familiar with, including Kerberos. Workbench can also authenticate to a Hadoop Distributed File System (HDFS) using Kerberos when Kerberos Impersonation is enabled.

Selecting a Spark template when creating a project will connect users to the remote Spark cluster where Livy is installed. They can use the Python libraries available through the platform or package a specific environment for the job. For more information, see *Hadoop / Spark*.

#### **Tested Versions:**

Workbench has been verified against the following versions.

Software	Version
Hadoop (Includes YARN and HDFS)	3.1.1
Spark	2.4.7
Hive	3.1.3000
Impala	3.4.0
Livy	0.7.1-incubating

**Note:** Workbench has also been verified against Cloudera Data Platform 7.1.7.

# 3.6.7 Configuring Livy server for Hadoop Spark access

Review the *Apache Livy requirements* before you begin the configuration process. There are three main configuration settings you must update on your Apache Livy server to allow Data Science & AI Workbench users access to Hadoop/Spark clusters:

- Livy impersonation
- Cluster access
- · Project access

If the Hadoop cluster is configured to use Kerberos authentication, you'll need to *allow Livy to access the services*. Additionally, you can configure Livy as a secure endpoint. For more information, see *Configuring Livy to use HTTPS* below.

## **Configuring Livy impersonation**

To enable users to run Spark sessions within Workbench, they need to be able to log in to each machine in the Spark cluster. The easiest way to accomplish this is to configure Livy impersonation as follows:

- 1. Add Hadoop.proxyuser.livy to your authenticated hosts, users, or groups.
- 2. Check the option to Allow Livy to impersonate users and set the value to all (\*), or a list of specific users or groups.

If impersonation is *not* enabled, the user executing the livy-server (livy) must exist on every machine. You can add this user to each machine by running the following command on each node:

```
sudo useradd -m livy
```

**Note:** If you have any problems configuring Livy, try setting the log level to DEBUG in the conf/log4j.properties file.

### Configuring cluster access

Livy server enables users to submit jobs from any remote machine or analytics cluster—even where a Spark client is not available—without requiring you to install Jupyter and Anaconda directly on an edge node in the Spark cluster.

To configure Livy server, put the following environment variables into a user's .bashrc file, or the conf/livy-env.sh file that's used to configure the Livy server.

These values are accurate for a Cloudera install of Spark with Java version 1.8:

```
SPARK_HOME=/opt/cloudera/parcels/CDH-7.1.7-1.cdh7.1.7.p0.15945976/lib/spark
LIVY_LOG_DIR=/var/log/livy2
LIVY_PID_DIR=/var/run/livy2
JAVA_HOME=/usr/java/jdk1.8.0_232-cloudera/
```

(continues on next page)

(continued from previous page)

```
SPARK_CONF_DIR=/etc/spark/conf
HADOOP_HOME=/opt/cloudera/parcels/CDH-7.1.7-1.cdh7.1.7.p0.15945976/lib/hadoop
HADOOP_CONF_DIR=/etc/hadoop/conf
```

Note that the port parameter that's defined as livy.server.port in conf/livy-env.sh is the same port that will generally appear in the Sparkmagic user configuration.

The minimum required parameter is livy.spark.master. Other possible values include the following:

- local[\*]—for testing purposes
- yarn-cluster—for using with the YARN resource allocation system
- a full spark URI like spark://masterhost:7077—if the spark scheduler is on a different host.

Example with YARN:

```
livy.spark.master = yarn-cluster
```

The YARN deployment mode is set to cluster for Livy. The livy.conf file, typically located in \$LIVY\_HOME/conf/livy.conf, may include settings similar to the following:

```
# What host address to start the server on. By default, Livy will bind to all network_
interfaces.
livy.server.host = 0.0.0.0

# What port to start the server on.
livy.server.port = 8998

# What spark master Livy sessions should use.
livy.spark.master = yarn

# What spark deploy mode Livy sessions should use.
livy.spark.deploy-mode = cluster
```

Restart Livy server once configuration is complete.

Anaconda recommends using a process control mechanism to restart your Livy server to ensure that it's reliably restarted in the event of a failure.

**Note:** The above example is to be used as a template only. Anaconda cannot assist with configuring Apache Livy for your organization.

## Using Livy with Kerberos authentication

If the Hadoop cluster is configured to use Kerberos authentication, you'll need to do the following to allow Livy to access the services:

1. Generate two keytabs for Apache Livy using kadmin.local.

Caution: The keytab principals for Livy must match the hostname that the Livy server is deployed on, or you'll see the following exception: GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos credentials).

These are hostname and domain dependent, so edit the following example according to your Kerberos settings:

```
$ sudo kadmin.local
kadmin.local: addprinc livy/<HOSTNAME>
kadmin.local: xst -k livy-<HOSTNAME>.keytab livy/<HOSTNAME>@<REALM>
...
kadmin.local: addprinc HTTP/<HOSTNAME>
kadmin.local: xst -k HTTP-<HOSTNAME>.keytab HTTP/<HOSTNAME>@<REALM>
...
```

This will generate two files: livy-<HOSTNAME>.keytab and HTTP-<HOSTNAME>.keytab.

- 2. Change the permissions of these two files so they can be read by livy-server.
- 3. Enable Kerberos authentication and reference these two keytab files in the conf/livy.conf configuration file, as shown:

Note: The hostname and domain are not the same—verify that they match your Kerberos configuration.

**Note:** The above example is to be used as a template only. Anaconda cannot assist with configuring Kerberos for your organization.

### **Configuring Livy to use HTTPS**

If you want to use Sparkmagic to communicate with Livy via HTTPS, you need to do the following to configure Livy as a secure endpoint:

- Generate a keystore file, certificate, and truststore file for the Livy server—or use a third-party SSL certificate.
- Update Livy with the keystore details.
- Update your Sparkmagic configuration.
- Restart the Livy server.

## If you're using a self-signed certificate

1. Generate a keystore file for Livy server using the following command:

2. Create a certificate:

```
keytool -export -alias <host> -keystore <keystore_file> -rfc -file <cert_file> -

⇒storepass <StorePassword>
```

3. Create a truststore file:

4. Update livy.conf with the keystore details. For example:

```
livy.keystore = <FILEPATH>/keystore.jks
livy.keystore.password = anaconda
livy.key-password = anaconda
```

5. Update ~/.sparkmagic/config.json. For example:

```
"kernel_python_credentials" : {
    "username": "",
    "password": "",
    "url": "https://<IP>:8998",
    "auth": "None"
    },
    "ignore_ssl_errors": true,
```

**Note:** In this example, ignore\_ssl\_errors is set to true because this configuration uses self-signed certificates. Your production cluster setup may be different.

**Caution:** If you misconfigure a .json file, all Sparkmagic kernels will fail to launch. You can test your Sparkmagic configuration by running the following Python command in an interactive shell: python -m json.tool config.json.

If you have formatted the JSON correctly, this command will run without error. Additional edits may be required, depending on your Livy settings.

6. Restart the Livy server.

The Livy server should now be accessible over https. For example, https://<livy host>:<livy port>.

To test your SSL-enabled Livy server, run the following Python code in an interactive shell to create a session:

```
livy_url = "https://<livy host>:<livy port>/sessions"
data = {'kind': 'spark', 'numExecutors': 1}
```

(continues on next page)

(continued from previous page)

Run the following Python code to verify the status of the session:

```
session_url = "https://<livy host>:<livy port>/sessions/0"
headers = {'Content-Type': 'application/json'}
r = requests.get(session_url, headers=headers, auth=HTTPKerberosAuth(mutual_
authentication=REQUIRED, sanitize_mutual_error_response=False), verify=False)
r.json()
```

Then submit the following statement:

## If you're using a third-party certificate

**Note:** Ensure that Java JDK is installed on the Livy server.

1. Create the keystore.p12 file using the following command:

```
openssl pkcs12 -export -in [path to certificate] -inkey [path to private key] -

→certfile [path to certificate] -out keystore.p12
```

2. Use the following command to create the keystore.jks file:

```
keytool -importkeystore -srckeystore keystore.p12 -srcstoretype pkcs12 -

→destkeystore keystore.jks -deststoretype JKS
```

3. If you don't already have the rootca.crt, you can run the following command to extract it from your Workbench installation:

```
kubectl get secrets anaconda-enterprise-certs -o jsonpath="{.data[`rootca\.crt`]}"

→| base64 -d > /ext/share/rootca.crt
```

4. Add the rootca.crt to the keystore.jks file:

```
keytool -importcert -keystore keystore.jks -storepass <password> -alias rootCA - \rightarrow file rootca.crt
```

5. Add the keystore.jks file to the livy.conf file. For example:

```
livy.keystore = <FILEPATH>/keystore.jks
livy.keystore.password = anaconda
livy.key-password = anaconda
```

- 6. Restart the Livy server.
- 7. Run the following command to verify that you can connect to the Livy server (using your actual host and port):

```
openssl s_client -connect anaconda.example.com:8998 -CAfile rootca.crt
```

If running this command returns **0**, you've successfully configured Livy to use HTTPS.

#### To add the trusted root certificate to the Workbench server

1. Install the ca-certificates package:

```
yum install ca-certificates
```

2. Enable dynamic CA configuration:

```
update-ca-trust force-enable
```

3. Add your rootca.crt as a new file:

```
cp rootca.crt /etc/pki/ca-trust/source/anchors
```

4. Update the certificate authority trust:

```
update-ca-trust extract
```

## To connect to Livy within a session

Open the project and run the following command in an interactive shell:

```
import os
os.environ['REQUESTS_CA_BUNDLE'] = /path/to/root.ca
```

You can also edit the anaconda-project.yml file for the project and set the environment variable there. See *Hadoop / Spark* for more information.

## **Configuring project access**

After you've installed Livy and configured cluster access, some additional configuration is required before you, as a Workbench user, can connect to a remote Hadoop Spark cluster from within your projects. For more information, see *Connecting to the Hadoop Spark ecosystem*.

• If the Hadoop installation used Kerberos authentication, add the krb5.conf to the global configuration using the following command:

```
anaconda-enterprise-cli spark-config --config /etc/krb5.conf krb5.conf
```

• To use Sparkmagic, pass two flags to the previous command to configure a Sparkmagic configuration file:

```
anaconda-enterprise-cli spark-config --config /etc/krb5.conf krb5.conf --config /

→opt/continuum/.sparkmagic/config.json config.json
```

This creates a yaml file—anaconda-config-files-secret.yaml—with the data converted for Workbench.

Use the following command to upload the yaml file to the server:

```
sudo kubectl replace -f anaconda-config-files-secret.yaml
```

To update the Workbench server with your changes, run the following command to identify the pod associated with the workspace services:

```
kubectl get pods
```

Restart the workspace services by running:

```
kubectl delete pod anaconda-enterprise-ap-workspace-<unique ID>
```

Now, whenever a new project is created, /etc/krb5.conf will be populated with the appropriate data.

# 3.6.8 Adding Anaconda Assistant to Workbench

Anaconda Assistant is the AI pair programmer developed by Anaconda to assist you with coding in your Jupyter Notebooks. You can install the Anaconda Assistant as an optional component of the Data Science & AI Workbench.

## **Installing Anaconda Assistant**

- 1. Open a browser and log in to Workbench as the anaconda-enterprise user.
- 2. Open any existing project and view its settings.
- 3. If necessary, change the **Default Editor** to *JupyterLab*.
- 4. Open a project session.
- 5. Open a terminal window in your project session.
- 6. Download the Anaconda Assistant tarball and its checksum by running the following commands:

```
curl -0 https://airgap-svc.s3.amazonaws.com/misc/jupyter-as-tgz-0.3.5_4.tgz curl -0 https://airgap-svc.s3.amazonaws.com/misc/jupyter-as-tgz-0.3.5_4.tgz.

→sha256
```

7. Verify the checksum of the files you downloaded by running the following command:

```
sha256sum -check jupyter-as-tgz-0.3.5_4.tgz.sha256
```

8. Unpack the tarball you just downloaded into the tools directory by running the following command:

```
tar xvzf jupyter-as-tgz-0.3.5_4.tgz -C /tools
```

**Tip:** Keep this session open while you enable the Anaconda Assistant.

## **Enabling the Anaconda Assistant**

## Obtaining an anaconda.cloud token

To enable the Anaconda Assistant, you must have an Anaconda Cloud account. If you do not have an Anaconda Cloud account, create one now.

Note: You must obtain your token from a machine that has connectivity to https://anaconda.cloud/.

Anaconda Assistant utilizes a token that must be generated using the anaconda-cloud-cli package, which is available in Anaconda's default channel. This token provides your Workbench instance with access to the Anaconda Assistant API.

- 1. Open a terminal application *outside* of Workbench on your machine.
- 2. Create a conda environment with the packages you need by running the following command:

```
conda create -n anaconda-cloud -y anaconda-cloud-auth anaconda-cloud-cli
```

3. Activate your new environment by running the following command:

```
conda activate anaconda-cloud
```

4. Log in to anaconda.cloud by running the following command:

```
anaconda login
```

- 5. Select anaconda.cloud and press Enter/return.
- 6. Use the browser window that appears to log in to Anaconda Cloud using your account credentials.

```
Tip: You can close the window once you are logged in.
```

7. Return to your terminal application (outside of Workbench) and enter a python command prompt by running the following command:

```
python
```

8. View your token by running the following commands:

```
from anaconda_cloud_auth.token import TokenInfo
TokenInfo.load('id.anaconda.cloud').api_key
```

Note: Store this token in a secure location for now.

## Setting environment variables

- 1. Connect to your Workbench Kubernetes cluster.
- 2. View a list of your configmaps by running the following command:

```
kubectl get cm
```

3. Edit the anaconda-enterprise-env-var-config.yml file.

```
kubectl edit cm anaconda-enterprise-env-var-config
```

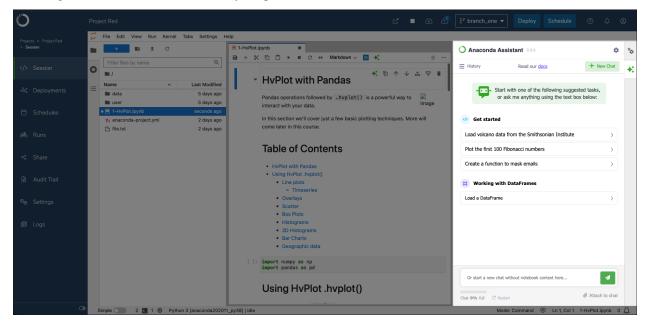
4. Include the following lines:

```
# Replace <CLOUD_API_TOKEN> with the your cloud API token
ANACONDA_AE5_CLOUD_TOKEN: <CLOUD_API_TOKEN>
ANACONDA_ASSISTANT_ENVIRONMENT_TYPE: enterprise-notebooks-prod
```

- 5. Save your work and close the file.
- 6. Update Workbench with your changes and restart services by running the following command:

## Verifying installation

With the tarball extracted into the /tools volume and your anaconda.cloud API key stored as a system variable, return to Workbench and create a new project with JupyterLab as its default editor. You will see the Anaconda Assistant on the right hand side of the screen when you open a notebook (.ipynb) file.



For more information about how to use the Assistant, see the Anaconda Assistant quickstart guide.

# 3.6.9 Adding VSCode to Workbench

You can install VSCode as an optional component of Data Science & AI Workbench. Technically, the stock Microsoft version of VSCode will not run in a browser-based environment, so Anaconda relies on code-server, a patched version of VSCode that allows it to be run in the browser.

**Note:** Anaconda recommends installing VSCode during a scheduled maintenance interval to prevent users from creating new sessions during installation. Existing sessions or deployments do not need to be halted.

## **Installing VSCode**

- 1. Download the VSCode tarball file.
- 2. Open a browser and log in to Workbench as the anaconda-enterprise user.
- 3. Open any existing project and view its settings.
- 4. If necessary, change the **Default Editor** to *JupyterLab*.
- 5. Open a project session.
- 6. Upload the VSCode tarball you just downloaded into the project.
- 7. Open a terminal for your session.
- 8. Unpack the tarball you just downloaded into the tools directory by running the following command:

```
tar xzvf vscode.tar.gz -C /tools
```

### **Enabling VSCode as an editor option**

Once you have unpacked the tarball, you must enable VSCode as an editor option for project sessions.

- 1. Access your Kubernetes cluster resource management console.
- 2. Open the anaconda-enterprise-anaconda-platform.yml file.
- 3. Find the tools: section of the file.
- 4. Add the following line, nested under tools::

```
vscode: VSCode
```

- 5. Save your work and close the file.
- 6. Open a terminal window and restart the workspace and ui pods by running the following command:

```
kubectl delete --wait=false $(kubectl get pods -o name|grep -E 'worksapce|ui')
```

## Removing VSCode as an editor option

- 1. Access your Kubernetes cluster resource management console.
- 2. Open the anaconda-enterprise-anaconda-platform.yml file.
- 3. Search for the tools: section of this file and remove the following line:

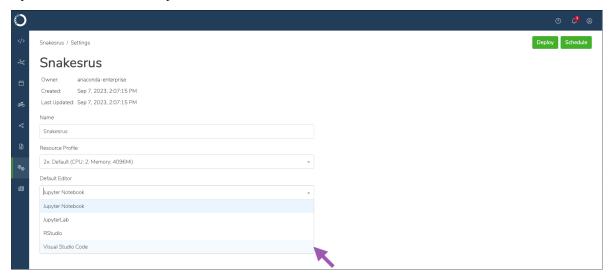
```
vscode: VSCode
```

- 4. Save your work and close the file.
- 5. Open a terminal and restart the workspace and ui pods by running the following command:

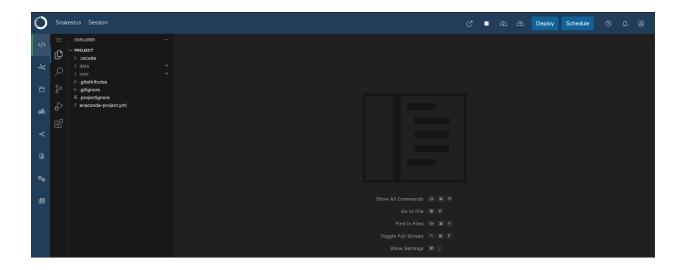
```
kubectl delete --wait=false $(kubectl get pods -o name|grep -E 'worksapce|ui')
```

## Verifying your installation

- 1. Return to Workbench in your browser and navigate to the projects grid.
- 2. Create a new project.
- 3. Open your project and view its settings.
- 4. Open the **Default Editor** dropdown menu and select *Visual Studio Code*.



- 5. Click Save.
- 6. Open a session for your project.



## **Uninstalling VSCode**

To completely remove VSCode from your instance of Workbench:

1. Stop any existing project session that is currently using VSCode. These sessions must also choose a different default editor, such as JupyterLab.

Caution: If you do not choose a different default editor, sessions will fail to start for these projects.

- 2. Access your Kubernetes cluster resource management console.
- 3. Open the anaconda-enterprise-anaconda-platform.yml file.
- 4. Search for the tools: section of this file and remove the following line:

```
vscode: VSCode
```

- 5. Save your work and close the file.
- 6. Open a terminal and restart the workspace and ui pods by running the following command:

```
kubectl delete --wait=false $(kubectl get pods -o name|grep -E 'worksapce|ui')
```

7. Once the cluster is stable, you can remove VSCode from the /tools volume by running the following command:

```
rm -r /tools/vscode
```

# Upgrading to a new version of Workbench

Upgrading Workbench will not affect the VSCode installation. However, the upgrade process will re-hide VSCode as an editor option. For more information, see *Enabling VSCode as an editor option*.

## **Updating the VSCode installation**

Anaconda recommends performing a fresh installation if you need to upgrade VSCode, as this minimizes the risk of unexpected issues.

- 1. Stop any existing project session that is currently using VSCode.
- 2. Download the latest VSCode installation project and save it locally.
- 3. Open a browser and log in to Workbench as an administrator with managed persistence permissions.

**Tip:** The anaconda-enterprise user account has these permissions.

- 4. Open a session for your VSCode installation project.
- 5. Open a terminal in your session.
- 6. Move the existing VSCode installation aside in case you need it by running the following command:

```
mv /tools/vscode /tools/vscode.old
```

**Tip:** If an error occurs, you can restore this file to avoid downtime.

7. Remove the directory by running the following command:

```
rm -r /tools/vscode
```

- 8. Upload your new VSCode tarball file to the project.
- 9. Install VSCode by running the following command:

```
bash install_vscode.sh
```

10. Verify your installation by running the following command:

```
bash /tools/vscode/start_vscode.sh
```

If your command returns an "address already in use" error, your installation was successful.

11. Remove the /tools/vscode.old directory once the update is complete.

```
rm -rf /tools/vscode.old
```

### Installing VSCode extensions

Anaconda relies on the Open VSX Registry as a source for VSCode extensions.

## Installing an extension into a project

- 1. Open a session in your project using the VSCode editor.
- 2. Click the extensions icon in the left-hand navigation.
- 3. Enter an extension's name in the search field.
- 4. Click **Install** next to the extension you want.



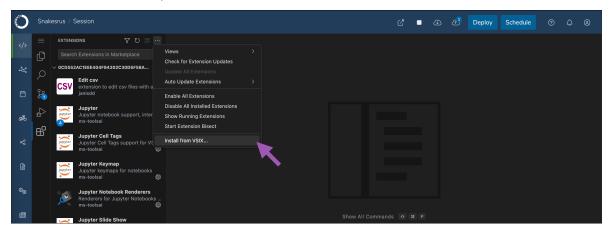
If you do not have a connection to the registry, you will need to obtain the extension's .vsix file manually, then upload it to the project you want to use it in.

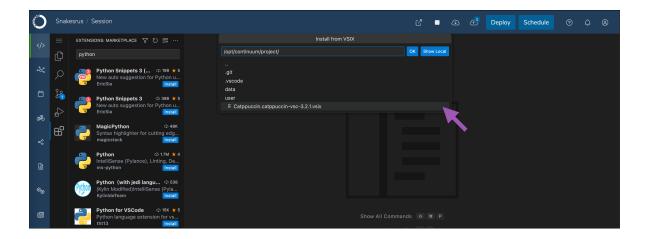
## Installing an extension manually

1. Open a browser and log in to Workbench as an administrator with managed persistence permissions.

**Tip:** The anaconda-enterprise user account has these permissions.

- 2. Open a project session using the VSCode editor.
- 3. Upload your .vsix file to the project.
- 4. Click the extensions icon in the left-hand navigation.
- 5. Click the more actions icon, then select **Install from VSIX...**.





# 3.6.10 Adding RStudio to Workbench

This topic covers how to install RStudio and use it within Data Science & AI Workbench.

**Note:** Anaconda recommends installing RStudio during a scheduled maintenance interval to prevent users from creating new sessions during installation. Existing sessions or deployments do not need to be halted.

## **Prerequisites**

These instructions are written for Workbench version 5.5.2 or later, and require the /tools volume from managed persistence to be enabled. For more information, see Mounting an external file share.

## **Installing RStudio**

# Launching the RStudio installer project

- 1. Download the RStudio installer project and save it to your machine.
- 2. Log in to Workbench as a user with read/write access to the /tools volume.

**Tip:** The anaconda-enterprise user has the correct permissions.

- 3. Use the **Create+/Upload Project** dialog to upload the RStudio installer project you just downloaded. Anaconda recommends using the JupyterLab editor for this task.
- 4. Open a session for the RStudio installer project.

## **Obtaining the RStudio Server binaries**

Now that your project is created, you need to obtain and install the Red hat Package Manager (RPM) files and pull them into the project session. There are multiple methods for accomplishing this.

## **Downloading RStudio directly from Workbench**

Anaconda recommends users with clusters that have internet access download RStudio directly through Workbench.

- 1. From your RStudio project session, open a new terminal window.
- 2. If you need to set proxy variables manually to access the internet, do so now.
- 3. Run the command:

```
bash download_rstudio.sh
```

## Here is an example output from the download script:

```
+----+
| AE5 RStudio Downloader |
+----+
- Target version: 2023.12.0-369
- Downloading into the data directory
- Downloading RHEL9 version only
- Downloading RHEL9/CentOS9 RPM file to data/rs-centos9.rpm
- URL: https://download2.rstudio.org/server/rhe19/x86_64/rstudio-server-
\rightarrowrhel-2023.12.0-369-x86 64.rpm
% Total
         % Received % Xferd Average Speed
                                          Time
                                                  Time
                                                          Time _
→Current
                             Dload Upload Total
                                                   Spent
                                                           Left Speed
100
   108M 100 108M
                           0 18.4M 0 0:00:05 0:00:05 --:-- 23.
\hookrightarrow 2M
- Verifying data/rs-centos9.rpm
+----+
The RStudio binaries have been downloaded.
You may now proceed with the installation step.
See the README.md file for more details.
+----+
```

### Using the download script on another Unix server

If you need to download the binaries on a separate machine first, use the download\_rstudio.sh script.

- Download the script file download\_rstudio.sh from the project to your desktop.
- 2. Move the script file to a machine that can run bash and curl and has connectivity to download2.rstudio.org.
- 3. Run the command:

```
bash download_rstudio.sh
```

- 4. If necessary, transfer the binaries rs-centos7.rpm and rs-centos8.rpm to the machine from which you access Workbench.
- 5. Use the JuptyerLab **upload** button to upload both binaries into the RStudio installer project.

#### Running the installation script

Now that you have obtained the RStudio binaries and uploaded the rs-centos7.rpm and rs-centos8.rpm files to your RStudio installer project:

- 1. Open a terminal window.
- If you have previously installed content into /tools/rstudio, remove it now by running the following command:

```
rm -r /tools/rstudio
```

3. Install RStudio by running the following command:

```
bash install_rstudio.sh
```

**Note:** The script will verify that all prerequisites are met before performing any file modifications.

## Here is an example output from the RStudio installer:

```
+----+
| AE5 RStudio Installer |
+----+
- Install prefix: /tools/rstudio
- Verifying data/rs-centos9.rpm
- Creating directory /tools/rstudio
- Unpacking RHEL9/CentOS9 package
1177155 blocks
- Moving files into final position
- Installing support files
+----+
RStudio installation is complete.
Once you have verified the installation, feel free to
shut down this session and delete the project.
+----+
WARNING: Many R packages make use of Java, and it seems
not to be present on this installation of AE5. To make
Java available to all AE5 users, run install_java.sh, or
manually download a JDK Linux x64 archive and unpack its
contents into the directory /tools/java.
+----+
```

## **Enabling RStudio as an editor**

With installation complete, the next step is to add RStudio to the editor selection list within Workbench's User Interface (UI).

- 1. Access your Kubernetes cluster resource management console.
- 2. Open the anaconda-enterprise-anaconda-platform.yml ConfigMap and search for the tools: section of the file.
- 3. Add the following line, nested under tools::

```
rstudio: Rstudio
```

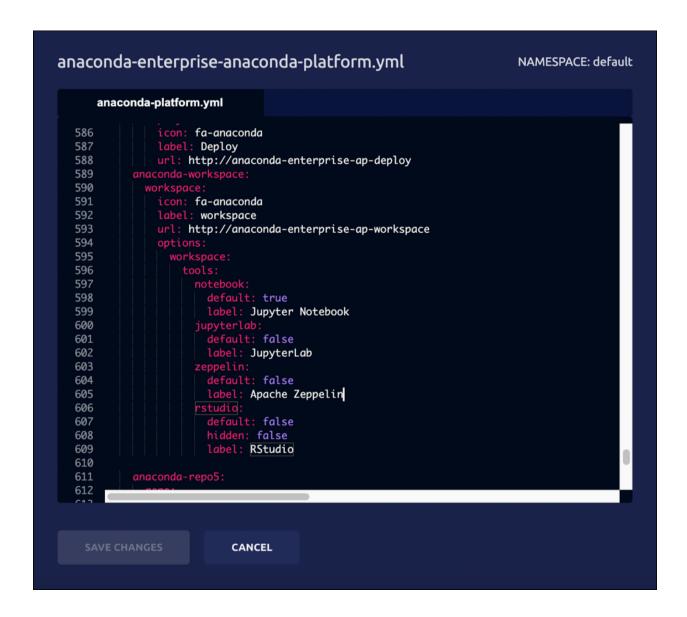
- 4. Save your work and close the file.
- 5. Open a terminal and restart the workspace and ui pods by running the following command:

```
kubectl delete --wait=false (kubectl get pods -o name|grep -E 'worksapce|ui <math>\rightarrow')
```

**Note:** Minor disruptions in UI responsiveness may occur while the pods are stabilizing. If you have allowed users to continue working during this installation, they may need to refresh their browsers.

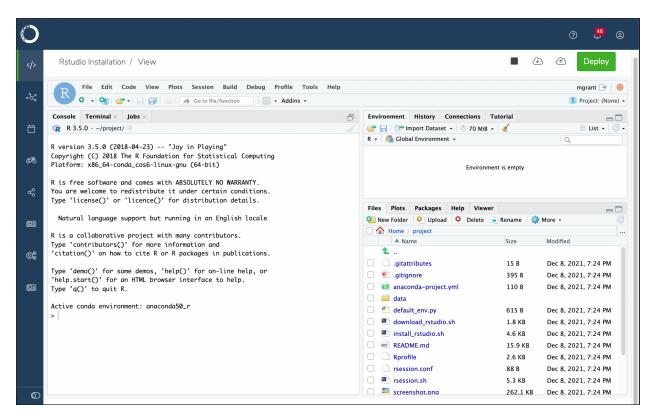
Once the cluster pods stabilize, the RStudio editor will be present in the **Default Editor** dropdown on any project's **Settings** page.

With RStudio enabled as an editor, your Op-Center will typically look like this:



### Verifying your installation

- 1. If necessary, stop the session for your RStudio installer project.
- 2. Go to the RStudio installer project's **Settings** page.
- 3. Open the **Default Editor** dropdown, select *RStudio*, and then click **Save**.
- 4. Start a new session and wait for the editor to launch.



#### If your installation is unsuccessful:

- 1. Click on the **Logs** tab.
- 2. Search the editor logs for a section titled Workbench R Session Manager and look for errors there and below.
- 3. Copy the full content of this log so it can be shared with Anaconda.
- 4. Stop the session completely.
- 5. Go to the project's **Settings** page, and change the **Default Editor** back to JupyterLab.
- 6. If your errors are obvious to you and understand what corrective action is needed, open a new session to make those changes.

Please reach out to Anaconda support if this occurs. Make sure to include a copy of the editor logs you obtained above.

## **Disabling RStudio**

To remove RStudio as an editor option for new projects:

- 1. Log in to your Kubernetes administrative console.
- 2. Open the anaconda-enterprise-anaconda-platform.yml ConfigMap file.
- 3. Search for the tools: section of this file and remove the following line:

rstudio: Rstudio

- 4. Save your work and close the file.
- 5. Open a terminal and restart the workspace and ui pods by running the following command:

```
kubectl delete --wait=false (kubectl get pods -o name|grep -E 'worksapce|ui <math>\hookrightarrow')
```

**Note:** Existing projects that are using RStudio will still have access to the editor.

## **Uninstalling RStudio**

**Caution:** If you need to permanently remove /tools/rstudio, instruct all users to stop all sessions currently using RStudio and choose a different editor, such as JupyterLab, before continuing. If they do not, sessions will fail to fully start.

To permanently uninstall RStudio, you need to remove the /tools/rstudio directory. Anaconda recommends removing the directory from outside Workbench, if possible. If necessary, instructions are provided to remove RStudio using Workbench.

## **Uninstalling RStudio using Workbench**

1. Log in to Workbench as a user with read/write access to the /tools volume.

**Tip:** The anaconda-enterprise user has the correct permissions.

2. Remove /tools/rstudio from within a Workbench session by running the following command:

rm -r /tools/rstudio

## Upgrading to a new version of Workbench

Upgrading to a new version of Workbench will not affect your RStudio installation; however, the upgrade process will cause the editor to be hidden. Once the upgrade is complete, you'll need to *enable RStudio as an editor* again.

## **Upgrading RStudio**

Anaconda recommends performing a fresh installation when upgrading RStudio.

- 1. Launch the RStudio installation project.
- 2. Obtain the RStudio server binaries.
- 3. (Optional) Rename your existing RStudio directory. If your installation is unsuccessful for any reason, you can rename the existing RStudio directory back to its original state to avoid downtime. Run the command:

mv /tools/rstudio /tools/rstudio.old

- 4. Instruct all users to stop all sessions currently using RStudio.
- 5. Run the installation script.
- 6. Enable RStudio as an editor.

- 7. Verify your installation.
- 8. If necessary, remove the old installation /tools/rstudio.old. Run the command:

```
rm -r /tools/rstudio.old
```

9. Log out of Workbench.

## Installing RStudio on additional Workbench instances

Once you have successfully installed RStudio on one instance, you can copy and move the /tools/rstudio directory to another instance, reducing the amount of work involved for a second install.

## Create an archive of an existing installation

- 1. Log in to an instance of Workbench with a running RStudio installation.
- 2. Open a session using the JupyterLab editor.
- 3. Open a terminal window.
- 4. Create the archive. Run the command:

```
tar cfz rstudio.tar.gz -C /tools rstudio
```

5. Download this file to your desktop. Once you have done so, you can remove the file from your Workbench session.

### Move the archive to a new instance

- 1. Open any session using the JupterLab editor.
- 2. Upload the archive rstudio.tar.gz into the project.
- 3. Open a terminal window.
- 4. (Optional) Rename your existing RStudio directory. If your installation is unsuccessful for any reason, you can rename the existing RStudio directory back to its original state to avoid downtime. Run the command:

```
mv /tools/rstudio /tools/rstudio.old
```

5. Install the archive. Run the command:

```
tar xfz rstudio.tar.gz -C /tools
```

- 6. Verify your installation.
- 7. If necessary, remove the old installation /tools/rstudio.old. Run the command:

```
rm -r /tools/rstudio.old
```

- 8. Enable RStudio as an editor.
- 9. Log out of Workbench.

# 3.6.11 Adding MLflow to Workbench

You can install MLflow as an optional component of Data Science & AI Workbench.

## **Prerequisites**

You must have managed persistence enabled.

#### Set environment variables

Setting environment variables allows MLflow to be accessible from all sessions, deployments, and schedules. This also sets the deployment-wide values for the MLflow tracking server endpoint.

- 1. Connect to your instance of Workbench.
- 2. View a list of your configmaps by running the following command:

```
kubectl get cm
```

3. Edit the anaconda-enterprise-env-var-config.yml file.

```
kubectl edit cm anaconda-enterprise-env-var-config
```

4. Include the following lines:

```
# Replace <WORKBENCH_FQDN> with your Workbench fully qualified domain name MLFLOW_DISABLE_ENV_MANAGER_CONDA_WARNING: TRUE MLFLOW_TRACKING_URI: https://mlflow.<WORKBENCH_FQDN> MLFLOW_REGISTRY_URI: https://mlflow.<WORKBENCH_FQDN>
```

**Note:** If your ENV\_VAR\_PLACEHOLDER: foo entry still exists, replace it now.

Here is an example of what your file might look like when complete:

- 5. Save your work and close the file.
- 6. Update Workbench with your changes and restart services by running the following command:

```
kubectl get pods | grep 'ap-deploy\|ap-workspace\|ap-ui' | cut -d' ' -f1 | xargs_ \rightarrowkubectl delete pods
```

### **Download MLflow**

- 1. Download the latest MLflow release.
- 2. Extract all files from the tarball you just downloaded.

**Tip:** Keep these somewhere that is easy for you to locate.

#### **Install MLflow**

- 1. Open a browser and navigate to Workbench.
- 2. Log in as an administrator account with managed persistence permissions.
- 3. Click Create +, then select Upload Project from the dropdown menu.
- 4. Click Browse.
- 5. Locate the extracted files from your download and select the MLflowTracking ServerProject-<VERSION>. tar.gz file.
- 6. Provide a name for your MLflow Server project.
- 7. Click Upload.



- 8. Open a session for your new MLflow Server project.
- 9. Upload the migrate-<VERSION>.py and anaconda-platform-mlflow-runtime-<VERSION>.tar.gz files to the root of the project.

**Caution:** Do not commit changes to the project until instructed. If you attempt to commit too early, you will receive an error due to the size of the runtime file.

10. Open a terminal in your session, and run the following command:

```
# Replace <VERSION> with your release version
python migrate-<VERSION>.py
```

11. Activate your new environment by running the following command:

```
conda activate /tools/mlflow/mlflow_env/
```

12. Verify your installation was successful by running the following command:

```
mlflow
```

Tip: If your install was successful, your command returns a list of available mlflow arguments.

- 13. Open the anaconda-project.yml file in the project with your preferred file editor.
- 14. Update the MLFLOW\_TRACKING\_GC\_TTL value to something that makes sense for your use case.

```
name: adsp_mlflow_appliance_runtime

variables:

# Do NOT change these unless instructed to do so.

SSL_CA_CERT_FILENAME: /run/secrets/kubernetes.io/serviceaccount/ca.crt

NAMESPACE_FILENAME: /run/secrets/kubernetes.io/serviceaccount/namespace

TOKEN_FILENAME: /run/secrets/kubernetes.io/serviceaccount/token

MLFLOW_TRACKING_SERVER_FORT: 8086

MLFLOW_TRACKING_SERVER_HOST: 0.0.0.0

MLFLOW_ARTIFACTS_DESTINATION: data/mlflow/artifacts

MLFLOW_DEFAULT_ARTIFACT_ROOT: mlflow-artifacts:/

DB_NAME: anaconda_mlflow

# Allow insecure TLS. (Do not run this way in production)

# MLFLOW_TRACKING_INSECURE_TLS: "true"

# Customer_Deployment_Configurable

MLFLOW_TRACKING_GC_TTL: 0dOhlOmOs
```

**Note:** The MLFLOW\_TRACKING\_GC\_TTL variable instructs MLflow to perform garbage collection on deleted artifacts that have reached a specified age.

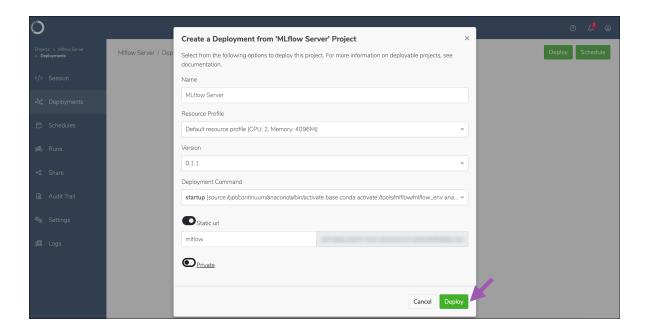
15. Commit the changes you've made to the project.

**Note:** It is not necessary to commit the migrate-<VERSION>.py file to the project. Once installation is complete, you can safely delete this file.

16. Stop the project session.

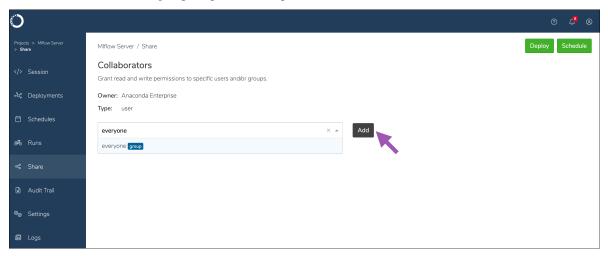
### **Deploy MLflow**

- 1. Select **Deployments** from the left-hand navigation.
- 2. Click Deploy.
- 3. Enter a name for the deployment, set the static url to the same value used in your anaconda-enterprise-env-var-config.yml file (https://mlflow.<AE5\_FQDN>), and then click **Deploy**.



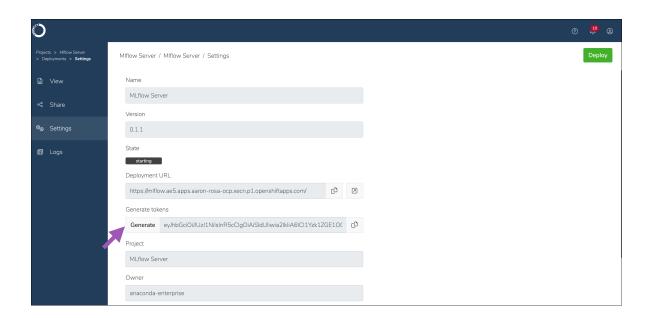
#### **Provide Access**

- 1. Select the deployment you just created.
- 2. Select **Share** from the left-hand navigation menu.
- 3. Enter the names of users or groups to provide with permissions to access MLflow, then click Add.



Note: This list populates from Keycloak.

- 4. Select **Settings** from the left-hand menu.
- 5. Click Generate to create a token for this deployment.



Note: Every user who needs API access to MLflow also requires this token. You must share this token manually.

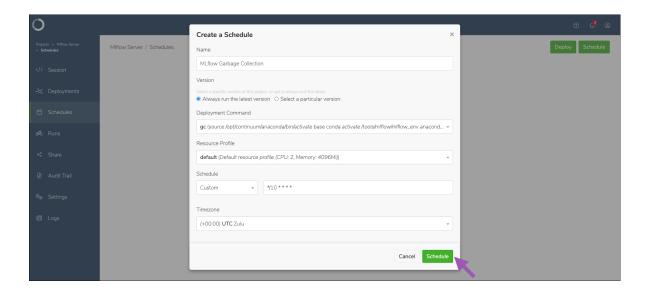
The administrator of the MLFlow Tracking Server *must* generate and provide the access token *each time* the server is restarted.

## Set up garbage collection

When a client deletes a resource, MLflow transitions the resource into a deleted lifecycle state and hides it in the UI, but does not purge it from the system. Deleted resources will block creation of new resources with the same name until the garbage collection process has purged it.

The garbage collection process works in tandem with the MLFLOW\_TRACKING\_GC\_TTL variable that is set in the anaconda-platform.yml project file. When a resource reaches the age specified by the MLFLOW\_TRACKING\_GC\_TTL variable AND the garbage collection process runs, it will be purged.

- 1. Create a schedule within the MLflow Server project.
- 2. Name the schedule MLflow Garbage Collection.
- 3. Open the **Deployment Command** dropdown menu and select **gc**.
- 4. Schedule an interval to run the garbage collection. Custom schedules utilize cron expressions.
- 5. Click Schedule.



## **Upgrading MLflow**

- 1. Download the latest MLflow release.
- 2. Open a browser and navigate to Workbench.
- 3. Log in as an administrator account with managed persistence permissions.
- 4. Open your MLflow Server project.
- 5. Select **Deployments** from the left-hand navigation.
- 6. Terminate your current deployment.
- 7. Select **Schedules** from the left-hand navigation.
- 8. Pause all scheduled runs.
- 9. Start a new session in your MLflow Server project.
- 10. Upload your newly obtained migrate-<VERSION>.py and anaconda-platform-mlflow-runtime-<VERSION>. tar.gz files to the root of the project.
- 11. Open a terminal in your project and run the following command:

```
# Replace <VERSION> with your release version
python migrate-<VERSION>.py
```

- 12. Redeploy your MLflow Server project.
- 13. Generate a new token to share your deployment.

Note: Remember, you *must* generate an access token and provide it to users *each time* the server is restarted!

14. Restart all scheduled runs.

# 3.6.12 Connecting to an external version control repository

Data Science & AI Workbench supports the use of an external version control repository for user-created projects. Anaconda recommends connecting to this external version control repository *at time of installation*, however you are also able to migrate from one version control repository to another after installation is complete.

- To provide permission granularity and maintain parity with your external version control repository, Workbench will grant individual platform users access to individual repositories. To prevent default permissions being applied to all users within a group, users cannot belong to the given organization or group.
- Platform users will be prompted for their access token before they create their first project in Workbench. Anaconda recommends you advise users to create an ever-lasting token, to retain permanent access to their files from within Workbench. The specific auth token permissions required for each repository are outlined here.

## Make sure you have the following prerequisites before you begin:

- The fully qualified domain name (FQDN) of your versions control server.
- The organization, team or group name associated with your service account.
- The username of the Administrator for the organization, team or group. This user will require full Admin permissions.
- The personal access token or password required to connect to your version control repository.

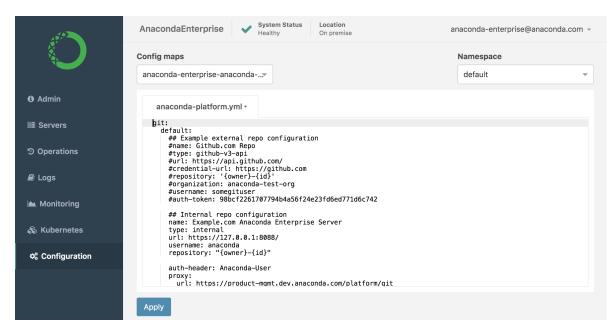
## Supported external git versions

Workbench supports integration with the following external repositories:

External repository	Supported versions	
GitHub Server	2.15, through 3.4.1	
GitHub Cloud	github.com	
Bitbucket Server/DC	5.9.1 through 7.21.0	
Bitbucket Cloud	bitbucket.org	
GitLab Self-Managed	10.4.2 through 14.9.2	
GitLab Cloud	gitlab.com	

### **Github Enterprise Edition (Server)**

- 1. Create a backup of the anaconda-enterprise-anaconda-platform.yml ConfigMap.
- 2. Open the anaconda-enterprise-anaconda-platform.yml ConfigMap for editing.
- 3. Locate the git section (pictured below)



- 4. *Uncomment* the Example external repo configuration section.
- 5. *Comment* the Internal repo configuration section.
- 6. Configure the Example external repo configuration.

name = A descriptive name for the service your organization uses.

type = The type of version control repository your organization uses: github-v3-api.

url = The URL of the API https://<FQDN>.com/api/v3/.

credential-url = The URL to authenticate against for repository operations such as cloning and pushing https:/
/<FQDN>.com/api/v3/.

repository = Must be '{owner}-{id}' encased in single quotes.

organization = The name of your Github organization.

username = The username associated with the Administrator account **This account must have full Admin permissions**.

auth-token = The personal access token for the Administrator account associated with the username.

#### **EXAMPLE:**

```
git:
    default:
        ## Example external repo configuration
    name: github-server
    type: github-v3-api
    url: https://<FQDN>/api/v3/
    http-timeout: 60
    credential-url: https://<FQDN>/api/v3/
    repository: '{owner}-{id}'
    organization: <github organization>
    username: <admin>
    auth-token: <token>
```

(continues on next page)

(continued from previous page)

```
## Internal repo configuration
#name: Example.com Workbench Server
#type: internal
#url: http://127.0.0.1:8088/
#http-timeout: 60
#username: anaconda
```

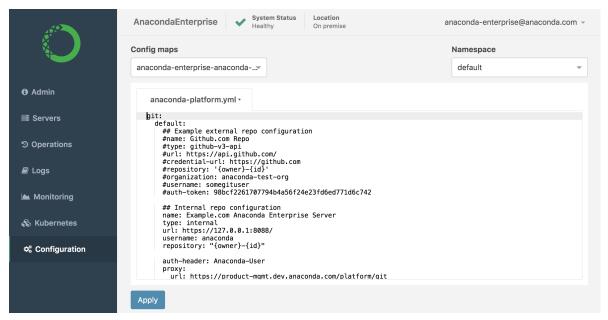
- 7. Save your changes.
- 8. Restart system pods with the following command for changes to take effect:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

Once all pods have returned to a running state, users should now be prompted to add their personal access token after logging into the platform.

## **Github Enterprise Edition (Cloud)**

- 1. Create a backup of the anaconda-enterprise-anaconda-platform.yml ConfigMap.
- 2. Open the anaconda-enterprise-anaconda-platform.yml ConfigMap for editing.
- 3. Locate the git section (pictured below)



- 4. *Uncomment* the Example external repo configuration section.
- 5. Comment the Internal repo configuration section.
- 6. Configure the Example external repo configuration.

name = A descriptive name for the service your organization uses.

type = The type of version control repository your organization uses: github-v3-api.

url = The URL of the API https://api.github.com/.

credential-url = The URL to authenticate against for repository operations such as cloning and pushing https://
/github.com/<github-organization>.

repository = Must be '{owner}-{id}' encased in single quotes.

organization = The name of your Github organization.

username = The username associated with the Administrator account **This account must have full Admin permissions**.

auth-token = The personal access token for the Administrator account associated with the username.

**EXAMPLE:** 

```
git:
  default:
    ## Example external repo configuration
   name: github-cloud
   type: github-v3-api
   url: https://api.github.com/
   http-timeout: 60
   credential-url: https://github.com/<github-organization>
   repository: '{owner}-{id}'
   organization: <github organization>
   username: <admin>
    auth-token: <token>
   ## Internal repo configuration
    #name: Example.com Workbench Server
    #type: internal
    #url: http://127.0.0.1:8088/
    #http-timeout: 60
    #username: anaconda
```

- 7. Save your changes.
- 8. Restart system pods with the following command for changes to take effect:

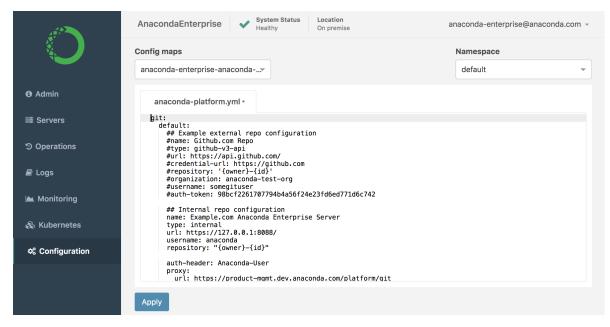
```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

Once all pods have returned to a running state, users should now be prompted to add their personal access token after logging into the platform.

## Bitbucket Server/Data Center

**Caution:** By default, Bitbucket uses master as its default Git branch. Workbench uses main as its default Git branch. You must update the system-wide default branch name in Bitbucket to target main. If you do not, your deployments will fail.

- 1. Create a backup of the anaconda-enterprise-anaconda-platform.yml ConfigMap.
- 2. Open the anaconda-enterprise-anaconda-platform.yml ConfigMap for editing.
- 3. Locate the git section (pictured below)



- 4. *Uncomment* the Example external repo configuration section.
- 5. Comment the Internal repo configuration section.
- 6. Configure the Example external repo configuration.

name = A descriptive name for the service your organization uses.

type = The type of version control repository your organization uses: bitbucket-v1-api.

url = The URL of the API https://<FQDN>:7990.

credential-url = The URL to authenticate against for repository operations such as cloning and pushing https:/
/<FQDN>:7990.

repository = Must be '{owner}-{id}' encased in single quotes.

organization = The name of your Bitbucket team.

username = The username associated with the Administrator account **This account must have full Admin permissions**.

auth-token = The Bitbucket app password for the Administrator account associated with the username.

#### **EXAMPLE:**

```
git:
    default:
        ## Example external repo configuration
    name: bitbucket-server
    type: bitbucket-v1-api
    url: https://<FQDN>:7990
    http-timeout: 60
    credential-url: https://<FQDN>:7990
    repository: '{owner}-{id}'
    organization: <bitbucket team>
    username: <admin>
    auth-token: <token>
```

(continues on next page)

(continued from previous page)

```
## Internal repo configuration
#name: Example.com Workbench Server
#type: internal
#url: http://127.0.0.1:8088/
#http-timeout: 60
#username: anaconda
```

- 7. Save your changes.
- 8. Restart system pods with the following command for changes to take effect:

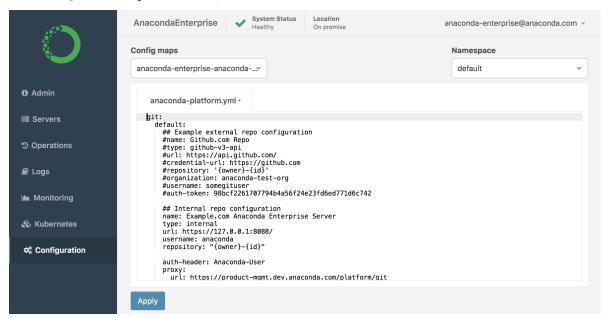
```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

Once all pods have returned to a running state, users should now be prompted to add their app password after logging into the platform.

#### **Bitbucket Cloud**

**Note:** Bitbucket.com does not support versioning of archive downloads and app deployments. In other words, **the latest revision will always be downloaded or deployed**.

- 1. Create a backup of the anaconda-enterprise-anaconda-platform.yml ConfigMap.
- 2. Open the anaconda-enterprise-anaconda-platform.yml ConfigMap for editing.
- 3. Locate the git section (pictured below)



- 4. Uncomment the Example external repo configuration section.
- 5. Comment the Internal repo configuration section.
- Configure the Example external repo configuration.
   name = A descriptive name for the service your organization uses.

```
type = The type of version control repository your organization uses: bitbucket-v2-api.
```

url = The URL of the API https://api.bitbucket.org.

credential-url = The URL to authenticate against for repository operations such as cloning and pushing
https://bitbucket.org.

repository = Must be '{owner}-{id}' encased in single quotes.

organization = The name of your Bitbucket team.

username = The username associated with the Administrator account **This account must have full Admin permissions**.

auth-token = The Bitbucket app password for the Administrator account associated with the username.

#### **EXAMPLE:**

```
git:
 default:
    ## Example external repo configuration
   name: bitbucket-cloud
    type: bitbucket-v2-api
   url: https://api.bitbucket.org
   http-timeout: 60
    credential-url: https://bitbucket.org/
    repository: '{owner}-{id}'
    organization: <bitbucket team>
    username: <admin>
    auth-token: <token>
    ## Internal repo configuration
    #name: Example.com Workbench Server
    #type: internal
    #url: http://127.0.0.1:8088/
    #http-timeout: 60
    #username: anaconda
```

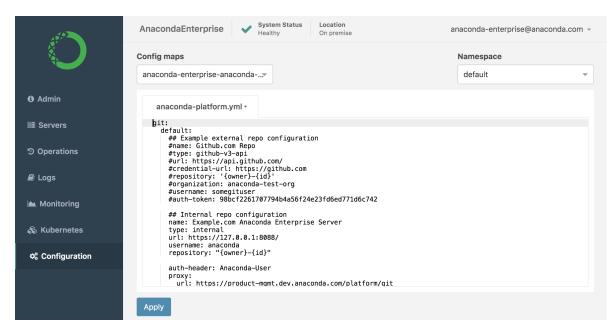
- 7. Save your changes.
- 8. Restart system pods with the following command for changes to take effect:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

Once all pods have returned to a running state, users should now be prompted to add their app password after logging into the platform.

#### **Gitlab Enterprise Edition (Self-Managed)**

- 1. Create a backup of the anaconda-enterprise-anaconda-platform.yml ConfigMap.
- 2. Open the anaconda-enterprise-anaconda-platform.yml ConfigMap for editing.
- 3. Locate the git section (pictured below)



- 4. *Uncomment* the Example external repo configuration section.
- 5. Comment the Internal repo configuration section.
- 6. Configure the Example external repo configuration.

name = A descriptive name for the service your organization uses.

type = The type of version control repository your organization uses: gitlab-v4-api.

url = The URL of the API https://<FQDN>.com.

credential-url = The URL to authenticate against for repository operations such as cloning and pushing https://<FQDN>.com.

repository = Must be '{owner}-{id}' encased in single quotes.

organization = The name of your GitLab group.

username = The username associated with the Administrator account **This account must have full Admin permissions**.

auth-token = The access token for the Administrator account associated with the username.

#### **EXAMPLE:**

```
git:
    efault:
        ## Example external repo configuration
    name: gitlab-server
    type: gitlab-v4-api
    url: https://<FQDN>.com
    http-timeout: 60
    credential-url: https://<FQDN>.com
    repository: '{owner}-{id}'
    organization: <gitlab group>
    username: <admin>
    auth-token: <token>
```

(continues on next page)

(continued from previous page)

```
## Internal repo configuration
#name: Example.com Workbench Server
#type: internal
#url: http://127.0.0.1:8088/
#http-timeout: 60
#username: anaconda
```

- 7. Save your changes.
- 8. Restart system pods with the following command for changes to take effect:

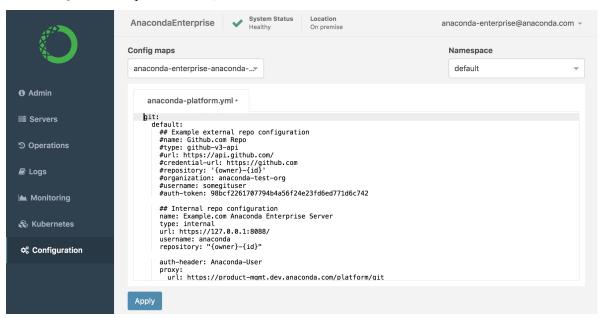
```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

Once all pods have returned to a running state, users should now be prompted to add their access token after logging into the platform.

#### **Gitlab Enterprise Edition (Cloud)**

**Note:** GitLab.com does not support versioning of archive downloads and app deployments. In other words, **the latest revision will always be downloaded or deployed**.

- 1. Create a backup of the anaconda-enterprise-anaconda-platform.yml ConfigMap.
- 2. Open the anaconda-enterprise-anaconda-platform.yml ConfigMap for editing.
- 3. Locate the git section (pictured below)



- 4. Uncomment the Example external repo configuration section.
- 5. Comment the Internal repo configuration section.
- 6. Configure the Example external repo configuration.

name = A descriptive name for the service your organization uses.

```
type = The type of version control repository your organization uses: gitlab-v4-api.
```

url = The URL of the API https://gitlab.com.

credential-url = The URL to authenticate against for repository operations such as cloning and pushing
https://gitlab.com.

repository = Must be '{owner}-{id}' encased in single quotes.

organization = The name of your GitLab group.

username = The username associated with the Administrator account **This account must have full Admin permissions**.

auth-token = The access token for the Administrator account associated with the username.

#### **EXAMPLE:**

```
git:
 default:
    ## Example external repo configuration
   name: gitlab-cloud
    type: gitlab-v4-api
   url: https://gitlab.com
   http-timeout: 60
    credential-url: https://gitlab.com
   repository: '{owner}-{id}'
    organization: <gitlab group>
    username: <admin>
    auth-token: <token>
    ## Internal repo configuration
    #name: Example.com Workbench Server
    #type: internal
    #url: http://127.0.0.1:8088/
    #http-timeout: 60
    #username: anaconda
```

- 7. Save your changes.
- 8. Restart system pods with the following command for changes to take effect:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

Once all pods have returned to a running state, users should now be prompted to add their access token after logging into the platform.

### 3.6.13 Migrating projects between version control repositories

If your organization has changed Git hosting services, and you therefore need to migrate projects from one *supported* version control repository to another, Anaconda recommends you follow this high-level process:

- 1. Perform pre-migration setup.
- 2. Run the project migration script.
- 3. Perform post migration cleanup.
- 4. Adding collaborators.

#### **Prerequisites:**

- *Update the Data Science & AI Workbench config map* with the information required to connect to the external version control repository.
- To run the project migration script, you'll need Administrator access to a command line tool that can run bash or Python scripts *on the master node of the Workbench cluster*.
- Ensure a recent version of git is installed on the master node
- You'll also need the origin Git host token/password, and destination Git host token/password.

### **Pre-migration setup**

1. If you haven't already done so, *on the master node*, change to the directory of the unpacked Workbench installer and install the bootstrap conda environment:

```
bash conda-bootstrap.sh
```

- 2. After the environment is finished installing, you may need to log out and log back in to activate the conda environment.
- 3. Temporarily disable reverse proxy authentication by adding the following key-value pair to the git section (outside of the storage section in the config map) of the anaconda-enterprise-anaconda-platform.yml file used to configure the platform to use an external version control repository:

```
reverse-proxy-auth: false
```

This should look similar to the following:

```
451 git:
452 url: https://aaron-541-timeout.svc.anaconda.com/platform/git # externally visible URL of the git server
host: aaron-541-timeout.svc.anaconda.com # full hostname of the git server
port: 8088
https:
455 key: /etc/secrets/certs/tls.key
456 key: /etc/secrets/certs/tls.key
457 certificate: /etc/secrets/certs/tls.crt
458 db:
459 database: anaconda_git
460 directory: /export # directory where git server will store its data
461 username: anaconda # OS username that the git server should run under
462 reverse-proxy-auth: false
463 lfs-secret: AohzzmIZVHYSTY]7MMIEIGMhjRYCTcfLdxHHGR8fKCM # LFS authentication token secret. Should be uniquely generated for each installation.
465 mox-commit-file-size: 500000000
```

4. Run the following command to restart the associated pod on the master node:

```
kubectl delete pod -l 'app=ap-git-storage'
```

5. Create a user mappings file that maps Workbench user IDs to Git user IDs. This is a colon-separated text file where the first field is the Workbench user name, and the second field is the corresponding Git user name. For example:

```
ae-admin:git-admin
ae-user1:git-user1
ae-user2:git-user2
```

**Note:** If you intend on migrating to or from a Bitbucket repository, you must use your Bitbucket account ID instead of your Bitbucket username in the user mappings file.

### Using the migration tool

Caution: Using the migration tool with https instead of http for the internal storage may result in an SSL error.

The migration tool is a Python script, migrate\_projects.py, found in the Workbench installation tarball. It can be used in the following ways:

```
usage: migrate_projects.py [-h] [--parallel PARALLEL] [--log-file LOG_FILE]
                      [--force-migrate] [--scratch-dir SCRATCH_DIR]
                      --postgres-host POSTGRES_HOST
                      [--postgres-user POSTGRES_USER]
                      [--postgres-passwd POSTGRES_PASSWD]
                      [--origin-api-type {internal,bitbucket-v1-api,bitbucket-v2-api,

→ github-v3-api, gitlab-v4-api}]
                      --origin-api-url ORIGIN_API_URL
                      [--origin-username ORIGIN_USERNAME]
                      [--origin-token ORIGIN_TOKEN]
                      [--origin-organization ORIGIN_ORGANIZATION]
                      [--dest-api-type {internal,bitbucket-v1-api,bitbucket-v2-api,
⇒github-v3-api,gitlab-v4-api}]
                      --dest-api-url DEST_API_URL
                      [--dest-username DEST_USERNAME]
                      [--dest-token DEST_TOKEN]
                      [--dest-organization DEST_ORGANIZATION]
                      --dest-user-mappings DEST_USER_MAPPINGS
optional arguments:
-h, --help
                      show this help message and exit
--cloud
                      Set if using Gitlab Cloud, or older version of Gitlab On-Prem
--parallel PARALLEL
                      Number of parallel migration jobs to spawn
--log-file LOG_FILE
                      Path prefix to log directory, suffixed with a
                    timestamp, e.g. migrate-projects-
                    log-1559234750640867208
                      Forces migration by replacing local and destination
--force-migrate
                    repositories
--scratch-dir SCRATCH_DIR
                    The scratch directory for cloning project repositories
--postgres-host POSTGRES_HOST
                    Hostname of Workbench Postgres DB
--postgres-user POSTGRES_USER
                    Username of Workbench postgres DB
--postgres-passwd POSTGRES_PASSWD
                    Password of Workbench postgres DB
--origin-api-type {internal,bitbucket-v1-api,bitbucket-v2-api,github-v3-api,gitlab-v4-
→api}
                    Origin git host API type
--origin-api-url ORIGIN_API_URL
                    Origin git host API URL
--origin-username ORIGIN_USERNAME
                    Origin git host username
--origin-token ORIGIN_TOKEN
```

(continues on next page)

(continued from previous page)

```
Origin git host auth token
--origin-organization ORIGIN_ORGANIZATION
                    Origin git host organization
--dest-api-type {internal,bitbucket-v1-api,bitbucket-v2-api,github-v3-api,gitlab-v4-api}
                    Destination git host API type
--dest-api-url DEST_API_URL
                    Destination git host API URL
--dest-username DEST_USERNAME
                    Destination git host username
--dest-token DEST_TOKEN
                    Destination git host auth token
--dest-organization DEST_ORGANIZATION
                    Destination git host organization
--dest-user-mappings DEST_USER_MAPPINGS
                    Colon-separated Workbench-to-git-host mappings file, e.g. ae-
                    user1:github-user1
```

For example, the tool can be used in the following way:

```
python migrate_projects.py \
    --postgres-host localhost --origin-api-url http://localhost:8443/ \
    --origin-username root --dest-api-type gitlab-v4-api \
    --dest-api-url https://mbrock-gitlab.anacondaenterprise.com/ \
    --dest-username root --dest-organization demo --dest-user-mappings \
    user-mappings-gitea-to-gitlab.txt --force-migrate --parallel 4
```

To ensure tokens are not visible in bash history, they can be omitted and can be entered via stdin when running the script.

**Note:** The postgres password can be left blank. When migrating *from* Workbench, the origin-token can be left blank. When migrating *to* Workbench, the dest-token can be left blank.

**Note:** When migrating to Gitlab Cloud, please use the --cloud flag.

#### **Post-migration cleanup**

After the script finishes migrating the projects, re-enable reverse proxy authentication by editing the key-value pair you previously added to the git section of the anaconda-enterprise-anaconda-platform.yml file, so it looks like the following:

```
reverse-proxy-auth: true
```

**Caution:** If you do not re-enable reverse proxy authentication, Workbench will not work.

To verify that the new repository is being used by Workbench, edit an existing project and commit your changes to it.

### **Adding collaborators**

If you've migrated to https://github.com, whenever a user is added to a project as a collaborator, they'll be sent an invitation to collaborate via email. They'll need to accept this invitation to be able to commit changes to the repository associated with the project. This does not apply to Github Enterprise.

## 3.6.14 Mounting an external file share

Data Science & AI Workbench provides a mechanism to mount shared volumes into sessions, deployments, and jobs, using the volumes: section of the platform configuration YAML file. In Workbench 5.4 and earlier, this was limited strictly to NFS volumes.

With Workbench version 5.5+ and the Bring Your Own Kubernetes (BYOK8s) option, this capability has been extended to support standard Kubernetes Persistent Volumes (PVs) and Persistent Volume Claims (PVCs). This addition allows sessions, deployments, and pods to access a much wider variety of shared volumes. The specific set choices available to you will depend on the specific Kubernetes implementation and host. See this section of the Kubernetes documentation for a list of volume types—but again, check the documentation for your specific Kubernetes provider for the subset of those choices available to you.

#### Before you begin:

Adding a new file share requires editing the Workbench ConfigMap and restarting the workspace and deploy pods. For that reason, Anaconda recommends scheduling a maintenance window for this task and backing up the current version of anaconda-enterprise-anaconda-platform.yml before making changes to it.

#### Creating the volume list

Volume specifications are provided to Workbench in the volumes: section of the ConfigMap, as seen in the following example:

```
56
    volumes:
57
         /absolute/path1:
58
             subPath: ""
             readOnly: false
59
             groupID: 1234
60
61
             pvc: efs-volume1
62
         relative/path2:
             subPath: ""
63
64
             readOnly: true
65
             groupID: 1001
             nfs:
66
67
                 server: 10.123.23.45
                 path: /exported/volume
68
                 readOnly: true
69
70
```

This section is a dictionary with one entry for each mount.

The dictionary key is the *mount* path for the volume. If the path does not begin with a forward slash /, then it is assumed to be *relative* to the /data subdirectory. So, for instance, the second mount path above will be /data/relative/path2.

subPath (optional): the subfolder of the shared filesystem to mount. Any leading slashes are ignored, and the path is assumed relative to the root of the exported mount. This is a somewhat advanced configuration; see *SubPaths and Template Expansion* section below for an advanced use case that can take advantage of this option.

readOnly (optional): if true, the file share will be mounted as read-only; that is, sessions, deployments, and jobs will not be able to write to the volume. If false, or omitted entirely, the file share will be mounted read-write.

**Note:** For back-compatibility, the readOnly flag can be provided in two different locations: either within the nfs specification or outside of it. The volume will be mounted read-only if either value is true.

groupID (optional): a Unix GID which has read and/or write access to the volume. See the section *Ensuring Access* below for more information.

Finally, the volume specification itself is given by *one* of the two entries: pvc or nfs.

pvc: the name of the Kubernetes Persistent Volume Claim. This PVC must meet the following criteria:

- The PVC must be *pre-created*. Workbench will *not* auto-create a PVC for you. Thus the PVC and the Persistent Volume (PV) it is bound to must already exist.
- If readOnly is false, it must have a ReadWriteMany access policy. If readOnly is true, it must be either ReadWriteMany or ReadOnlyMany.

nfs: this is an NFS import specification, and its syntax has not changed from Workbench 5.4—and remains the only option for multi-node, Gravity-based clusters.

- server: the FQDN or IP address of the NFS server.
- path: the exported path from the NFS server.
- readOnly: see above.

#### **Ensuring access**

For all mounted volumes, it is important to ensure that both read and write permissions are aligned with those of the session, deployment, and job containers. By default, containers run with a fixed, nonzero UID—identical across all users—and a GID of 0. Kubernetes also provides the facility to add supplemental groups to the containers, and we have added a limited ability to utilize this capability. To that end, here are the scenarios that Workbench supports in this standard case:

If the GID of the content on the volume is 0, then sessions, deployments, and pods will be able to access it without further configuration.

If the GID of the content is a single, nonzero value, then sessions, deployments, and pods can be given access to it by taking one of the following steps:

- Specify the GID using a pv.beta.kubernetes.io/gid Persistent Volume annotation. (This must be added to the PV, *not* to the PVC.) Many persistent volume providers do this automatically—if so, Workbench 5.5 will pick this up automatically as well, and no further configuration is needed.
- Specify the GID for the volume using the groupID option above.

In all of these scenarios, only one GID is allowed per volume. So, for instance, in the example configuration above, the GID list will be 0, 1001, 1234. Files accessible by any of these three GIDs will be accessible.

To access volumes with more complex Unix access permissions, including individual user permissions, the Authenticated NFS integration pattern will likely be necessary. See the dedicated documentation for that pattern for details.

#### Adding the volume list to the ConfigMap

**Note:** The example provided in this section is for Gravity installs. Bring your own Kubernetes installs do not have the Operations Center and need to use a different method of managing the ConfigMap.

- 1. Log in to Workbench, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
- 4. Select **Kubernetes** from the menu on the left.

**Note:** At this point Anaconda recommends copying the existing configuration into a backup text file, should you need to restore it.

5. Under the ConfigMaps tab, select the anaconda-enterprise-anaconda-platform.yml configuration file.



- 6. If the volume section is commented out, uncomment it. Whether it's commented out or not, replace it with the new volume configuration according to the *volume section* above. It is very important to respect the indentation of the volume section, so confirm that before saving.
- 7. Click **Apply** to save your changes to the ConfigMap.

8. To update the Workbench server with your changes, restart services by running these commands *on the master node*:

```
sudo gravity enter kubectl get pods | grep 'workspace\|deploy' | cut -d' ' -f1 | xargs kubectl delete...
```

- 9. Wait about 10 seconds, and then run kubectl get pods. Examine the workspace and deploy pods. You may still see the old versions of the pods shutting down, which is expected. The new pods should be running as expected. If for some reason there are errors, it is likely because your ConfigMap is improperly formatted. Double check your editing and repeat steps 5 through 7 above to correct the issue.
- 10. To verify that Workbench users can access the volumes you add, start a new project editing session or restart an existing one and confirm that the volumes are listed as available directories. See *Loading data* for more information.

### Addressing file share downtime

If a server that you have configured as a volume mount in Workbench goes offline, editor sessions, applications and deployment jobs that use it will not start. If downtime occurs, Anaconda recommends disabling the mount until the server resumes proper functioning.

### Disabling a volume mount

- 1. Stop any sessions, deployments, or jobs that rely on that shared volume.
- 2. Open the ConfigMap and comment out the volume section.
- 3. Restart the workspace and deploy pods as described in step 8 above.

#### ADVANCED: SubPaths and Template Expansion

**Note:** The capability described in this section should be considered advanced and/or experimental in nature. Most customers can safely ignore this section. Instead, they should consider the new Managed Persistence feature, which leverages these capabilities in a very specific and tested manner. If you do find a novel use case for subPaths and templating, please share it with us!

The subPath option can be used to mount just a *subdirectory* of a shared volume at multiple mount points in the session. Anything above that subdirectory would not be visible within the container. For example, consider a shared volume with the following fileset:

```
/testA
/user1/testB
/user2/testC
```

Suppose this volume is exported via a PVC called pv1, and consider the following mount configuration:

```
mount1:
    pvc: pv1
    subPath: user1
mount2/mount3:
    pvc: pv1
    subPath: user2
```

If nothing else is mounted into the /data path, the container would see the following tree:

```
/data/mount1/testB
/data/mount2/mount3/testC
```

Notice that testA is not accessible at all, and the user1 and user2 are not even at the same "depth" of the directory tree anymore. As a result, sub path mounting offers a simple form of access control, allowing only portions of a shared volume to be made available to sessions, deployments, and jobs.

The usefulness of this subpath functionality has been enhanced with simple *templating* functionality. Specifically, the subPath value can include one or more of the following strings (braces included), and Workbench will dynamically substitute a relevant value in its place:

- {user}: the username of the person running the session, deployment, or job
- {owner}: the username of the owner of the project associated with the session, deployment, or job. These will differ in a collaboration scenario: if userA created a project and shared it with userB, who is now running a session, {user} would be userB and {owner} would be userA.
- {id}: the 32-character uuid of the session, deployment, or job.
- {project}: the 32-character unid of the project itself. This will be the same across all sessions, deployments, and jobs associated with that project.

Using these templates, we can construct mounting configurations that dynamically select subdirectories based upon the context of the running container. For example, consider the following mount configuration:

```
shared_data:
    pvc: pv1
    subPath: owner/{owner}/{project}
unshared_data:
    pvc: pv1
    subPath: user/{user}/{project}
scratch:
    pvc: pv1
    subPath: scratch/{id}
```

Then given a project owned by userA, and a session being run by userB, the three subpaths might be

```
/data/shared_data -> owner/userA/23dc83c697ca4c21bdc3c96e6b7d2c86
/data/unshared_data -> user/userB/23dc83c697ca4c21bdc3c96e6b7d2c86
/data/scratch -> scratch/bf38438b5a8044d190de3c2899d6da9e
```

depending, of course, on the precise UUID values for the project and session.

As mentioned in the note in the beginning of this section, subpaths and templating represent an advanced feature and Anaconda recommends it only for experimental use. But they do sit at the core of our Managed Persistence support, so we are ensuring this functionality is tested and supported for that use case. If you find a new use for this capability, please let us know so that we can ensure it is included in our support.

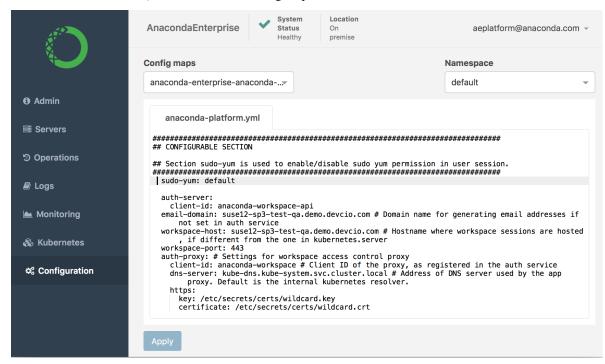
## 3.6.15 Disabling sudo for yum

By default, sudo access for yum is enabled on the Data Science & AI Workbench platform. You can easily disable it, however, if your organization requires it.

- 1. Log in to Workbench, select the **Menu** icon in the top right corner, and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
- 4. Select **Configuration** from the menu on the left.
- 5. Verify that the anaconda-enterprise-anaconda-platform.yml configuration file is selected in the **Config** map drop-down menu.

**Note:** Anaconda recommends that you make a backup copy of this file since you will be editing it directly.

6. Scroll down to the sudo-yum section of the Config map:



7. Change the setting from default to disable:

```
sudo-yum: disable
```

- 8. Click **Apply** to save your changes.
- 9. To update the Workbench server with your changes, restart services by running these commands *on the master node*:

```
sudo gravity enter
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

To re-enable sudo yum, simply change this Config map setting back to default, save your changes, and restart services.

## 3.6.16 Specifying alternate wildcard domains

By default, Data Science & AI Workbench expects the wildcard domain to be the same for the primary platform server and the application domain.

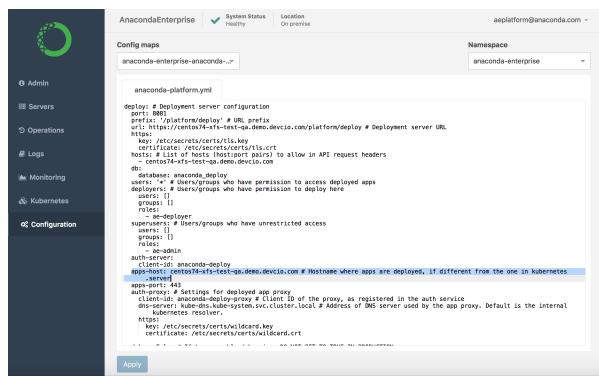
If your particular implementation uses different domains, you'll need to update the configuration file for the platform with the fully qualified domain name (FQDN) for each server.

**Note:** Make sure the wildcard domain has a TLS cert and DNS entry that meets *these requirements* before you follow the process below to specify it as an apps-host or workspace-host.

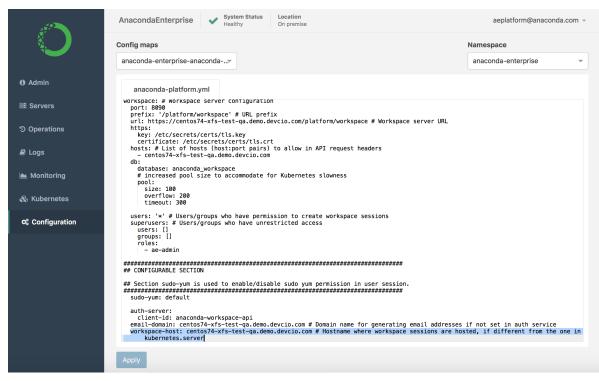
- 1. Log in to Workbench, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
- 4. Select **Configuration** from the menu on the left.
- 5. Verify that the anaconda-enterprise-anaconda-platform.yml configuration file is selected in the **Config** map drop-down menu.

**Note:** Any changes you make will impact how Workbench functions, so **we strongly recommend that you save a copy of the original file** before making any changes.

6. Scroll down to the Deployment server configuration section of the Config map:



- 7. Search for and update the apps-host setting with the FQDN of the host server you'll be deploying apps to, if it's different than the default Kubernetes server.
- 8. Scroll down to the Workspace server configuration section of the Config map:



- 9. Update the workspace-host setting with the FQDN of the host server you'll be using as a workspace server, if it's different than the default Kubernetes server.
- 10. Click **Apply** to save your changes.
- 11. To update the Workbench server with your changes, restart services by running these commands *on the master node*:

```
sudo gravity enter kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

# 3.6.17 Setting global config variables

Data Science & AI Workbench provides a secondary config map (named anaconda-enterprise-env-var-config) that you can use to configure the platform. Any environment variables that you add to this config map will be available to sessions, deployments and schedules. This is a convenient alternative to using the Workbench CLI, as you can add any variable supported by conda configuration.

- 1. Log in to Workbench, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
- 4. Select **Configuration** from the menu on the left.

- 5. Use the **Config map** drop-down menu to select the anaconda-enterprise-env-var-config.yml configuration file. The default config map contains a placeholder only: ENV\_VAR\_PLACEHOLDER: foo.
- 6. To add an environment variable, replace this placeholder with an actual entry. For example, to configure Workbench to *use a proxy for conda packages*, you might add entries that resemble the following:

```
HTTP_PROXY: proxy.url.com:3128
NO_PROXY: anaconda-test.url.com
```

- 7. Click **Apply** to save your changes.
- 8. To update Workbench with your changes, restart services by running these commands on the master node:

```
sudo gravity enter kubectl get pods | grep 'ap-deploy\|ap-workspace' | cut -d' ' -f1 | xargs kubectl →delete pods
```

# 3.6.18 Connecting to external Postgres

**Caution:** Anaconda recommends that you connect to your external Postgres instance immediately after installation completes. If you do not, you will need to create a dump file from the internal Workbench Postgres instance, then import it into your own Postgres instance, ensuring that it mirrors the internal setup.

You can connect Workbench to your organization's pre-existing Postgres instance for both BYOK8s and Gravity installations. Connecting to an external Postgres instance enables you to take advantage of multiple High Availability/Disaster Recovery (HA/DR) options your organization may already have in place.

Workbench also supports connections to external Postgres instances that are hosted within the same overarching cluster environment. For example, your organization can have a multi-namespace/region deployment of Postgres within your Kubernetes cluster that connects to Workbench via the Service name. Combining this with dynamic block Persistent Storage and Managed Persistence enables Workbench to rapidly relocate deployed projects from failing nodes to healthy ones utilizing standard Kubernetes pod scheduling.

#### **Connecting to external Postgres**

To connect to your external Postgres instance:

- 1. Create a Workbench account with read/write access and set the account's password.
- 2. Open your anaconda-enterprise-anaconda-platform.yml file and edit the following section:

```
db: # Database client configuration
host: <IP/Hostname> # Database hostname
port: 5432
username: "<postgres-account>"
password: "<password>"
```

3. Save your changes, then restart all pods:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

4. Once the pods are in a running state, confirm that the following databases have been created in Postgres.

```
anaconda_auth
anaconda_auth_escrow
anaconda_deploy
anaconda_git
anaconda_operation_controller
anaconda_repository
anaconda_storage
anaconda_ui
anaconda_workspace
```

**Caution:** If a pod shows a crashloop state, inspect the pod's logs. The most common cause of a crashloop state is a misconfiguration.

**Caution:** If the anaconda\_auth database is not created in Postgres, you must update the following environmental variables in the anaconda-enterprise-ap-auth deployment:

- # Replace <POSTGRES> with the hostname or service name of the Postgres.  $\rightarrow$  instance
- # Replace <POSTGRES-ACCOUNT> with the user account name for your\_
- →Postgres instance
- # Replace <PASSWORD> with the user account password for your Postgres\_
- →instance

#### env:

- name: WAIT\_FOR\_IT

value: '<POSTGRES>:5432'

- name: DB\_USER

value: <POSTGRES-ACCOUNT>

name: DB\_PASSWORD value: <PASSWORD>name: DB\_ADDR value: <POSTGRES>

## 3.6.19 Migrating from Anaconda Enterprise 4 to Workbench

The process of migrating from Anaconda Enterprise 4 to Data Science & AI Workbench involves the following tasks:

#### For Administrators:

- Export all packages and package information from your Anaconda Enterprise 4 Repository.
- Import the packages into Workbench.

### For Notebook users:

- Export each project environment to a .yml file.
- Convert each project into a format compatible with Workbench.
- Upload each project into Workbench.

Due to architectural changes between versions of the platform, there are some additional steps you may need to follow to *migrate code* between Anaconda Enterprise 4 and Workbench. These steps vary, based your current and new platform configurations.

### **Exporting packages**

Workbench enables you to create a site dump of all packages used by your organization, including the owners and permissions associated with each package.

- 1. Log in to the Anaconda Enterprise 4 Repo and switch to the anaconda-server user.
- 2. To export your packages, run the following command on the server hosting your Anaconda Enterprise 4 Repository:

```
anaconda-server-admin export-site
```

Running this command creates a directory structure containing all files and user information from your Anaconda Enterprise 4 Repository. For example:

Each subdirectory of site-dump contains the contents of the Repository *as it pertains to a particular user*. For example anaconda-user-1 has two packages, moto and pysocks. The meta. j son file in each user directory contains information about any groups of end users that user belongs to, as well as their organizations.

Package directories contain the package files, prefixed with the id of the database. The meta.json file in each package directory contains metadata about the packages, including version, build number, dependencies, and build requirements.

**Note:** Other files included in the site-dump—such as projects and environments—are NOT imported by the package import tool. That's why users have to *export their Notebook projects* separately.

### Importing packages

You can choose whether to import packages into Workbench by username or organization, or import all packages.

#### Before you begin:

- Anaconda recommends you compare the import options before proceeding, so you can choose the option that most closely aligns with the desired outcome for your organization.
- You'll be using the Workbench command line interface (CLI) to import the packages you exported, so be sure to *install the Workbench CLI* if you haven't already.
- 1. Log in to the command line interface using the following command:

```
anaconda-enterprise-cli login
```

2. Follow the instructions below for the method you want to use to import packages.

#### To import packages by username or organization:

As you saw in the example above, the packages for each user are put in a separate directory in the site-dump. This means that the import process is the same whether you specify a directory based on a username or organization.

Import a single directory from the site-dump using the following command:

```
anaconda-enterprise-cli admin import site-dump/name
```

Replacing name with the actual name of the directory you want to import.

Note: You can also pass a list of directories to import.

#### To import all packages:

Run the following command to import all packages in the site dump:

```
anaconda-enterprise-cli admin import site-dump/*
```

#### How channels of imported packages are named

When you import packages by username, a new channel is created for each unique label the user has applied to their packages, using the username as a prefix. (The default package label "main" is not included in channel names.)

For example, if anaconda-user-1 has the following packages:

- moto-0.4.31-2.tar.bz2 with label main
- pysocks-1.6.6-py35\_0.tar.bz2 with label test

The following channels are created:

- anaconda-user-1 containing the package file moto-0.4.31-2.tar.bz2
- anaconda-user-1/test containing the package file pysocks-1.6.6-py35\_0.tar.bz2

When you import all packages in an organization, a new channel is created for each organization, group, and label. The script appends any groups associated with the organization to the channel name it creates. (The default package label "main" and default organization label "Owner" are not included in channel names.)

For example, if anaconda-organization includes a group called Devs, and the site dump for anaconda-organization contains a package file named xz-5.2.2-1.tar.bz2 with the label Test, running the script will create the following channels:

- anaconda-organization This channel contains all packages that the organization owner can access.
- anaconda-organization/Devs This channel contains all packages that the Dev group can access.
- anaconda-organization/Devs/Test This channel contains all packages labeled Test that the Dev group can access.

#### Granting access to channels and packages

After everything is uploaded, each channel created as part of the import process is shared with the appropriate users and groups. In the case of the example above, ``anaconda-user-1`` is granted *read-write* access to the anaconda-user-1 and anaconda-user-1/test channels, and all members of the Devs group will have *read* permission for everything in the Devs channel.

You can change these access permissions as needed using the Workbench UI or CLI. See *Managing channels and packages* for more information.

### Migrating Anaconda Enterprise 4 Notebook Projects

#### Before you begin:

- If your project refers to channels in your on-premises repository or other channels in anaconda.org, ask you System Administrator to *mirror those channels* and make them available to you in Workbench.
- If your project use non-conda packages, you'll need to *upload those packages* to Workbench.
- If your notebook refers to multiple kernels or environments, set the kernel to a single environment.
- If your project contains several notebooks, verify that they all are using the same kernel or environment.

#### **Exporting your project**

Exporting a project creates a yml file that includes all the environment information for the project.

- 1. Log in to your Anaconda Enterprise 4 Notebooks server.
- 2. Open a terminal window and activate conda environment 2.6 for your project.
- 3. Install anaconda project in the environment:

```
conda install anaconda-project=0.6.0
```

If you get a not found message, install it from anaconda.org:

```
conda install -c anaconda anaconda-project=0.6.0
```

4. Export your environment to a file:

```
conda env export -n default -f _env.yml
```

<default> is the name of the environment where the notebook runs.

5. Verify that the format of the environment file looks similar to the following, and that the dependencies for each notebook in the project are listed:

```
yaml
channels:
- wakari
- r
```

(continues on next page)

(continued from previous page)

```
- https://conda.anaconda.org/wakari
- defaults
- anaconda-adam
prefix: /projects/anaconda/MigrationExample/envs/default
dependencies:
- _license=1.1=py27_1
- accelerate=2.3.1=np111py27_0
- accelerate_cudalib=2.0=0
- alabaster=0.7.9=py27_0
# ... etc ...
```

If it contains any warning messages, run this script to modify the encoding and remove the warnings:

```
import ruamel_yaml
with open("_env.yml") as env_fd:
    env = ruamel_yaml.load(env_fd)
with open("environment.yml", "w") as env_fd:
    ruamel_yaml.dump(env, env_fd, Dumper=ruamel_yaml.RoundTripDumper)
```

### **Converting your project**

To create a project that's compatible with Workbench, perform these steps:

1. Run the following command from an interactive shell:

```
anaconda-project init
```

2. Anaconda Enterprise 4 supports Linux only, so run the following command to remove the Windows and MacOS platforms from the project's anaconda-project.yml configuration file:

```
anaconda-project remove-platforms win-64 osx-64
```

3. Run the following command to verify the platforms were removed:

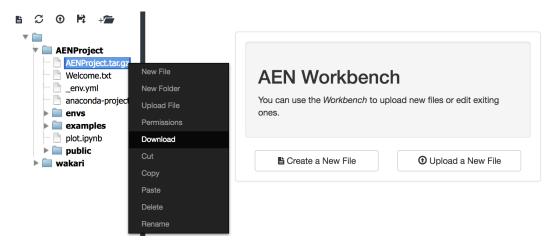
```
anaconda-project list-platforms
```

- 4. Add /.indexer.pid and .git to the .projectignore file.
- 5. Run the following command to compress your project:

```
# Replace <FILENAME> with a name for your project archive file anaconda-project archive <FILENAME>.tar.gz
```

**Caution:** Project names cannot contain spaces or special characters. There is a 1GB size limit for project archive files.

6. In Anaconda Enterprise 4 Notebooks, from your project home page, open the AEN Workbench. Locate your project file (e.g., AENProject.tar.gz in the image below) in the file list, right-click and select **Download**.



Now your project is ready to be uploaded into Workbench.

### Uploading your project to Workbench

Log in to Workbench and upload your project archive file <FILENAME>.tar.gz. See Working with projects for help.

**Note:** To maintain performance, there is a 1GB file size limit for project files you upload. Workbench projects are based on Git, so Anaconda recommends you commit only text-based files relevant to a project, and keep them under 100MB. Binary files are difficult for version control systems to manage, so Anaconda recommends using storage solutions designed for that type of data, and connecting to those data sources from within your sessions.

#### Migrating code

Anaconda Enterprise 4 and Workbench are based on a different architecture. This means that code inside your Anaconda Enterprise 4 notebooks might not run as expected in Workbench. Anaconda Enterprise 4 sessions ran directly on the host filesystem, where the libraries, drivers, packages, and connectors required to run them were available. Workbench sessions run in isolated containers with their own independent file system, so they don't necessarily have access to everything on the host.

This difference in architecture primarily impacts the following:

#### Connecting to external data sources

If you currently rely on ODBC/JDBC drivers to connect to specific databases such as Oracle and Impala, Anaconda recommends you use services that support this, such as Apache Impala and Apache Hive, instead. Additionally, using a language and platform agnostic connector such as Thrift allows you to create reproducible code that is more portable.

For best practices on how to connect to different external systems inside Workbench, see *Connecting to the Hadoop* and *Spark ecosystem*.

Service/System	Recommended
Apache Impala	impyla
Apache Hive	pyhive
Oracle	build conda package with their driver

If this is not possible, Anaconda recommends you obtain or *build conda packages* for the connectors and drivers you need. This enables you to *add them as package dependencies for your project* that will be installed when you start a Notebook session or deploy the project.

This has the added benefit of enabling you to update dependencies on connectors on a per-project basis.

#### Sharing custom Python and R libraries

It's quite common to share custom libraries by adding them to a location in the filesystem where all users can access the libraries they need. Workbench sessions and deployments run in isolated containers, so users cannot use this method to access shared libraries.

Instead, Anaconda recommends you *create a conda package for each library*. This enables you to control access to each package library and version—both essential to managing software at the enterprise level.

After you create the package, *upload it to the internal Workbench repository*, where it can be *shared with users* and *included as a dependency* in user sessions and deployments.

#### Installing external dependencies

If you typically install dependencies using system package managers such as apt and yum, you can continue to do so in Workbench. Dependencies installed from the command line are available during the current session only, however.

If you want them to persist across project sessions and deployments, add them as packages in the project's anaconda-project.yml configuration file. See *Configuring project settings* for more information.

If your project depends on package that is not available in your internal Workbench Repository, search anaconda.org or build your own conda package using conda-build, then *upload the conda package* to the Workbench repository.

If you don't have the expertise required to build the custom packages your organization needs, consider engaging our consulting team to make your mission-critical analytics libraries available as conda packages.

Data Science Al Workbench Documentation, Release 5.7.1								

## **USING WORKBENCH**

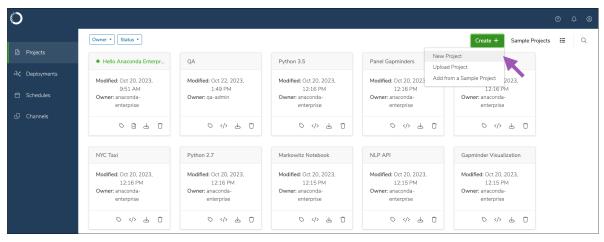
# 4.1 Projects

In Data Science & AI Workbench, a project is a structured environment for developing, managing, and deploying data science and machine learning applications. This includes relevant packages, channels, scripts, notebooks, files, environment variables, services and commands, and a *core configuration file* named anaconda-project.yml. For more information, see *Developing a project*.

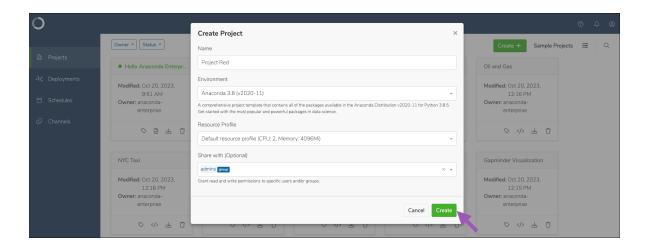
Project components are archived into a .tar.bz2, .tar.gz, or .zip file for portability purposes, allowing you to store the project and share it with others.

## 4.1.1 Creating a new project

1. From the projects page, click Create +, then select New Project.

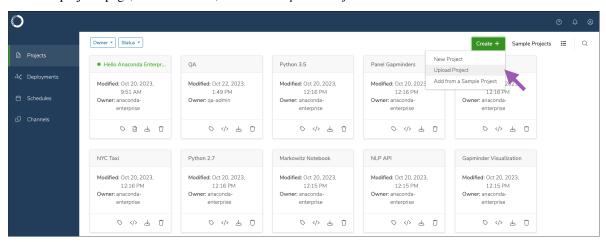


- 2. Enter a name for your project.
- 3. Select an environment for your project.
- 4. Select a resource profile for your project.
- 5. If necessary, add any users or groups as collaborators.
- 6. Click Create.

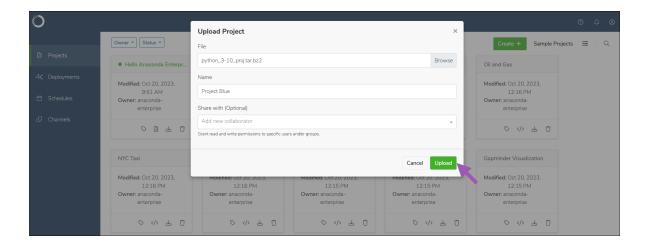


# 4.1.2 Uploading an existing project

1. From the projects page, click **Create +**, then select *Upload Project*.



- 2. Click **Browse** and upload your project archive file to Workbench.
- 3. Enter a name for your project.
- 4. If necessary, add any users or groups as collaborators.
- 5. Click Upload.



# 4.1.3 Using sample projects

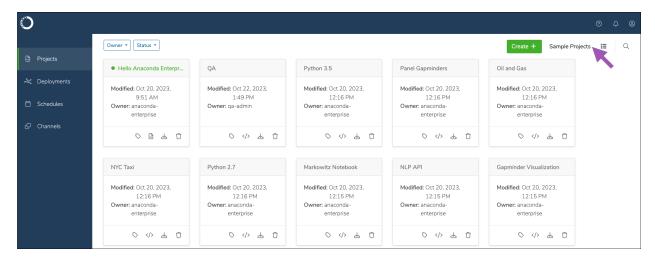
Anaconda provides you with a gallery of sample projects that include:

- Several base Anaconda Distribution for Python environments
- Minimal Python environments
- R notebooks and R Shiny apps
- Simplified Jupyter Notebooks for Matplotlib and HvPlot
- Dashboards for the Gapminder dataset, oil and gas, NYC taxi, and attractor equations
- TensorFlow apps for Flask, Tornado, and MINST trained data
- And tutorial projects for:
  - Using Intake to distribute and ingest data
  - Database access using SQLAlchemy, Pandas, and Dask
  - Visualizations with HoloViz.org tools
  - Building time-series forecasting models using statsmodels

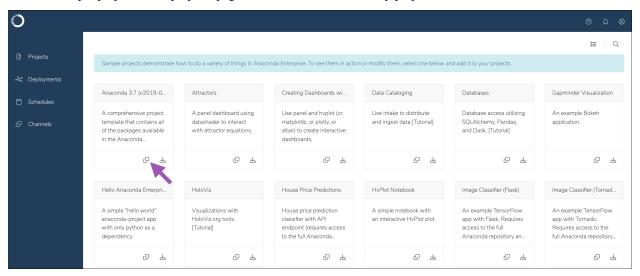
These projects are intended to aid you with your own projects by demonstrating how to do a variety of things in Workbench.

To access the sample project gallery, navigate to the **Projects** page, then click **Sample Projects**.

4.1. Projects 239



To add a sample project to the projects page, click the clone icon for any project.



## 4.1.4 Deleting a project

- 1. From the projects page, open the project you want to delete.
- 2. Click **Delete** at the bottom of the project settings page.
- 3. Click **Delete** again to confirm that you want to delete the project and end any active sessions and deployments from the project.

**Warning:** Deleting a project is irreversible. Project owners and administrators are the only users that can delete shared projects.

### Working with projects

To perform work within a Data Science & AI Workbench project, open it from the **Projects** page. The project's **Settings** page opens by default. Use the left-hand navigation from here to work with and manage different aspects of your project:

**Session** - Sessions provide an Integrated Development Environment (IDE) for project development. They enable you to write code, explore and visualize data, and develop and evaluate models in a collaborative environment, incorporating familiar Git-based branching and merging workflows for version management.

**Deployments** - Deployments make developed applications or models accessible for end-user interaction, or for automated tasks, such as a web service, REST API, or scheduled job. For more information, see *Deployments*.

**Schedules** - Setting a schedule for your project instructs it to perform specific tasks automatically at predetermined times. You can update data, generate a report, execute a pipeline, run backup operations, and more. For more information, see *Scheduling deployments*.

**Runs** - In Workbench, a run refers to the execution of a part of your project, similar to running a script or a program. When you initiate a run, Workbench allocates resources to execute the task. This could be any automated task you have established in your project.

**Share** - Sharing projects with collaborators in Workbench facilitates teamwork by ensuring that all team members have the ability to contribute and have access to the project's contents. Each project in Workbench has its own internal Git repository that is accessible by the project creator and collaborators that are added to the project. For more information, see *Sharing a project*.

**Note:** If your organization would prefer to use its own supported external version control repository, your administrator can configure Workbench to use that repository instead of the internal GitHub server. For more information, see *Connecting to an external version control repository*.

Once complete, you will be prompted for your personal access token before you create your first project in Workbench. For more information, see *Configuring access to version control*.

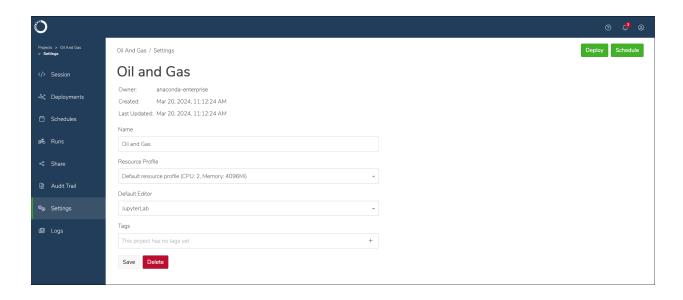
**Audit Trail** - The audit trail is like a project history timeline. It shows detailed records of all activities and changes made within a project. This includes user actions, version changes, access logs, deployment records, resource usage, and system notifications. The audit trail provides transparency, accountability, and is a useful resource when diagnosing issues within a project.

Settings - Project settings allow you to configure various aspects of your project. From this page, you can:

- Set your project's name.
- Select a resource profile. *This is a critical step for meeting computational demands of your project.* For more information, see *Understanding resource profiles*.
- Choose a default editor for your project. The default selection is *JupyterLab*.
- Assign tags to your project. For more information, see *Project tagging*.
- Delete the project. For more information, see *Deleting a project*.`

**Logs** - Project logs provide comprehensive insights into the operational health of the project's components. Each project runs an editor, sync, and proxy container, and logs for each container are accessible to aid in troubleshooting issues you may encounter with your project. For more information, see *Project logs*.

4.1. Projects 241



### Working offline

To work on a project offline, download an archived version (tar.gz) of the project from the projects page. When you are ready, you can upload the files you've modified back to the project in Workbench and commit your changes to make them available for other project collaborators. For more information, see *Saving and committing changes in a project*.

### **Branching and Merging**

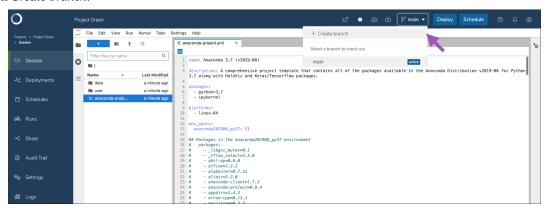
Each project in Data Science & AI Workbench has its own repository that the files for the project are stored in. You can create branches off of your project to develop changes locally, then commit and push them to a remote collaborative project repository, where other users will then be able to pull your changes.

#### Creating a branch

New branches are always created off of the currently active branch.

From the projects page:

- 1. Open your project and start a session.
- 2. Open the branch dropdown menu.
- 3. Click Create branch.



- 4. Provide a name for your branch.
- 5. Click Create.

Tip: The current active branch and its parent branch are always identified in the branch dropdown.

### Switching between branches

You can switch between branches that exist in your project using the branch dropdown menu.

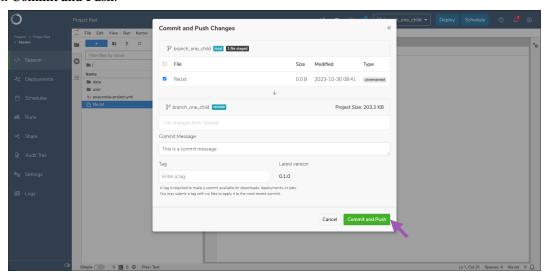
- 1. Open the branch dropdown menu.
- 2. Select the branch you want to work in.

**Caution:** Your working branch must be clean to switch between branches. *Commit and push* your work before switching branches!

**Note:** You may need to refresh your editor view for it to detect the most recent changes.

### Pushing changes to a branch

- 1. Click the **Commit and Push** icon.
- 2. Select the files you want to commit to the remote branch from your local branch.
- 3. Enter a commit message for your changes.
- 4. If necessary, tag a version for your commit.
- 5. Click Commit and Push.

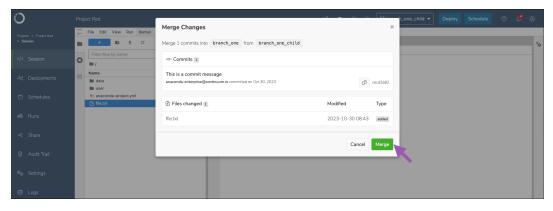


4.1. Projects 243

### Merging changes to a parent branch

When the changes on your branch are ready, you can merge them into the parent branch using the branch dropdown menu.

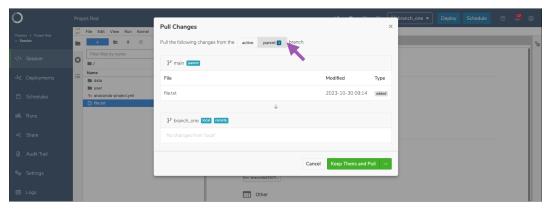
- 1. Open the branch dropdown menu.
- 2. Click Merge changes into parent branch.
- 3. Click Merge.



### Pulling changes from a parent or remote branch

If the parent or remote branch has changes applied to it that the active branch does not, you must pull the changes into your local branch to avoid merge conflicts before you push your commits.

- 1. Click the **Pull** icon.
- 2. Select the branch that contains changes you need to pull.



3. Open the **Pull** dropdown and select a pull method:

### **Keep Theirs and Pull**

Discards your local changes in favor of content on the parent branch. Your changes will be lost.

#### **Keep Mine and Pull**

Discards changes on the parent branch in favor of your local changes. Changes on the parent branch will be overwritten.

#### **Keep Both and Pull**

Saves the conflicting files with different file names so you can compare their content and decide how you want to reconcile the differences manually.

### **Deleting a branch**

There is currently no method for deleting a branch from the UI. This feature will be added in a later release. However, you can still delete branches from your project manually if necessary. To delete a branch from a project:

- 1. Open your project and start a session.
- 2. Open a terminal in your project session.
- 3. Run the following commands:

```
# Replace <BRANCH> with the name of the branch you want to delete
git branch -D <BRANCH>
git push origin :<BRANCH>
```

Caution: If you attempt to delete a branch that another user currently has active, the branch will not be deleted.

### **Project tagging**

Tag projects you own to include additional options for project filtering, allowing you to efficiently locate projects when you need them.

Note: Administrator users can manage tags for any project.

### Adding tags to a project

Projects can have up to four tags each. To add a tag to a project:

- 1. Click the Tag icon of any project to view its tags.
- 2. Click Manage Tags.
- 3. Open the **Tags** dropdown, then select existing tags and/or enter a new tag for your project.
- 4. Click Save Tags.

**Tip:** You can also add tags to a project from the project's settings page.

#### **Managing tags**

To manage project tags:

- 1. From the projects page, click the Tag icon of any project to view its tags.
- 2. Click Manage Tags.
- 3. Add or remove tags as needed.
- 4. Click Save Tags.

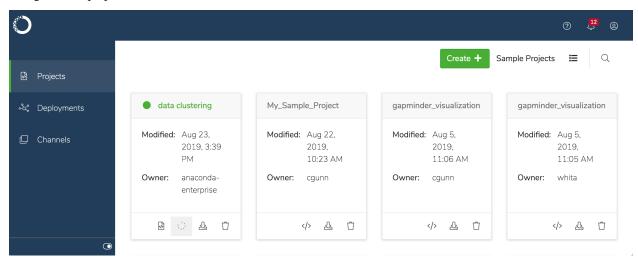
4.1. Projects 245

### Filtering tagged projects

Once you have added tags to your projects, you can use them to filter your projects and help you locate them efficiently. From the projects page, open the Tag filter dropdown, then select the tags you need. The project list will filter to display only projects that contain those tags.

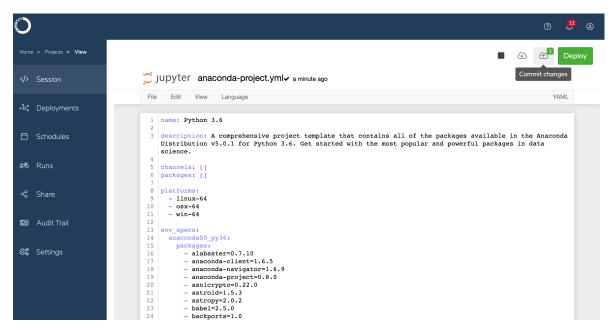
### **Editing a project**

After you have created your project so that it appears in your **Projects** list, you can open a project session to make changes to the project.



#### To edit a project:

- 1. Click the **Open session** icon </>
  to open the project in the editor specified the project.
- 2. Make your changes to the project, and save them locally. A badge is displayed on the **Commit Changes** icon to indicate that you've made changes that haven't been committed to the server.



- 3. When you're ready to update the repository with your changes, click the icon to *commit your changes*. If the project is shared, others will then be able to to access your changes, See *collaborating on projects* for important things to consider when working with others on shared projects.
- 4. When you're done working with the project, click the **Stop session** icon . The session is listed in the **Audit Trail** for the project.

You can also leave a project session open, and click the **Return to session** when you're ready to resume work.

**Tip:** See *Developing your project* to learn how to manage the dependencies for your project, so you can run it and deploy it.

### **Developing a project**

To enable Data Science & AI Workbench to manage the dependencies for your project—so you can run it and deploy it—you need to configure the following settings for each project you create or upload:

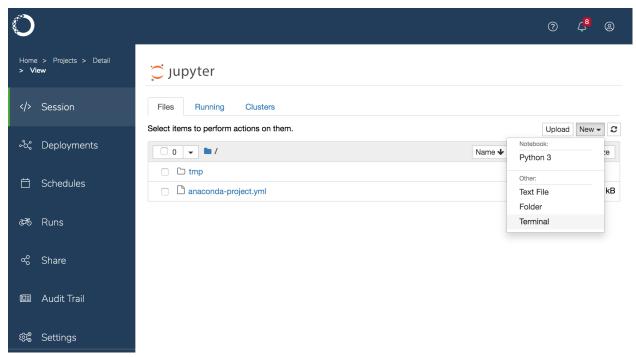
- *Include all the packages used by the project* (e.g., conda, pip, system).
- Create a custom Conda environment.
- *Specify the deployment command required to run the project.*
- Specify environment variables to use in editor sessions and deployments.

All dependencies are tracked in a project's anaconda-project.yml file. While there are various ways to modify this file—using the user interface or a command-line interface—any changes to a project's configuration will persist for future project sessions and deployments, regardless of the method you use.

**Note:** This is different than using conda install to add a package using the conda environment during a session, as that method impacts the project temporarily, *during the current session only*.

4.1. Projects 247

**Jupyter Notebook** supports anaconda-project commands only. You'll need to run these commands in a terminal. To open a terminal window within a Jupyter Notebook editor session, click **New**, then select **Terminal** in the dropdown:



### Adding packages to a project

Workbench offers several ways to add packages to a project, so you can choose the method you prefer:

• In a JupyterLab editing session, click the **Project** tab on the far left and click the Edit pencil icon in the **PACK-AGES** field. Add your packages and click **Save**.

-or-

In a terminal run anaconda-project add-packages followed by the package names and optionally the versions.

EXAMPLE: anaconda-project add-packages hvplot pandas=0.25

The command may take a moment to run as it collects the dependencies and downloads the packages. The packages will be visible in the project's anaconda-project.yml file. If this file is already open, close it and reopen it to see your changes.

To install packages from a specific channel:

EXAMPLE: anaconda-project add-packages -c conda-forge tranquilizer

**Caution:** anaconda-project commands must be run from the lab\_launch environment. This is the default environment when using the Jupyter Notebook terminal. For JupyterLab, it will be the first terminal on left. If your terminal prompt is not (lab\_launch), you can activate it with the command conda activate lab\_launch.

**Note:** The default channel\_alias for conda in Workbench is configured to point to *the internal package repository*, which means that short channel names will refer to channels in the internal package repository.

To use packages from an external or online package repository, you will need to specify the full channel URL such as anaconda-project add-packages bokeh -c https://conda.anaconda.org/pyviz in a command or in anaconda-project.yml. The channel\_alias can be *customized by an administrator*, which affects all sessions and deployments.

**If you are working in an air-gapped environment** (without internet access), your Administrator will need to *mirror* the packages into your organization's internal package repository for you to be able to access them.

## To install pip packages:

List the packages in the pip: section of anaconda-project.yml. For example:

After editing the anaconda-project.yml file to include the pip packages you want to install, run the anaconda-project prepare command to install the packages.

#### To install system packages:

In a terminal, run sudo yum install followed by the package name.

EXAMPLE: sudo yum install sqlite

**Note:** Any system packages you install from the command line are available *during the current session only*. If you want them to persist, add them to the project's anaconda-project.yml file. **The system package must be available in a Workbench channel** for it to be installed correctly via the anaconda-project.yml file.

## **Custom project environment**

**Note:** Each project only supports the use of a single environment.

For the *standard template projects* the Conda environments have been pre-built as a bootstrap to reduce initialization time when additional packages are added as described above. However, you may wish to create a custom environment specification.

You can use the terminal and command line to specify the environment for a project. For example, to create an environment called new\_env with notebook, pandas, and bokeh:

```
anaconda-project add-env-spec --name new_env anaconda-project add-packages --env-spec new_env notebook pandas=0.25 panel=0.6
```

Remove the original environment that corresponds to the template you chose when you initially created the project. For example, to remove the Python 3.6 environment:

4.1. Projects 249

anaconda-project remove-env-spec anaconda50\_py36

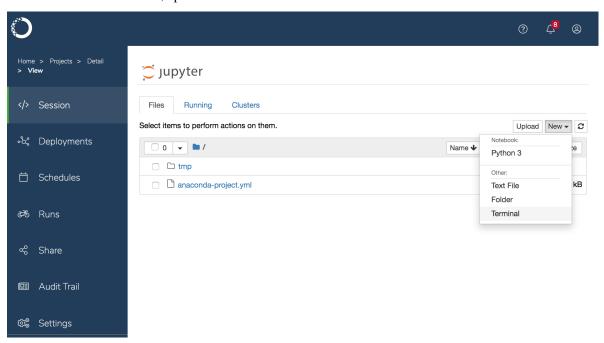
For your changes to take effect, you must commit all changes to the project, then stop and re-start the project.

**Note:** You must include the **notebook** package for the environment to edit and run notebooks in either the Jupyter Notebook or JupyterLab editors.

**Tip:** Using the anaconda-project command ensures that the environment will prepare correctly when the session is restarted. For more information about anaconda-project commands type anaconda-project --help.

#### To verify whether an environment has been initialized for a Notebook session:

1. Within the Notebook session, open a terminal window:



2. Run the following commands to list the contents of the parent directory:



```
(lab_launch) pwd
/opt/continuum/project
(lab_launch) cd ..
(lab_launch) ls
anaconda downloads prepare.log project
(lab_launch)
```

If the environment is being initialized, you'll see a file named preparing. When the environment has finished initializing, it will be replaced by a file named prepare.log.

**Tip:** If you need to troubleshoot session startup, you can use a terminal to view the session startup logs. When

session startup begins, the output of the anaconda-project prepare command is written to /opt/continuum/preparing, and when the command completes, the log is moved to /opt/continuum/prepare.log.

#### Adding deployment commands to a project

You can use Workbench to deploy projects containing notebooks, Bokeh applications, and generic scripts or web frameworks. Before you can *deploy a project*, it needs to have an appropriate deployment command associated with it.

Each of the following methods can be used to add a deployment command in the project's config file anaconda-project.yml:

• In a JupyterLab editing session, click the **Project** tab on the far left and click the plus sign to the right of the **COMMANDS** field. Add information about the command and click **Save**.

**Note:** This method is available within the JupyterLab editor only, so you'll need to set that as your default editor—in the project's **Settings**—and restart the project session to see this option in the user interface. The two methods described below do not show notifications in the user interface.

-or-

• Use the command line interface:

EXAMPLE: anaconda-project add-command --type notebook default data-science-notebook. ipynb

The following are example deployment commands you can use:

### For a Notebook:

```
commands:
    default:
    notebook: your-notebook.ipynb
```

For a project with a Bokeh (version 0.12) app defined in a main.py file:

```
commands:
    default:
        bokeh_app: .
        supports_http_options: True
```

For a Panel dashboard (panel must be installed in your project):

```
commands:
    default:
    unix: panel serve script-or-notebook-file
    supports_http_options: True
```

For a generic script or web framework, including Python or R:

```
commands:
    default:
    unix: bash run.sh
    supports_http_options: true
```

4.1. Projects 251

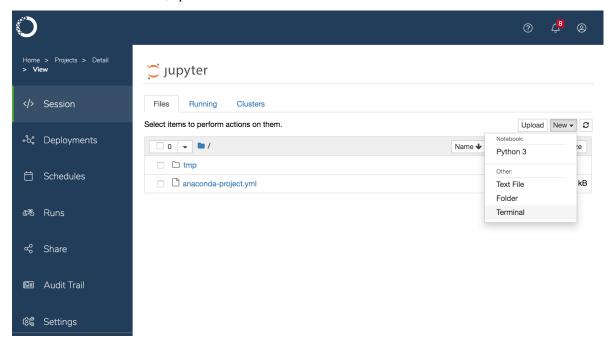
```
commands:
    default:
        unix: python your-script.py
        supports_http_options: true

commands:
    default:
        unix: Rscript your-script.R
        supports_http_options: true
```

**Note:** For deployment commands that can handle anaconda-project-- arguments (like Panel) supports\_http\_options: True must be added to the command.

To validate your anaconda-project.yml and verify your project will deploy successfully:

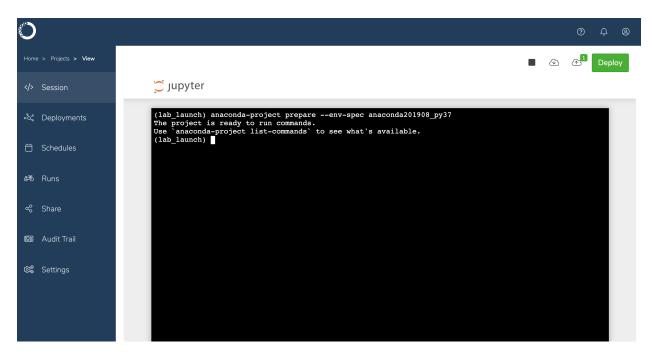
1. Within the Notebook session, open a terminal window:



2. Run the following command, replacing anaconda44\_py35 with the name of your environment, if it's different:

```
anaconda-project prepare --env-spec anaconda44_py35
```

If the environment includes everything needed to deploy the project, you'll see a message like the following:



Otherwise, any errors preventing a successful deployment will be identified.

If you want to test the deployment immediately after preparing the environment, run the following command instead:

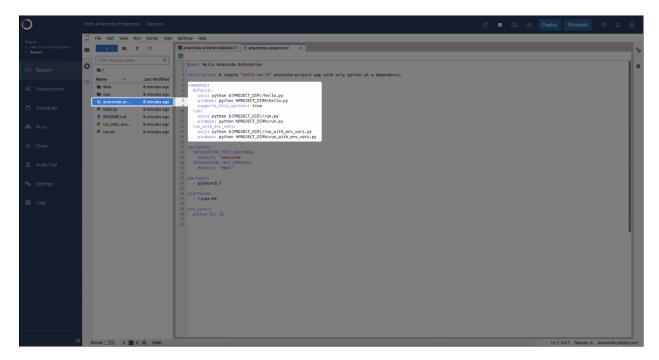
```
anaconda-project run <command-name>
```

If there are any errors preventing a successful deployment, they will be displayed in the terminal.

## **Testing project deployments**

Once deployment commands have been added to your project, you can test the deployment using the test\_deployment command. You can view the deployment commands for your project from the projects .yml configuration file.

4.1. Projects 253



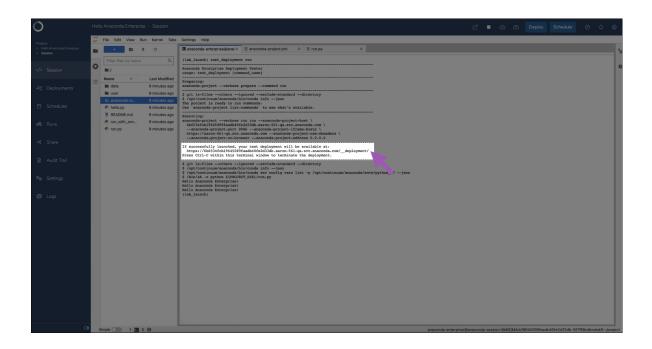
To test a project deployment, complete the following steps:

- 1. Open a project session.
- 2. Open a terminal.
- 3. Test a deployment command you've added to your project by running the following command:

```
# Replace <COMMAND> with an available deployment command
test_deployment <COMMAND>
```

**Note:** If you do not supply a deployment command to test, the first command listed under the commands: section of the projects .yml configuration file will be run.

4. Navigate to the web address returned by the command to verify your project deployed successfully.



#### **Environment variables**

You can add environment variables that will be set when you run notebooks in an editor session and at the start of a deployment command.

• In a JupyterLab editing session, click the **Project** tab on the far left and click the + button next to **VARIABLES**. Provide the name, description and default value of all variables you require.

-or-

 You can use the terminal and command line. For example, to add an environment variable that sets MY\_VAR to hello.:

```
anaconda-project add-variable --default hello MY_VAR
```

## Saving and committing changes in a project

Saving changes to files within an editor or project is different from committing those changes to the Data Science & AI Workbench server. For example, when you select **File > Save** within an editor, you *save your changes to your local copy of the file* to preserve the work you've done.

Caution: Project file names cannot contain unicode characters (special characters, punctuation, or symbols).

When you're ready to update the server with your changes, you *commit your changes to the project*. This also allows others to access your changes, if the project is shared. See *collaborating on projects* for important things to consider when working with others on shared projects.

**Caution:** If the total size of the files stored in your project's Git repository exceeds 1GB, your project will experience slowdowns or become unresponsive. Anaconda advises maintaining an individual file size of less than 50MB each. If you need to work with files that exceed this threshold, speak with your cluster administrator.

4.1. Projects 255

Furthermore, binary files are difficult for version control systems to manage, so Anaconda recommends using storage solutions designed for that type of data, and connecting to those data sources from within your Workbench sessions.

#### To commit your changes:



- 1. Click the Commit Changes icon.
- 2. Select the files you have modified and want to commit. If a file that you changed isn't displayed in this list, make sure you saved it locally.

**Note:** Editors create temporary files that may be displayed in the file list. For example, Jupyter Notebook and JupyterLab both create a hidden folder named .ipynb\_checkpoints for each notebook project you create. This folder is hidden because the editor uses it internally, to capture the state of your .ipynb file between auto-save operations. Anaconda recommends you add this and any other hidden folders to your .gitignore file, so they are excluded from the list of project files that are checked into version control.

- 3. Enter a message that briefly describes the changes you made to the files or project. This information is useful for differentiating your commit from others.
- 4. Enter a meaningful label or version number for your project commit in the **Tag** field. You can use tags to create multiple versions of a single project so that you—or collaborators—can easily deploy a specific version of the project. See *deploying a project* for more information.
- 5. Click Commit.

#### Collaborating on projects

Effective collaboration is key to the success of any enterprise-level project, so it's essential to understand how to work well with others in shared projects.

To give others access to a project that you've created, you can add them as a *collaborator*.

When you add a user (or group of users) as collaborators on a project, it means that they have permission to edit the project *files* and commit changes to the master copy on the server while you may be actively working on a local copy. The only project *setting* they'll be able to change is the default editor—all other project settings will be disabled for editing.

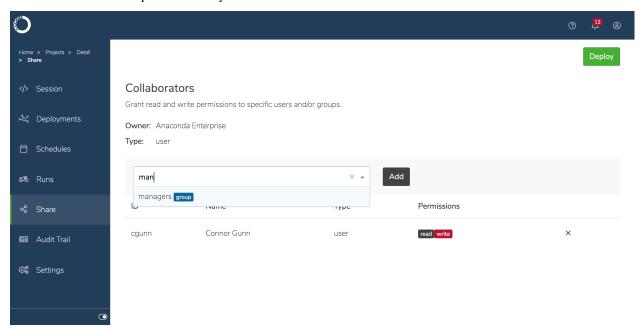
**Note:** Data Science & AI Workbench creates a repository for each project that you create, and will authorize only those users who have been explicitly added as project collaborators to update *the version control repository configured* for your organization with their changes to the project.

Workbench tracks all changes to a project and lets you know when files have been updated, so you can choose which version to use.

## Sharing a project

You can share a project with specific users or groups of users

- 1. Click **Projects** to view all of your projects.
- 2. Click the project you want to share and select **Share** in the left menu.
- 3. Start typing the name of the user or group in the Add New Collaborator drop-down to search for matches. Select the one that corresponds to what you want and click **Add**.



To unshare—or remove access to—a project, check the large X next to the collaborator you want to remove and click **Remove** to confirm your selection.

**Note:** If you remove a collaborator from a project *while they have a session open for that project*, they might see a 500 Internal Server Error message. To avoid this, ask them to close their running session before you remove them from the project.

Any collaborators you share your project with will see the project in their **Projects** list when they log in to Workbench, and if others share their projects with you, they'll appear in yours.

## Getting updates from other users

When a collaborator makes a change to the project, a badge will appear beside the **Fetch Changes** icon



Click this icon to pull changes from the server and update your local copy of the project with any changes made by other collaborators.

Workbench compares the copy of the project files you have locally with those on the server and notifies if any files have a conflict. If there is no file conflict, your local copies are updated.

Note: Fetching the latest changes may overwrite or delete your local copy of files without warning if a collaborator

4.1. Projects 257

has committed changes to the server and you have not made changes to the same files, as there is no file conflict.

#### **EXAMPLE:**

- Alice and Bob are both collaborators a project that includes file1.txt.
- Alice deletes file1.txt from her local copy of the project and commits her changes to the server.
- Bob pulls the latest changes from the server. Bob hasn't edited file1.txt, so file1.txt is deleted from Bob's local version of the project. Bob's local copy of the project and the version on the server now match exactly.

If the updates on the server conflict with changes you have made locally, you can choose one of the following options:

- · Cancel the Pull.
- Keep theirs and Pull—discards your local changes in favor of theirs. Your changes will be lost.
- Keep mine and Pull—discards changes on the server in favor of your local changes. Their changes will be overwritten.
- **Keep both and Pull**—saves the conflicting files with different filenames so you can compare the content of the files and decide how you want to reconcile the differences. See *resolving file conflicts* below for more information.

**Note:** If you have a file open that has been modified by fetching changes, close and reopen the file for the changes to be reflected. Otherwise, the next time you save the file, you may see a "File has been overwritten on disk" alert in JupyterLab. This alert lets you choose whether to cancel the save, discard the current version and open the version of the file on disk, or overwrite the file on disk with the current version.

## **Committing your changes**

After you have saved your changes locally, click the **Commit Changes** icon to update the master copy on the server with your changes.

If your changes conflict with updates made by other collaborators, a list of the files impacted will be highlighted in red. You may choose how you want to proceed from the following options:

- · Cancel the Commit.
- **Proceed with the Commit**—overwrites your collaborators' changes. Proceed with caution when choosing this option. Collaborators may not appreciate having their work overwritten, and important work may be lost in the process.
- Selectively Commit—commit only those files which don't have conflicts by unchecking the ones highlighted in red

Committing changes to the server involves a full sync, so any changes that have been made to the project on the server—that do not conflict with your changes—are pulled in the process. This means that after committing your changes, your local copy will match the master copy on the server.

## Resolving file conflicts

File conflicts result whenever you have updated a file locally, while a collaborator has changed that same file in their copy of the project and committed their changes to the master copy on the server.

In these cases, you may want to select **Keep both and Pull** to save the conflicting files with different filenames. This enables you to compare the content of the files and decide the best approach to take to reconcile the differences. *The solution will likely involve manually editing the file* to combine both sets of changes and then committing the file.

EXAMPLE: If a file is named Some Data.txt and Alice has committed updates to that file on the server, your new local copy of the file from the server—containing Alice's changes—will be named Some Data.txt (Alice's conflicted file). Your local copy named Some Data.txt will not change.

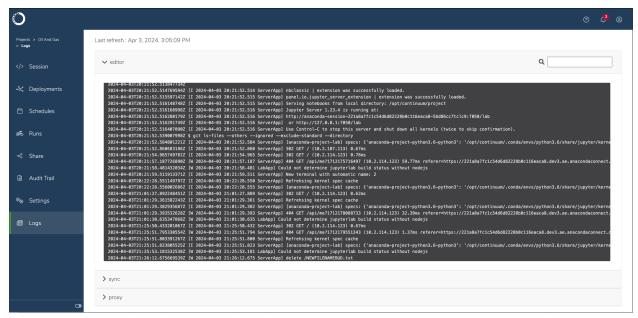
#### **Project logs**

Because Data Science & AI Workbench is built on top of a Kubernetes cluster, each project session is created as a pod. Each project session pod has an editor, sync, and proxy container, and logs for each container are accessible to aid you in troubleshooting issues you may encounter with your project. These logs can provide insights to the operational health of your project, possible configuration errors, and potential security issues.

## Viewing project logs

To view a project session's logs:

- 1. Log in to Workbench.
- 2. Open a project.
- 3. If necessary, start a session in the project.
- 4. Select **Logs** from the left-hand navigation.
- 5. Open a container dropdown to view the logs for that container.



4.1. Projects 259

**Tip:** Use the search function to help you efficiently locate information in the container logs.

#### **Editor container logs**

The editor container is where the integrated development environment (IDE) set by the project configurations runs. Logs from this container can provide insights into the following:

- User actions performed within the editor, such as file edits, saves, and execution of code cells
- Errors or warnings related to the editor software, as opposed to Workbench software
- Problems that are caused by editor plug-ins or extensions

## Sync container logs

The sync container is responsible for synchronizing project files and code between persistent storage and the editor. Logs from this container can provide insights into the following:

- · Successes or failures when syncing files between the local user repository and the projects remote repository
- Issues encountered during file creation or deletion, or file modification attempts
- Problems accessing external resources the project relies on

## **Proxy container logs**

The proxy container handles network traffic to and from the project's environment, and facilitates access to web interfaces and APIs. Logs from this container can provide insights into the following:

- Incoming and outgoing network requests, which is useful for identifying unauthorized access attempts
- Errors related to network connectivity, DNS issues, or failed attempts to connect to external services
- Information on request handling times and bottlenecks in network communication

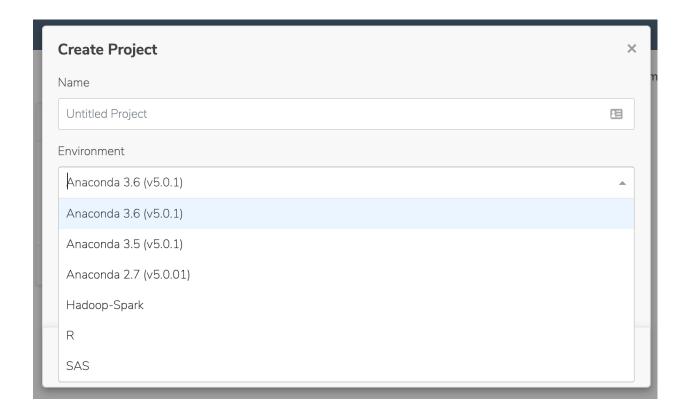
# 4.2 Using project templates

In addition to sample projects, Data Science & AI Workbench provides project *templates* to help you get started with configuring and developing in your project. Project templates provide pre-built Conda environments in your project editor session where a number of packages have already been installed.

Templates are provided for common environments such as Python, R, and Spark and Hadoop.

Each template environment includes many of the post popular data science packages You can use anaconda-project commands to *customize them as needed*.

To use one of the available templates, simply select it from the **Environment** list when you create your project:



## 4.2.1 Python templates

Data Science & AI Workbench templates environments are provided Python versions 2.7, 3.5, and 3.6.

In a running project session the Python 2.7 environment includes all of the packages in the Anaconda distribution for Python 2.7 with a check mark in the **In Installer** column. The same is true for the Python 3.5 and Python 3.6 environments.

Additional Conda and pip packages can be added using the process described in *Developing a project*.

For example, to upgrade to a newer version of Pandas and add the HvPlot package, run the following in a terminal

```
anaconda-project add-packages pandas=0.25 hvplot
```

Python notebooks can be edited with any of the editors provided with Workbench: Jupyter Notebooks, JupyterLab, or Apache Zeppelin. To change the default editor for your Python project, click on it or select **View details** from its menu in the list view. Then click **Settings** to select your preferred editor. For more information, see *Projects*.

## 4.2.2 R templates

Data Science & AI Workbench provides users with the R Essentials bundle. The bundle contains around 80 of the most commonly-used packages for data science, including r-base version 3.4.2, caret, dplyr, ggplot2, glmnet, irkernel, rbokeh, shiny, tidyverse, and many more packages!

If you need packages not included in the R Essentials bundle, you can *add packages to your projects as described here*. You'll need to connect to a repository containing the packages you require. If you are unable to connect to a repository containing the packages you require, contact your Administrator and request that they *mirror the packages to a channel you can access*.

You can edit R notebooks with any of the editors provided by Workbench: Jupyter Notebooks, JupyterLab, or Apache Zeppelin. To edit notebooks using RStudio, see *adding RStudio to Workbench*.

## 4.2.3 Hadoop / Spark

If your Data Science & AI Workbench Administrator has configured Livy server for Hadoop and Spark access, you'll be able to access them within the platform.

The Hadoop/Spark project template includes sample code to connect to the following resources, with and without *Kerberos authentication*:

- Spark
- Hadoop Distributed File System (HDFS)
- Hive
- Impala

In the editor session there are two environments created. anaconda50\_hadoop contains the packages consistent with the Python 3.6 template plus additional packages to access Hadoop and Spark resources. The anaconda50\_impyla environment contains packages consistent with the Python 2.7 template plus additional packages to access Impala tables using the Impyla Python package.

### **Using Kerberos authentication**

If the Hadoop cluster is configured to use Kerberos authentication—and your Administrator has configured Workbench to work with Kerberos—you can use it to authenticate yourself and gain access to system resources. The process is the same for all services and languages: Spark, HDFS, Hive, and Impala.

**Note:** You'll need to contact your Administrator to get your Kerberos *principal*, which is the combination of your username and security domain.

To perform the authentication, open an environment-based terminal in the interface. This is normally in the Launchers panel, in the bottom row of icons, and is the right-most icon.

When the interface appears, run this command:

kinit myname@mydomain.com

Replace myname@mydomain.com with the Kerberos *principal*, the combination of your username and security domain, which was provided to you by your Administrator.

Executing the command requires you to enter a password. If there is no error message, authentication has succeeded. You can verify by issuing the klist command. If it responds with some entries, you are authenticated.

You can also use a keytab to do this. Upload it to a project and execute a command like this:

kinit myname@mydomain.com -kt mykeytab.keytab

**Note:** Kerberos authentication will lapse after some time, requiring you to repeat the above process. The length of time is determined by your cluster security administration, and on many clusters is set to 24 hours.

For deployments that require Kerberos authentication, Anaconda recommends generating a shared Kerberos keytab that has access to the resources needed by the deployment, and adding a kinit command that uses the keytab as part of the deployment command.

Alternatively, the deployment can include a form that asks for user credentials and executes the kinit command.

#### **Using Spark**

Apache Spark is an open source analytics engine that runs on compute clusters to provide in-memory operations, data parallelism, fault tolerance, and very high performance. Spark is a general purpose engine and highly effective for many uses, including ETL, batch, streaming, real-time, big data, data science, and machine learning workloads.

**Note:** Using Workbench with Spark requires Livy and Sparkmagic. The Hadoop/Spark project template includes Sparkmagic, but your Administrator must have configured Workbench to work with a Livy server.

## **Supported versions**

The following combinations of the multiple tools are supported:

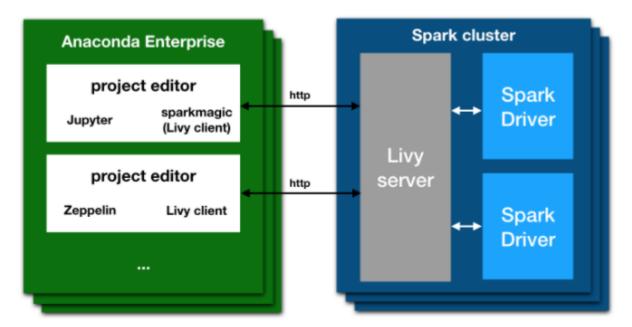
- Python 2 and 3
- Apache Livy 0.7.1-incubating
- Apache Spark 2.4.7

#### Livy

Apache Livy is an open source REST interface to submit and manage jobs on a Spark cluster, including code written in Java, Scala, Python, and R. These jobs are managed in Spark contexts, and the Spark contexts are controlled by a resource manager such as Apache Hadoop YARN. This provides fault tolerance and high reliability as multiple users interact with a Spark cluster concurrently.

With Workbench, you can connect to a remote Spark cluster using Apache Livy with any of the available clients, including Jupyter notebooks with Sparkmagic. Workbench provides Sparkmagic, which includes Spark, PySpark, and SparkR notebook kernels for deployment.

The Apache Livy architecture gives you the ability to submit jobs from any remote machine or analytics cluster, even where a Spark client is not available. It removes the requirement to install Jupyter and Anaconda directly on an edge node in the Spark cluster.



Livy and Sparkmagic work as a REST server and client that:

- · Retains the interactivity and multi-language support of Spark
- Does not require any code changes to existing Spark jobs
- · Maintains all of Spark's features such as the sharing of cached RDDs and Spark Dataframes, and
- Provides an easy way of creating a secure connection to a Kerberized Spark cluster.

When Livy is installed, you can connect to a remote Spark cluster when creating a new project by selecting the Spark template.

## **Kernels**

When you copy the project template "Hadoop/Spark" and open a Jupyter editing session, you will see several kernels such as these available:

- Python 3
- PySpark
- PySpark3
- R
- Spark
- SparkR
- Python 2

To work with Livy and Python, use PySpark. Do not use PySpark3.

To work with Livy and Scala, use Spark.

You can use Spark with Workbench in two ways:

1. Starting a notebook with one of the Spark kernels, in which case all code will be executed on the cluster and not locally.

Note that a connection and all cluster resources will be assigned as soon as you execute any ordinary code cell, that is, any cell not marked as <code>%%local</code>.

2. Starting a normal notebook with a Python kernel, and using %load\_ext sparkmagic.magics. That command will enable a set of functions to run code on the cluster. See examples (external link).

To display graphical output directly from the cluster, you must use SQL commands. This is also the only way to have results passed back to your local Python kernel, so that you can do further manipulation on it with pandas or other packages.

In the common case, the configuration provided for you in the Session will be correct and not require modification. However, in other cases you may need to use sandbox or ad-hoc environments that require the modifications described below.

## Overriding session settings

Certain jobs may require more cores or memory, or custom environment variables such as Python worker settings. The configuration passed to Livy is generally defined in the file ~/.sparkmagic/conf.json.

You may inspect this file, particularly the section "session\_configs", or you may refer to the example file in the spark directory, sparkmagic\_conf.example.json. Note that the example file has not been tailored to your specific cluster.

In a Sparkmagic kernel such as PySpark, SparkR, or similar, you can change the configuration with the magic %%configure. This syntax is pure JSON, and the values are passed directly to the driver application.

#### **EXAMPLE:**

```
%%configure -f
{"executorMemory": "4G", "executorCores":4}
```

To use a different environment, use the Spark configuration to set spark.driver.python and spark.executor.python on all compute nodes in your Spark cluster.

#### **EXAMPLE:**

If all nodes in your Spark cluster have Python 2 deployed at /opt/anaconda2 and Python 3 deployed at /opt/anaconda3, then you can select Python 2 on all execution nodes with this code:

If all nodes in your Spark cluster have Python 2 deployed at /opt/anaconda2 and Python 3 deployed at /opt/anaconda3, then you can select Python 3 on all execution nodes with this code:

If you are using a Python kernel and have done %load\_ext sparkmagic.magics, you can use the %manage\_spark command to set configuration options. The session options are in the "Create Session" pane under "Properties".

Overriding session settings can be used to target multiple Python and R interpreters, including Python and R interpreters coming from different Anaconda parcels.

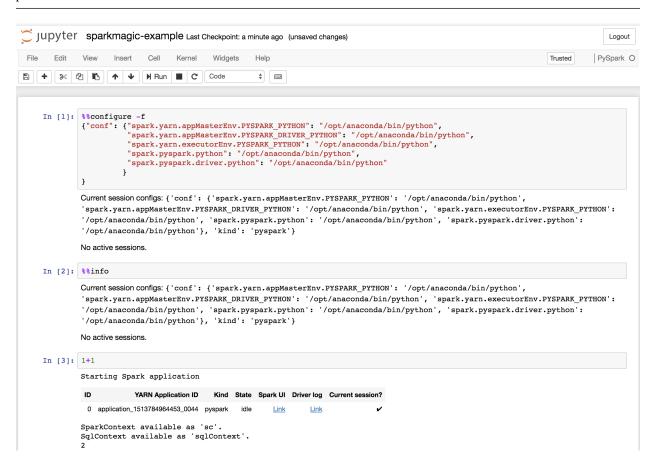
## Using custom Anaconda parcels and management packs

Workbench Administrators can generate custom parcels for CDP or custom management packs for Hortonworks Data Platform (HDP) to distribute customized versions of Anaconda across a Hadoop/Spark cluster using Cloudera Manager for CDP or Apache Ambari for HDP. See *Using installers, parcels and management packs* for more information.

As a platform user, you can then select a specific version of Anaconda and Python on a per-project basis by including the following configuration in the first cell of a Sparkmagic-based Jupyter Notebook.

For example:

**Note:** Replace /opt/anaconda/ with the prefix of the name and location for the particular parcel or management pack.



## Overriding basic settings

In some more experimental situations, you may want to change the Kerberos or Livy connection settings. This could be done when first configuring the platform for a cluster, usually by an administrator with intimate knowledge of the cluster's security model.

Users could override basic settings if their administrators have not configured Livy, or to connect to a cluster other than the default cluster.

In these cases, Anaconda recommends creating a krb5.conf file and a sparkmagic\_conf.json file in the project directory so they will be saved along with the project itself. An example Sparkmagic configuration is included, sparkmagic\_conf.example.json, listing the fields that are typically set. The "url" and "auth" keys in each of the kernel sections are especially important.

The krb5.conf file is normally copied from the Hadoop cluster, rather than written manually, and may refer to additional configuration or certificate files. These files must all be uploaded using the interface.

To use these alternate configuration files, set the KRB5\_CONFIG variable default to point to the full path of krb5. conf and set the values of SPARKMAGIC\_CONF\_DIR and SPARKMAGIC\_CONF\_FILE to point to the Sparkmagic config file. You can set these either by using the Project pane on the left of the interface, or by directly editing the anaconda-project.yml file.

For example, the final file's variables section may look like this:

```
variables:
   KRB5_CONFIG:
    description: Location of config file for kerberos authentication
    default: /opt/continuum/project/krb5.conf

SPARKMAGIC_CONF_DIR:
    description: Location of sparkmagic configuration file
    default: /opt/continuum/project

SPARKMAGIC_CONF_FILE:
    description: Name of sparkmagic configuration file
    default: sparkmagic_conf.json
```

Note: You must perform these actions before running kinit or starting any notebook/kernel.

**Caution:** If you misconfigure a .json file, all Sparkmagic kernels will fail to launch. You can test your Sparkmagic configuration by running the following Python command in an interactive shell: python -m json.tool sparkmagic\_conf.json.

If you have formatted the JSON correctly, this command will run without error. Additional edits may be required, depending on your Livy settings. See *Apache Livy and Workbench* and *Configuring Livy server for Hadoop Spark access* for information on installing and configuring Livy.

## **Python**

Example code showing Python with a Spark kernel:

```
sc
data = sc.parallelize(range(1, 100))
data.mean()
import pandas as pd
df = pd.DataFrame([("foo", 1), ("bar", 2)], columns=("col1", "col2"))
sparkdf = sqlContext.createDataFrame(df)
sparkdf.select("col1").show()
sparkdf.filter(sparkdf['col2'] == 2).show()
```

## **Using HDFS**

The Hadoop Distributed File System (HDFS) is an open source, distributed, scalable, and fault tolerant Java based file system for storing large volumes of data on the disks of many computers. It works with batch, interactive, and real-time workloads.

## **Dependencies**

• python-hdfs

## **Supported versions**

- Python 2 or 3
- Hadoop 3.1.1

## **Kernels**

• [anaconda50\_hadoop] Python 3

## Connecting

To connect to an HDFS cluster you need the address and port to the HDFS Namenode, normally port 50070.

To use the hdfscli command line, configure the ~/.hdfscli.cfg file:

```
[global]
default.alias = dev

[dev.alias]
url = http://<Namenode>:port
```

Once the library is configured, you can use it to perform actions on HDFS with the command line by starting a terminal based on the [anaconda50\_hadoop] Python 3 environment and executing the hdfscli command. For example:

```
$ hdfscli
Welcome to the interactive HDFS python shell.
The HDFS client is available as `CLIENT`.

In [1]: CLIENT.list("/")
Out[1]: ['hbase', 'solr', 'tmp', 'user']
```

## **Python**

Sample code showing Python with HDFS without Kerberos:

```
from hdfs import InsecureClient

client = InsecureClient('http://<Namenode>:50070')
client.list("/")
```

Python with HDFS with Kerberos:

```
from hdfs.ext.kerberos import KerberosClient

client = KerberosClient('http://<Namenode>:50070')
client.list("/")
```

## **Using Hive**

Hive is an open source data warehouse project for queries and data analysis. It provides an SQL-like interface called HiveQL to access distributed data stored in various databases and file systems.

Hive is very flexible in its connection methods and there are multiple ways to connect to it, such ODBC and Thrift.

## **Dependencies**

• pyhive

## **Supported versions**

- Python 2 or 3
- Hive 3.1.3000

#### **Kernels**

• [anaconda50\_hadoop] Python 3

#### **Drivers**

#### **ODBC**

Using ODBC requires downloading a driver for the specific version of Hive that you are using. This driver is also specific to the vendor you are using.

Cloudare EXAMPLE:

- Hive ODBC Connector 2.5.4 Download
- Hive ODBC Connection 2.5.4 Documentation

We recommend downloading the respective ODBC drivers to either your managed persistence directory, or to an NFS directory mounted in the session.

#### Connecting

To connect to a Hive cluster you need the address and port to a running Hive Server 2, normally port 10000.

To use PyHive, open a Python notebook based on the [anaconda50\_hadoop] Python 3 environment and run:

```
from pyhive import hive
conn = hive.connect('<Hive Server 2>', port=10000)
cursor = conn.cursor()
cursor.execute('SHOW DATABASES')
cursor.fetchall()
```

## **Python**

Anaconda recommends the Thrift method to connect to Hive from Python. With Thrift you can use all the functionality of Hive, including security features such as SSL connectivity and Kerberos authentication. Thrift does not require special drivers, which improves code portability.

Instead of using an ODBC driver for connecting to the SQL engines, a Thrift client uses its own protocol based on a service definition to communicate with a Thrift server. This definition can be used to generate libraries in any language, including Python.

Hive using PyHive:

Hive using ODBC with Kerberos:

```
cnxnstr = 'Driver={/opt/cloudera/hiveodbc/lib/64/libclouderahiveodbc64.so};
HIVESERVERTYPE=2;
HOST=<HOSTNAME>;
PORT=<PORT>;
UID=1002;
PWD=password;
Schema=<SCHEMA>;
AuthMECH=4;
HTTPPath=http://<FQDN>;
AllowSelfSignedServerCert=1;
CAIssuedCertNamesMismatch=1'
```

**Note:** The output will be different, depending on the tables available on the cluster.

#### **Using Impala**

Apache Impala is an open source, native analytic SQL query engine for Apache Hadoop. It uses massively parallel processing (MPP) for high performance, and works with commonly used big data formats such as Apache Parquet.

Impala is very flexible in its connection methods and there are multiple ways to connect to it, such as ODBC and Thrift.

## **Dependencies**

- impyla
- implyr

## **Supported versions**

- Python 2 or 3
- Impala 3.4.0

## Connecting

To connect to an Impala cluster you need the address and port to a running Impala Daemon, normally port 21050.

To use Impyla, open a Python Notebook based on the Python 2 environment and run:

```
from impala.dbapi import connect
conn = connect('<Impala Daemon>', port=21050)
cursor = conn.cursor()
cursor.execute('SHOW DATABASES')
cursor.fetchall()
```

## **Python**

Anaconda recommends the Thrift method to connect to Impala from Python. With Thrift you can use all the functionality of Impala, including security features such as SSL connectivity and Kerberos authentication. Thrift does not require special drivers, which improves code portability.

Instead of using an ODBC driver for connecting to the SQL engines, a Thrift client uses its own protocol based on a service definition to communicate with a Thrift server. This definition can be used to generate libraries in any language, including Python.

Impala using Impyla:

**Note:** The output will be different, depending on the tables available on the cluster.

# 4.3 Working with packages

Data Science & AI Workbench uses *packages* to bundle software files and information about the software—such as its name, specific version and description—into a single file that can be easily installed and managed.

Packages are distributed via *channels*. Channels may point to a cloud-based repository or a private location on a remote or local repository that you or someone else in your organization created. For more information, see *Configuring channels and packages*.

**Note:** Workbench supports the use of both conda and pip packages in its repository. To create and share channels and packages from your Workbench Repository using conda commands, first *install anaconda-enterprise-cli* and log in to your Workbench instance.

Creating a package requires familiarity with the conda package manager and command line interface (CLI), so not all Workbench users will create packages and channels.

Many Workbench users may interact with packages primarily within the context of *projects* and *deployments*. In this case, they will likely do the following:

- Access and download any packages and installers they need from the list of those available under Channels.
- Work with the contents of the package as they create models and dashboards, then
- Add any packages the project depends on to the project before deploying it.

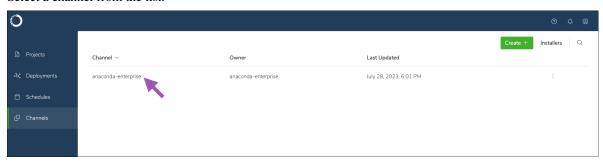
Other users may primarily build packages, upload them to channels and share them with others to access and download.

## 4.3.1 Viewing package information

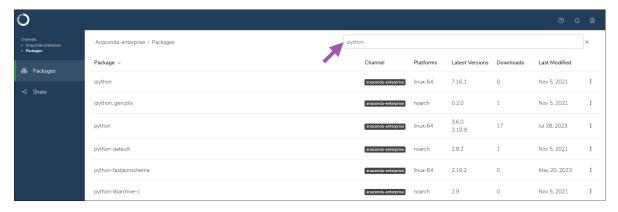
Each package in Data Science & AI Workbench presents valuable information pertaining to the package, such as its platform architecture, version, license, number of downloads, and the last time the package was updated. You can also find a command on this page to assist you with installing the package in your environment.

To view a package's information:

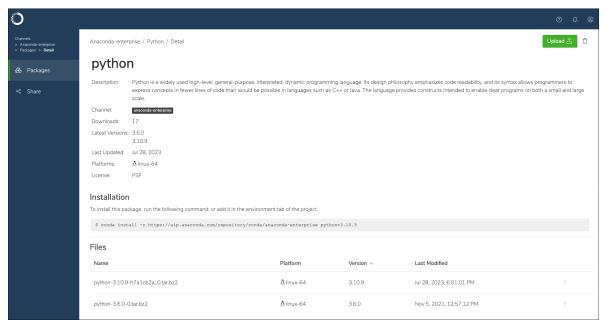
- 1. Log in to Workbench.
- 2. Select **Channels** from the left-hand navigation.
- 3. Select a channel from the list.



4. Click the search icon, then enter the name of the package into the search bar.



5. Select the package from the list to view its information.



# 4.3.2 Building a conda package

You can build a conda package to bundle software files and information about the software—such as its name, specific version and description—into a single file that can be easily installed and managed.

Building a conda package requires installing conda build and creating a conda build recipe. You then use the conda build command to build the conda package from the conda recipe.

Tip: If you are new to building packages with conda, here are some video tutorials that you may find helpful:

## • Production-grade Packaging with Anaconda | AnacondaCon 2018

This 41-minute presentation by Mahmoud Hashemi covers using conda and conda environments to build an OS package (RPM) and Docker images.

## • The Sheer Joy of Packaging | SciPy 2018 Tutorial

This 210-minute presentation by Michael Sarahan, Filipe Fernandes, Chris Barker, Matt Craig, Matt McCormick, Jean-Christophe Fillion-Robin, Jonathan Helmus, and Ray Donnelly provides end-to-end examples of packaging with PyPI and conda. You can find materials from the tutorial here.

#### • Making packages and packaging "just work" | PyData 2017 Tutorial

This 40-minute presentation by Michael Sarahan walks you through critical topics such as the anatomy of a Python package, tools available to make packaging easier, plus how to automate builds and why you might want to do so.

You can build conda packages from a variety of source code projects, most notably Python. For help packaging a Python project, see the Setuptools documentation.

**Note:** Setuptools is a package development process library designed to facilitate packaging Python projects, and is not part of Anaconda, Inc. Conda-build uses the build system that's native to the language, so in the case of Python that's setuptools.

After you build the package, you can *upload it to a channel* for others to access.

## 4.3.3 Uploading a conda package

After you build a conda package, you can upload it to a channel to make it available for others to use.

A channel is a specific location for storing packages, and may point to a cloud-based repository or a private location on a remote or local repository that you or your organization created. See *Accessing remote package repositories* for more information.

**Note:** There is a 1GB file size limit for package files you upload.

## To add a package to an existing channel:

- 1. Click **Channels** in the top menu to display your existing channels.
- 2. Select the specific channel you want to add your package to—information about any packages already in the channel is displayed.
- 3. Click **Upload**, browse for the package and click **Upload**. The package is added to the list.

Now you can share the channel and packages with others.

#### To create a new channel to add packages to:

1. Click Create in the upper right corner, enter a meaningful name for the channel and click Create.

**Note:** Channels are Public—accessible by non-authenticated users—by default. To make the channel Private, and therefore available to authenticated users only, disable the toggle to switch the channel setting from Public to Private.

2. Click Upload to add your package(s) to the channel.

#### **Using the CLI:**

You can also create a channel by running the following in a terminal window:

anaconda-enterprise-cli channels create <channelname>

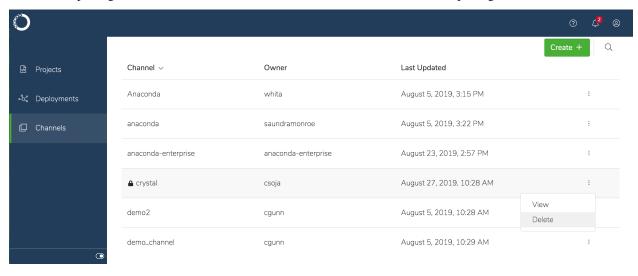
**Note:** The channel name < channel name > you enter must not already exist.

Now you can upload a package to the channel by entering the following:

anaconda-enterprise-cli upload path/to/pkgs/notebookname.tar.bz2 --channel <channelname>

Replacing path/to/pkgs/notebookname.tar.bz2 with the actual path to the package you want to upload, and <channelname> with the actual channel name.

To remove a package from a channel, select **Delete** from the command menu for the package:



Note: If the Delete command is not available, you don't have permission to remove the package from the channel.

### Setting a default channel

There is no default\_channel in a fresh install, so you'll have to enter a specific channel each time.

If you don't want to enter the --channel option with each command, you can set a default channel:

anaconda-enterprise-cli config set default\_channel <channelname>

To display your current default channel:

\$ anaconda-enterprise-cli config get default\_channel
'<channelname>'

After setting the default channel, upload to your default channel:

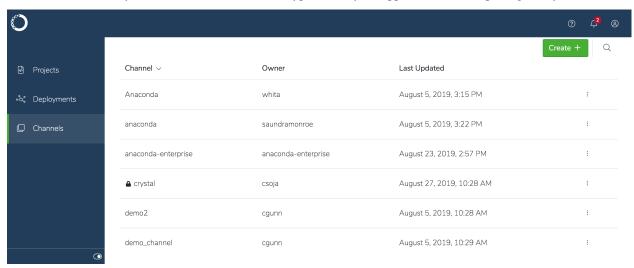
anaconda-enterprise-cli upload <path/to/pkgs/packagename.tar.bz2>

Replacing path/to/pkgs/packagename.tar.bz2> with the actual path to the package you want to upload.

## 4.3.4 Sharing channels and packages

After you build a package and upload it to a channel, you can enable others to access it by sharing the channel with them. You can share a channel with specific users, or groups of users.

To share multiple packages with the same set of users, you can upload all of the packages to a channel and share that channel. This enables you to create channels for each type of user you support, and add the packages they need to each.

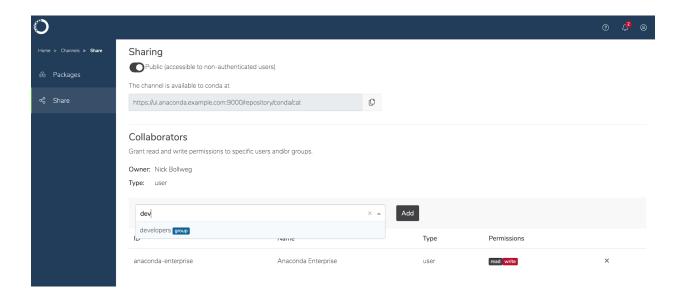


Anyone you share the channel with will see it in their **Channels** list when they log in to Data Science & AI Workbench. They can then download the packages in the channel they want to work with, and *add any packages their project depends on* to their project before *deploying it*.

**Note:** The default is to grant collaborators *read-write* access, so if you want to prevent them from adding and removing packages from the channel, be sure they have *read-only* access. You'll need to *use the CLI* to make a channel read-only.

#### To share a channel with unauthenticated users:

- 1. Select the channel in the **Channels** list, and verify that the packages in the channel are all appropriate to share.
- 2. Click **Share** in the left menu.
- 3. Ensure the channel is set to Public, copy the URL location of the channel, and distribute it to the people with whom you want to share the channel.

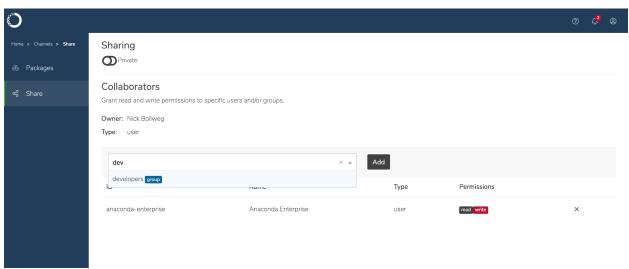


## To share a channel with other platform users:

- 1. Select the channel in the **Channels** list and verify that all the packages you want to share are listed.
- 2. Click **Share** in the left menu.

**Note:** Channels are Public—accessible by non-authenticated users—by default. To make the channel Private, and therefore available to authenticated users only, disable the toggle to switch the channel setting from Public to Private.

- 3. Start typing the name of the user or group in the **Add New Collaborator** drop-down to search for matches. Select the option that corresponds to what you want. You can add multiple users or groups at the same time.
- 4. Click **Add** when you're satisfied with your selections.



To "unshare" a channel with a collaborator, simply click the large X next to the right of their name in the **Collaborator** list.

## **Using the CLI:**

Get a list of all the channels on the platform with the channels list command:

```
anaconda-enterprise-cli channels list
```

Share a channel with a specific user using the share command:

```
anaconda-enterprise-cli channels share --user username --level r <channelname>
```

You can also share a channel with an existing group created by your Administrator:

```
anaconda-enterprise-cli channels share --group GROUPNAME --level r <channelname>
```

Replacing GROUPNAME with the actual name of your group.

**Note:** Adding --level r grants this group read-only access to the channel.

You can "unshare" a channel using the following command:

```
anaconda-enterprise-cli channels share --user <username> --remove <channelname>
```

Run anaconda-enterprise-cli channels --help to see more information about what you can do with channels.

For help with a specific command, enter that command followed by --help:

```
anaconda-enterprise-cli channels share --help
```

## 4.3.5 Configuring conda

If you are familiar with conda and want to use it to install the packages you need, you can configure conda to search a specific set of channels for packages. Listing channel locations in the .condarc file overrides conda defaults, causing conda to search only the channels listed, in the order specified.

The channels you specify can be *public* or *private*. Private channels will require you to authenticate before you can conda install packages from them.

If your organization has configured conda *at the system level* to limit platform users to only access packages in your on-premises repository, **this will override your user-level configuration file**.

To configure conda, create or update your ~.condarc configuration file in the root directory of your local machine to include your preferred repository channels. For example:

#### channels:

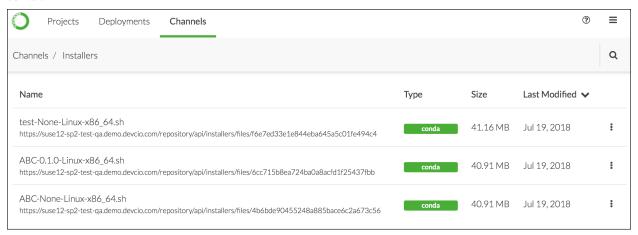
- <anaconda\_dot\_org\_username>
- http://some.custom/channel
- file:///some/local/directory
- defaults

For more information, see this section of the conda docs.

# 4.4 Using installers, parcels and management packs

In addition to Anaconda and Miniconda installers, your administrator may create custom installers, Cloudera Manager parcels, or Hortonworks Data Manager management packs for you and your colleagues to use. They make these specific packages and their dependencies available to you via *channels*.

To view the installers available to you, navigate to the **Channels** menu, then click the **Installers** link in the top-right corner.



To download an installer, click on its name in the list.

**Note:** If you don't see an installer that you expected to see, please contact your administrator and ask them to *generate* the installer you need.

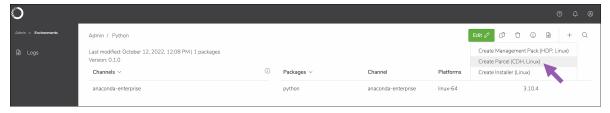
# 4.4.1 Working with Cloudera Data Platform (CDP) parcels

When working with CDP, create a parcel containing the conda environment you have created, then import the parcel into your CDP cluster.

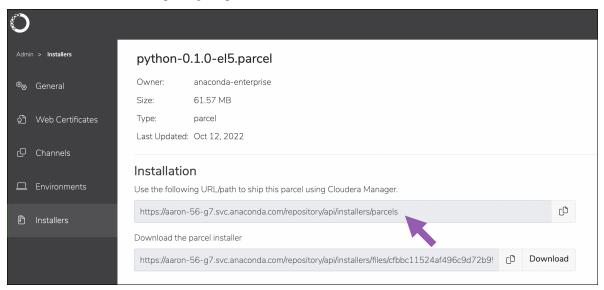
1. Create the environment.



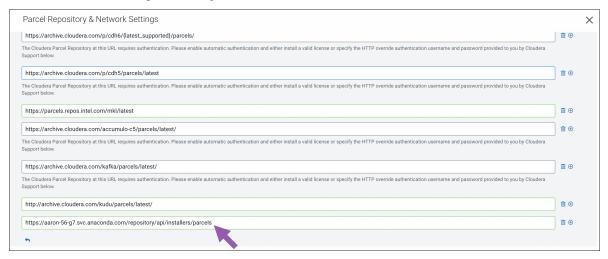
2. Create the parcel.



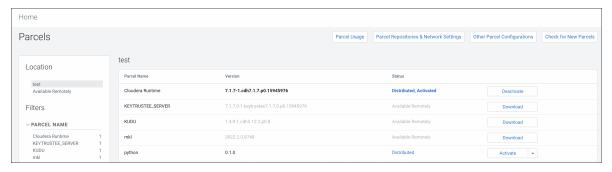
3. Retrieve the URL used for importing the parcel into CDP.



4. Add the URL to the CDP parcel configuration.



5. Save your changes, check for new parcels, and download the new parcel.



# 4.5 Working with data

## 4.5.1 Loading data into your project

Data Science & AI Workbench uses *projects* to encapsulate all of the components necessary to use or run an application: the relevant packages, channels, scripts, notebooks and other related files, environment variables, services and commands, along with a configuration file named anaconda-project.yml.

You can also access and load data in a variety of formats, stored in common sources including the following:

- File systems
- NFS shared drives
- Databases
- Hadoop and Spark clusters
- Distributed version control repositories such as Git and Bitbucket (if configured by your Administrator).

The amount of data you read into your project will impact the resources required to successfully run the project, whether in a notebook session or deployment. See the following section on *understanding resource profiles* to learn more.

## **Understanding resource profiles**

Resource profiles are used to limit the amount of CPU cores and RAM available for use when running a project session or *deployment*.

**Note:** Choosing a resource profile with a greater number of available cores is not guaranteed to improve performance—it will also depend on whether the libraries used by the project can take advantage of multiple cores, for example.

Memory limits are enforced by the Linux kernel, so when the memory limit is exceeded the most recent process will crash. Be sure to select a resource profile that offers sufficient runtime resources required by your project to avoid such errors. A best practice recommendation is to **choose a resource profile with roughly double the amount of memory required by the size of data you need to read**.

To see the total memory in use, open a terminal and run the following command:

```
cat /sys/fs/cgroup/memory/memory.usage_in_bytes | awk '{print $1/1024/1024}'
```

#### Uploading files to a project

Open an editing session for the project, then choose the file you want to upload. The process of uploading files varies slightly, based on the editor used:

- In Jupyter Notebook, click **Upload** and select the file to upload. Then click the blue **Upload** button displayed in the file's row to add the file to the project
- In JupyterLab, click the Upload files icon and select the file. In the top right corner, click Commit Changes to add the file to your project.
- In Zeppelin, use the **Import note** feature to select a JSON file or add data from a URL.

Once a file is in the project, you can use code to read it. For example, to load the iris dataset from a comma separated value (CSV) file into a pandas DataFrame:

```
import pandas as pd
irisdf = pd.read_csv('iris.csv')
```

## **Accessing NFS shared drives**

After your Administrator has *configured Workbench to mount an NFS share*, you'll be able to access it from within your notebooks. You'll just need to know the name of the volume, so you can access it. For example, if they named the configuration file section myvolume, the share will be mounted at /data/myvolume.

From a notebook you can use code such as this to read data from the share:

```
import pandas as pd
irisdf = pd.read_csv('/data/myvolume/iris.csv')
```

## Accessing data stored in databases

You can also connect to the following database engines to access data stored within them:

- · Cassandra
- · Cockroach
- Cosmos
- Couchbase
- Db2
- · Elasticsearch
- MariaDB
- MLDB
- MongoDB
- MS SQL
- MySQL
- Neo4j
- Oracle
- PostgreSQL
- · Redis
- S3
- Snowflake
- Vertica

See *Storing secrets* for information about adding credentials to the platform, to make them available in your projects. Any secrets you add will be available across all sessions and deployments associated with your user account.

### Hadoop Distributed File System (HDFS), Spark, Hive, and Impala

Loading data from HDFS, Spark, Hive, and Impala is discussed in *Hadoop / Spark*.

## 4.5.2 Exploring project data

With Data Science & AI Workbench, you can explore project data using visualization libraries such as Bokeh and Matplotlib, and numeric libraries such as NumPy, SciPy, and Pandas.

Use these tools to discover patterns and relationships in your datasets, and develop approaches for your analysis and deployment pipelines.

The following examples use the Iris flower data set, and this mini customer data set (customers.csv):

```
customer_id,title,industry
1,data scientist,retail
2,data scientist,academia
3,compiler optimizer,academia
4,data scientist,finance
5,compiler optimizer,academia
6,data scientist,academia
7,compiler optimizer,academia
8,data scientist,retail
9,compiler optimizer,finance
```

1. Begin by importing libraries, and reading data into a Pandas DataFrame:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
irisdf = pd.read_csv('iris.csv')
customerdf = pd.read_csv('customers.csv')

%matplotlib inline
```

2. Then list column / variable names:

3. Summary statistics include minimum, maximum, mean, median, percentiles, and more:

```
print('length:', len(irisdf)) # length of data set
print('shape:', irisdf.shape) # length and width of data set
print('size:', irisdf.size) # length * width
print('min:', irisdf['sepal_width'].min())
```

(continues on next page)

```
length: 150
shape: (150, 5)
size: 750
min: 2.0
max: 4.4
mean: 3.057333333333337
median: 3.0
50th percentile: 3.0
5th percentile: 2.34499999999998
10th percentile: 2.5
95th percentile: 3.8
```

#. Use the value\_counts function to show the number of items in each category, sorted from largest to smallest. You can also set the ascending argument to True to display the list from smallest to largest.

```
print(customerdf['industry'].value_counts())
print()
print(customerdf['industry'].value_counts(ascending=True))
```

```
academia 5
finance 2
retail 2
Name: industry, dtype: int64

retail 2
finance 2
academia 5
Name: industry, dtype: int64
```

### **Categorical variables**

In statistics, a categorical variable may take on a limited number of possible values. Examples could include blood type, nation of origin, or ratings on a Likert scale.

Like numbers, the possible values may have an order, such as from disagree to neutral to agree. The values cannot, however, be used for numerical operations such as addition or division.

Categorical variables tell other Python libraries how to handle the data, so those libraries can default to suitable statistical methods or plot types.

The following example converts the class variable of the Iris dataset from object to category.

```
print(irisdf.dtypes)
print()
irisdf['class'] = irisdf['class'].astype('category')
print(irisdf.dtypes)
```

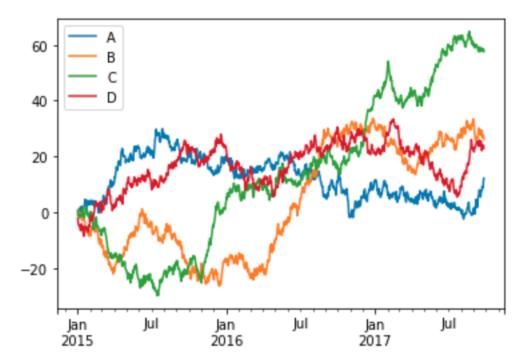
```
float64
sepal_length
                float64
sepal_width
petal_length
                float64
petal_width
                float64
class
                 object
dtype: object
sepal_length
                 float64
sepal_width
                 float64
petal_length
                 float64
petal_width
                 float64
class
                category
dtype: object
```

Within Pandas, this creates an array of the possible values, where each value appears only once, and replaces the strings in the DataFrame with indexes into the array. In some cases, this saves significant memory.

A categorical variable may have a logical order different than the lexical order. For example, for ratings on a Likert scale, the *lexical* order could alphabetize the strings and produce agree, disagree, neither agree nor disagree, strongly agree, strongly disagree. The *logical* order could range from most negative to most positive as strongly disagree, disagree, neither agree nor disagree, agree, strongly agree.

### Time series data visualization

The following code sample creates four series of random numbers over time, calculates the cumulative sums for each series over time, and plots them.

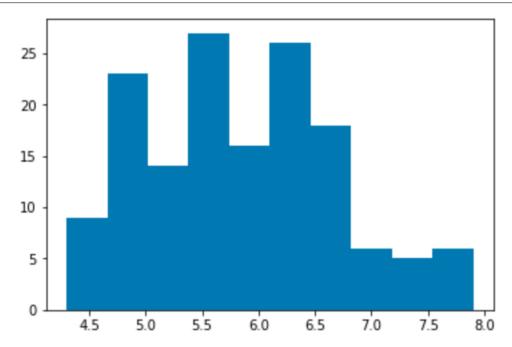


This example was adapted from http://pandas.pydata.org/pandas-docs/stable/visualization.html.

# **Histograms**

This code sample plots a histogram of the sepal length values in the Iris data set:

```
plt.hist(irisdf['sepal_length'])
plt.show()
```



### **Bar charts**

The following sample code produces a bar chart of the industries of customers in the customer data set.

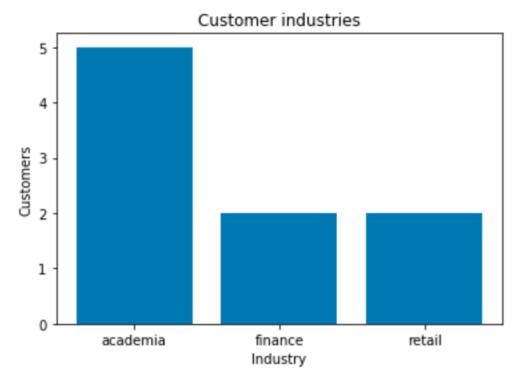
```
industries = customerdf['industry'].value_counts()

fig, ax = plt.subplots()

ax.bar(np.arange(len(industries)), industries)

ax.set_xlabel('Industry')
ax.set_ylabel('Customers')
ax.set_title('Customer industries')
ax.set_xticks(np.arange(len(industries)))
ax.set_xticklabels(industries.index)

plt.show()
```

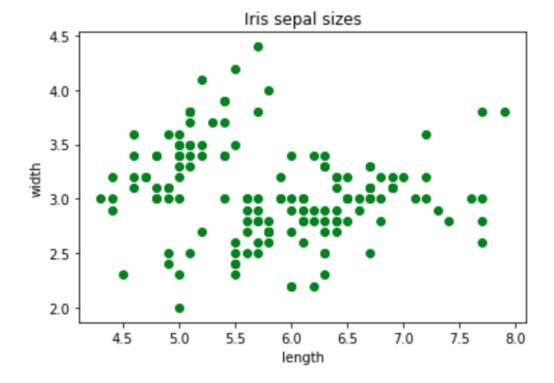


This example was adapted from https://matplotlib.org/gallery/statistics/barchart\_demo.html.

# **Scatter plots**

This code sample makes a scatter plot of the sepal lengths and widths in the Iris data set:

```
fig, ax = plt.subplots()
ax.scatter(irisdf['sepal_length'], irisdf['sepal_width'], color='green')
ax.set(
    xlabel="length",
    ylabel="width",
    title="Iris sepal sizes",
)
plt.show()
```



# **Sorting**

To show the customer data set:

customerdf

row	customer_id	title	industry
0	1	data scientist	retail
1	2	data scientist	academia
2	3	compiler optimizer	academia
3	4	data scientist	finance
4	5	compiler optimizer	academia
5	6	data scientist	academia
6	7	compiler optimizer	academia
7	8	data scientist	retail
8	9	compiler optimizer	finance

To sort by industry and show the results:

```
customerdf.sort_values(by=['industry'])
```

row	customer_id	title	industry
1	2	data scientist	academia
2	3	compiler optimizer	academia
4	5	compiler optimizer	academia
5	6	data scientist	academia
6	7	compiler optimizer	academia
3	4	data scientist	finance
8	9	compiler optimizer	finance
0	1	data scientist	retail
7	8	data scientist	retail

To sort by industry and then title:

```
customerdf.sort_values(by=['industry', 'title'])
```

row	customer_id	title	industry
2	3	compiler optimizer	academia
4	5	compiler optimizer	academia
6	7	compiler optimizer	academia
1	2	data scientist	academia
5	6	data scientist	academia
8	9	compiler optimizer	finance
3	4	data scientist	finance
0	1	data scientist	retail
7	8	data scientist	retail

The sort\_values function can also use the following arguments:

- axis to sort either rows or columns
- ascending to sort in either ascending or descending order
- inplace to perform the sorting operation in-place, without copying the data, which can save space
- kind to use the quicksort, merge sort, or heapsort algorithms
- na\_position to sort not a number (NaN) entries at the end or beginning

### Grouping

customerdf.groupby('title')['customer\_id'].count() counts the items in each group, excluding missing values such as not-a-number values (NaN). Because there are no missing customer IDs, this is equivalent to customerdf.groupby('title').size().

```
print(customerdf.groupby('title')['customer_id'].count())
print()
print(customerdf.groupby('title').size())
print()
print(customerdf.groupby(['title', 'industry']).size())
```

```
print()
print(customerdf.groupby(['industry', 'title']).size())
```

```
title
compiler optimizer
data scientist
                      5
Name: customer_id, dtype: int64
title
compiler optimizer
                      4
                      5
data scientist
dtype: int64
title
                    industry
                                3
compiler optimizer academia
                    finance
                                1
data scientist
                    academia
                                2
                    finance
                                1
                    retail
dtype: int64
industry title
academia compiler optimizer
                                3
                                2
          data scientist
finance
          compiler optimizer
                                1
          data scientist
                                1
retail
          data scientist
                                2
dtype: int64
```

By default groupby sorts the group keys. You can use the sort=False option to prevent this, which can make the grouping operation faster.

### **Binning**

Binning or bucketing moves continuous data into discrete chunks, which can be used as ordinal categorical variables.

You can divide the range of the sepal length measurements into four equal bins:

```
pd.cut(irisdf['sepal_length'], 4).head()
```

```
0  (4.296, 5.2]
1  (4.296, 5.2]
2  (4.296, 5.2]
3  (4.296, 5.2]
4  (4.296, 5.2]
Name: sepal_length, dtype: category
Categories (4, interval[float64]): [(4.296, 5.2] < (5.2, 6.1] < (6.1, 7.0] < (7.0, 7.9]]</pre>
```

Or make a custom bin array to divide the sepal length measurements into integer-sized bins from 4 through 8:

```
custom_bin_array = np.linspace(4, 8, 5)
custom_bin_array
```

```
array([4., 5., 6., 7., 8.])
```

Copy the Iris data set, and apply the binning to it:

```
iris2=irisdf.copy()
iris2['sepal_length'] = pd.cut(iris2['sepal_length'], custom_bin_array)
iris2['sepal_length'].head()
```

```
0 (5.0, 6.0]

1 (4.0, 5.0]

2 (4.0, 5.0]

3 (4.0, 5.0]

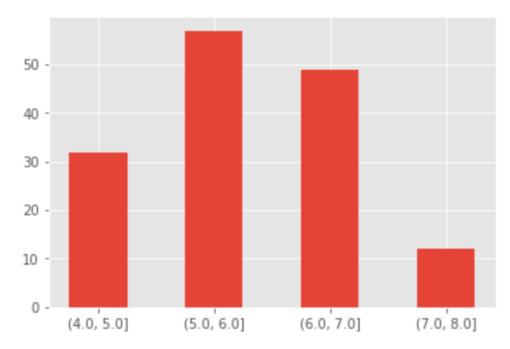
4 (4.0, 5.0]

Name: sepal_length, dtype: category

Categories (4, interval[float64]): [(4.0, 5.0] < (5.0, 6.0] < (6.0, 7.0] < (7.0, 8.0]]
```

Then plot the binned data:

```
plt.style.use('ggplot')
categories = iris2['sepal_length'].cat.categories
ind = np.array([x for x, _ in enumerate(categories)])
plt.bar(ind, iris2.groupby('sepal_length').size(), width=0.5, label='Sepal length')
plt.xticks(ind, categories)
plt.show()
```



This example was adapted from http://benalexkeen.com/bucketing-continuous-variables-in-pandas/.

# 4.5.3 Data preparation

Data Science & AI Workbench supports data preparation using numeric libraries such as NumPy, SciPy, and Pandas.

These examples use this small data file vendors.csv:

```
Vendor Number, Vendor Name, Month, Day, Year, Active, Open Orders, 2015, 2016, Percent Growth "104.0", ACME Inc, 2, 15, 2014, "Y", 200, "$45,000.00", $54000.00, 20.00% 205, Apogee LTD, 8, 12, 2015, "Y", 150, "$29,000.00", "$30,450.00", 5.00% 143, Zenith Co, 4, 5, 2014, "Y", 290, "$18,000.00", $23400.00, 30.00% 166, Hollerith Propulsion, 9, 25, 2015, "Y", 180, "$48,000.00", $48960.00, 2.00% 180, Airtek Industrial, 8, 2, 2014, "N", Closed, "$23,000.00", $17250.00, -25.00%
```

The columns are the vendor ID number, vendor name, month day and year of first purchase from the vendor, whether the account is currently active, the number of open orders, purchases in 2015 and 2016, and percent growth in orders from 2015 to 2016.

### **Converting data types**

Computers handle many types of data, including integer numbers such as 365, floating point numbers such as 365.2425, strings such as "ACME Inc", and more.

An operation such as division may work for integers and floating point numbers, but produce an error if used on strings.

Often data libraries such as pandas will automatically use the correct types, but they do provide ways to correct and change the types when needed. For example, you may wish to convert between an integer such as 25, the floating point number 25.0, and strings such as "25", "25.0", or "\$25.00".

Pandas data types or dtypes correspond to similar Python types.

Strings are called str in Python and object in pandas.

Integers are called int in Python and int64 in pandas, indicating that pandas stores integers as 64-bit numbers.

Floating point numbers are called float in Python and float64 in pandas, also indicating that they are stored with 64 bits

A boolean value, named for logician George Boole, can be either True or False. These are called bool in Python and bool in pandas.

Pandas includes some data types with no corresponding native Python type: datetime64 for date and time values, timedelta[ns] for storing the difference between two times as a number of nanoseconds, and category where each item is one of a list of strings.

Here we import the vendor data file and show the dtypes:

```
import pandas as pd
import numpy as np
df = pd.read_csv('vendors.csv')
df.dtypes
```

Vendor Number	float64
Vendor Name	object
Month	int64
Day	int64
Year	int64
Active	object

```
Open Orders object
2015 object
2016 object
Percent Growth object
dtype: object
```

Try adding the 2015 and 2016 sales:

```
df['2015']+df['2016']
```

```
0 $45,000.00$54000.00

1 $29,000.00$30,450.00

2 $18,000.00$23400.00

3 $48,000.00$48960.00

4 $23,000.00$17250.00

dtype: object
```

These columns were stored as the type "object", and concatenated as strings, not added as numbers.

Examine more information about the DataFrame:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 10 columns):
Vendor Number
                 5 non-null float64
Vendor Name
                  5 non-null object
                  5 non-null int64
Month
Day
                  5 non-null int64
Year
                 5 non-null int64
Active
                 5 non-null object
Open Orders
                 5 non-null object
2015
                 5 non-null object
2016
                 5 non-null object
Percent Growth
                 5 non-null object
dtypes: float64(1), int64(3), object(6)
memory usage: 480.0+ bytes
```

Vendor Number is a float and not an int. 2015 and 2016 sales, percent growth, and open orders are stored as objects and not numbers. The month, day, and year values should be converted to datetime64, and the active column should be converted to a boolean.

The data can be converted with the astype() function, custom functions, or pandas functions such as to\_numeric() or to\_datetime().

### astype()

The astype() function can convert the Vendor Number column to int:

```
df['Vendor Number'].astype('int')
```

```
0 104
1 205
2 143
3 166
4 180
Name: Vendor Number, dtype: int64
```

astype() returns a copy, so an assignment statement will convert the original data. This can be checked by showing the dtypes.

```
df['Vendor Number'] = df['Vendor Number'].astype('int')
df.dtypes
```

```
Vendor Number
                    int64
Vendor Name
                   object
Month
                    int64
Day
                    int64
Year
                    int64
Active
                  object
Open Orders
                  object
2015
                  object
2016
                  object
Percent Growth
                  object
dtype: object
```

However, trying to convert the 2015 column to a float or the Open Orders column to an int returns an error.

```
df['2015'].astype('float')
```

```
ValueError: could not convert string to float: '$23,000.00'
```

```
df['Open Orders'].astype('int')
```

```
ValueError: invalid literal for int() with base 10: 'Closed'
```

Even worse, trying to convert the Active column to a bool completes with no errors, but converts both Y and N values to True.

```
df['Active'].astype('bool')
```

```
0 True
1 True
2 True
3 True
4 True
Name: Active, dtype: bool
```

astype() works if the data is clean and can be interpreted simply as a number, or if you want to convert a number to a string. Other conversions require custom functions or pandas functions such as to\_numeric() or to\_datetime().

### **Custom conversion functions**

This small custom function converts a currency string like the ones in the 2015 column to a float by first removing the comma (,) and dollar sign (\$) characters.

```
def currency_to_float(a):
    return float(a.replace(',','').replace('$',''))
```

Test the function on the 2015 column with the apply() function:

```
df['2015'].apply(currency_to_float)
```

```
0 45000.0
1 29000.0
2 18000.0
3 48000.0
4 23000.0
Name: 2015, dtype: float64
```

Convert the 2015 and 2016 columns and show the dtypes:

```
df['2015'] = df['2015'].apply(currency_to_float)
df['2016'] = df['2016'].apply(currency_to_float)
df.dtypes
```

```
Vendor Number
                    int64
Vendor Name
                    object
Month
                    int64
Day
                    int64
Year
                    int64
Active
                    object
Open Orders
                   object
2015
                   float64
2016
                  float64
Percent Growth
                    object
dtype: object
```

Convert the Percent Growth column:

```
def percent_to_float(a):
    return float(a.replace('%',''))/100
df['Percent Growth'].apply(percent_to_float)
```

```
0 0.20
1 0.05
2 0.30
3 0.02
4 -0.25
Name: Percent Growth, dtype: float64
```

```
df['Percent Growth'] = df['Percent Growth'].apply(percent_to_float)
df.dtypes
```

```
Vendor Number
                     int64
Vendor Name
                    object
Month
                     int64
                     int64
Day
Year
                     int64
Active
                    object
Open Orders
                    object
                   float64
2015
2016
                   float64
Percent Growth
                   float64
dtype: object
```

NumPy's np.where() function is a good way to convert the Active column to bool. This code converts "Y" values to True and all other values to False, then shows the dtypes:

```
np.where(df["Active"] == "Y", True, False)
```

```
array([ True, True, True, False])
```

```
df["Active"] = np.where(df["Active"] == "Y", True, False)
df.dtypes
```

```
Vendor Number
                     int64
Vendor Name
                    object
Month
                     int64
Day
                     int64
Year
                     int64
Active
                      bool
Open Orders
                    object
2015
                   float64
2016
                   float64
Percent Growth
                   float64
dtype: object
```

### **Pandas helper functions**

The Open Orders column has several integers, but one string. Using astype() on this column would produce an error, but the pd.to\_numeric() function built in to pandas will convert the numeric values to numbers and any other values to the "not a number" or "NaN" value built in to the floating point number standard:

```
pd.to_numeric(df['Open Orders'], errors='coerce')
```

```
0 200.0

1 150.0

2 290.0

3 180.0

4 NaN

Name: Open Orders, dtype: float64
```

In this case, a non-numeric value in this field indicates that there are zero open orders, so we can convert NaN values to zero with the function fillna():

```
pd.to_numeric(df['Open Orders'], errors='coerce').fillna(0)
```

```
0 200.0

1 150.0

2 290.0

3 180.0

4 0.0

Name: Open Orders, dtype: float64
```

Similarly, the pd.to\_datetime() function built in to pandas can convert the Month Day and Year columns to datetime64[ns]:

```
pd.to_datetime(df[['Month', 'Day', 'Year']])
```

```
0 2014-02-15

1 2015-08-12

2 2014-04-05

3 2015-09-25

4 2014-08-02

dtype: datetime64[ns]
```

Use these functions to change the DataFrame, then show the dtypes:

```
df['Open Orders'] = pd.to_numeric(df['Open Orders'], errors='coerce').fillna(0)
df['First Purchase Date'] = pd.to_datetime(df[['Month', 'Day', 'Year']])
df.dtypes
```

```
Vendor Number
                                 int64
Vendor Name
                                object
Month
                                 int64
Day
                                 int64
Year
                                 int64
Active
                                  bool
                               float64
Open Orders
2015
                               float64
2016
                               float64
Percent Growth
                               float64
First Purchase Date
                       datetime64[ns]
dtype: object
```

# Converting data as it is read

You can apply dtype and converters in the pd.read\_csv() function. Defining dtype is like performing astype() on the data.

A dtype or a converter can only be applied once to a specified column. If you try to apply both to the same column, the dtype is skipped.

After converting as much of the data as possible in pd.read\_csv(), use code similar to the previous examples to convert the rest.

```
Vendor Number
                            Vendor Name Month Day Year
                                                             Active Open Orders
                                                                                        2015
    2016 Percent Growth First Purchase Date
                                                                             200.0 45000.0 _
              104
                                ACME Inc
                                                   15
                                                       2014
                                                                True
→54000.0
                      0.20
                                     2014-02-15
                                                                                    29000.0 🚨
1
             205
                              Apogee LTD
                                               8
                                                   12 2015
                                                                True
                                                                             150.0
\rightarrow 30450.0
                      0.05
                                     2015-08-12
              143
                               Zenith Co
                                               4
                                                       2014
                                                                True
                                                                             290.0
                                                                                    18000.0 🚨
                      0.30
\rightarrow23400.0
                                     2014-04-05
3
              166 Hollerith Propulsion
                                                   25 2015
                                                                True
                                                                             180.0
                                                                                    48000.0 _
                      0.02
                                     2015-09-25
48960.0
              180
                      Airtek Industrial
                                                    2 2014
                                                               False
                                                                                    23000.0
\rightarrow 17250.0
                     -0.25
                                     2014-08-02
```

```
df2.dtypes
```

```
Vendor Number
                                 int64
Vendor Name
                                 object
Month
                                 int64
Day
                                 int64
Year
                                 int64
Active
                                  bool
Open Orders
                               float64
                               float64
2015
2016
                               float64
Percent Growth
                               float64
First Purchase Date
                        datetime64[ns]
dtype: object
```

We thank http://pbpython.com/pandas\_dtypes.html for providing data preparation examples that inspired these examples.

# Merging and joining data sets

You can use pandas to merge DataFrames:

```
C
  kev
        Α
             В
  K0
       A0
            B0
                C0
                    D0
  K1
       A1
            B1
                C1
                     D1
                C2
                     D2
   K2
       A2
            B2
3
   К3
       A3
            B3
                C3
                     D3
```

The available merge methods are left to use keys from the left frame only, right to use keys from the right frame only, outer to use the union of keys from both frames, and the default inner to use the intersection of keys from both frames.

This merge using the default inner join omits key combinations found in only one of the source DataFrames:

```
C
  key1 key2
               Α
                    В
                             D
    K0
          K0
              A0
                  B0
                       C0
                           D0
1
    Κ1
          Κ0
              A2
                   B2
                       C1
                           D1
2
    K1
                       C2
                           D2
          Κ0
              A2
                  B2
```

This example omits the rows with key1 and key2 set to K0, K1, K2, K1, or K2, K0.

Joins also copy information when necessary. The left DataFrame had one row with the keys set to K1, K0 and the right DataFrame had two. The output DataFrame has two, with the information from the left DataFrame copied into both rows.

The next example shows the results of a left, right, and outer merge on the same inputs. Empty cells are filled in with NaN values.

```
pd.merge(left, right, how='left', on=['key1', 'key2'])
```

```
key1 key2
                     В
                            C
                                  D
                 Α
                           C<sub>0</sub>
                                 D0
0
    K0
          K0
               Α0
                    B0
1
    K0
          Κ1
               Α1
                    В1
                         NaN
                               NaN
2
    K1
          K0
               A2
                    B2
                           C1
                                 D1
3
    K1
           Κ0
               A2
                    B2
                           C2
                                 D2
    K2
          K1 A3
                    В3
                         NaN
                               NaN
```

```
pd.merge(left, right, how='right', on=['key1', 'key2'])
```

```
key1 key2
                      В
                           C
                                D
                Α
    K0
          K0
                Α0
                     B0
                          C0
                              D0
    Κ1
                A2
                          C1
                              D1
1
          K0
                     B2
2
    Κ1
          K0
                A2
                     B2
                          C2
                              D2
    K2
                          C3
          Κ0
              NaN
                    NaN
                              D3
```

```
pd.merge(left, right, how='outer', on=['key1', 'key2'])
```

```
C
  key1 key2
                  Α
                        В
                                     D
    K0
           K0
                 A0
                       B0
                             C<sub>0</sub>
                                   D0
1
    K0
           Κ1
                 A1
                       В1
                            NaN
                                  NaN
2
    K1
           K0
                 A2
                       B2
                             C1
                                   D1
3
                       B2
                             C2
                                   D2
    Κ1
           K0
                 A2
    K2
           Κ1
                 A3
                       В3
                            NaN
                                  NaN
    K2
           Κ0
               NaN
                             C3
                                   D3
                      NaN
```

If a key combination appears more than once in both tables, the output will contain the Cartesian product of the associated data.

In this small example a key that appears twice in the left frame and three times in the right frame produces six rows in the output frame.

```
left = pd.DataFrame({'A' : [1,2], 'B' : [2, 2]})
right = pd.DataFrame({'A' : [4,5,6], 'B': [2,2,2]})
pd.merge(left, right, on='B', how='outer')
```

```
A_y
   A_x B
0
     1
        2
              4
     1
        2
               5
1
2
     1
        2
              6
3
     2
        2
              4
4
     2
        2
              5
5
     2
        2
```

To prevent very large outputs and memory overflow, manage duplicate values in keys before joining large DataFrames.

While merging uses one or more columns as keys, joining uses the indexes, also known as row labels.

Join can also perform left, right, inner, and outer merges, and defaults to left.

```
В
               C
                     D
     Α
K0
    Α0
        B0
              C0
                    D0
    A1
        В1
             NaN
                   NaN
K2
    A2
              C2
                    D2
        B2
```

```
left.join(right, how='outer')
```

```
Α
            В
                   C
                        D
                 C0
K0
     A0
           B0
                       D0
Κ1
     A1
           B1
                NaN
                      NaN
K2
     A2
           B2
                 C2
                       D2
К3
    NaN
          NaN
                 C3
                       D3
```

```
left.join(right, how='inner')
```

```
A B C D
K0 A0 B0 C0 D0
K2 A2 B2 C2 D2
```

This is equivalent to using merge with arguments instructing it to use the indexes:

```
pd.merge(left, right, left_index=True, right_index=True, how='inner')
```

```
A B C D
K0 A0 B0 C0 D0
K2 A2 B2 C2 D2
```

You can join a frame indexed by a join key to a frame where the key is a column:

```
B key
            C
              D
 Α
Α0
    B0
       K0
           C0
              D0
    B1 K1
           C1 D1
Α1
    B2 K0
           C0 D0
A2
А3
    B3 K1
           C1 D1
```

You can join on multiple keys if the passed DataFrame has a MultiIndex:

```
C D

K0 K0 C0 D0

K1 K0 C1 D1

K2 K0 C2 D2

K1 C3 D3
```

```
left.join(right, on=['key1', 'key2'])
```

```
B key1 key2
                         C
    Α
0
  Α0
       B0
             K0
                  K0
                        C0
                             D0
  A1
       B1
             K0
                  Κ1
                       NaN
                            NaN
       B2
             K1
                        C1
   A2
                  K0
                             D1
3
       B3
                  Κ1
                        C3
   А3
             K2
                             D3
```

Note that this defaulted to a left join, but other types are also available:

```
left.join(right, on=['key1', 'key2'], how='inner')
```

```
B key1 key2
                       C
                           D
    Α
                      C0 D0
  A0
       B0
            K0
                  K0
       B2
                      C1
                          D1
2
  A2
             K1
                  Κ0
3
   A3
       В3
            K2
                  Κ1
                      C3
                          D3
```

For more information, including examples of using merge to join a single index to a multi-index or to join two multi-indexes, see the pandas documentation on merging.

When column names in the input frames overlap, pandas appends suffixes to disambiguate them. These default to \_x and \_y but you can customize them:

```
left = pd.DataFrame({'k': ['K0', 'K1', 'K2'], 'v': [1, 2, 3]})
right = pd.DataFrame({'k': ['K0', 'K0', 'K3'], 'v': [4, 5, 6]})
pd.merge(left, right, on='k')
```

```
k v_x v_y
0 K0 1 4
1 K0 1 5
```

```
pd.merge(left, right, on='k', suffixes=['_1', '_r'])
```

```
k v_l v_r
0 K0 1 4
1 K0 1 5
```

Join has similar arguments lsuffix and rsuffix.

```
left = left.set_index('k')
right = right.set_index('k')
left.join(right, lsuffix='_l', rsuffix='_r')
```

```
      v_1
      v_r

      k
      K0
      1
      4.0

      K0
      1
      5.0

      K1
      2
      NaN

      K2
      3
      NaN
```

You can join a list or tuple of DataFrames on their indexes:

```
right2 = pd.DataFrame({'v': [7, 8, 9]}, index=['K1', 'K1', 'K2'])
left.join([right, right2])
```

```
v_x v_y
                v
Κ0
      1
         4.0
              NaN
K0
      1
         5.0
              NaN
Κ1
      2
         NaN
              7.0
Κ1
      2
         NaN
              8.0
K2
      3 NaN 9.0
```

If you have two frames with similar indices and want to fill in missing values in the left frame with values from the right frame, use the combine\_first() method:

```
0 1 2
0 NaN 3.0 5.0
1 -4.6 NaN -8.2
2 -5.0 7.0 4.0
```

The method update() overwrites values in a frame with values from another frame:

```
df1.update(df2)
df1
```

```
0 1 2
0 NaN 3.0 5.0
1 -42.6 NaN -8.2
2 -5.0 1.6 4.0
```

The pandas documentation on merging has more information, including examples of combining time series and other ordered data, with options to fill and interpolate missing data.

We thank the pandas documentation for many of these examples.

### Filtering data

This example uses a vendors DataFrame similar to the one we used above:

```
'FirstPurchaseDate': ['2014-02-15', '2015-08-12', '2014-04-05', '2015-09-

→25', '2014-08-02']})

df['FirstPurchaseDate'] = df['FirstPurchaseDate'].astype('datetime64[ns]')

df
```

	VendorNumber	VendorName	Active	OpenOrders	Purchases2015	Purchases2016 _
-→ <b>F</b>	PercentGrowth	FirstPurchaseDate				
0	104	ACME Inc	True	200	45000.0	54000.0 🚨
$\hookrightarrow$	0.20	2014-02-15				
1	205	Apogee LTD	True	150	29000.0	30450.0 🚨
$\hookrightarrow$	0.05	2015-08-12				
2	143	Zenith Co	True	290	18000.0	23400.0 🚨
$\hookrightarrow$	0.30	2014-04-05				
3	166	Hollerith Propulsion	True	180	48000.0	48960.0 🚨
$\hookrightarrow$	0.02	2015-09-25				
4	180	Airtek Industrial	False	0	23000.0	17250.0 🚨
$\hookrightarrow$	-0.25	2014-08-02				

To filter only certain rows from a DataFrame, call the query method with a boolean expression based on the column names.

```
df.query('0penOrders>160')
```

	VendorNumber	VendorName	Active	OpenOrders	Purchases2015	Purchases2016 _
←J	PercentGrowth	FirstPurchaseDate				
0	104	ACME Inc	True	200	45000.0	54000.0 🚨
$\hookrightarrow$	0.20	2014-02-15				
2	143	Zenith Co	True	290	18000.0	23400.0 🚨
$\hookrightarrow$	0.30	2014-04-05				
3	166	Hollerith Propulsion	True	180	48000.0	48960.0 🚨
$\hookrightarrow$	0.02	2015-09-25				

Filtering can be done with indices instead of queries:

```
df[(df.0penOrders < 190) & (df.Active == True)]</pre>
```

	VendorNumber	VendorName	Active	OpenOrders	Purchases2015	Purchases2016 _
_	PercentGrowth	FirstPurchaseDate				
1	205	Apogee LTD	True	150	29000.0	30450.0 🚨
_	0.05	2015-08-12				
3	166	Hollerith Propulsion	True	180	48000.0	48960.0 🚨
<u>د</u>	0.02	2015-09-25				

# 4.5.4 Using statistics

Data Science & AI Workbench supports statistical work using the R language and Python libraries such as NumPy, SciPy, Pandas, Statsmodels, and scikit-learn.

The following Jupyter notebook Python examples show how to use these libraries to calculate correlations, distributions, regressions, and principal component analysis.

These examples also include plots produced with the libraries seaborn and Matplotlib.

We thank these sites, from whom we have adapted some code:

- $\bullet\ https://stackoverflow.com/questions/25571882/pandas-columns-correlation-with-statistical-significance/49040342$
- https://joomik.github.io/Housing/

Start by importing necessary libraries and functions, including Pandas, SciPy, scikit-learn, Statsmodels, seaborn, and Matplotlib.

This code imports load\_boston to provide the Boston housing dataset from the datasets included with scikit-learn.

```
import pandas as pd
import seaborn as sns
from scipy.stats import pearsonr
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.formula.api as sm
%matplotlib inline
```

Load the Boston housing data into a Pandas DataFrame:

```
#Load dataset and convert it to a Pandas dataframe
boston = load_boston()
df = pd.DataFrame(boston.data, columns=boston.feature_names)
df['target'] = boston.target
```

In the Boston housing dataset, the target variable is MEDV, the median home value.

Print the dataset description:

```
#Description of the dataset
print(boston.DESCR)
```

```
:Number of Instances: 506
:Number of Attributes: 13 numeric/categorical predictive
:Median Value (attribute 14) is usually the target
:Attribute Information (in order):
              per capita crime rate by town
    - CRIM
              proportion of residential land zoned for lots over 25,000 sq.ft.
    - ZN
   - INDUS
              proportion of non-retail business acres per town
   - CHAS
              Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
   - NOX
              nitric oxides concentration (parts per 10 million)
    - RM
               average number of rooms per dwelling
   AGE
              proportion of owner-occupied units built prior to 1940
   - DIS
              weighted distances to five Boston employment centres
    - RAD
              index of accessibility to radial highways
   - TAX
               full-value property-tax rate per $10,000
   - PTRATIO pupil-teacher ratio by town
              1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
   - B
    - LSTAT
              % lower status of the population
    MEDV
              Median value of owner-occupied homes in $1000's
:Missing Attribute Values: None
:Creator: Harrison, D. and Rubinfeld, D.L.
This is a copy of UCI ML housing dataset.
http://archive.ics.uci.edu/ml/datasets/Housing
This dataset was taken from the StatLib library which is maintained at Carnegie Mellon.
→University.
The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
                     Used in Belsley, Kuh & Welsch, 'Regression diagnostics
vol.5, 81-102, 1978.
...', Wiley, 1980. N.B. Various transformations are used in the table on
pages 244-261 of the latter.
The Boston house-price data has been used in many machine learning papers that address.
→regression
problems.
**References**
- Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and.
→Sources of Collinearity', Wiley, 1980. 244-261.
- Quinlan, R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings
→on the Tenth International Conference of Machine Learning, 236-243, University of
→Massachusetts, Amherst. Morgan Kaufmann.
- many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)
```

Show the first five records of the dataset:

```
#Check the first five records df.head()
```

```
row CRIM
            ZN
                 INDUS CHAS NOX
                                   RM
                                         AGE DIS
                                                      RAD TAX
                                                                PTRATIO B
                                                                                LSTAT target
    0.00632 18.0 2.31
                       0.0
                             0.538 6.575 65.2 4.0900 1.0 296.0 15.3
                                                                         396.90 4.98
                                                                                      24.0
                             0.469 6.421 78.9 4.9671 2.0 242.0 17.8
1
   0.02731 0.0
                 7.07
                       0.0
                                                                         396.90 9.14
                                                                                      21.6
                 7.07
                       0.0
                             0.469 7.185 61.1 4.9671 2.0 242.0 17.8
                                                                         392.83 4.03
                                                                                      34.7
3
                             0.458 6.998 45.8 6.0622 3.0 222.0 18.7
    0.03237 0.0
                 2.18
                       0.0
                                                                         394.63 2.94
                                                                                      33.4
                 2.18
                             0.458 7.147 54.2 6.0622 3.0 222.0 18.7
                                                                         396.90 5.33
                       0.0
                                                                                      36.2
```

Show summary statistics for each variable: count, mean, standard deviation, minimum, 25th 50th and 75th percentiles, and maximum.

```
#Descriptions of each variable
df.describe()
```

```
CRIM
                                                                                                      INDUS
                                                                                                                                                                                                                             RM
                                                                                                                                                                                                                                                                      AGE
stat
                                                              ZN
                                                                                                                                             CHAS
                                                                                                                                                                                     NOX
                                                                                                                                                                                                                                                                                                             DIS
                                                                                                         PTRATIO
                                                                                                                                                В
                                                                                                                                                                                         LSTAT
                         RAD
                                                                 TAX
                                                                                                                                                                                                                                 target
 count 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000 506.
  \color{red} \color{red} \color{blue} \color{blu
                    3.593761
                                                              11.363636
                                                                                                   11.136779
                                                                                                                                          0.069170
                                                                                                                                                                                      0.554695
                                                                                                                                                                                                                              6.284634
                                                                                                                                                                                                                                                                      68.574901
 →795043
                                                                                                                                                                                                                                               22.532806
                                        9.549407
                                                                                408.237154 18.455534
                                                                                                                                                               356.674032 12.653063
std
                      8.596783
                                                              23.322453 6.860353
                                                                                                                                             0.253994
                                                                                                                                                                                      0.115878
                                                                                                                                                                                                                              0.702617
                                                                                                                                                                                                                                                                      28.148861
                                        8.707259
 \hookrightarrow 105710
                                                                                168.537116 2.164946
                                                                                                                                                                91.294864
                                                                                                                                                                                                       7.141062
                                                                                                                                                                                                                                                9.197104
min
                      0.006320
                                                              0.000000
                                                                                                      0.460000
                                                                                                                                             0.000000
                                                                                                                                                                                      0.385000
                                                                                                                                                                                                                              3.561000
                                                                                                                                                                                                                                                                      2.900000
                                                                                                                                                                                                                                                                                                             1.
 →129600
                                        1.000000
                                                                                187.000000 12.600000
                                                                                                                                                               0.320000
                                                                                                                                                                                                        1.730000
                                                                                                                                                                                                                                                5.000000
                      0.082045
                                                              0.000000
                                                                                                      5.190000
                                                                                                                                             0.000000
                                                                                                                                                                                      0.449000
                                                                                                                                                                                                                              5.885500
                                                                                                                                                                                                                                                                      45.025000
                                                                                                                                                                                                                                                                                                             2.
  →100175
                                        4.000000
                                                                                279.000000 17.400000
                                                                                                                                                                375.377500 6.950000
                                                                                                                                                                                                                                                17.025000
50%
                     0.256510
                                                              0.000000
                                                                                                      9.690000
                                                                                                                                             0.000000
                                                                                                                                                                                     0.538000
                                                                                                                                                                                                                              6.208500
                                                                                                                                                                                                                                                                      77.500000
  <del>-207450</del>
                                        5.000000
                                                                                330.000000 19.050000
                                                                                                                                                                391.440000 11.360000
                                                                                                                                                                                                                                               21.200000
75%
                      3.647423
                                                              12.500000 18.100000 0.000000
                                                                                                                                                                                      0.624000
                                                                                                                                                                                                                              6.623500
                                                                                                                                                                                                                                                                      94.075000
  →188425
                                        24.000000
                                                                               666.000000 20.200000
                                                                                                                                                                396.225000 16.955000
                                                                                                                                                                                                                                               25.000000
                      88.976200
                                                             100.000000 27.740000 1.000000
                                                                                                                                                                                                                                                                      100.000000 12.
max
                                                                                                                                                                                     0.871000
                                                                                                                                                                                                                              8.780000
  →126500 24.000000 711.000000 22.000000
                                                                                                                                                           396.900000 37.970000 50.000000
```

#### **Correlation matrix**

The correlation matrix lists the correlation of each variable with each other variable.

Positive correlations mean one variable tends to be high when the other is high, and negative correlations mean one variable tends to be high when the other is low.

Correlations close to zero are weak and cause a variable to have less influence in the model, and correlations close to one or negative one are strong and cause a variable to have more influence in the model.

```
#Here shows the basic correlation matrix
corr = df.corr()
corr
```

```
DIS
variable CRIM
                      ZN
                                 INDUS
                                             CHAS
                                                        NOX
                                                                    RM
                                                                                AGE
→RAD
             TAX
                         PTRATIO B
                                                LSTAT
                                                            target
CRIM
          1.0000000 \quad -0.199458 \quad 0.404471 \quad -0.055295 \quad 0.417521 \quad -0.219940 \quad 0.350784 \quad -0.377904 \\ \_
\multimap 0.622029 0.579564 0.288250 \multimap 0.377365 0.452220 \multimap 0.385832
          -0.199458\ 1.000000\ -0.533828\ -0.042697\ -0.516604\ 0.311991\ -0.569537\ 0.664408\ {}_{\square}
\rightarrow -0.311948 -0.314563 -0.391679 0.175520 -0.412995 0.360445
          0.404471 - 0.533828 \ 1.000000 \ 0.062938 \ 0.763651 - 0.391676 \ 0.644779 - 0.708027 
TNDUS
\leftarrow 0.595129 \quad 0.720760 \quad 0.383248 \quad -0.356977 \quad 0.603800 \quad -0.483725
CHAS
          -0.055295 -0.042697 0.062938 1.000000 0.091203 0.091251 0.086518 -0.099176
\rightarrow -0.007368 -0.035587 -0.121515 0.048788 -0.053929 0.175260
          0.417521 - 0.516604 \ 0.763651 \ 0.091203 \ 1.000000 - 0.302188 \ 0.731470 - 0.769230
NOX
\multimap 0.611441 0.668023 0.188933 \multimap 0.380051 0.590879 \multimap 0.427321
          -0.219940 \ 0.311991 \ -0.391676 \ 0.091251 \ -0.302188 \ 1.000000 \ -0.240265 \ 0.205246 \ {\color{red} \sqcup}
\hookrightarrow -0.209847 -0.292048 -0.355501 0.128069 -0.613808 0.695360
          0.350784 - 0.569537 \ 0.644779 \ 0.086518 \ 0.731470 - 0.240265 \ 1.000000 - 0.747881
-0.456022 0.506456 0.261515 -0.273534 0.602339 -0.376955
DIS
          -0.377904\ 0.664408\ -0.708027\ -0.099176\ -0.769230\ 0.205246\ -0.747881\ 1.000000\ 
\rightarrow -0.494588 -0.534432 -0.232471 0.291512 -0.496996 0.249929
          0.622029 - 0.311948 \ 0.595129 - 0.007368 \ 0.611441 - 0.209847 \ 0.456022 - 0.494588
RAD
\hookrightarrow 1.000000 0.910228 0.464741 -0.444413 0.488676 -0.381626
          0.579564 - 0.314563 \ 0.720760 - 0.035587 \ 0.668023 - 0.292048 \ 0.506456 - 0.534432
\rightarrow 0.910228 1.000000 0.460853 -0.441808 0.543993 -0.468536
PTRATIO 0.288250 -0.391679 0.383248 -0.121515 0.188933 -0.355501 0.261515 -0.232471
\hookrightarrow 0.464741 0.460853 1.000000 -0.177383 0.374044 -0.507787
          -0.377365 0.175520 -0.356977 0.048788 -0.380051 0.128069 -0.273534 0.291512 _{\square}
\hookrightarrow -0.444413 -0.441808 -0.177383 1.000000 -0.366087 0.333461
          0.452220 - 0.412995 \ 0.603800 - 0.053929 \ 0.590879 - 0.613808 \ 0.602339 - 0.496996
\leftarrow 0.488676 \quad 0.543993 \quad 0.374044 \quad -0.366087 \quad 1.000000 \quad -0.737663
          -0.385832 0.360445 -0.483725 0.175260 -0.427321 0.695360 -0.376955 0.249929 _
\hookrightarrow -0.381626 -0.468536 -0.507787 0.333461 -0.737663 1.000000
```

### Format with asterisks

Format the correlation matrix by rounding the numbers to two decimal places and adding asterisks to denote statistical significance:

```
def calculate_pvalues(df):
    df = df.select_dtypes(include=['number'])
    pairs = pd.MultiIndex.from_product([df.columns, df.columns])
    pvalues = [pearsonr(df[a], df[b])[1] for a, b in pairs]
    pvalues = pd.Series(pvalues, index=pairs).unstack().round(4)
    return pvalues

# code adapted from https://stackoverflow.com/questions/25571882/pandas-columns-
    correlation-with-statistical-significance/49040342

def correlation_matrix(df,columns):
    rho = df[columns].corr()
    pval = calculate_pvalues(df[columns])
    # create three masks
```

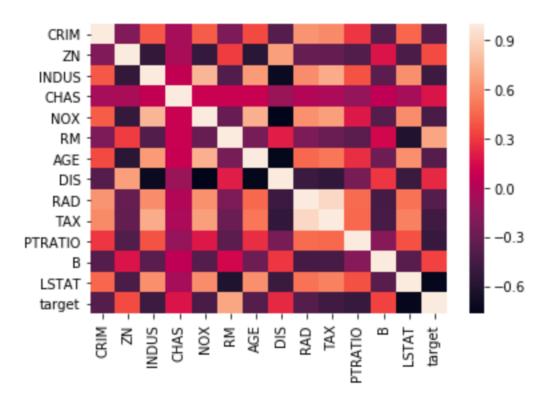
```
r0 = rho.applymap(lambda x: '{:.2f}'.format(x))
r1 = rho.applymap(lambda x: '{:.2f}*'.format(x))
r2 = rho.applymap(lambda x: '{:.2f}**'.format(x))
r3 = rho.applymap(lambda x: '{:.2f}**'.format(x))
# apply marks
rho = rho.mask(pval>0.01,r0)
rho = rho.mask(pval<=0.1,r1)
rho = rho.mask(pval<=0.05,r2)
rho = rho.mask(pval<=0.01,r3)
return rho

columns = df.columns
correlation_matrix(df,columns)</pre>
```

variable			INDUS			RM	AGE	DIS	RAD
→ TAX	PTRATI	0 B	LSTAT	targe	t				
=======			=======	======	======	=======	======	=======================================	-
·	1 00444						0.25444	0 20444	0 (2444
	1.00***					-0.22***	0.35***	-0.38***	0.62***
	-0.20***					<b>Λ</b> 21***	0 E7***	0 66***	0
	0.31*** -					0.31	-0.37	0.00	-0.
	0.40***					_N 30***	0 64***	_0 71***	0 60***
	* 0.38**					0.55	0.01	0.71	0.00
			0.06			0.09**	0.09*	-0.10**	-0.01
→ -0.04			-0.05			0.03	0.03	0110	0.01
	0.42***				1.00***	-0.30***	0.73***	-0.77***	0.61***
→ 0.67**	* 0.19**	* -0.38	*** 0.59*	** -0.43	***				_
RM	-0.22***	0.31***	-0.39***	0.09**	-0.30***	1.00***	-0.24***	0.21***	-0.
<b>→21***</b> -	0.29*** -	0.36***	0.13***	-0.61***	0.70***				
AGE	0.35***	-0.57***	0.64***	0.09*	0.73***	-0.24***	1.00***	-0.75***	0.46***
→ 0.51**	* 0.26**	* -0.27	*** 0.60*	** -0.38	***				
220	-0.38***		0112	0.120	0	0.21***	-0.75***	1.00***	-0.
, 15	0.53*** -	00	0.20	0.00	0.20				
	0.62***					-0.21***	0.46***	-0.49***	1.00***
	* 0.46**								
	0.58***					-0.29***	0.51***	-0.53***	0.91***
	* 0.46**					0 20444	0 20444	0 22444	0.46444
	0.29***					-0.36***	0.26***	-0.23***	0.46***
	* 1.00** -0.38***					0.13***	0 27***	N 20***	•
_	0.44*** -					0.15	-0.27	0.29	-0.
,	0.44***	0.20		0.0.	0.00	_0 61***	0 60***	_0 50***	0 40***
	* 0.37**					0.01	<b>U.</b> UU	<b>0.</b> Ju	V:43 L
	-0.39***					0.70***	-0.38***	0.25***	-0.
-	0.47*** -					0110	3.30	0.23	•
, , , ,									

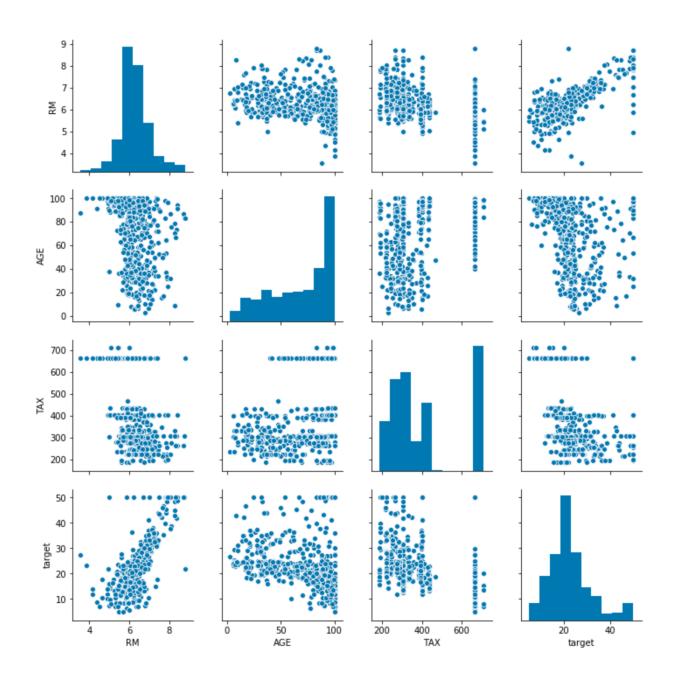
# Heatmap

Heatmap of the correlation matrix:



## Pairwise distributions with seaborn

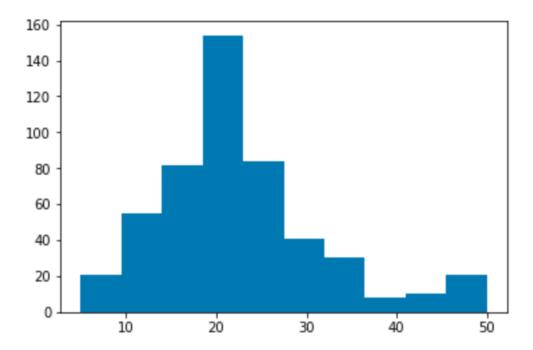
```
sns.pairplot(df[['RM', 'AGE', 'TAX', 'target']])
```



# **Target variable distribution**

Histogram showing the distribution of the target variable. In this dataset this is "Median value of owner-occupied homes in 1000's", abbreviated MEDV.

```
plt.hist(df['target'])
plt.show()
```



### Simple linear regression

The variable MEDV is the target that the model predicts. All other variables are used as predictors, also called features.

The target variable is continuous, so use a linear regression instead of a logistic regression.

```
# Define features as X, target as y.
X = df.drop('target', axis='columns')
y = df['target']
```

Split the dataset into a training set and a test set:

A linear regression consists of a coefficient for each feature and one intercept.

To make a prediction, each feature is multiplied by its coefficient. The intercept and all of these products are added together. This sum is the predicted value of the target variable.

The residual sum of squares (RSS) is calculated to measure the difference between the prediction and the actual value of the target variable.

The function fit calculates the coefficients and intercept that minimize the RSS when the regression is used on each record in the training set.

```
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
# The intercept
```

```
Intercept:
36.98045533762056
Coefficients:
CRIM
            -0.116870
ZN
            0.043994
INDUS
          -0.005348
CHAS
           2.394554
NOX
         -15.629837
RM
            3.761455
AGE
           -0.006950
DIS
           -1.435205
RAD
           0.239756
TAX
           -0.011294
PTRATIO
           -0.986626
           0.008557
LSTAT
           -0.500029
Name: coefficients, dtype: float64
```

Now check the accuracy when this linear regression is used on new data that it was not trained on. That new data is the test set.



This scatter plot shows that the regression is a good predictor of the data in the test set.

The mean squared error quantifies this performance:

```
# The mean squared error as a way to measure model performance.
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
```

```
Mean squared error: 29.79
```

### Ordinary least squares (OLS) regression with Statsmodels

OLS Regression Resu	ılts		
Dep. Variable:	======================================	 R-squared:	0.741
Model:	OLS	Adj. R-squared:	0.734
Method:	Least Squares	F-statistic:	108.1
Date:	Thu, 23 Aug 2018	<pre>Prob (F-statistic):</pre>	6.95e-135
Time:	07:29:16	Log-Likelihood:	-1498.8
No. Observations:	506	AIC:	3026.
Df Residuals:	492	BIC:	3085.
Df Model:	13		
Covariance Type:	nonrobust		

/	1.0		,
(confini	ied fron	ı previous	nage
(COIITIII)	ica mon	i pievious	page,

coef s	td err				0.975]	
Intercept	36.4911					
AGE	0.0008	0.013	0.057	0.955	-0.025	0.027
В	0.0094	0.003	3.500	0.001	0.004	0.015
CHAS	2.6886	0.862	3.120	0.002	0.996	4.381
CRIM	-0.1072	0.033	-3.276	0.001	-0.171	-0.043
DIS	-1.4758	0.199	-7.398	0.000	-1.868	-1.084
INDUS	0.0209	0.061	0.339	0.735	-0.100	0.142
LSTAT	-0.5255	0.051	-10.366	0.000	-0.625	-0.426
NOX	-17.7958	3.821	-4.658	0.000	-25.302	-10.289
PTRATIO	-0.9535	0.131	-7.287	0.000	-1.211	-0.696
RAD	0.3057	0.066	4.608	0.000	0.175	0.436
RM	3.8048	0.418	9.102	0.000	2.983	4.626
TAX	-0.0123	0.004	-3.278	0.001	-0.020	-0.005
ZN	0.0464	0.014	3.380	0.001	0.019	0.073
mnibus:		178.0				1.078
Prob(Omnil	ous):	0.0	000 Jarqı	ue-Bera (JB	3):	782.015
Skew:		1.	521 Prob	(JB):		1.54e-170
Kurtosis:		8.2		No.		1.51e+ <b>0</b> 4

# Principal component analysis

The initial dataset has a number of feature or predictor variables and one target variable to predict.

strong multicollinearity or other numerical problems.

Principal component analysis (PCA) converts these features into a set of principal components, which are linearly uncorrelated variables.

The first principal component has the largest possible variance and therefore accounts for as much of the variability in the data as possible.

Each of the other principal components is orthogonal to all of its preceding components, but has the largest possible variance within that constraint.

Graphing a dataset by showing only the first two or three of the principal components effectively projects a complex dataset with high dimensionality into a simpler image that shows as much of the variance in the data as possible.

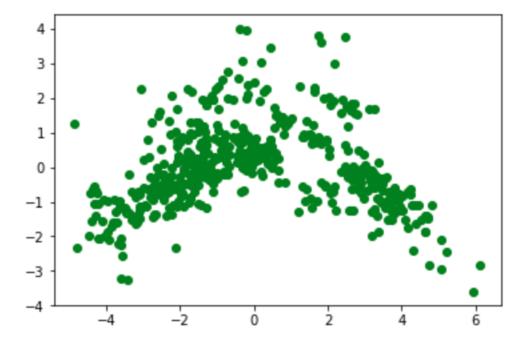
PCA is sensitive to the relative scaling of the original variables, so begin by scaling them:

```
# Feature Scaling
x = StandardScaler().fit_transform(X)
```

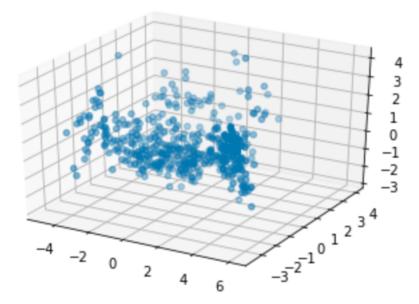
Calculate the first three principal components and show them for the first five rows of the housing dataset:

row	principal component 1	principal component 2	principal component 3
0	-2.097842	0.777102	0.335076
1	-1.456412	0.588088	-0.701340
2	-2.074152	0.602185	0.161234
3	-2.611332	-0.005981	-0.101940
4	-2.457972	0.098860	-0.077893

# Show a 2D graph of this data:



# Show a 3D graph of this data:



Measure how much of the variance is explained by each of the three components:

```
# Variance explained by each component
explained_variance = pca.explained_variance_ratio_
explained_variance
```

```
array([0.47097344, 0.11015872, 0.09547408])
```

Each value will be less than or equal to the previous value, and each value will be in the range from 0 through 1.

The sum of these three values shows the fraction of the total variance explained by the three principal components, in the range from 0 (none) through 1 (all):

```
sum(explained_variance)
```

```
0.6766062376563704
```

Predict the target variable using only the three principal components:

Plot the predictions from the linear regression in green again, and the new predictions in blue:

```
fig, ax = plt.subplots()
ax.scatter(y_test, y_pred, color='skyblue')
ax.scatter(y_test_linear, y_pred_linear, color = 'green')
```

```
ax.set(
    xlabel="Prices: $Y_i$",
    ylabel="Predicted prices: $\hat{Y}_i$",
    title="Prices vs Predicted prices: $Y_i$ vs $\hat{Y}_i$",
)
plt.show()
```



The blue points are somewhat more widely scattered, but similar.

Calculate the mean squared error:

```
Linear regression mean squared error: 29.79
PCA mean squared error: 43.49
```

# 4.6 Deployments

Deployments are methods for sharing parts of your data science projects and machine learning models with other users. You can create deployments for microservices such as interactive dashboard visualizations, web applications, REST APIs, or Jupyter Notebooks.

When you deploy a project, Data Science & AI Workbench finds and builds all of the software dependencies (the packages and libraries the project needs to operate properly) and encapsulates them into a self-contained artifact.

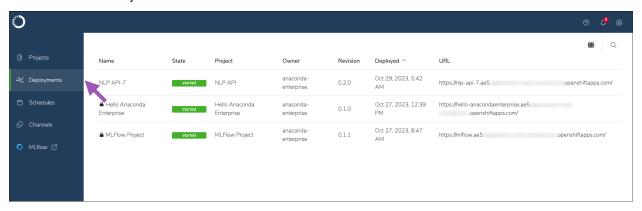
You can create multiple deployments from a single project. Each deployment can be a different version and can be shared with others.

# 4.6.1 Deployment commands

Deployments are defined as commands in your anaconda-project.yml file. Each project in Workbench can define any number of deployable commands. See *Adding deployment commands to a project* for more information.

# 4.6.2 Viewing all deployments

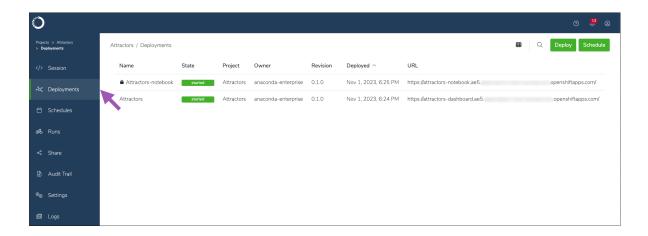
Select **Deployments** from the left-hand navigation to view a list of deployments across all projects you own or that have been shared with you.



# 4.6.3 Viewing a project's deployments

To view the deployments specific to a project:

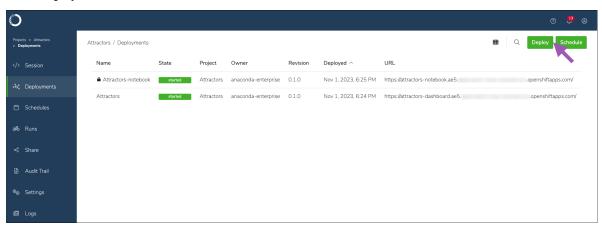
- 1. From the Projects page, select your project.
- 2. Select **Deployments** from the left-hand navigation.



# 4.6.4 Deploying a project

Deployments are run independently of development sessions for projects, and can be deployed to different hardware resource profiles.

- 1. From the Projects page, select the project you want to deploy.
- 2. Select **Deployments** from the left-hand menu.
- 3. Click Deploy.



- 4. Enter a name for your deployment.
- 5. Select a resource profile to run your deployment.

**Note:** The configurations present in this dropdown are controlled by your Administrator. Check with them if you are unsure which profile to choose.

- 6. Select the project version associated with the deployment command you are running.
- 7. Select a deployment command.

Caution: If there is no deployment command listed, you cannot deploy your project.

8. Set the URL for your deployment.

**Note:** This is the URL you'll use to access your deployment from a web browser, and therefore *it must be unique*. Toggle **Static URL** 0FF if you want Workbench to automatically generate a URL for your deployment.

- 9. Share your deployment with any other users or groups.
- 10. Set your deployment to private or public access:

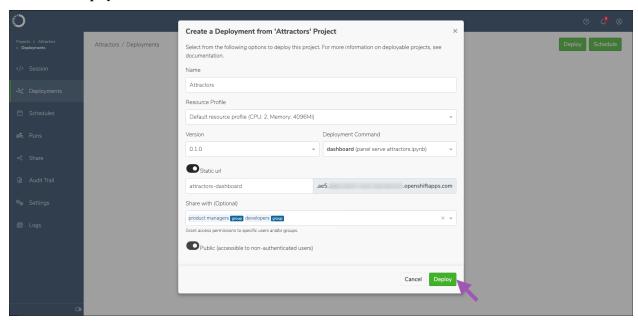
#### **Private access**

Restricts access to authenticated platform users only. Private deployments display a lock icon to indicate their secure status.

#### **Public access**

Opens access to non-authenticated users.

#### 11. Click Deploy.

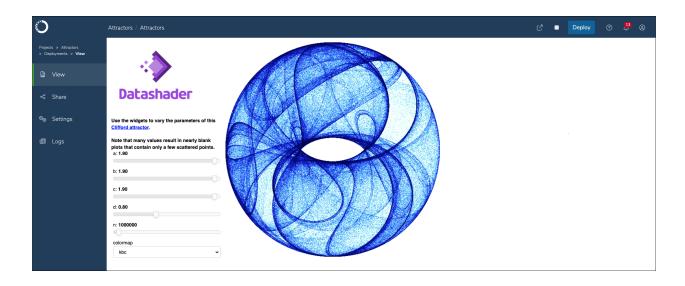


Note: It can take several minutes to obtain and build all the dependencies for the project deployment.

You can also schedule a project to be deployed on a regular basis or at a specific time. For more information, see *Scheduling deployments*.

# 4.6.5 Interacting with deployments

To interact with your deployment, select it from the list of project deployments, or navigate to its URL in a web browser.



# **Deploying a REST API**

Data Science & AI Workbench enables you to deploy your machine learning or predictive models as a REST API endpoint. You can then share your deployment with colleagues and have them query your model as you continue to update, improve, and redeploy as needed.

There are many tools available in the open-source market for building REST APIs within the Python and R ecosystems. This topic explains how to generate a REST API endpoint from a python function you've created within a Jupyter Notebook.

**Note:** REST API endpoints deployed using Workbench are secure and only accessible to users that you've shared the deployment with or users that have generated a token that can be used to query the REST API endpoint outside of Workbench.

#### **Install Tranquilizer**

Open a session and run the following command to install Tranquilizer:

anaconda-project add-packages tranquilizer nbconvert

There are three essential skills you must understand in order to use Tranquilizer effectively:

- Functions Defined sections of code that perform specific tasks.
- Type Hints A method to statically indicate the type of a value within your Python code.
- **Docstrings** Comments in the code that document what a given function does.

Note: Tranquilizer reads the function's docstring to automatically generate Swagger documentation for the REST API.

### **Decorate your function**

Decorated functions are identified by the Tranquilizer server. These functions are executed when HTTP requests are made to the server.

Here is an example of a temperature conversion function with the tranquilizer decoration:

```
from tranquilizer import tranquilize

@tranquilize(method='get')
def to_celsius(fahrenheit: float):
    "Convert degrees Fahrenheit to degrees Celsius

    :param fahrenheit: Degrees fahrenheit"'

return (fahrenheit - 32) * 5/9
```

**Note:** When the Tranquilizer server starts, the entire Jupyter Notebook is executed. Consider carefully how you arrange operations that are heavy on CPU or memory usage, like reading data or loading models. Computations for reusable objects should be performed only once, rather than every time the function is called.

#### Add the command

With tranquilizer downloaded and your function decorated, you must now prepare the deployment command. Open your anaconda-project.yml file and add the following code:

```
commands:
    celsius_api:
    unix: tranquilizer converter.ipynb --name "Temperature Conversion"
    supports_http_options: True
```

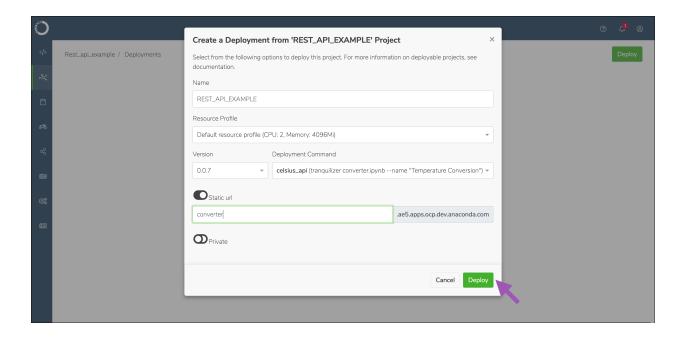
Caution: Make sure your notebook file name is spelled correctly! If it is not, your commands will not work.

Alternatively, you can run the following command in the terminal to add the command to the anaconda-project.yml file:

```
anaconda-project add-command --type unix --supports-http-options celsius_api
→"tranquilizer converter.ipynb --name 'Temperature Conversion'"
```

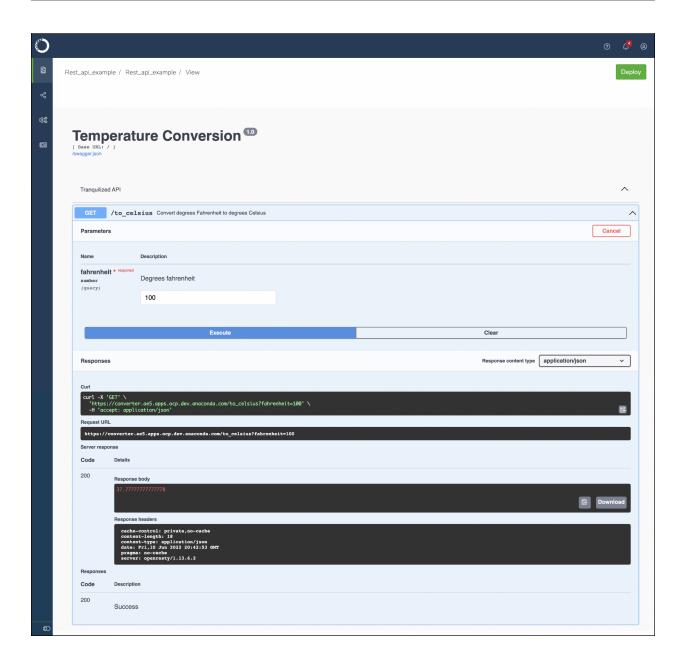
### Commit and deploy

Commit all changes to your project and deploy the command. *Choose a unique URL for your deployment*.



# **Verify deployment**

Once the deployment has started, go to the deployments **View** page and use swagger to verify that the API performs the correct function.



# Deploy a REST API with R

You can use the R Plumber package to build and deploy a REST API in R.

Run the following command to install R Plumber:

```
anaconda-project add-packages -c r r-plumber
```

The API is defined in a file called api.R

```
library(plumber)

#* @apiTitle Temperature Converter
```

(continues on next page)

(continued from previous page)

```
#* Convert degrees Fahrenheit to Celsius
#* @param fahrenheit:numeric Degrees Fahrenheit
#* @get /to_celsius
function(fahrenheit) {
    (as.numeric(fahrenheit) - 32) * 5/9
}

#* Redirect to Swagger documentation
#* @get /
function(res) {
    res$body <- "<html><meta http-equiv='refresh' content=\"0; URL='/__swagger__/'\" /></
    html>"
    res
}
```

To separate your code into api.r and run.R scripts, add your command specification in the anaconda-project.yml file:

```
commands:
api:
unix: Rscript run.R
```

Then format your run.R script as follows:

```
plumber::plumb("api.R")$run(port = 8086, host = '0.0.0.0', swagger = TRUE)
```

Alternatively, you can place the contents of the the run.R script directly in the anaconda-project.yml file:

```
commands:
    api:
    unix: R -e 'plumber::plumb("api.R")$run(port = 8086, host = "0.0.0.0", swagger = □
    →TRUE)'
```

### Deploying a Flask application

The process of deploying a Flask application (website and REST APIs) on Data Science & AI Workbench involves the following:

- 1. Configuring Flask to run behind a proxy
- 2. Enabling Anaconda Project HTTP command-line arguments
- 3. Running Flask on the deployed host and port

Here is a small Flask application that includes the call to .run(). The file is saved to server.py.

This Flask application was written using Blueprints, which is useful for separating components when working with a large Flask application.

Here, the nested block in if \_\_name\_\_ == '\_\_main\_\_' could be in a separate file from the 'hello' Blueprint.

```
from flask import Flask, Blueprint
hello = Blueprint('hello', __name__)
```

4.6. Deployments 327

(continues on next page)

(continued from previous page)

```
@hello.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app = Flask(__name__)
    app.register_blueprint(hello, url_prefix='/')

app.run()
```

### Running behind an HTTPS proxy

Workbench maintains all HTTPS connections into and out of the server and deployed instances. When writing a Flask app, you only need to inform it that will be accessed from behind the proxy provided by Workbench.

The simplest way to do this is with the ProxyFix function from werkzeug. More information about proxies is provided here.

```
from flask import Flask, Blueprint
from werkzeug.contrib.fixers import ProxyFix

hello = Blueprint('hello', __name__)

@hello.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app = Flask(__name__)
    app.register_blueprint(hello, url_prefix='/')

app.wsgi_app = ProxyFix(app.wsgi_app)
    app.run()
```

#### **Enabling command-line arguments**

In your anaconda-project.yml file, you define a deployable command as follows:

```
commands:
    default:
    unix: python ${PROJECT_DIR}/server.py
    supports_http_options: true
```

The flag supports\_http\_options means that server.py is expected to act on the following command line arguments defined in the Anaconda Project Reference.

This is easily accomplished by adding the following argparse code before calling app.run() in server.py

```
import sys
from argparse import ArgumentParser
```

(continues on next page)

(continued from previous page)

```
# ... the Flask application blueprint
if __name__ == '__main__':
    # arg parser for the standard anaconda-project options
   parser = ArgumentParser(prog="hello_world",
                            description="Simple Flask Application")
   parser.add_argument('--anaconda-project-host', action='append', default=[],
                        help='Hostname to allow in requests')
   parser.add_argument('--anaconda-project-port', action='store', default=8086,_
→type=int,
                        help='Port to listen on')
   parser.add_argument('--anaconda-project-iframe-hosts',
                        action='append',
                        help='Space-separated hosts which can embed us in an iframe per_
→our Content-Security-Policy')
    parser.add_argument('--anaconda-project-no-browser', action='store_true',
                        default=False.
                        help='Disable opening in a browser')
   parser.add_argument('--anaconda-project-use-xheaders',
                        action='store_true'.
                        default=False.
                        help='Trust X-headers from reverse proxy')
   parser.add_argument('--anaconda-project-url-prefix', action='store', default='',
                        help='Prefix in front of urls')
   parser.add_argument('--anaconda-project-address',
                        action='store',
                        default='0.0.0.0',
                        help='IP address the application should listen on.')
    args = parser.parse_args()
```

### **Running your Flask application**

The final step is to configure the Flask application with the Anaconda Project HTTP values and call app.run(). Note that registering the Blueprint provides a convenient way to deploy your application without having to rewrite the routes.

```
# ... the flask application blueprint

if __name__ == '__main__':
    # ... parse command line arguments

app = Flask(__name__)
    app.register_blueprint(hello, url_prefix=args.anaconda_project_url_prefix)

app.config['PREFERRED_URL_SCHEME'] = 'https'

app.wsgi_app = ProxyFix(app.wsgi_app)
    app.run(host=args.anaconda_project_address, port=args.anaconda_project_port)
```

Here is the complete code for the Hello World application.

```
import svs
from flask import Flask, Blueprint
from argparse import ArgumentParser
from werkzeug.contrib.fixers import ProxyFix
hello = Blueprint('hello', __name__)
@hello.route('/')
def hello_world():
   return "Hello, World!"
if __name__ == '__main__':
    # arg parser for the standard anaconda-project options
   parser = ArgumentParser(prog="hello_world",
                            description="Simple Flask Application")
   parser.add_argument('--anaconda-project-host', action='append', default=[],
                        help='Hostname to allow in requests')
   parser.add_argument('--anaconda-project-port', action='store', default=8086,
→type=int,
                        help='Port to listen on')
   parser.add_argument('--anaconda-project-iframe-hosts',
                        action='append',
                        help='Space-separated hosts which can embed us in an iframe per_
→our Content-Security-Policy')
   parser.add_argument('--anaconda-project-no-browser', action='store_true',
                        default=False,
                        help='Disable opening in a browser')
   parser.add_argument('--anaconda-project-use-xheaders',
                        action='store_true',
                        default=False,
                        help='Trust X-headers from reverse proxy')
   parser.add_argument('--anaconda-project-url-prefix', action='store', default='',
                        help='Prefix in front of urls')
   parser.add_argument('--anaconda-project-address',
                        action='store',
                        default='0.0.0.0',
                        help='IP address the application should listen on.')
   args = parser.parse_args()
    app = Flask(__name__)
    app.register_blueprint(hello, url_prefix = args.anaconda_project_url_prefix)
    app.config['PREFERRED_URL_SCHEME'] = 'https'
   app.wsgi_app = ProxyFix(app.wsgi_app)
    app.run(host=args.anaconda_project_address, port=args.anaconda_project_port)
```

### **Sharing deployments**

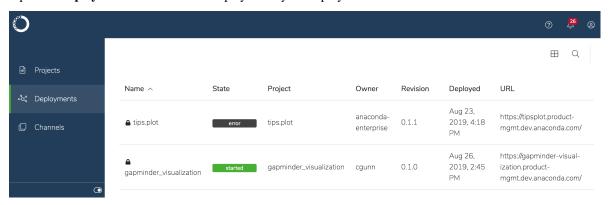
After you have *deployed a project*, you can *share* the deployment with others. You can share a deployment publicly, with other Data Science & AI Workbench users, or both.

Any *collaborators* you add to your deployment will see your deployment in *their* **Deployments** list when they log in to Workbench.

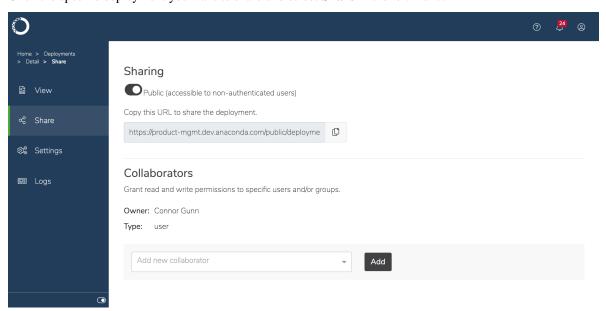
**Note:** Your Workbench Administrator creates the users and groups with whom you can share your deployments, so check with them if you need a new group created.

### To share a deployment:

1. If you're already working with the associated project, select **Deployments** in the left menu. Otherwise, click the top-level **Deployments** menu item to display all of your deployments.



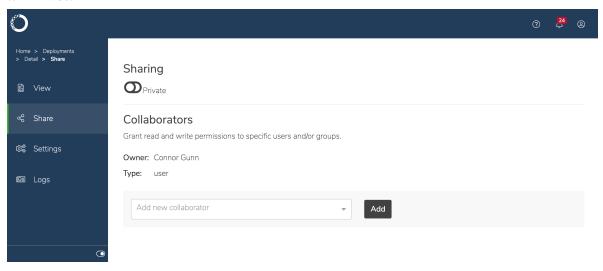
2. Click the specific deployment you want to share and select **Share** in the left menu.



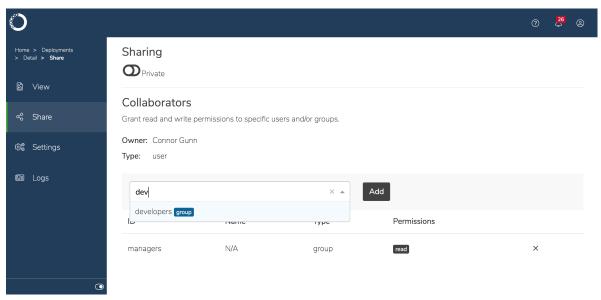
3. The deployment is Public, or accessible to any one who has the deployment URL, by default. You can copy and distribute the unique URL that's generated when you *deploy the project* to others with whom you want to share the deployment.

**Note:** If the deployment is going to be used as an endpoint that's called by other code (e.g., a REST API), you'll want to **provide a static URL** when *deploying the project*, and NOT use the generated URL displayed here. For more information, see *the instructions below*.

To limit access to the deployment to only those users with an access token, enable the Public toggle so it switches to Private.



4. To share the deployment with other users of the Workbench platform, start typing the name of the user or group in the **Collaborators** drop-down to search for matches. Select the one that corresponds to what you want, and click **Add**.



To remove collaborator access to a deployment, check the  $\mathbf{X}$  next to the user or group you want to remove as collaborators and click **Remove** to confirm your selection.

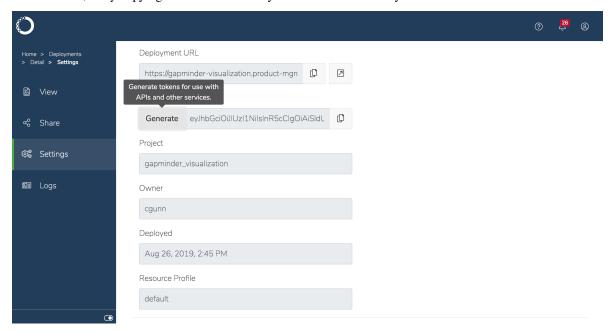
### To enable others to reference a deployment from within their code:

Rather than sharing your model with other data scientists and having them run it, you can give them an endpoint to query the model, which you can continue to update, improve and redeploy as needed.

**Note:** If the deployment is going to be used as an endpoint that's called by other code, you'll want to **provide a static URL** when *deploying the project*, and NOT use an auto-generated URL.

If your deployment is Private, you'll also need to *generate a token* that can be used to connect to the associated Notebooks, APIs or other running code. People will need both the deployment URL and the token to access a private deployment. **Tokens are powerful and should be protected like passwords.** 

- 1. Click the deployment you want to generate a token for and select **Settings** in the left menu.
- 2. Scroll to the **Generate Tokens** setting and click **Generate**. Copy the token that's generated to the clipboard with the icon, or by copying it with mouse or keyboard shortcuts like any other text.



You can then share this token, and the **Deployment URL**, with others to enable them to connect to the deployment from within Notebooks, APIs and other running code.

To remove a deployment from the server—thereby making it unavailable to yourself and others—you *terminate the deployment*. This also frees up its resources.

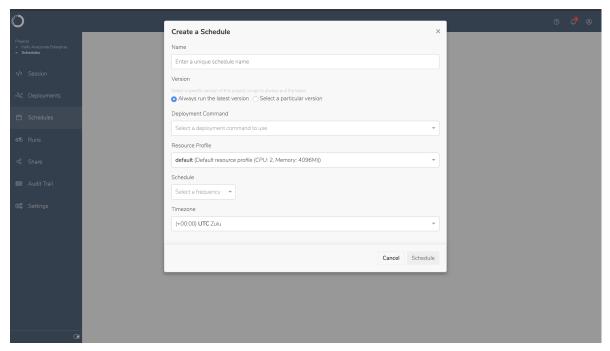
#### Scheduling deployments

If you want to *deploy a project* on a regular basis, Data Science & AI Workbench enables you to schedule the deployment. For example, you can schedule a deployment that's resource intensive to run after regular business hours, or to import new data on a weekly basis.

**Note:** A task that's run via a scheduled deployment can read data previously committed to the project from an editor session, but cannot be used to commit any new data to it. Any data written to a scheduled deployment's container will be deleted immediately after the scheduled task runs, so Anaconda recommends that you ensure data is read from and written to *external data sources*.

#### To schedule a deployment:

- 1. Open the project you want to schedule a deployment for by clicking on it in the **Projects** list.
- 2. Click **Schedules** in the menu on the left.
- 3. Click **Create a Schedule** if it's the first schedule to be created for the project, or the Schedule button if there are existing schedules.



- 4. Give the schedule a meaningful name to help differentiate it from any other schedules.
- 5. Specify whether you want to deploy the latest version of the project, or select a particular version.
- 6. Specify the **Deployment Command** to use to deploy the project. Schedules are intended for automatic or non-interactive execution of script files or notebooks, therefore only unix: commands are supported. See an example here.

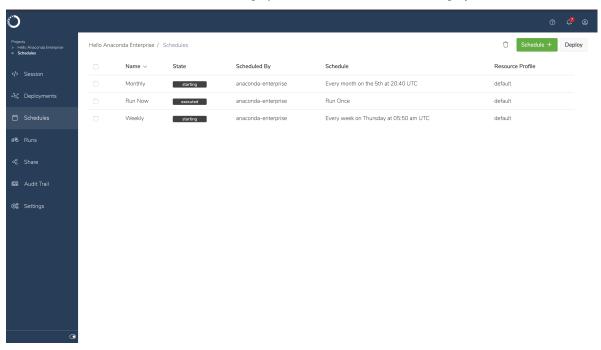
**Note:** If there is no deployment command listed, you cannot deploy the project. Return to the project and add a deployment command, or ask the project owner to do so if it's not your project. See *Configuring project settings* for more information about adding deployment commands.

- 7. Choose the runtime resources your project requires to run from the **Resource Profile** drop-down, or accept the default. Your Administrator configures the options in this list, so check with them if you aren't sure.
- 8. Use the controls to specify how often and when you want to schedule the deployment, or select **Custom** and enter a valid cron expression. To help ensure your schedule runs when you intend it to, Anaconda recommends you verify your cron expression before saving your schedule.

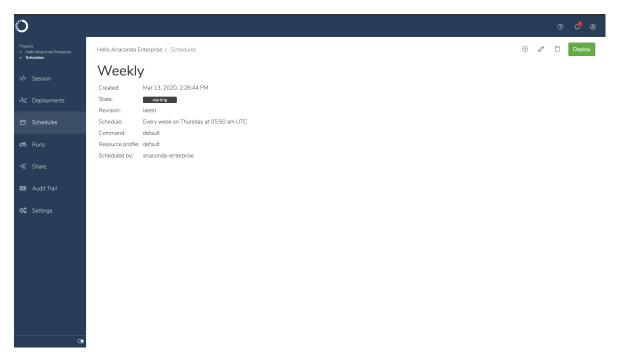
**Note:** All scheduled times are in UTC (Coordinated Universal Time).

Alternatively, if you want it to run now—instead of scheduling it—select Run Now.

9. Click **Schedule** to create the schedule, and display it in the list of schedules for the project.



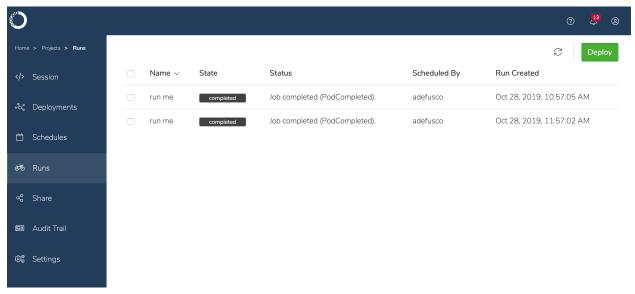
10. Click on a schedule in the list to view and edit its details.



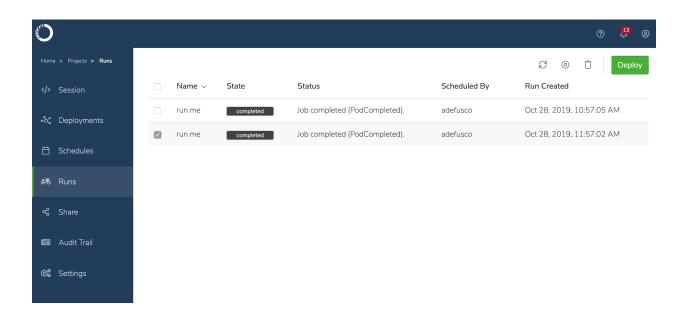
11. Use the controls above the schedule to pause, edit, or delete a selected schedule.

**Note:** If you attempt to delete a schedule that is currently running or is scheduled to run, you will be prompted to confirm that you want to force the deletion.

To view a list of all the scheduled deployments that are currently running or have already run, click **Runs** in the menu on the left.



Select a specific run in the list to enable the controls to refresh, stop or delete it.

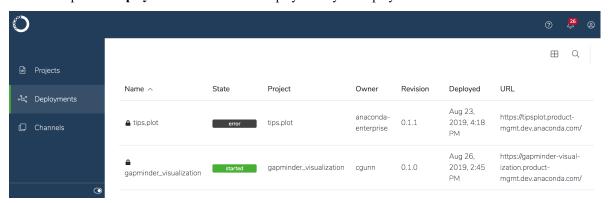


### Terminating a deployment

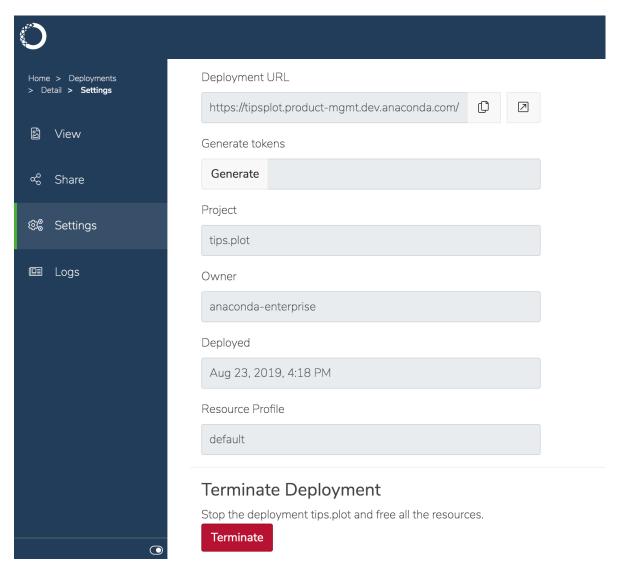
When a deployment is no longer required, you can terminate it to stop it from running. This will remove it from the server and free up the resources it's currently using. Terminating a deployment does not affect the original *project* from which the deployment was created—only the deployment. It does make the deployment unavailable to *any users you had shared it with*, however.

#### To terminate a deployment:

1. Click the top-level **Deployments** menu item to display all of your deployments.



2. Click the specific deployment you want to terminate, and click **Settings** in the menu on the left.



- 3. Scroll down until the **Terminate** button is visible, and click it.
- 4. Confirm that you want to stop the deployment. The deployment stops, and is removed from the list of deployments.

### **Deployment logs**

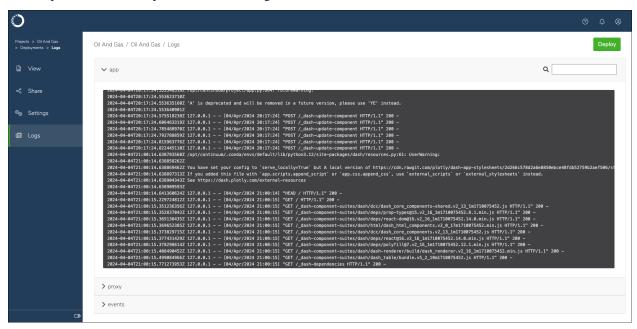
When you "deploy" your project, Data Science & AI Workbench creates various Kubernetes resources, including a Kubernetes Deployment, which manages the lifecycle of pods—including their creation, scaling, and updating as necessary. Each project deployment is created as a Kubernetes pod.

Each pod has an app and proxy container, and logs for each container are accessible to assist you in troubleshooting issues that can occur within your project's deployment. These logs can provide insights into the operational health of your deployment, possible configuration errors, and detecting potential security issues. You can also access Kubernetes event logs to aid you in troubleshooting your cluster.

## Viewing deployment logs

To view a deployment's logs:

- 1. Log in to Workbench.
- 2. Open a project.
- 3. Select **Deployments** from the left-hand navigation.
- 4. If necessary, deploy your project.
- 5. Select your project deployment.
- 6. Select **Logs** from the left-hand navigation.
- 7. Open a container dropdown to view the logs for that container.



Tip: Use the search function to help you efficiently locate information in the container logs.

## **App container logs**

The app container is where your project application code runs. Logs from this container can provide insights into the following:

- Standard output (stdout) and standard error (stderr) from your application
- Error messages and stack traces when exceptions occur
- Information related to the applications resource usage and processing times

### **Proxy container logs**

The proxy container handles network traffic to the application from both the internet and from other internal Kubernetes services. Logs from this container can provide insights into the following:

- Records of incoming and outgoing requests, including timestamps, HTTP status codes, URLs, and client IP addresses
- Records of unauthorized access attempts to help identify and address irregularities in network traffic
- Information about network-related errors, such as failed connections, timeouts, or DNS issues

### **Events logs**

The events logs contain actions taken by Kubernetes as it schedules pods for creation on the node (as they consume cluster resources), pulls the image, and then starts each of the pod's app and proxy containers. These logs provide insights into the following:

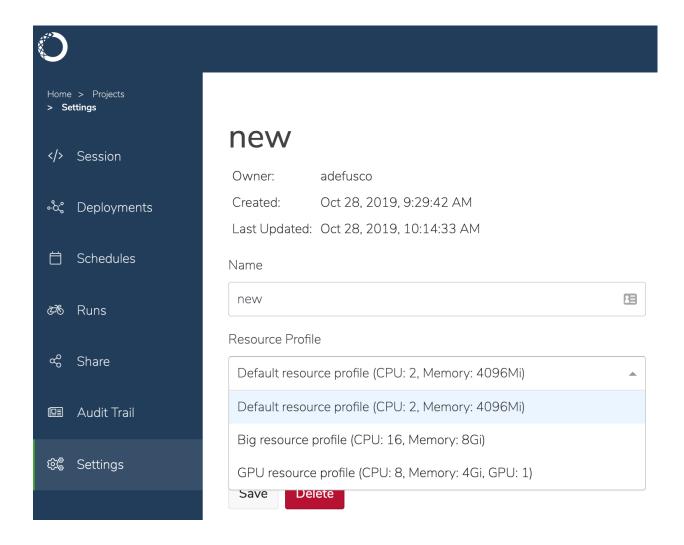
- · Information about the automatic scaling actions taken by Kubernetes in response to workload demands
- Records of new version deployments, configuration updates, and their success or failure
- Results of health-check events to ensure your application is healthy

# 4.7 Using GPUs in sessions and deployments

Data Science & AI Workbench enables you to leverage the compute power of graphics processing units (GPUs) from within your editor sessions. To do so, you can select a resource profile that features a GPU when you first create the project, or use the project's **Settings** tab to select a resource profile after the project is created.

To enable access to a GPU while running a *deployed application*, select the appropriate resource profile when you *deploy the associated project*.

In either case, if the resource profile you need isn't listed, ask your Administrator to configure one for you to use.



# 4.8 Configuring your user settings

Data Science & AI Workbench maintains settings related to your user account, based on how the system was configured by your Administrator. There are times when you may need to update the information related to your user account—to change your password, *add credentials required to access a version control repository*, or *add secrets* that can be used to access file systems, data stores and other resources implemented by your organization, for example.

To access your account settings, click the **User** icon in the upper-right corner and select the **Settings** option in the pull-down.

Click **Advanced Settings** to configure the following settings for your Workbench account:

- To change the email or name associated with your account, edit the associated field for the Account.
- To change the password you use to log in to Workbench, select **Password**.
- To enable two-factor authentication for your account, select Authenticator.
- To view a history of your sessions using Workbench, select **Sessions**. You can also log out of all sessions in one click here.

- To view a list of Workbench applications currently running and the permissions you have been granted, select Applications.
- To view a log of all activity related to your account, select **Log**.

**Note:** Fields that you are not permitted to edit appear grayed / disabled.

# 4.8.1 Configuring access to version control

If your Administrator has configured Workbench to use a supported version control repository other than the internal GitHub server, you'll need to provide your credentials to be able to access that repository. *Anaconda recommends you create an ever-lasting token, so you can retain permanent access to your files from within Workbench.* 

Your auth token must also have the following permissions:

External Repository	Permissions Required
Bitbucket Enterprise	Admin access for Projects and Repositories
GitHub Enterprise	repo:status, repo_deployment, public_repo, repo:invite, and delete_repo
GitLab Enterprise	Check the api access check box when creating your access token.

**Note:** You'll be prompted to configure your personal access token when you attempt to create your first project in Workbench, if you haven't already done so.

- 1. Under External Version Control Credentials, click Add.
- 2. Enter the username and personal access token you use to access the repository in the relevant fields.
- 3. Click **Add** to update the platform with your credentials.

To manage credentials that you've added, click on the command menu for the credentials, then choose whether you want to edit or delete them.

Now that you've configured access, you'll be able access the repository within your sessions and deployments without having to leave the platform. Workbench creates a repository for each *project that you create*.

### 4.8.2 Storing secrets

Workbench enables you to securely store information such as user names, passwords, API keys, or authentication tokens. Any secrets you add will be available across sessions and deployments for all projects associated with your account—but the values are not shared with other users.

Secrets are mounted into deployments and sessions as files, where the name of the file matches the name of the secret. Each file stores the value provided for that secret. You can access the contents of these files from within your projects, to access file systems, data stores and other resources implemented by your organization.

**Note:** We highly recommend you use the secrets store over including credentials in your project, due to the potential security risk associated with storing them in version control.

1. Under Secrets, click Add.

2. Enter a Name and Value for the secrets you want to store, then click Add.

Note: Secret names can contain alphanumeric characters and underscores only—not special characters or paths.

Any secrets you add are listed by name. To manage your secrets, click on the command menu choose whether you want to edit, delete or copy the name of the secret.

### To access credentials you've added within a session, deployment, or scheduled job:

- 1. Open a new terminal window.
- 2. Change directory to the location where the secrets are stored: /var/run/secrets/user\_credentials/.
- 3. Run cat <credential\_key>—replacing credential\_key with the actual key name—to display the value you entered when you added the secret.
- 4. Use the value to access the file system, data store or other resource as needed. See *Loading data* for more information.

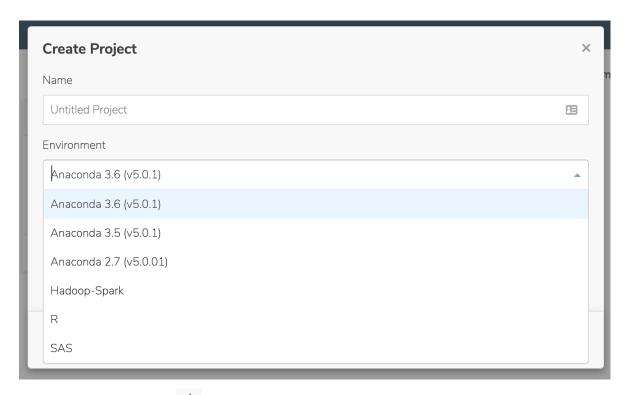
# 4.9 Visualizations and dashboards

Data Science & AI Workbench makes it easy for you to create and share interactive data visualizations, live notebooks or machine learning models built using popular libraries such as Bokeh and HoloViews.

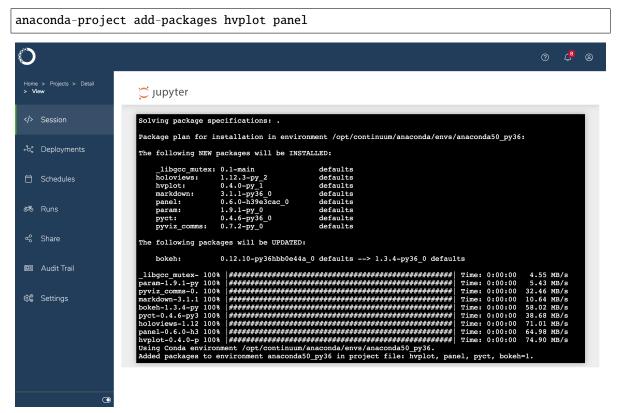
To get you started quickly, Workbench provides sample projects of Bokeh applications for clustering and cross filtering data. There are also several examples of Workbench projects that use HoloViz here.

Follow these steps to create an interactive plot:

1. From the **Projects** view, select **Create + > New Project** and create a project from the **Anaconda** 3.6 (v5.0.1) template:



2. Open the project in a session , select **New > Terminal** to open a terminal, and run the following command to install packages for hvplot and panel:



3. Select **New > Python 3** to create a new notebook, rename it tips.ipynb, and add the following code to create an interactive plot:

```
import pandas as pd
import hvplot.pandas
import panel

panel.extension()

df = pd.read_csv('http://bit.ly/tips-csv')
p = df.hvplot.scatter(x='total_bill', y='tip', hover_cols=['sex','day','size'])
pn.panel(p).servable()
```

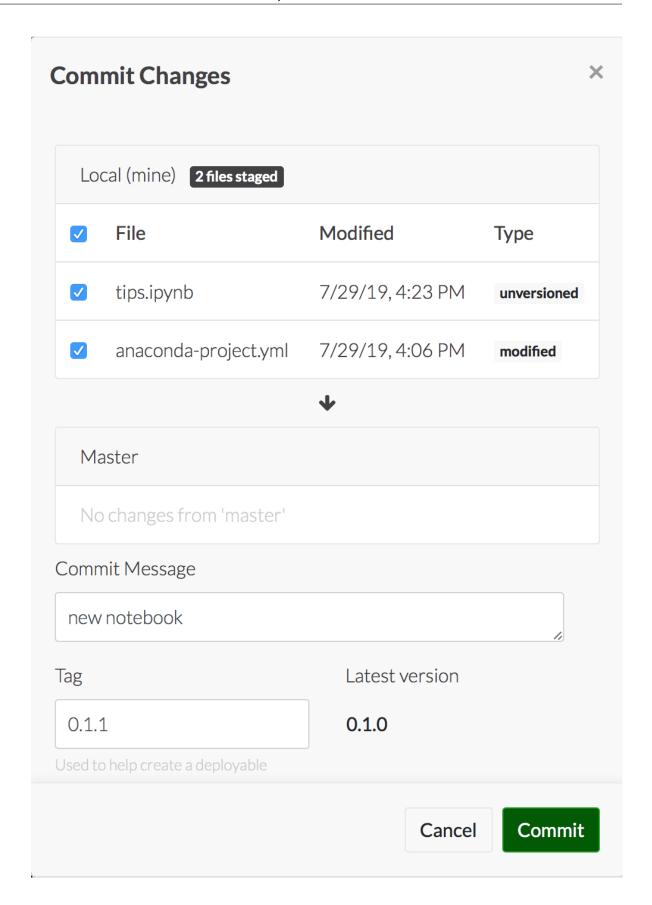
**Note:** In this example, the data is being read from the Internet. Alternatively, you could download the .csv and upload it to the project.

4. Open the project's anaconda-project.yml file, and add the following lines after the description. This is the deployment command that Workbench will use when you deploy the notebook

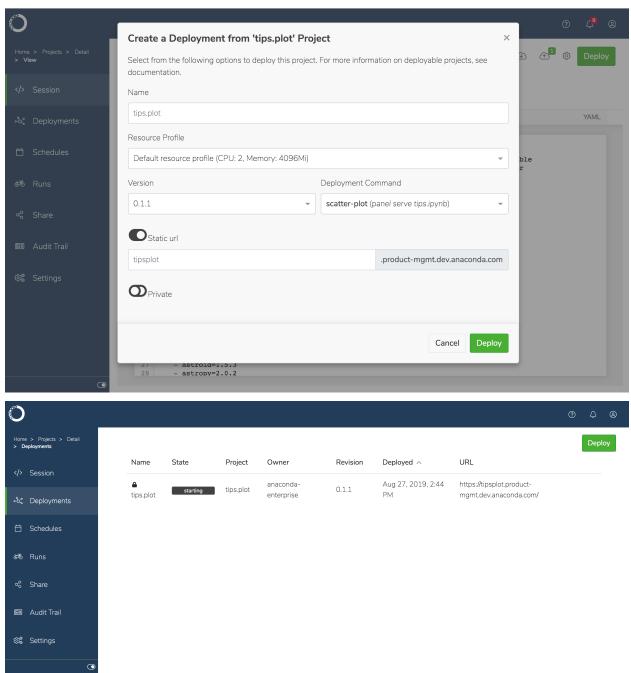
```
commands:
    scatter-plot:
    unix: panel serve tips.ipynb
    supports_http_options: True
```

5. Save and commit your changes.

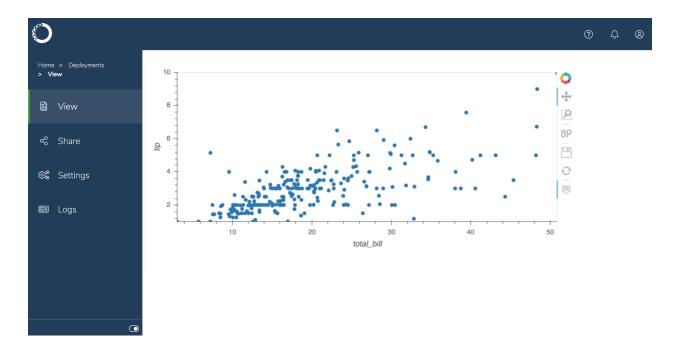
```
Home > Projects > View
                                                   Ç jupyter anaconda-project.yml✔ a minute ago
                                                             Edit View Language
                                                                                                                                                                                                                           YAML
                                                     1 name: Python 3.6
                                                         description: A comprehensive project template that contains all of the packages available in the Anac Distribution v5.0.1 for Python 3.6. Get started with the most popular and powerful packages in data
& Runs
                                                     8 platforms:
≪ Share
                                                            - linux-64
Audit Trail
                                                   12 env_specs:
14 anaconda50_py36:
15 packages:
16 - alabaster=0
17 - anaconda-c:
ම්දී Settings
                                                                   - alabaster=0.7.10
                                                                   - anaconda-client=1.6.5
                                                                   - anaconda-navigator=1.6.9
- anaconda-project=0.8.0
- asn1crypto=0.22.0
- astroid=1.5.3
                                                                   - astropy=2.0.2
- babel=2.5.0
```



6. Now you're ready to deploy the project.

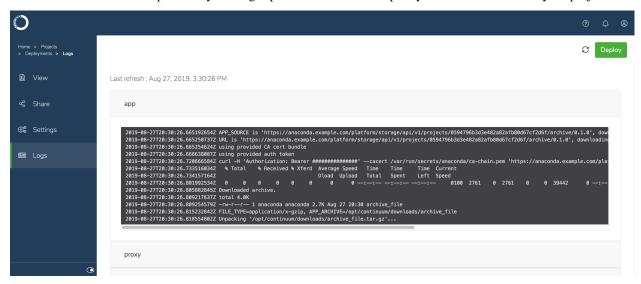


To interact with the notebook—executing its cells without making changes to it—click the deployment's name.



**Tip:** To dive deeper into the world of data visualization, follow this HoloViz tutorial.

To view and monitor the logs for the deployment while it's running, click **Logs** in the left menu. The **app** section records the initialization steps and any messages printed to standard output by the command used in your project.



You can also share the deployment with others.

# 4.10 Machine learning and deep learning

Data Science & AI Workbench facilitates machine learning and deep learning by enabling you to *develop models*, *train them*, and *deploy them*. You can also use Workbench to *query and score models* that have been *deployed as a REST API*.

To help get you started, Workbench includes several sample notebooks for common repetitive tasks. You can access them from the gallery of Sample Projects available from **Projects**. See *Projects* for more information.

We've also provided a walkthrough of the process for creating an interactive data visualization.

# 4.10.1 Developing models

Data Science & AI Workbench makes it easy for you to create models that you can *train* to make predictions and facilitate machine learning based on deep learning neural networks.

You can deploy your trained model as a REST API, so that it can be queried and scored.

The following libraries are available in Workbench to help you develop models:

- Scikit-learn-for algorithms and model training.
- TensorFlow-to express numerical computations as stateful dataflow graphs.
- XGBoost—a gradient boosting framework for C++, Java, Python, R and Julia.
- Theano-expresses numerical computations & compiles them to run on CPUs or GPUs.
- Keras—contains implementations of commonly used neural network building blocks to make working with image and text data easier.
- Lasagne—contains recipes for building and training neural networks in Theano.
- Neon-deep learning framework for building models using Python, with Math Kernel Library (MKL) support.
- MXNet-framework for training and deploying deep neural networks.
- Caffe-deep learning framework with a Python interface geared towards image classification and segmentation.
- CNTK-cognitive toolkit for working with massive datasets to facilitate distributed deep learning. Describes neural networks as a series of computational steps via a directed graph.

# 4.10.2 Training models

Data Science & AI Workbench provides machine learning libraries such as scikit-learn and Tensorflow that you can use to train the models you create.

#### To train a model:

When you are ready to run an algorithm against your model and tune it, download the scikit-learn or Tensorflow package from the anaconda channel. If you don't see this channel or these packages in your **Channels** list, contact your Administrator to *mirror these packages* to make them available to you.

### Serializing your model:

When you are ready to convert your model or application into a format that can be easily distributed and reconstructed by others, use Workbench to *deploy it*.

- YAML supports non-hierarchical data structures & scalar data
- JSON for client-server communication in web apps

• HD5 – designed to store large amounts of hierarchical data; works well for time series data (stored in arrays)

**Note:** Your model or app must be written in a programming language that supports object serialization, such as Python, PHP, R or Java.

# 4.10.3 Deploying models as endpoints

Data Science & AI Workbench enables you to deploy machine learning models as endpoints to make them available to others, so the models can be *queried and scored*. You can then save users' input data as part of the training data, and retrain the model with the new training dataset.

#### Versioning your model:

To enable you to test variations of a model, you can *deploy multiple versions of the model*. You can then direct different sets of users to each of the versions, to facilitate A/B testing.

#### Deploying your model as an endpoint:

Deploying a model as an endpoint involves these simple steps:

- 1. Create a project to tell Workbench where to look for the artifacts that comprise the model.
- 2. *Deploy the project* to build the model and all of its dependencies. Now you—and others with whom you *share the deployment*—can interact with the app, and select different datasets and algorithms.

# 4.10.4 Querying and scoring models

Data Science & AI Workbench enables you to query and score models that have been created in Python, R, or another language such as Curl, CLI, Java or Javascript. The model doesn't have to have been created using Workbench, as long as the model has been *deployed as an endpoint*.

Scoring can be incredibly useful to an organization, including the following "real world" examples:

- By financial institutions, to determine the level of risk that a loan applicant represents.
- By debt collectors, to predict the likelihood of a debtor to repay their debt.
- By marketers, to predict the likelihood of a subscriber list member to respond to a campaign.
- By retailers, to determine the probability of a customer to purchase a product.

A scoring engine calculates predictions or makes recommendations based on your model. A model's score is computed based on the model and query operators used:

- · Boolean queries—specify a formula
- Vector space queries—support free text queries (with no query operators necesssarily connecting them)
- Wildcard queries—match any pattern

## Using an external scoring engine

Advanced scoring techniques used in machine learning algorithms can automatically update models with new data gathered. If you have an external scoring engine that you prefer to use on your models, you can do so within Workbench.

# **TROUBLESHOOTING**

Anaconda's Data Science & AI Workbench provides detailed logs and monitoring information related to the Kubernetes services and containers it uses. Gravity-based installations can use the provided Operations Center and Kubernetes CLI to access this information, to help diagnose and debug errors that you or other users may encounter while using the platform.

# 5.1 Gravity troubleshooting

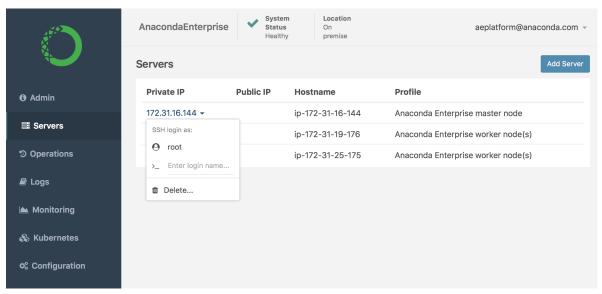
Administrators can access the Operations Center to configure, monitor, and troubleshoot the platform.

#### To access the Operations Center:

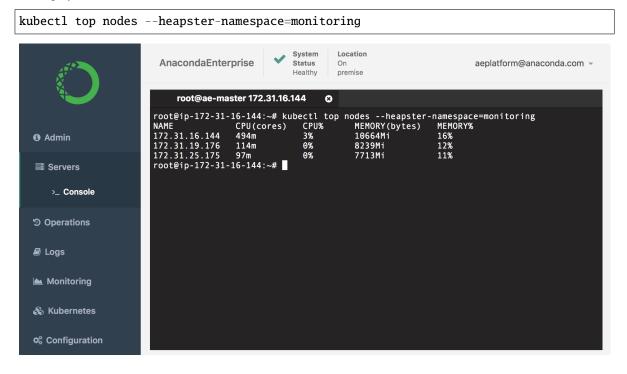
- 1. Log in to Workbench, select the **Menu** icon in the top right corner, and click the **Administrative Console** link displayed at the bottom of the slide out window.
- 2. Click Manage Resources.
- 3. Login to the Operations Center using the Administrator credentials *configured after installation*.

#### To view resource utilization:

- 1. Select **Servers** in the menu on the left.
- 2. Click on the Private IP address of the Workbench master node, and select SSH login as root.



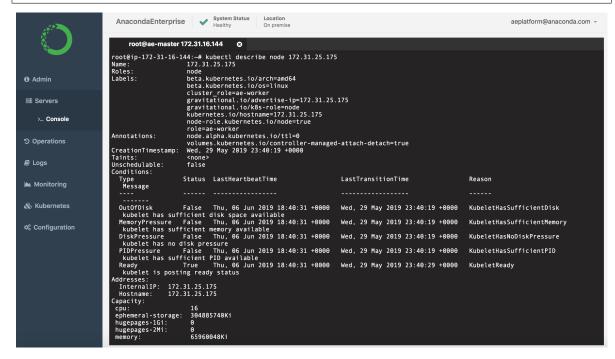
3. To display the current resource utilization of each *node* in the cluster, run this command:



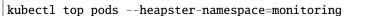
**Note:** This is actual resource utilization, not limits or requests.

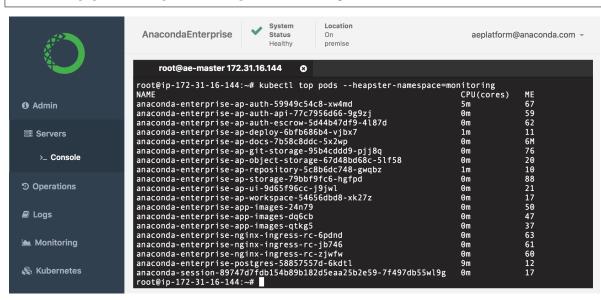
4. To view utilization *and requests* for a particular node, run the kubectl describe node command against the IP address for the node (listed under NAME). For example:

kubectl describe node 172.31.25.175

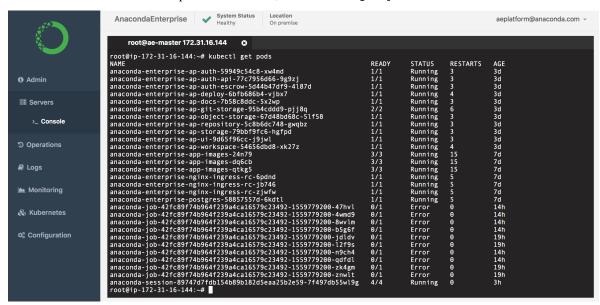


5. To view the resource utilization per *pod*, run this command:





6. To view the current status of all pods in the cluster, run kubectl get pods.

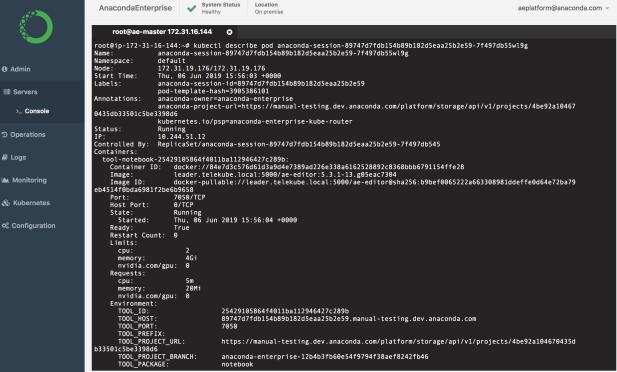


The following table summarizes common pod states:

Status	Description
Running	The pod has been bound to a node, and at least one container is running.
Pending	The pod is waiting for one or more container images to be created.
Terminating	The pod is in the process of being terminated.
Error	An error has occurred with the pod.
Init:CrashLoopBackoff	The pod failed to start, and will make another attempt in a few minutes.

7. To view information for a particular pod, run the kubectl describe pod command against the pod (listed under NAME). For example:

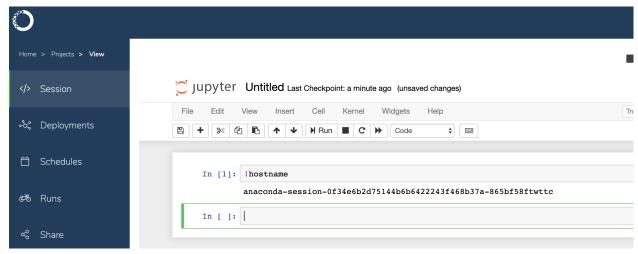


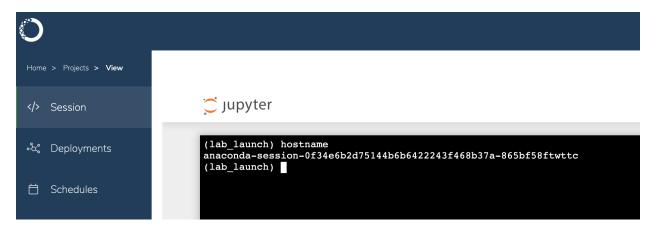


You can also use the Operations Center **Logs** to gain insights into pod behavior and troubleshoot issues. See *logging* for more information.

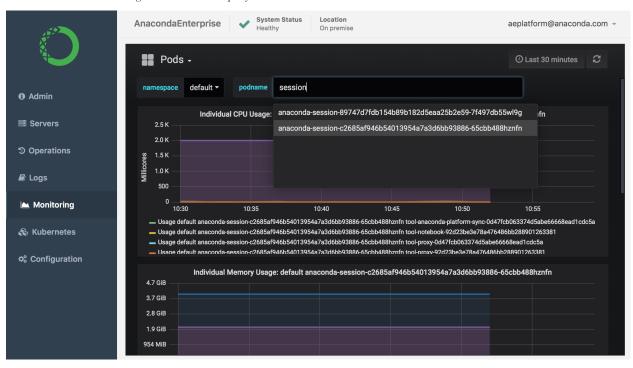
# 5.2 User errors

If a user experiences issues within a Notebook session, have them send you the name of the pod associated with their project session. They can obtain this information by running the hostname command from within a Jupyter Notebook or terminal window.





You can then use the commands described above or the Operation Center's **Monitoring** and **Logs** features to investigate the issue. See *Monitoring sessions and deployments* for more information.

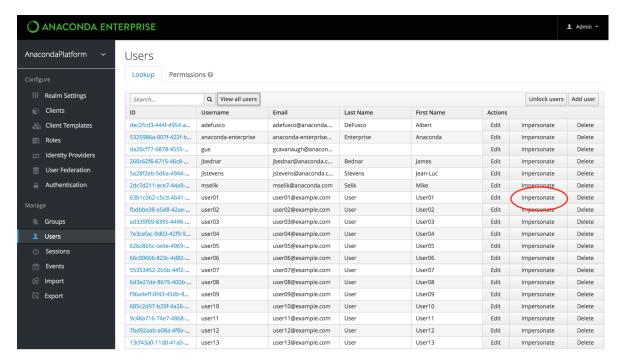


**Tip:** As an Administrator, you can also use the Authentication Center to impersonate a user to try to reproduce the problem they are experiencing.

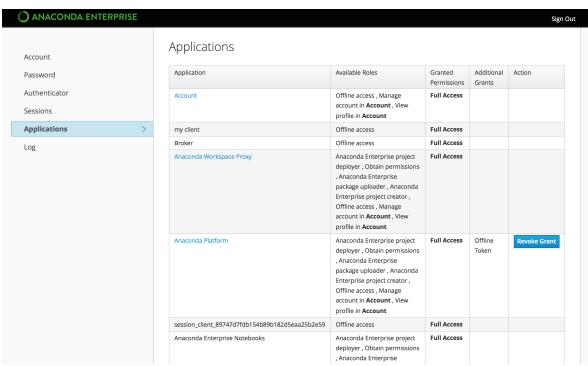
#### To access the Authentication Center:

- 1. Log in to Workbench, click the **Menu** icon in the top right corner, then click the **Administrative Console** link in the bottom of the slideout menu.
- 2. Click Manage Users.
- 3. In the Manage menu on the left, click **Users**.
- 4. On the **Lookup** tab, click **View all users** to list every user in the system, or search the user database for all users that match the criteria you enter, based on their first name, last name, or email address.

5.2. User errors 355



5. Click **Impersonate** in the row of **Actions** for the user to display a table of all **Applications** this user has interacted with on the platform, including editor sessions and deployments.



6. Click the Anaconda Platform link to interact with Workbench as the user.

See Managing users for more information on managing users.

# 5.3 Editor sessions

To help you troubleshoot issues with editor sessions, it might be helpful to understand what is happening "behind the scenes".

- When a user starts a session, Workbench launches the appropriate editor for them to work with their project files. In the background, the editor environment and other services are running in Docker containers.
- To improve startup time for projects, the editor container includes conda environments for each of the *project tem*plate environments provided by the platform. These environments are stored in /opt/continuum/anaconda/ envs, along with any custom environments created during the editor session.
- The project repository is cloned into /opt/continuum/project. (Only changes to files in this directory can be saved to the repository.)
- The anaconda-project prepare command runs, scans the project's anaconda-project.yml file for new packages and environments, and installs them into the running session.

During this phase, you can monitor the progress by watching the output of /opt/continuum/preparing.

When this process completes, the /opt/continuum/prepare.log is created.

**Caution:** Any changes made to the container image will be lost when the session stops, so any packages installed from the command line are available during the current session only. To persist package installs across sessions, they must be added to the project's anaconda-project.yml file.

5.3. Editor sessions 357

# REFERENCE MATERIALS

The following information is provided for your reference, to help you understand some of the core terminology used in Data Science & AI Workbench, and what changes were made between releases.

We also include answers to common questions you may have, and workarounds for known issues you may encounter while using the platform.

Additional information to help you get the most out of Anaconda features is available at https://support.anaconda.com/.

# 6.1 Release notes

The following notes are provided to help you understand the major changes made between releases, and therefore may not include minor bug fixes and updates. If you are experiencing issues using Data Science & AI Workbench, consider *reviewing the known issues documented here* to find workarounds.

### 6.1.1 Data Science & Al Workbench 5.7.1

Released: April 5, 2024

#### What's New

- Significant improvements have been made to the log viewing interface. Project and deployment logs now have a
  set window height, automatically scroll as information comes in, and provide search functionality, allowing you
  to locate information efficiently.
- Existing user secrets are now editable from the **Advanced Settings** page.

#### **Improvements**

- Pod metadata now contains more useful information, such as project name, project-session, project-deployment, and jobs.
- The Scheduling tab is now only available to users who have the ae-deployer role assigned to them.

### **Security Updates**

- System pods have been updated to ubi9.3-1610.
- Keycloak has been updated to version 23.0.7.
- Angular has been updated to version 16.
- Incorporated security updates for a number of our core dependencies, including sqlalchemy, nginx-ingress, pillow, and libwebp.

#### What's Fixed?

- Fixed a bug preventing notebook files from opening in VSCode.
- Fixed a bug that caused intermittent issues with scheduled jobs.

### 6.1.2 Data Science & Al Workbench 5.7.0

Released: November 7, 2023

#### What's New

- Anaconda Enterprise is now "Data Science & AI Workbench", or more simply, "Workbench".
- The Anaconda Assistant is available to install and use with your Jupyter Notebooks.
- You can now create branches within projects in Workbench for additional utility in collaborating on work.
- You can add tags to your projects and locate them quickly using the project tag filter.

#### **Improvements**

- Collaborators can now be added to projects and deployments when they are created.
- · Additional UI has been added to include MLFlow in the left-hand navigation once installed.
- Projects can be configured to have reloadable MLFlow model endpoints so that the model can be retrained without having to restart the deployment.

### **Security Updates**

- System pods have been updated to ubi9.
- Keycloak has been updated to version 22.0.5.
- Redis has been removed from the system images.

### What's Fixed?

- Fixed a bug that prevented scheduled jobs from converting back to run-once jobs.
- The Manage Resources button has been removed from BYOK8s installations.
- The UI pod recovers more gracefully if Redis is down.
- Various updates have been made to UI notification messages for consistency and clarity.

# 6.1.3 Anaconda Enterprise 5.6.2

Released: Aug 4, 2023

### What's New

- MLFlow can now be installed as an optional component.
- Package license information is now available in the UI.
- You can now filter projects by owner.

# **Improvements**

 Keycloak has been configured to apply the 'secure' flag to its cookies to prevent them from being sent over insecure HTTP connections.

### **Security Updates**

- Several system images have been updated:
  - Python 3.11
  - Kubernetes 1.26
  - Keycloak 21.1.1
  - Angular 13.4.0
  - NodeJS 16.13.1
  - Postgres 12.15
  - Redis 7.0.11

#### What's Fixed?

- Various UI bugs have been fixed.
- · Fixed a bug that prevented collaborators from being removed from deployments/projects correctly.
- Fixed a bug that prevented users from saving changes to their projects from Jupyter lab sessions.

# 6.1.4 Anaconda Enterprise 5.6.1

Released: February 28, 2023

### What's New

- Introducing: Project filtering!
  - Use the newly added filter on the project dashboard to sort your projects by owner and/or show only projects with active sessions.
  - Applied filters will also limit the results returned from using the **Search** field.
  - Applied filters persist as you navigate Anaconda Enterprise, and as you create and delete projects!
  - Hide the filter if you don't need it! You will still be informed if filters are being applied to your view.
- Projects dashboard improvements:
  - Newly created/uploaded/added projects are moved to the top left of the projects dashboard for 60 seconds.
     Once you refresh the dashboard, the project will be sorted into the "active sessions" group at the top of the dashboard, alphanumerically, according to its title.
  - Whenever you start a session for an existing project it is sorted into the "active sessions" group at the top
    of the dashboard, alphanumerically, according to its title. You will be returned to the top of the dashboard
    after starting the session.

# **Improvements**

- Improved integration with Anaconda Server!
  - Use the same credentials to sign in to both Anaconda Server and Anaconda Enterprise!
  - Create secure channels by applying filters to mirrors in Anaconda Server, then use those channels with projects in Anaconda Enterprise!

### **Security Updates**

- Several system images have been updated:
  - Base image to ubi8.7-1054
  - Postgres to 12.13

- Redis to 7.0.8
- Keycloak to 20.0.3

#### What's Fixed?

- Groups with an external version control system are no longer available when adding collaborators to a project from the project's **Share** page.
- The "Oil and Gas" sample project can now be deployed successfully.
- External version control credentials are now properly saved when entered.

# 6.1.5 Anaconda Enterprise 5.6.0

Released: October 3, 2022

#### What's New

- Keycloak has been upgraded to version 18.0.2 to resolve known vulnerabilities in the vendor-supplied containers
  for Keycloak. We've also eliminated AE5's direct access to the postgres database storing Keycloak's data. This
  information is now retrieved using Keycloak API calls.
- Documentation Updates:
  - Added instructions on how to use RStudio as an in browser IDE.
  - Updated instructions on how to wrap your python functions using tranquilizer and deploy them as REST API endpoints within AE5.
  - Updated instructions for upgrading between versions of AE5.
- JupyterLab has been updated to version 3.4.

### **Improvements**

- Added a dashboard to view all of a user's scheduled jobs.
- A Python 3.10 environment has been added to the sample projects dashboard.
- Updated openssl, ca-certificates, certifi, and the Python minor version in each existing environment.
- UI improvements have been made for an overall better experience when deploying a project or scheduling a project for deployment.
- European date format is now accepted when committing to a job.

### **Security Updates**

- Java has been removed from the editor image.
- Several system images have been updated:
  - Base image to ubi8.6-855
  - Postgres to 12.12
  - Redis to 6.2.7
  - Keycloak to 18.0.2

#### What's fixed?

- Users can now edit and delete jobs with an empty schedule (a.k.a. run once jobs). When a job is edited, it can be converted into a cronjob.
- User now has the ability to tag a commit in the UI after they have already committed all the code changes.

- Deleting a project now deletes all its job run records as well.
- Updated the error message that displays when a user attempts to use an invalid certificate.

# 6.1.6 Anaconda Enterprise 5.5.2

Released: March 10, 2022

#### What's New

- Added libnsl.so.1 to the user container to enable database connections.
- Added Docker-free RStudio installation method by mounting a shared volume at /usr/lib/rstudio-server, writable (temporarily) by a standard Anaconda Enterprise user.
- Ensured that all of the following repo configurations can persist: anaconda-client, conda-repo-cli, and conda-token.
- Eliminated the isolated package cache for deployments and jobs.
  - Deployments and jobs get their own package cache, and therefore do not reuse the shared package cache users rely on for deployments, which slows down the conda install process for deployments.
- Added clear instructions for platform administrators to add projects to samples/templates, with the ability to rebuild index.json to include customer-driven indexing of the sample projects.

### **Improvements**

- Properly initialize storage and persistence subdirectories via Helm.
  - The Helm install succeeds with managed persistence enabled, with empty storage and persistence directories.
  - Once the pods stabilize all subdirectories should be properly created and permissioned:
    - \* storage: git, object, pgdata
    - \* persistence: projects, environments, gallery
- Ability to update the helm chart without forcing certs to be modified.
  - Allows existing secrets to be preserved; eliminated kube-system ssl secret
  - Added three different certificate generation modes:
    - \* generate or boolean True: use our generateCerts helper
    - \* load or boolean False: load certificates from the certs/ subdirectory of the helm chart
    - \* any other string: do nothing
- Updated "concurrencyPolicy" for cronjobs to prevent a cronjob from scheduling a new job if one is still running, by setting the "concurrencyPolicy" to "Forbid" for scheduled jobs.
  - This will alleviate a scheduled job that had inadvertently used a deployment command instead of a
    batch job command, resulting in the job creating a new pod for each run of the job, all of which just
    stayed running eventually causing the total requests for these jobs to fill available memory.
- CVE-2020-36242 Upgraded package cryptography to version 3.3.2 or above to address vulnerability.
- Zeppelin has been updated to version 0.9.0 and made an installable tool
  - https://zeppelin.apache.org/download.html
  - Note that Zeppelin seems not to support Python 3.8 or later yet.
- Updated "Run stopped" notification when a run is deleted, notification is now "Run Deleted successfully"

- Improved behavior of the UI when the operation-controller API fails, so that when the UI is assembling the project grid it queries the operation-controller for the status of created jobs.
  - If the operation-controller pod is restored to health, the full project list comes back and all is well.
  - If an error is detected, the user is notified that there was an error querying the project creation queue.
- · Quickly deleted projects will no longer show up in the project grid.
  - The operation controller will omit from the list the jobs that are completed and removed from the project grid.
- Removed the free channel from the default condarc.
- Updated the project creation jobs to respect affinity settings, which are now present in the yaml file.

### **Bug Fixes**

- Corrected job scheduling with Kubernetes Since Kubernetes behaves according to its default behavior, a job that exits with a non-zero exit code causes Kubernetes to retry the exact same task over again.
  - Currently, AE5 looks at the job and only runs it once (success or failure) since our backend monitors the execution of jobs, and if a re-attempt is launched, AE5 actually detects that and shuts it down.
  - This can cause an issue since there's a chance that user code is run twice and the user may be unaware
    of this.
  - Setting the Kubernetes "backoffLimit" parameter to 0 will ensure that the retries are fully suppressed.
- Corrected the api.py loop in anaconda-enterprise-cli to perform retries with backoff in response to "SSLError" or "ConnectionError" exceptions.
- Updated cas-mirror to correct error when updating an existing channel with new packages via mirroring; "Error DECRYPTION FAILED OR BAD RECORD MAC"
- Whitelisted a set of environment variables and placed them in .Renviron. This whitelist currently does not include admin-supplied variables such as HTTPS\_PROXY.
  - RStudio does not pass proxy variables through to the user.
  - RStudio Server filters out all of the environment variables before starting a session.
- · Corrected corrupt anaconda-platform.yml preventing git commit.
  - When the anaconda-platform.yml file within a tagged commit cannot be parsed for any reason, the attempted git commit will fail.
  - This is due to an unexpected error in the post-metadata.py file.
  - User must remove the offending tags (this must be done even if the corruption is detected and fixed; commits will still fail).
- Corrected the AE5 deployment restart to respect git version tag.
  - The version is the same as the original deployment versus changing to latest.
- Corrected issue where deleting a deployment would lead to UI timeout errors; "HTTP 599 Timeout error"
  - This is caused because of the shutil.rmtree call inside the directory deletion process.
  - Corrected by discarding the directory and let user session CPU cycles clean them up on background.
- Corrected the ability to edit a scheduled job so that, when you perform a new run, the scheduled job will use the new version you specify.
- Corrected managed persistence GID to be preserved by changing the primary group of the user process from 0 to the GID of /opt/continuum/project.

- GID 0 should still be offered as a supplemental group to ensure that all existing files are accessible.
- Corrected uploading large size files. The commit should now be successful.
  - max-commit-file-size is 50000000.
- Updated the openIdap library.
  - Optionally, you can install openIdap 2.4 in the R conda environment to fix the RStudio issue due to broken openIdap package.

# 6.1.7 Anaconda Enterprise 5.5.1

Released: August 23, 2021

### **Expanded support for Bring Your Own Kubernetes**

- Updated internal Kubernetes API support to include versions up to 1.21
  - Previous versions of AE5 were limited to Kubernetes 1.15 and earlier, which are no longer available in most environments.
  - Verified on multiple Kubernetes offerings
    - \* Google Kubernetes Engine (GKE)
    - \* Amazon Elastic Kubernetes Service (EKS)
    - \* OpenShift (up to OCP 4.7)
  - Tested with Anthos/GKE and OpenShift on-prem
- Expanded support for Managed Persistence volumes
  - AE 5.5.1 allows the administrator to specify a custom group ID (GIDs) for the mounted volume, in order to support a wider variety of attached storage providers

### **User-facing changes**

- Improved upon the "session failed" notification: when a session is started on the project grid page, a spinner now shows the session is working in the background.
- Collaborator can be successfully removed from a project (from the UI), and a project with at least one collaborator can be successfully deleted.
- If you use an internal proxy for accessing external resources, you can (and should) set the global proxy in anaconda-enterprise-env-var-config and restart the workspace pod. This will enable the session to start successfully.
- Changed the value of kubernetes.run\_as\_root from *false* to *true* in the configmap, so that sessions, deployments, and jobs will run as UID from the start. This enables features like authenticated NFS and the sparkconfig script. This also resolved the issue with sessions taking a long time to open.
- Values you put in conda.other\_variables appear in the log list and are available in sessions, deployments, and job.
- JupyterLab and Jupyter Notebook both notify the user that the session is ready instead of the "failed to start session" message.
- Corrected an issue in the UI pod where, when a user deletes a session, the UI will still ask for file changes (e.g. the git commit UI) before it completely cleans out references to that session, causing a traceback.

- Corrected the traceback that occurred whenever a user closed or navigated away from a window with JupyterLab running.
- Fixed the dnf issue related to the new repo added for sssd.
  - Launch a session
  - Open a terminal
  - Run sudo dnf list, which should not cause errors:
    - \* sudo dnf install --installroot=/opt vim 2
    - \* Sudo dnf list
- Ability to successfully commit, push and/or revert the commit from cli.
- Resolved an issue with JupyterLab staying on screen when a session is stopped. Users are unlikely to have two JupyterLab windows open, but even when a user does, and they stop the underlying session in one window, it should be understood that the other window will lose its connection as well. The main window will go back to the "No open sessions" page.
- Resolved an issue around config file secrets, where run-once and scheduled jobs were not getting the spark-config config file secret. As a result, the sssd.conf was not set up properly for authenticated NFS on jobs.
- Modified the startup logic so that 1) all project sessions can share a common package cache, and 2) deployments and jobs get their own separate package cache.
- A proper validation message will now be displayed when saving web certificates to inform the user which field the error is coming from.
- Ability to allow /opt/continuum/project to persist, allowing the existing volume mount support to apply to /opt/continuum/project and not lose uncommitted data. This will ensure any changes the user has made to the conda environment will be properly encapsulated into changes to anaconda-project.yml.
- Documented a new volume configuration syntax: Mounting an external file share.
- Clarified steps for migrating projects to or from Bitbucket repository in the instructions for *migrating projects* between repositories. If a user wants to migrate to or from Bitbucket repository, they must use their Bitbucket account ID instead of Bitbucket username in the user mappings file.
- In sessions and deployments, you now have the ability to comment out environment variables or add a digit in the name of the global variable within env-var-config.yml.

# 6.1.8 Anaconda Enterprise 5.5.0

Released: April 14, 2021

# Administrator-facing changes

- Support for installation on customer-supplied Kubernetes platforms
  - Initial release: support for OpenShift 4.2 and GKE with k8s 1.15 or earlier
  - Additional platforms and OCP/k8s versions to come in a fast-follow release
- Managed Persistence: leverages a persistent, shared volume, allowing administrators to easily customize:
  - Sample and template projects
  - Pre-baked conda environments

### **User-facing changes**

- Managed Persistence: Users benefit from persistent storage of relevant content, with appropriate user-level and project-level access controls:
  - Project session code, data, and conda environments
  - Server and database credentials, software preferences
- Worked on several key areas to ensure the release meets the requirements outlined by customers:
  - Error and warning notification
  - Scheduling workflow
  - Log workflow
  - User Interface
  - Improved experience with projects, search, authentication
  - Improved support and updates which include validation of sample projects along with template environments & lab\_launch environment

# 6.1.9 Anaconda Enterprise 5.4.1

Released: April 15, 2020

### Administrator-facing changes

- Updated minimum and recommended requirements
- You can now **configure size limits for files** (Default value of 50MB) being committed into the internal git by changing the related values on the *config map flag*. This ensures that projects don't get bogged down by oversized internal storage. Anaconda recommends keeping files below 50MB and using external file storage for large data sets. (AENT-5922)
- You can now set the number of max concurrent queue jobs and enable/disable project creation with a queue using a *config map flag*. By implementing a queue, Kubernetes jobs for project creation are performed only when resources are available, ensuring that project creation doesn't fail due to lack of cluster resources. (AENT-5801)
- Default **SSO Timeout increased** to 1 day.

### **User-facing changes**

- You now have the ability to **see whether your project is in the queue** or actively being created.
- You will now be alerted when your commits fail, saving time and work.
- You can now **schedule your deployment in multiple timezones** via a dropdown in the Scheduler UI. Note that these scheduled deployment times will be displayed in UTC.
- You can now access public channels and deployments if not added as collaborators.
- CRON string validation has been added to schedules UI.

### **Backend improvements (non-visible changes)**

- There was an issue with users trying to create multiple projects at a time, overwhelming the cluster resources and ultimately causing some projects to fail to create. We've fixed that by **implementing a job queue**, limiting the number of simultaneous project creations based on configuration and available system resources.
- GPU support fixed, built on CUDA 10.x.
- Job pods automatically clean up upon completion of jobs.

# 6.1.10 Anaconda Enterprise 5.4.0

Released: October 31, 2019

### Administrator-facing changes

- Updated minimum and recommended requirements
- Added support for installing the Anaconda Enterprise cluster on Centos/RHEL 7.7, 8.0
- Upgraded Gravity to version 6.1.9 (and Kubernetes 1.15.05) with updated monitoring dashboards
- Ability to configure external Postgres database
- Authenticated NFS mounts
- Customize Sample Gallery and new project template collection. Requires AE5 Tools.

## **User-facing changes**

- · New UI look-and-feel
- New sample gallery projects
- Fixed JupyterLab and Jupyter Notebook timeout
- Upgraded Conda to version 4.6.14 and anaconda-project to version 0.8.3. Provide faster package installs and improved error messages
- NFS mounts now work with scheduled jobs

### **Backend improvements (non-visible changes)**

• Upgraded nginx to version 1.17.2, which uses nginx-ingress version 1.5.2, to address CVEs.

# 6.1.11 Anaconda Enterprise 5.3.1

Released: July 17, 2019

#### Administrator-facing changes

- Added support for using *on-premises versions of Bitbucket and GitLab*, and removed the previous requirement to connect to your repository endpoint over SSL.
- Added support for installing the Anaconda Enterprise cluster on RHEL/CentOS 7.6.
- Added support for NVIDIA CUDA 10.0 drivers on GPU worker nodes.
- Added the ability to set global environment variables via a new configuration file, making them available across all containers. This method can be used to address the issue where values in custom .condarc files could be overwritten if the file was placed in a directory of "lower priority" that the user's home directory.

# **User-facing changes**

- Patched JupyterLab and Jupyter Notebook to address "session timeout" and "failed to fetch" issues. Users may still see an error, but if they reload their notebook, they can continue working without losing any work.
- Fixed issue where users were being asked to confirm the environment when creating a project from the Hadoop-Spark template.
- Fixed issue where the UI makes it appear that changes made by collaborators on a project have not been committed, when they have been, leading the user to believe that an error has occurred.
- Improved the usability of the *Schedules UI*.

#### **Backend improvements (non-visible changes)**

• Upgraded Jupyter Notebook to version 5.7.8 to address CVEs.

# 6.1.12 Anaconda Enterprise 5.3.0

Released: March 22, 2019

### Administrator-facing changes

- Increased the *minimum and recommended disk space requirements* for the master node.
- Added recommendation to setup partitions on the master node using Logical Volume Management (LVM) to accommodate easier future expansion.
- Added noarch to the default platforms in the anaconda.yaml mirror config file.
- Added a bootstrap executable that you can run to install conda to the Anaconda Enterprise installer.
- Changed the process for installing and configuring the Anaconda Enterprise cli and cas-mirror slightly.

### **User-facing changes**

- Added ability to *deploy projects* to user-supplied, static URLs.
- · Improved UI notifications on behind-the-scenes processes, and added a Notification Center.
- Optimized database operations and made other performance improvements.
- Added a sample project for connecting to an S3 bucket.
- Fixed issue where users couldn't use Kerberos authentication (kinit) to access a Spark/Hadoop cluster from within a notebook.
- Fixed issue where incorrect default kernels were being used for projects created from the Hadoop-Spark template.
- Improved error message handling to clarify errors and provide instructions on how to workaround or recover from them.
- Added usability improvements related to scheduling deployment runs, audit trail logging, and session initialization.

# 6.1.13 Anaconda Enterprise 5.2.4

Released: January 21, 2019

### Administrator-facing changes

- Fixed issue where custom resource profiles weren't being captured during in-place upgrades.
- · Added security fixes.

# 6.1.14 Anaconda Enterprise 5.2.3

Released: January 2, 2019

• Included fix to address a vulnerability in Kubernetes which allowed for permission escalation. You can learn more about the vulnerability here.

#### **User-facing changes**

- Added ability for users to store secrets that can be used to access file systems, data stores and other enterprise
  resources from within sessions and deployments. Any secrets added to the platform will be available across all
  projects associated with the user's account. For more information, see Storing secrets.
- Fixed issue that required users to modify the anaconda-project.yml file to make the Hadoop-Spark environment template work properly.
- Added ability to view each project's owner, and sort the list of projects based on this column.
- Fixed various issues to improve project and session performance.

# 6.1.15 Anaconda Enterprise 5.2.2

Released: October 10, 2018

### Administrator-facing changes

- Added ability to configure an external Git repository (instead of the internal Git repository) to store projects containing version-controlled notebooks, code, and other files. Supported external Git version control systems include Atlassian BitBucket, GitHub and GitHub Enterprise, and GitLab.
- Administrators can optionally configure GPU worker nodes to be used only for sessions and deployments that require a GPU (by preventing CPU-only sessions and deployments from accessing GPU resources).
- In-place upgrades can now be performed from AE 5.2.x to AE 5.2.2.
- Improved functionality in backup script related to backup location and disk capacity requirements.
- Implemented multiple security enhancements related to cache control headers, HTTP strict transport security, and default ciphers and protocols across all services.
- Administrators no longer need to generate separate TLS/SSL certificates for the Operations Center.
- Improved validation of custom TLS/SSL certificates in the Administrator Console.
- Administrators can now disable access to sudo yum operations in sessions across the platform.
- Fixed an issue related to orphaned clients for sessions and deployments not being removed from Authentication Center.
- Tokens for user notebook sessions and deployments are now stored in encrypted format.
- Renamed platform-wide conda settings to default\_channels, channel\_alias, ssl\_verify settings in the conda section of configmap to be consistent with conda configuration settings.
- Administrators can now specify the channel priority order when creating environments/installers.
- Fixed an issue related to sorting of package versions when creating environments/installers.
- Fixed an issue with download links for custom Anaconda parcels.
- Improved behavior of package mirroring tool to only remove existing packages when clean mode is active.
- Fixed an issue related to mirroring pip packages from PyPI repository.

- Added support for noarch packages in package mirroring tool.
- Improved logging and error handling in package mirroring tool.
- Fixed an issue related to projects failing to be created due to special characters in usernames.
- Fixed an issue related to authorization center errors when syncing large number of users from external identity providers.
- Added logout functionality to anaconda-enterprise-cli.

# **User-facing changes**

- Apache Zeppelin is now available as a notebook editor for projects (in addition to Jupyter Notebooks and Jupyter-Lab). Apache Zeppelin is a web-based notebook that enables data-driven, interactive data analytics and collaborative documents with interpreters for Python, R, Spark, Hive, HDFS, SQL, and more.
- Conda channels in the repository can be made publicly available (default), or access can be restricted to specific authenticated users or groups.
- A single notebook kernel (associated with the active conda environment used within a project) is now displayed by default in Jupyter Notebooks and JupyterLab.
- Collaborators can now select a different default editor for projects that have been shared with them.
- Implemented various fixes to configuration parameters for scheduled jobs within a project.
- Improved input/form validation related to projects, deployments, packages, and settings across the platform.
- Improved error messaging/handling across the platform, along with the ability to view errors and logs from underlying services.
- Improved notifications for tasks such as uploading projects and copying sample projects.
- Users are now prompted to delete all related sessions, deployments, jobs, and runs (including those used by collaborators) when deleting a project.
- Fixed an issue that caused numerous erroneous job runs to be spawned based on the default job scheduling parameters.

# 6.1.16 Anaconda Enterprise 5.2.1

Released: August 30, 2018

#### **User-facing changes**

- Fixed issue with loading spinner appearing on top of notebook sessions
- Fixed issue related to missing projects and copying sample projects when upgrading from AE 5.1.x
- Improved visual feedback when loading notebook sessions/deployments and performing actions such as creating/copying projects

# 6.1.17 Anaconda Enterprise 5.2.0

Released: July 27, 2018

#### Administrator-facing changes

- New administrative console with workflows for managing channels and packages, creating installers, and other distinct administrator tasks
- Added ability to mirror pip packages from PyPI repository
- Added ability to define custom hardware resource profiles based on CPU, RAM, and GPU for user sessions and deployments
- Added support for GPU worker nodes that can be defined in resource profiles
- Added ability to explicitly install different types of master nodes for high availability
- · Added ability to specify NFS file shares that users can access within sessions and deployments
- Significantly reduced the amount of time required for backup/restore operations
- Added channel and package management tasks to UI, including downloading/uploading packages, creating/sharing channels, and more
- · Anaconda Livy is now included in the Anaconda Enterprise installer to enable remote Spark connectivity
- All network traffic for services is now routed on standard HTTPS port 443, which reduces the number of external ports that need to be configured and accessed by end users
- Notebook/editor sessions are now accessed via subdomains for security and isolation
- Reworked documentation for administrator workflows, including managing cluster resources, configuring authentication, generating custom installers, and more
- Reduced verbosity of console output from anaconda-enterprise-cli
- Suppressed superfluous database errors/warnings

### **User-facing changes**

- Added support for selecting GPU hardware in project sessions and deployments, to accelerate model training and other computations with GPU-enabled packages
- Added ability to select custom hardware resource profiles based on CPU, RAM, and GPU for individual sessions and deployments
- Added support for scheduled and batch jobs, which can be used for recurring tasks such as model training or ETL pipelines
- Added support for connecting to external Git repositories in a project session or deployment using account-wide credentials (SSH keys or API tokens)
- New, responsive user interface, redesigned for data science workflows
- Added ability to share deployments with unauthenticated users outside of Anaconda Enterprise
- Changed the default editor in project sessions to Jupyter Notebooks (formerly JupyterLab)
- Added ability to specify default editor on a per-project basis, including Jupyter Notebooks and JupyterLab
- · Added ability to work with data in mounted NFS file shares within sessions and deployments
- Added ability to export/download projects from Anaconda Enterprise to local machine
- Added package and channel management tasks to UI, including uploading/downloading packages, creating/sharing channels, and more

- Reworked documentation for data science workflows, including working with projects/deployments/packages, using project templates, machine learning workflows, and more
- Added ability to use plotting/Javascript libraries in JupyterLab
- Added ability to force delete a project with running sessions, shared collaborators, etc.
- · Improved messaging when a session or deployment cannot be scheduled due to limited cluster resources
- The last modified date/time for projects now accounts for commits to the project
- · Unique names are now enforced for projects and deployments
- Fixed bug in which project creator role was not being enforced

### **Backend improvements (non-visible changes)**

- Updated to Kubernetes 1.9.6
- Added RHEL/CentOS 7.5 to supported platforms
- Added support for SELinux passive mode
- · Anaconda Enterprise now uses the Helm package manager to manage and upgrade releases
- New version (v2) of backend APIs with more comprehensive information around projects, deployments, packages, channels, credentials and more
- Fixed various bugs related to custom Anaconda installer builds
- Fixed issue with kube-router and a CrashLoopBackOff error

# 6.1.18 Anaconda Enterprise 5.1.3

Released: June 4, 2018

#### **Backend improvements (non-visible changes)**

- · Fixed issue when generating custom Anaconda installers that contain packages with duplicate files
- Fixed multiple issues related to memory errors, file size limits, and network transfer limits that affected the generation of large custom Anaconda installers
- · Improved logging when generating custom Anaconda installers

# 6.1.19 Anaconda Enterprise 5.1.2

Released: March 16, 2018

### Administrator-facing changes

• Fixed issue with image/version tags when upgrading AE

### **Backend improvements (non-visible changes)**

• Updated to Kubernetes 1.7.14

# 6.1.20 Anaconda Enterprise 5.1.1

Released: March 12, 2018

### Administrator-facing changes

- Ability to specify custom UID for service account at install-time (default UID: 1000)
- Added pre-flight checks for kernel modules, kernel settings, and filesystem options when installing or adding nodes
- Improved initial startup time of project creation, sessions, and deployments after installation. Note that all services will be in the ContainerCreating state for 5 to 10 minutes while all AE images are being pre-pulled, after which the AE user interface will become available.
- Improved upgrade process to automatically handle upgrading AE core services
- Improved consistency between GUI- and CLI-based installation paths
- · Improved security and isolation between internal database from user sessions and deployments
- · Added capability to configure a custom trust store and LDAPS certificate validation
- Simplified installer packaging using a single tarball and consistent naming
- Updated documentation for system requirements, including XFS filesystem requirements and kernel modules/settings
- Updated documentation for mirroring packages from channels
- Added documentation for configuring AE to point to online Anaconda repositories
- · Added documentation for securing the internal database
- Added documentation for configuring RBAC, role mapping, and access control
- · Added documentation for LDAP federation and identity management
- Improved documentation for backup/restore process
- Fixed issue when deleting related versions of custom Anaconda parcels
- Added command to remove channel permissions
- Fixed issue related to Ops Center user creation in post-install configuration
- Silenced warnings when using verify\_ssl setting with anaconda-enterprise-cli
- Fixed issue related to default admin role (ae-admin)
- Fixed issue when generating TLS/SSL certificates with FQDNs greater than 64 characters
- Fixed issue when using special characters with AE Ops Center accounts/passwords
- Fixed bug related to Administrator Console link in menu

### **User-facing changes**

- Improvements to collaborative workflow: Added notification when collaborators make changes to a project, ability to pull changes into a project, and ability to resolve conflicting changes when saving or pulling changes into a project.
- Additional documentation and examples for connecting to remote data and compute sources: Spark, Hive, Impala, and HDFS
- · Optimized startup time for Spark and SAS project templates

- Improved initial startup time of project creation, sessions, and deployments by pre-pulling images after installation.
- Increased upload limit of projects from 100 MB to 1 GB
- · Added capability to sudo yum install system packages from within project sessions
- Fixed issue when uploading projects that caused them to fail during partial import
- Fixed R kernel in R project template
- Fixed issue when loading sparklyr in Spark Project
- Fixed issue related to displaying kernel names and Spark project icons
- Improved performance when rendering large number of projects, packages, etc.
- · Improved rendering of long version names in environments and projects
- Render full names when sharing projects and deployments with collaborators
- Fixed issue when sorting collaborators and package versions
- · Fixed issue when saving new environments
- Fixed issues when viewing installer logs in IE 11 and Safari

# 6.1.21 Anaconda Enterprise 5.1.0

Released: January 19, 2018

### Administrator-facing changes

- New post-installation administration GUI with automated configuration of TLS/SSL certificates, administrator account, and DNS/FQDN settings; significantly reduces manual steps required during post-installation configuration process
- New functionality for administrators to generate custom Anaconda installers, parcels for Cloudera CDH, and management packs for Hortonworks HDP
- Improved backup and restore process with included scripts
- Switched from groups to roles for role-based access control (RBAC) for Administrator and superuser access to AE services
- Clarified system requirements related to system modules and IOPS in documentation
- Added ability to specify fractional CPUs/cores in global container resource limits
- Fixed consistency of TLS/SSL certificate names in configuration and during creation of self-signed certificates
- Changed use of verify\_ssl to ssl\_verify throughout AE CLI for consistency with conda
- Fixed configuration issue with licenses, including field names and online/offline licensing documentation

### User changes

- Updated default project environments to Anaconda Distribution 5.0.1
- Improved configuration and documentation on using Sparkmagic and Livy with Kerberos to connect to remote Spark clusters
- Fixed R environment used in sample projects and project template
- · Fixed UI rendering issue on package detail view of channels, downloads, and versions

- Fix multiple browser compatibility issues with Microsoft Edge and Internet Explorer 11
- Fixed multiple UI issues with Anaconda Project JupyterLab extension

# Backend improvements (non-visible changes)

- Updated to Kubernetes 1.7.12
- Updated to conda 4.3.32
- Added SUSE 12 SP2/SP3, and RHEL/CentOS 7.4 to supported platform matrix
- Implemented TLS 1.2 as default TLS protocol; added support for configurable TLS protocol versions and ciphers
- Fixed default superuser roles for repository service, which is used for initial/internal package configuration step
- Implemented secure flag attribute on all session cookies containing session tokens
- Fixed issue during upgrade process that failed to vendor updated images
- Fixed DiskNodeUnderPressure and cluster stability issues
- Fixed Quality of Service (QoS) issue with core AE services on under-resourced nodes
- · Fixed issue when using access token instead of ID token when fetching roles from authentication service
- Fixed issue with authentication proxy and session cookies

#### **Known issues**

- IE 11 compatibility issue when using Bokeh in notebooks (including sample projects)
- IE 11 compatibility issue when downloading custom installers

# 6.1.22 Anaconda Enterprise 5.0.6

Released: November 9, 2017

# 6.1.23 Anaconda Enterprise 5.0.5

Released: November 7, 2017

# 6.1.24 Anaconda Enterprise 5.0.4

Released: September 12, 2017

# 6.1.25 Anaconda Enterprise 5.0.3

Released: August 31, 2017 (General Availability Release)

# 6.1.26 Anaconda Enterprise 5.0.2

Released: August 15, 2017 (Early Adopter Release)

# 6.1.27 Anaconda Enterprise 5.0.1

Released: March 8, 2017 (Early Adopter Release)

#### Features:

- Simplified, one-click deployment of data science projects and deployments, including live Python and R note-books, interactive data visualizations and REST APIs.
- End-to-end secure workflows with SSL/TLS encryption.
- Seamlessly managed scalability of the entire platform
- Industry-grade productionization, encapsulation, and containerization of data science projects and applications.

# 6.2 Known issues

We are aware of the following issues using Data Science & AI Workbench. If you're experiencing other unexpected behavior, consider checking our Support Knowledge Base.

# 6.2.1 Project is stuck in loading state on creation when email address is used as username

You cannot use an email address for a username in Workbench. If your username is an email address and you attempt to create a project, it will appear to be stuck in a loading state. The project will disappear when you refresh the page.

### Workaround

Log in to your Keycloak admin console and update the username to some other entry that is not an email address. Usernames cannot be all numeric entries.

# 6.2.2 Subsequent Jobs using the "Run Now" schedule do not consume GPU resources

When scheduling a job using the "Run Now" schedule, the first run will consume GPU resources if a GPU resource profile is used. Each subsequent job after the first will not consume a GPU resource when the **Run schedule** button is used.

Please note that this only occurs when the **Run schedule** button is selected multiple times for a job that is scheduled using the "Run Now" schedule. This does not affect any job scheduled with a cron expression.

#### Workaround

Do not run the same job multiple times on the "Run Now" schedule, using the **Run schedule** button. Instead, schedule the job with a cron expression.

6.2. Known issues 377

# 6.2.3 Unable to obtain Zeppelin credentials

After selecting **Credential** and clicking the question mark icon in the Zeppelin editor, the user should be redirected to Zeppelin documentation explaining the process for obtaining credentials. However, that link is broken.

#### Workaround

Rather than committing something sensitive in your code/repository through Zeppelin, create a Kubernetes secret in JSON format.

# 6.2.4 Attempting to install new PyViz packages in JupyterLab results in error

The new PyViz libraries aren't compatible with the version of JupyterLab used in Workbench. For more information on PyViz compatibility, see https://github.com/pyviz/pyviz\_comms#compatibility.

#### Workaround

Open the project in Jupyter Notebook.

# 6.2.5 Unable to download files when running JupyterLab in Chrome browser

If you attempt to download a file from within a JupyterLab project running on Chrome, you may see a Failed/Forbidden error, preventing you from being unable to download the file.

#### Workaround

Open the project in Jupyter Notebook or another supported browser, such as Firefox or Safari, and download the file.

# 6.2.6 Unexpected metadata in a package breaks Workbench channel

The cspice and spiceypy packages mirrored from conda-forge include incompatible metadata, which causes a channeldata.json build failure, and makes the entire channel inaccessible.

### Workaround

Remove these packages from the Workbench channel, or update your conda-forge mirror to pull in the latest packages.

# 6.2.7 Custom conda configuration file may be overwritten

If you add a custom .condarc file to your project using the anaconda-enterprise-cli spark-config command, it may get overwritten with the default config options when you deploy the project.

### Workaround

Place the .condarc file in a directory other than your home directory (/opt/continuum/.condarc).

Note that the conda config settings are loaded from all of the files on the conda config search path. The config settings are merged together, with keys from higher priority files taking precedence over keys from lower priority files. If you need extra settings, start by adding the .condarc file to a lower priority file first and see if this works for you.

For more information on how directory locations are prioritized, see this blog post.

Starting in Workbench 5.3.1, you can also *set global config variables via a config map*, as an alternative to using the Workbench CLL.

# 6.2.8 Incorrect information in command output

When running the anaconda-enterprise-cli spark-config command to *connect to a remote Hadoop Spark cluster from within a project*, the output says you need to specify the namespace by including -n anaconda-enterprise.

#### Workaround

You must omit -n anaconda-enterprise from the command, as Workbench is installed in the default namespace.

# 6.2.9 Error creating an environment immediately after installation

At least one project must exist on the platform before you can *create an environment*. If you attempt to create an environment first, the logs will say that the associated job is running, and the container isn't ready.

#### Workaround

Create a project first. The environment creation process will continue and successfully complete after a few minutes.

# 6.2.10 Cluster performance may degrade after extended use

The default limit for max\_user\_watches may be insufficient, and can be increased to improve cluster longevity.

#### Workaround

Run the following command on each node in the cluster, to help the cluster remain active:

```
sysctl -w fs.inotify.max_user_watches=1048576
```

To ensure this change persists across reboots, you'll also need to run the following command:

# 6.2.11 Invalid issuer URL causes library to get stuck in a sync loop

When using the Workbench Operations Center to create an OIDC Auth Connector, if you enter an invalid issuer url in the spec, the go-oidc library can get stuck in a sync loop. This will affect *all* connectors.

### Workaround

On a single node cluster, you'll need to do the following shut down gravity:

1. Find the gravity services: systemctl list-units | grep gravity.

You will see output like this:

2. Shut down the teleport service:

6.2. Known issues 379

```
systemctl stop gravity__gravitational.io__teleport__2.3.5.service
```

3. Shut down the planet-master service:

```
systemctl stop gravity__gravitational.io__planet-master__0.1.87-1714.service
```

On a multi-node cluster, you'll need to shut down gravity AND all gravity-site pods:

```
kubectl delete pods -n kube-system gravity-site-XXXXX
```

In both cases, you'll need to restart gravity services:

```
systemctl start gravity__gravitational.io__planet-master__0.1.87-1714.service systemctl start gravity__gravitational.io__teleport__2.3.5.service
```

# 6.2.12 GPU affinity setting reverts to default during upgrade

When upgrading Workbench from a version that supports the ability to reserve GPU nodes to a newer version (e.g., 5.2.x > 5.2.3), the nodeAffinity setting reverts to the default value, thus allowing CPU sessions and deployments to run on GPU nodes.

#### Workaround

If you had commented out the nodeAffinity section of the Config map in your previous installation, you'll need to do so again after completing the upgrade process. See *Setting resource limits* for more information.

# 6.2.13 Install and post-install problems

#### Failed installations

If an installation fails, you can view the failed logs as part of the support bundle in the failed installation UI.

After executing sudo gravity enter you can check /var/log/messages to troubleshoot a failed installation or these types of errors.

After executing sudo gravity enter you can run journalctl to look at logs to troubleshoot a failed installation or these types of errors:

```
journalctl -u gravity-23423lkqjfefqpfh2.service
```

Note: Replace gravity-23423lkqjfefqpfh2.service with the name of your gravity service.

You may see messages in /var/log/messages related to errors such as "etcd cluster is misconfigured" and "etcd has no leader" from one of the installation jobs, particularly gravity-site. This usually indicates that etcd needs more compute power, needs more space or is on a slow disk.

Workbench is very sensitive to disk latency, so we usually recommend using a better disk for /var/lib/gravity on target machines and/or putting etcd data on a separate disk. For example, you can mount etcd under /var/lib/gravity/planet/etcd on the hosts.

After a failed installation, you can *uninstall Workbench* and start over with a fresh installation.

### Failed on pulling gravitational/rbac

If the node refuses to install and fails on pulling gravitational/rbac, create a new directory TMPDIR before installing and provide write access to user 1000.

### "Cannot continue" error during install

This bug is caused by a previous failure of a kernel module check or other preflight check and subsequent attempt to reinstall.

Stop the install, make sure the preflight check failure is resolved, and restart the install again.

#### Problems during post-install or post-upgrade steps

Post-install and post-upgrade steps run as Kubernetes jobs. When they finish running, the pods used to run them are **not** removed. These and other stopped pods can be found using:

```
kubectl get pods -A
```

The logs in each of these three pods will be helpful for diagnosing issues in the following steps:

Pod	Issues in this step
ae-wagonwheel	post-install UI
install	installation step
postupdate	post-update steps

### Post-install configuration doesn't complete

After completing the post-install steps, clicking **FINISH SETUP** may not close the screen, and prevent you from continuing.

You can complete the process by running the following commands within gravity.

To determine the site name:

```
SITE_NAME=$(gravity status --output=json | jq '.cluster.token.site_domain' -r)
```

To complete the post-install process:

```
gravity --insecure site complete
```

#### Re-starting the post-install configuration

In order to reinitialize the post-install configuration UI—to regenerate temporary (self-signed) SSL certificates or reconfigure the platform based on your domain name—you must re-create and re-expose the service on a new port.

First, export the deployment's resource manifest:

Edit wagon.yaml, replacing image: ae-wagonwheel:5.X.X with image: leader.telekube.local:5000/ae-wagonwheel:5.X.X

Then recreate the ae-wagonwheel deployment using the updated YAML file:

6.2. Known issues 381

**NOTE:** Replace 5.X.X with your actual version number.

To ensure the deployment is running in the system namespace, execute sudo gravity enter and run:

```
kubectl get deploy -n kube-system
```

One of these should be ae-wagonwheel, the post-install configuration UI. To make this visible to the outside world, run:

```
\label{local_potential} $$ kubectl expose deploy ae-wagonwheel --port=8000 --type=NodePort --name=post-install -n\_ \\ $$ $$ $$ $$ kube-system
```

This will run the UI on a new port, allocated by Kubernetes, under the name post-install.

To find out which port it is listening under, run:

```
kubectl get svc -n kube-system | grep post-install
```

Then navigate to http://<your domain>:<this port> to access the post-install UI.

# 6.2.14 Kernel parameters may be overwritten and cause networking errors

If networking starts to fail in Workbench, it may be because a kernel parameter related to networking was inadvertently overwritten.

#### Workaround

On the master node running Workbench, run gravity status and verify that all kernel parameters are set correctly. If the Status for a particular parameter is degraded, follow the instructions here to reset the kernel parameter.

# 6.2.15 Removing collaborator from project with open session generates error

If you remove a collaborator from a project while they have a session open for that project, they might see a 500 Internal Server Error message.

#### Workaround

Add the user as a collaborator to the project, have them stop their notebook session, then remove them as a collaborator. For more information, see *how to share a project*.

To prevent collaborators from seeing this error, ask them to close their running session before you remove them from the project.

#### Affected versions

5.2.x

# 6.2.16 Workbench auth pod throws OutOfMemory Error

If you see an exception similar to the following, Workbench has exceeded the maximum heap size for the JVM:

```
Exception: java.lang.OutOfMemoryError thrown from the UncaughtExceptionHandler in thread "default task-248"

2018-08-29 23:13:26.327 UTC ERROR XNIO001007: A channel event listener threw an exception: java.lang.OutOfMemoryError: Java heap space (default I/O-36) [org.xnio. listener]
```

(continues on next page)

(continued from previous page)

```
2018-08-29 23:12:32.823 UTC ERROR UT005023: Exception handling request to /auth/

→realms/AnacondaPlatform/protocol/openid-connect/token: java.lang.OutOfMemoryError:

→Java heap space (default task-86) [io.undertow.request]

2018-08-29 23:13:01.353 UTC ERROR XNIO001007: A channel event listener threw an →exception: java.lang.OutOfMemoryError: Java heap space
```

#### Workaround

Increase the JVM max heap size by doing the following:

1. Open the anaconda-enterprise-ap-auth deployment spec by running the following command in a terminal:

```
$ kubectl edit deploy anaconda-enterprise-ap-auth
```

2. Increase the value for JAVA\_OPTS (example below):

```
spec:
 containers:
  - args:
    - cp /standalone-config/standalone.xml /opt/jboss/keycloak/standalone/
&& /opt/jboss/keycloak/bin/standalone.sh -Dkeycloak.migration.action=import
     -Dkeycloak.migration.provider=singleFile -Dkeycloak.migration.file=/etc/
⇒secrets/keycloak/keycloak.json
     -Dkeycloak.migration.strategy=IGNORE_EXISTING -b 0.0.0.0
 command:
 - /bin/sh
 - -c
 env:
  - name: DB_URL
   value: anaconda-enterprise-postgres:5432
 - name: SERVICE_MIGRATE
   value: auth_quick_migrate
 name: SERVICE_LAUNCH
   value: auth_quick_launch
  name: JAVA_OPTS
   value: -Xms64m -Xmx2048m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m
```

#### Affected versions

5.2.1

# 6.2.17 Fetch changes behavior in Apache Zeppelin may not be obvious to new users

A Fetch changes notification appears, but the changes do not get applied to the editor. This is how Zeppelin works, but users unfamiliar with the editor may find it confusing.

If a collaborator makes changes to a notebook that's also open by another user, the user needs to pull the changes that the collaborator made AND click the small reload arrows to refresh their notebook with the changes (see below).



# Affected versions

6.2. Known issues 383

5.2.2

# 6.2.18 Apache Zeppelin can't locate conflicted files or non-Zeppelin notebook files

If you need to access files other than Apache Zeppelin notebooks within a project, you can use the %sh interpreter from within a Zeppelin notebook to work with files via bash commands, or use the **Settings** tab to change the default editor to Jupyter Notebooks or JupyterLab and use the file browser or terminal.

#### Affected versions

5.2.2

# 6.2.19 Create and Installer buttons are not visible on Channels page

When the Channels page is viewed initially, the Create and Installers buttons are not visible on the top right section of the screen. This prevents the user from creating channels or viewing a list of installers.

#### Workaround

To make the Create and Installer buttons visible on the Channels page, perform one of the following steps:

- Click on the top-level Channels navigation link again when viewing the Channels page
- Click on a specific channel to view its detail page, then return to the Channels page

#### Affected versions

5.2.1

# 6.2.20 Updating a package from the Anaconda metapackage

When updating a package dependency of a project, if that dependency is part of the Anaconda metapackage the package will be installed once but a subsequent anaconda-project call will uninstall the upgraded package.

#### Workaround

When updating a package dependency remove the anaconda metapackage from the list of dependencies at the same time add the new version of the dependency that you want to update.

### Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3

# 6.2.21 File size limit when uploading files

Unable to upload new files inside of a project that are larger than the current restrictions:

• The limit of file uploads in JupyterLab is 15 MB

#### Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3, 5.2.0, 5.2.1, 5.2.2, 5.2.3

# 6.2.22 IE 11 compatibility issue when using Bokeh in projects (including sample projects)

Bokeh plots and applications have had a number of issues with Internet Explorer 11, which typically result in the user seeing a blank screen.

#### Workaround

Upgrade to the latest version of Bokeh available. On Anaconda 4.4 the latest is 0.12.7. On Anaconda 5.0 the latest version of Bokeh is 0.12.13. If you are still having issues, consult the Bokeh team or support.

#### Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3

# 6.2.23 IE 11 compatibility issue when downloading custom Anaconda installers

Unable to download a custom Anaconda installer from the browser when using Internet Explorer 11 on Windows 7. Attempting to download a custom installer with this setup will result in an error that "This page can't be displayed".

#### Workaround

Custom installers can be downloaded by refreshing the page with the error message, clicking the "Fix Connection Error" button, or using a different browser.

#### Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3

# 6.2.24 Project names over 40 characters may prevent JupyterLab launch

If a project name is more than 40 characters long, launching the project in JupyterLab may fail.

#### Workaround

Rename the project to a name less than 40 characters long and launch the project in JupyterLab again.

### Affected versions

5.1.1, 5.1.2, 5.1.3

# 6.2.25 Long-running jobs may falsely report failure

If a job (such as an installer, parcel, or management pack build) runs for more than 10 minutes, the UI may falsely report that the job has failed. The apparent job failure occurs because the session/access token in the UI has expired.

However, the job will continue to run in the background, the job run history will indicate a status of "running job" or "finished job", and the job logs will be accessible.

### Workaround

To prevent false reports of failed jobs from occurring in the UI, you can extend the access token lifespan (default: 10 minutes).

To extend the access token lifespan, log in to the Workbench Authentication Center, navigate to Realm Settings > Tokens, then increase the Access Token Lifespan to be at least as long as the jobs being run (e.g., 30 minutes).

### Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3

6.2. Known issues 385

## 6.2.26 New Notebook not found on IE11

On Internet Explorer 11, creating a new Notebook in a Classic Notebook editing session may produce the error "404: Not Found". This is an artifact of the way that Internet Explorer 11 locates files.

#### Workaround

If you see this error, click "Back to project", then click "Return to Session". This refreshes the file list and allows IE11 to find the file. You should see the new notebook in the file list. Click on it to open the notebook.

### Affected versions

5.0.4, 5.0.5

# 6.2.27 Disk pressure errors on AWS

If your Workbench instance is on Amazon Web Services (AWS), overloading the system with reads and writes to the directory /opt/anaconda can cause disk pressure errors, which may result in the following:

- Slow project starts.
- · Project failures.
- Slow deployment completions.
- Deployment failures.

If you see these problems, check the logs to verify whether disk pressure is the cause:

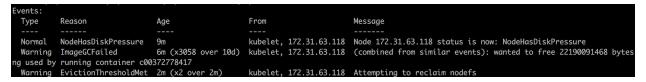
1. To list all nodes, run:

```
kubectl get node
```

2. Identify which node is experiencing issues, then run the following command against it, to view the log for that node:

```
kubectl describe node <master-node-name>
```

If there is disk pressure, the log will display an error message similar to the following:



#### Workaround

To relieve disk pressure, you can add disks to the instance by adding another Elastic Block Store (EBS) volume. If the disk pressure is being caused by a back up, you can move the backed up file somewhere else (e.g., to an NFS mount). See *Backing up and restoring Workbench* for more information.

To add disks to the instance by adding another Elastic Block Store (EBS) volume.

- 1. Open the AWS console and add a new EBS volume provisioned to 3000 IOPS. A typical disk size is 500 GB.
- 2. Attach the volume to your Workbench master.
- 3. To find your new disk's name run fdisk -1. Our example disk's name is /dev/nvme1n1. In the rest of the commands on this page, replace /dev/nvme1n1 with your disk's name.

4. Format the new disk: fdisk /dev/nvme1n1

To create a new partition, at the first prompt press n and then the return key.

Accept all default settings.

To write the changes, press w and then the return key. This will take a few minutes.

5. To find your new partition's name, examine the output of the last command. If the name is not there, run fdisk -1 again to find it.

Our example partition's name is /dev/nvme1n1p1. In the rest of the commands on this page, replace /dev/nvme1n1p1 with your partition's name.

- 6. Make a file system on the new partition: mkfs /dev/nvme1n1p1
- 7. Make a temporary directory to capture the contents of /opt/anaconda: mkdir /opt/aetmp
- 8. Mount the new partition to /opt/aetmp: mount /dev/nvmeln1p1 /opt/aetmp
- 9. Shut down the Kubernetes system.

Find the gravity services: systemctl list-units | grep gravity

You will see output like this:

Shut down the teleport service: systemctl stop gravity\_\_gravitational.io\_\_teleport\_\_2.3.5. service

```
Shut down the planet-master service: systemctl stop gravity__gravitational. io__planet-master__0.1.87-1714.service
```

- 10. Copy everything from /opt/anaconda to /opt/aetmp: rsync -vpoa /opt/anaconda/\* /opt/aetmp
- 11. Include the new disk at the /opt/anaconda mount point by adding this line to your file systems table at /etc/fstab:

```
/dev/nvme1n1p1 /opt/anaconda ext4 defaults 00
```

Use mixed spaces and tabs in this pattern: /dev/nvme1n1p1<tab>/opt/anaconda<tab>ext4<tab>defaults<tab>0<space>0

12. Move the old /opt/anaconda out of the way to /opt/anaconda-old: mv /opt/anaconda /opt/anaconda-old

If you're certain the rsync was successful, you may instead delete /opt/anaconda: rm -r /opt/anaconda

- 13. Unmount the new disk from the /opt/aetmp mount point: umount /opt/aetmp
- 14. Make a new /opt/anaconda directory: mkdir /opt/anaconda
- 15. Mount all the disks defined in fstab: mount -a
- 16. Restart the gravity services:

6.2. Known issues 387

```
systemctl start gravity__gravitational.io__planet-master__0.1.87-1714.service systemctl start gravity__gravitational.io__teleport__2.3.5.service
```

# 6.2.28 Disk pressure error during backup

If a disk pressure error occurs *while backing up your configuration*, the amount of data being backed up has likely exceeded the amount of space available to store the backup files. This triggers the Kubernetes eviction policy defined in the **kubelet** startup parameter and causes the backup to fail.

To check your eviction policy, run the following commands on the master node:

```
sudo gravity enter
systemctl status | grep "/usr/bin/kubelet"
```

#### Workaround

Restart the backup process, and specify a location with sufficient space (e.g., an NFS mount) to store the backup files. See *Backing up and restoring Workbench* for more information.

# 6.2.29 General diagnostic and troubleshooting steps

### **Entering a Gravity Workbench environment**

To enter the Workbench environment and gain access to kubectl and other commands within Workbench, use the command:

```
sudo gravity enter
```

### Moving files and data

Occasionally you may need to move files and data from the host machine to the Workbench environment. If so, there are two *shared mounts* to pass data back and forth between the two environments:

- host: /opt/anaconda/ -> Workbench environment: /opt/anaconda/
- host: /var/lib/gravity/planet/share -> Workbench environment: /ext/share

If data is written to either of the locations, that data will be available on both the host machine and within the Workbench environment

### **Debugging**

AWS Traffic needs to handle the public IPs and ports. You should either use a canonical security group with the proper ports opened or manually add the specific ports listed in *Network Requirements*.

### Problems during air gap project migration

The command anaconda-project lock over-specifies the channel list resulting in a conda bug where it adds defaults from the internet to the list of channels.

### Solution:

Add to the .condarc: "default\_channels". This way, when conda adds "defaults" to the command it is adding the internal repo server and not the repo.continuum.io URLs.

#### **EXAMPLE:**

```
default_channels:
    anaconda
channels:
    our-internal
    out-partners
    rdkit
    bioconda
    defaults
    r-channel
    conda-forge
channel_alias: https://:8086/conda
auto_update_conda: false
ssl_verify: /etc/ssl/certs/ca.2048.cer
```

### LDAP error in ap-auth

[LDAP: error code 12 - Unavailable Critical Extension]; remaining name 'dc=acme, dc=com'

This error can be caused when pagination is turned on. Pagination is a server side extension and is not supported by some LDAP servers, notably the Sun Directory server.

### Session startup errors

If you need to troubleshoot session startup, you can use a terminal to view the session startup logs. When session startup begins the output of the anaconda-project prepare command is written to /opt/continuum/preparing, and when the command completes the log is moved to /opt/continuum/prepare.log.

# 6.3 Frequently asked questions

### 6.3.1 General

#### When was the general availability (GA) release of Workbench v5?

Our GA release was August 31, 2017 (version 5.0.3). Our most recent version was released February 28, 2023 (version 5.6.1).

## Which integrated development environments (IDEs) can I use with Workbench?

Workbench supports the use of Jupyter Notebooks and JupyterLab, which are the most popular integrated data science environments for working with Python and R notebooks. You can also install and utilize both RStudio and VSCode for use in building your Workbench projects.

#### Can I deploy multiple data science applications to Workbench?

Yes, you can deploy multiple data science applications and languages across a Workbench cluster. Each data science application runs in a secure and isolated environment with all of the dependencies from Anaconda that it requires.

A single node can run multiple applications based on the amount of compute resources (CPU and RAM) available on a given node. Workbench handles all of the resource allocation and application scheduling for you.

#### Does Workbench support high availability deployments?

Partially. Some of the Workbench services and user-deployed apps will be automatically configured when installed to three or more nodes. Workbench provides several automatic mechanisms for fault tolerance and service continuity, including automatic restarts, health checks, and service migration.

For more information, see Fault tolerance in Workbench.

### Which identity management and authentication protocols does Workbench support?

Workbench comes with out-of-the-box support for the following:

- LDAP / AD
- SAML
- Kerberos

For more information, see Connecting to external identity providers.

### Does Workbench support two-factor authentication (including one-time passwords)?

Yes, Workbench supports single sign-on (SSO) and two-factor authentication (2FA) using FreeOTP, Google Authenticator or Google Authenticator compatible 2FA.

You can configure one-time password policies in Workbench by navigating to the authentication center and clicking on Authentication and then OTP Policy.

# 6.3.2 System requirements

### What operating systems are supported for Workbench?

Please see operating system requirements.

Note: Linux distributions other than those listed in the documentation can be supported on request.

### What are the minimum system requirements for Workbench nodes?

Please see system requirements.

### Which browsers are supported for Workbench?

Please see browser requirements.

### Does Workbench come with a version control system?

Yes, Workbench includes an internal Git server, which allows users to save and commit versions of their projects.

### Can Workbench integrate with my own Git server?

Yes, as described in Connecting to an external version control repository.

# 6.3.3 Installation

### How do I install Workbench?

The Workbench installer is a single tarball that includes Docker, Kubernetes, system dependencies, and all of the components and images necessary to run Workbench. The system administrator runs one command on each node.

### Can Workbench be installed on-premises?

Yes, including airgapped environments.

### Can Workbench be installed on cloud environments?

Yes, including Amazon AWS, Microsoft Azure, and Google Cloud Platform.

# Does Workbench support air gapped (off-line) environments?

Yes, the Workbench installer includes Docker, Kubernetes, system dependencies, and all of the components and images necessary to run Workbench on-premises or on a private cloud, with or without internet connectivity. We can deliver the installer to you on a USB drive.

## Can I build Docker images for the install of Workbench?

No. The installation of Workbench is supported only by using the single-file installer. The Workbench installer includes Docker, Kubernetes, system dependencies, and all of the components and images necessary for Workbench.

#### Can I install Workbench on my own instance of Kubernetes?

Yes, please refer to our BYOK8s environment preparation guide.

# Can I get the Workbench installer packaged as a virtual machine (VM), Amazon Machine Image (AMI) or other installation package?

No. The installation of Workbench is supported only by using the single-file installer.

### Which ports are externally accessible from Workbench?

Please see network requirements.

### Can I use Workbench to connect to my Hadoop/Spark cluster?

Yes. Workbench supports connectivity from notebooks to local or remote Spark clusters by using the Sparkmagic client and a Livy REST API server. Workbench provides Sparkmagic, which includes Spark, PySpark, and SparkR notebook kernels for deployment.

### How can I manage Anaconda packages on my Hadoop/Spark cluster?

An administrator can generate custom Anaconda parcels for Cloudera CDH or custom Anaconda management packs for Hortonworks HDP using Workbench. A data scientist can use these Anaconda libraries from a notebook as part of a Spark job.

## On how many nodes can I install Workbench?

You can install Workbench in the following configurations during the initial installation:

- One node (one master node)
- Two nodes (one master node, one worker node)
- Three nodes (one master node, two worker nodes)
- Four nodes (one master node, three worker nodes)

After the initial installation, you can add or remove worker nodes from the Workbench cluster at any time.

One node serves as the master node and writes storage to disk, and the other nodes serve as worker nodes. Workbench services and user-deployed applications run seamlessly on the master and worker nodes.

### Can I generate certificates manually?

Yes, if automatic TLS/SSL certificate generation fails for any reason, you can generate the certificates manually. Follow these steps:

1. Generate self-signed temporary certificates. On the master node, run:

```
cd path/to/Anaconda/Enterprise/unpacked/installer
cd DIY-SSL-CA
bash create_noprompt.sh DESIRED_FQDN
cp out/DESIRED_FQDN/secret.yaml /var/lib/gravity/planet/share/secrets.yaml
```

Replace DESIRED\_FQDN with the fully-qualified domain of the cluster to which you are installing Workbench.

Saving this file as /var/lib/gravity/planet/share/secrets.yaml on the Workbench master node makes it accessible as /ext/share/secrets.yaml within the Workbench environment which can be accessed with the command sudo gravity enter.

2. Update the certs secret

Replace the built-in certs secret with the contents of secrets.yaml. Enter the Workbench environment and run these commands:

```
$ kubectl delete secrets certs
secret "certs" deleted
$ kubectl create -f /ext/share/secrets.yaml
secret "certs" created
```

# 6.3.4 GPU Support

### How can I make GPUs available to my team of data scientists?

If your data science team plans to use version 5.2 of the Workbench AI enablement platform, here are a few approaches to consider when planning your GPU cluster:

• Build a dedicated GPU-only cluster.

If GPUs will be used by specific teams only, creating a separate cluster allows you to more carefully control GPU access.

• Build a heterogeneous cluster.

Not all projects require GPUs, so a cluster containing a mix of worker nodes—with and without GPUs—can serve a variety of use cases in a cost-effective way.

• Add GPU nodes to an existing cluster.

If your team's resource requirements aren't clearly defined, you can start with a CPU-only cluster, and add GPU nodes to create a heterogeneous cluster when the need arises.

Workbench supports heterogeneous clusters by allowing you to create different "resource profiles" for projects. Each resource profile describes the number of CPU cores, the amount of memory, and the number of GPUs the project needs. Administrators typically will create "Regular", "Large", and "Large + GPU" resource profiles for users to select from when running their project. If a project requires a GPU, Workbench will run it on only those cluster nodes with an available GPU.

### What software is GPU accelerated?

Anaconda provides a number of GPU-accelerated packages for data science. For deep learning, these include:

- Keras (keras-gpu)
- TensorFlow (tensorflow-gpu)
- Caffe (caffe-qpu)
- PyTorch (pytorch)
- MXNet (mxnet-gpu)

For boosted decision tree models:

• XGBoost (py-xgboost-qpu)

For more general array programming, custom algorithm development, and simulations:

• CuPy (cupy)

• Numba (numba)

**Note:** Unless a package has been specifically optimized for GPUs (by the authors) and built by Anaconda with GPU support, it will not be GPU-accelerated, even if the hardware is present.

#### What hardware does each of my cluster nodes require?

Anaconda recommends installing Workbench in a cluster configuration. Each installation should have an odd number of master nodes, and Anaconda recommends at least one worker node. The master node runs all Workbench core services and does not need a GPU.

Using EC2 instances, a *minimal configuration* is one master node running on a m4.4xlarge instance and one GPU worker node running on a p3.2xlarge instance. More users will require more worker nodes—and possibly a mix of CPU and GPU worker nodes.

See *Hardware requirements* for the baseline requirements for Workbench.

#### How many GPUs does my cluster need?

A best practice for machine learning is for each user to have exclusive use of their GPU(s) while their project is running. This ensures they have sufficient GPU memory available for training, and provides more consistent performance.

When a Workbench user launches a notebook session or deployment that requires GPUs, those resources are reserved for as long as the project is running. When the notebook session or deployment is stopped, the GPUs are returned to the available pool for another user to claim.

The number of GPUs required in the cluster can therefore be determined by the number of concurrently running note-book sessions and deployments that are expected. Adding nodes to a Workbench cluster is straightforward, so organizations can start with a conservative number of GPUs and grow as demand increases.

To get more out of your GPU resources, Workbench supports scheduling and running unattended jobs. This enables you to execute periodic retraining tasks—or other resource-intensive tasks—after regular business hours, or at times GPUs would otherwise be idle.

#### What kind of GPUs should I use?

Although the Anaconda Distribution supports a wide range of NVIDIA GPUs, Workbench deployments for data science teams developing models should use one of the following GPUs:

- Tesla V100 (recommended)
- Tesla P100 (adequate)

#### Can I mix GPU models in one cluster?

Kubernetes cannot currently distinguish between different GPU models in the same cluster node, so Workbench requires all GPU-enabled nodes *within a given cluster* to have the same GPU model (for example, all Tesla V100). Different clusters (e.g., "production" and "development") can use different GPU models, of course.

#### Can I use cloud GPUs?

Yes, Workbench 5.2 can be installed on cloud VMs with GPU support. Amazon Web Services (AWS), Google Cloud Platform, and Microsoft Azure all offer Tesla GPU options.

# 6.3.5 Anaconda Project

#### What operating systems and Python versions are supported for Anaconda Project?

Anaconda Project supports Windows, macOS and Linux, and tracks the latest Anaconda releases with Python 2.7, 3.5, 3.6, and 3.7.

# How is encapsulation with Anaconda Project different from creating a workspace or project in Spyder, Py-Charm, or other IDEs?

A workspace or project in an IDE is a directory of files on your desktop. Anaconda Project encapsulates those files, but also includes additional parameters to describe how to run a project with its dependencies. Anaconda Project is portable and allows users to run, share, and deploy applications across different operating systems.

#### What types of projects can I deploy?

Anaconda Project is very flexible and can deploy many types of projects with conda or pip dependencies. Deployable projects include:

- Notebooks (Python and R)
- Bokeh applications and dashboards
- REST APIs in Python and R (including machine learning scoring and predictions)
- Python and R scripts
- Third-party apps, web frameworks, and visualization tools such as Tensorboard, Flask, Falcon, deck.gl, plot.ly Dash, and more.

Any generic Python and R script or webapp can be configured to serve on port 8086, which will show the app in Workbench when deployed.

#### Does Workbench include Docker images for my data science projects?

Workbench includes data science application images for the editor and deployments. You can install additional packages in either environment using Anaconda Project. Anaconda Project includes the information required to reproduce the project environment with Anaconda, including Python, R, or any other conda package or pip dependencies.

#### After upgrading Workbench my projects no longer work

If you've upgraded to Workbench 5.4 and are getting package install errors you may need to re-write your anaconda-project.yml file.

If you were using modified template anaconda-project.yml files for Python 2.7, 3.5, or 3.6 it is best to leave the package list empty in the env\_specs section. Then you should add your required packages and their versions to the global package list.

Here's an example using the Python 3.6 template anaconda-project.yml file from Workbench version 5.3.1 where the package list has been removed from the env\_specs and the required packages added to the global list.

```
name: Python 3.6

description: A comprehensive project template that contains all of the packages_
    available in the Anaconda Distribution v5.0.1 for Python 3.6. Get started with the_
    most popular and powerful packages in data science.

channels: []
packages:
    python=3.6
    notebook
    pandas=0.25
```

(continues on next page)

(continued from previous page)

```
- psycopg2
- holoviews

platforms:
- linux-64
- osx-64
- win-64

env_specs:
anaconda50_py36:
packages: []
channels: []
```

#### 6.3.6 Notebooks

#### Are the deployed, self-service notebooks read-only?

Yes, the deployed versions of self-service notebooks are read-only, but they can be executed by collaborators or viewers. Owners of the project that contain the notebooks can edit the notebook and deploy (or re-deploy) them.

#### What happens when other people run the notebook? Does it overwrite any file, if notebook is writing to a file?

A deployed, self-service notebook is read-only but can be executed by other collaborators or viewers. If multiple users are running a notebook that writes to a file, the file will be overwritten unless the notebook is configured to write data based on a username or other environment variable.

#### Can I define environment variables as part of my data science project?

Yes, Anaconda Project supports environment variables that can be defined when deploying a data science application. Only project collaborators can view or edit environment variables, and they cannot be accessed by viewers.

#### How are Anaconda Project and Workbench available?

Anaconda Project is free and open-source. Workbench is a commercial product.

#### Where can I find example projects for Workbench?

Sample projects are included as part of the Workbench installation, which include sample workflows and notebooks for Python and R such as financial modeling, natural language processing, machine learning models with REST APIs, interactive Bokeh applications and dashboards, image classification, and more.

The sample projects include examples with visualization tools (Bokeh, deck.gl), pandas, scipy, Shiny, Tensorflow, Tensorboard, xgboost, and many other libraries. Users can save the sample projects to their Workbench account or download the sample projects to their local machine.

#### Does Workbench support batch scoring with REST APIs?

Yes, Workbench can be used to deploy machine learning models with REST APIs (including Python and R) that can be queried for batch scoring workflows. The REST APIs can be made available to other users and accessed with an API token.

#### Does Workbench provide tools to help define and implement REST APIs?

Yes, a data scientist can basically create a model without much work for the API development. Workbench includes an API wrapper for Python frameworks that builds on top of existing web frameworks in Anaconda, making it easy to expose your existing data science models with minimal code. You can also deploy REST APIs using existing API frameworks for Python and R.

# 6.3.7 Help and training

#### Do you offer support for Workbench?

Yes! You can submit support tickets to your Technical Account Manager (TAM) if you need assistance.

#### Do you offer training for Workbench?

Yes, we offer product training for collaborative, end-to-end data science workflows with Workbench.

# Do you have a question not answered here?

Please contact us for more information.

# 6.4 Understanding Workbench system requirements

Data Science & AI Workbench is a DS/AI/ML development and deployment platform built on top of the industry standard Kubernetes container orchestration system. Workbench leverages Kubernetes to create and manage sessions, deployments, and jobs. During normal operation, users and administrators are insulated from this underlying complexity. Workbench is truly at its best when running on a stable, performant Kubernetes cluster.

When issues arise that compromise the operation of Kubernetes, on the other hand, the operation of Workbench itself suffers. We have found that it is helpful to share with our customers more detail about the system requirements that Kubernetes demands. By doing so, we hope to clarify and motivate our documented requirements, and to help customers appreciate why the implementation process requires precision, persistence, and patience.

For our Gravity-based customers, the *installation process* hides much of that complexity. The main step in the installation process—the execution of the sudo ./gravity install command—is actually performing the following steps:

- · Perform a variety of pre-flight checks to verify the satisfaction of important system requirements
- · Install and configure Docker
- Install and configure Planet, a containerized implementation of Kubernetes bundled with a set of custom cluster management tools
- Install Helm, an industry standard tool for installing Kubernetes applications
- · Load the Workbench container images into the internal Docker registry
- Use a standard Helm process to install the Workbench application
- Run final Anaconda-specific application configuration tasks

Enumerating these steps helps to illustrate just how "thin" the steps specific to Workbench truly are. The bulk of the Gravity implementation effort involves the construction of a stable, performant Kubernetes cluster.

#### 6.4.1 Hardware considerations

#### **CPU and Memory: node considerations**

In our Gravity implementation, the primary node—where the central Kubernetes services and Workbench system containers run—does not host user workloads. Our standard recommendation of 16 cores and 64GB RAM provides ample headroom to ensure the correct operation of these functions.

For worker nodes, where user workloads (sessions, deployments, and jobs) are scheduled, the most important quantities are the *total number of cores and total amount of RAM across all worker nodes*. That said, nodes with more cores and

RAM are better than smaller nodes for two reasons: first, because it allows aggregate user workload to be accommodated with less total hardware; and second, so that the system can accommodate truly large-memory workloads when necessary.

#### **CPU** and **Memory**: user workloads

Do not compromise the compute resources offered to your users.

A decent data science laptop today ships with 6 cores and 16GB of RAM. While some of these resources are consumed by the operating system and other processes, their data science workloads are free to consume the vast majority.

Not all users are likely to be active at any given time, so it is not necessary to mirror this allocation on a 1:1 basis on your Workbench cluster. Kubernetes supports the notion of *oversubscription*, enabling CPU and memory allocations to exceed 100%. If we adopt a relatively standard oversubscription ratio of 4:1, we still need 75 cores and 200GB of RAM to support 50 users. Rounding that down to 64 cores and 192GB of RAM seems reasonable, at least to start. Economizing further will come at the cost of productivity—and additional resources tend to be significantly less expensive than the data scientists who will use them!

The laptop comparison is imperfect in a very important respect. On a laptop, *swap space* can be employed to temporarily allow memory consumption in excess of the physical limit. Not so with Kubernetes: a process will be *terminated* if it exceeds its memory limit, likely resulting in a loss of work. This further emphasizes the need to ensure that users are given a generous memory limit, determined not by their average usage, but rather their peak.

Some installations operate with just a single node—serving both control plane and user workload functions. For installations with a small number of simultaneous users, this is a feasible approach, as long as the node is sized aggressively—say, 64 cores and 256GB RAM.

#### VM QoS / Oversubscription / Overcommitment

Many on-premise data centers employ virtualization technology such as VMWare to better manage compute resources. A common practice in such scenarios is *oversubscription*—the ability to schedule a greater number of virtual CPUs (vCPUs) than the number of *physical* CPUs (pCPUs) present on the system. Oversubscription is an essential component of cost effective virtual machine management, since machines rarely see constant 100% usage levels.

Unfortunately, this approach is *not* necessarily compatible with Kubernetes. Kubernetes employs its *own* resource management strategy, including a notion of oversubscription. Our recommended practice for Workbench is to employ a ratio of 4:1 for user workloads. If this were compounded with, say, a 4:1 ratio at the virtual machine level, and the *true* overcommitment level is closer to 16:1. With no control over the *other* workloads sharing the same physical cores, there is a real risk of sporadic performance loss that impacts overall cluster health.

For this reason, we *strongly* recommend that any virtual machine intended to serve as a Workbench node be assigned to a *guaranteed* service class that ensures that its CPU and memory reservations are fully honored, with no oversubscription at the VM level. *Allow the Kubernetes layer to manage oversubscription exclusively.* 

#### **GPUs**

One of the more challenging aspects of implementation is the enablement of GPU computation within Workbench. It is our view that NVidia is still in the process of maturing their "story" around the use of GPUs within Docker containers in general and Kubernetes in particular. As of February 2022, the official Kubernetes documentation about GPU scheduling marks it as an "experimental" feature.

In our experience, customers *can* be successful deploying GPUs in Workbench. Workbench ships a standard CUDA 11 library in user-facing containers, and the underlying Planet implementation is built with NVidia support components. That said, our experience leads us to offer these cautions proactively.

- GPUs cannot be shared between sessions, deployments, and jobs. That means that if a user launches a session with a GPU resource profile, that GPU is reserved for their container, even if it is idle.
- Not all versions of the NVidia driver set are compatible with GPU container runtime.
- For *some* versions of the NVidia drivers, some manual rearrangement of the installed driver files are sometimes required in order for the Gravity/Planet container to "find" them.

In short, the enablement of GPUs within Workbench is a challenge, but one that many customers have nevertheless found worthwhile.

# 6.4.2 Operating system

In this section, we highlight a number of the important considerations for our Gravity-based offering. For BYOK8s customers, these types of concerns are likely "baked-in" to the general objective of standing up a performance cluster. However, customers who build their own on-premise Kubernetes clusters will likely encounter similar concerns.

# Kernel modules and settings

The system requirements provide sufficient detail on the kernel modules and other OS settings required to ensure effective operation of the Kubernetes layer. A common mistake is the failure to ensure that these settings are preserved upon reboot—so the cluster operates without incident until a system modification forces a reboot. System management software (see below) can often prevent these settings from persisting properly.

#### Firewall settings

Kubernetes itself actively manages the firewall settings on the master and worker nodes to ensure proper communication management between nodes and pods. Introducing additional firewall settings runs the risk of interrupting Workbench functionality. Please make sure that additional firewall configurations are disabled or confirmed to be compatible with Workbench. This is another common configuration that can be corrupted by automated system management tools.

#### The Linux audit daemon (auditd)

The Linux audit system provides a flexible method to detect and log a variety of system issues, and is a genuinely useful tool that is commonly enabled on the Kubernetes stack. For this reason, we have the following guidelines for exceptions and exclusions:

- /var/lib/gravity *must* be excluded from auditd monitoring.
- /opt/anaconda *should* be excluded as well. That said, we do not have strong evidence that system instability can be tied to monitoring of that directory.
- If managed persistence is hosted on the master node, then we encourage the exclusion of that directory as well. Conda environment management performs a significant number of disk operations, and slowing these operations can significantly diminish the user experience.

#### Antivirus / antimalware

Our customers utilize a variety of Linux antivirus and antimalware scanning tools, some of which include an on-demand scanning component. As with auditd, this scanning introduces a significant burden on proper Kubernetes operation. For this reason, our guidance for on-demand scanning mirrors that of auditd. In particular, /var/lib/gravity *must* be excluded from on-demand scanning.

#### System management software

One frequent culprit involved in sudden loss of Workbench functionality are system management tools such as Chef or Puppet. Tools such as these are designed to automate and simplify the management of large numbers of servers. Where they run afoul of Workbench is when the application requires *exceptions* to configurations enforced by these tools. It is *essential* that those exceptions are properly enabled. Otherwise, these tools can make fatal modifications to the underlying operating system unannounced: removing necessary kernel modules, reinstalling firewall rules, removing auditd exceptions, and others. If your organization uses tools such as these, please review the Workbench system requirements with them and confirm that the necessary exceptions are *permanently* engaged, with clear documentation as to why. Otherwise, we find that customers will eventually encounter administrators who remove these important configuration details and thereby disrupt the operation of Workbench.

#### **Backup solutions**

Many organizations will employ backup solutions on any server running critical applications, or production environments. It is important to exclude Gravity from any scheduled backup as this will cause severe disk pressure. Workbench has its own scripts that can be used to make a backup of the application on a regular basis.

#### **6.4.3 Disks**

#### Disk space

The disk space requirements specified for Gravity installations for /var/lib/gravity, /opt/anaconda, and /tmp must be respected. The installer includes disk space checks in its pre-flight checks.

With managed persistence, generous disk space allocations are even more important. This disk holds a copy of every project (and one copy for each collaborator), and every custom conda environment created by users. A single conda environment can consume multiple gigabytes. For this reason, we encourage that the size of this disk should *start* at 1TB, and preferably support live resizing.

### I/O performance

Low disk latency and high throughput in the /var/lib/gravity directory is essential for the stability of the platform. In particular, the master node hosts the Kubernetes etcd key-value store there.

In practice, we have found that the use of platter disks for /var/lib/gravity is a primary cause of system instability. *Use of an SSD for this directory is effectively required*. Direct-attached storage is preferred whenever possible, but we do believe that a sufficiently performant network-attached storage volume for /opt/anaconda is acceptable. Indeed, our positive experience with shared storage for BYOK8s installations validate this belief.

#### Auditing and antivirus software

As mentioned above, auditd daemons and antivirus software can significantly impact effective disk performance. For this reason, we mention here as well that the guidelines listed above for these tools must be honored.

#### Managed persistence

The new Managed Persistence functionality of Workbench requires the use of a shared volume that is accessible from all nodes, master and worker. So far, our customers have found that a performant enterprise NAS offers sufficient performance for their needs.

In theory, it is possible to export a directory from the master node via NFS. If an independent file sharing option is available, Anaconda recommends that instead, to ensure that the master node may focus on Kubernetes-related duties. But we have multiple successful implementations using this approach.

As our real-world experience with this feature is more limited, we will update these recommendations as more information comes in.

#### Cloud-specific concerns

Ensuring sufficient disk I/O performance is essential for a successful cloud-based implementation of Workbench. Fortunately, the common cloud providers make this a relatively straightforward thing to achieve. If possible, select VMs with attached SSDs large enough to hold /var/lib/gravity. When it is necessary to use additional attached block volumes, respect the IOPS recommendations in our system requirements. Each cloud provider offers different mechanisms for ensuring disk performance.

- In practice, the larger the disk, the higher the base IOPS performance. If you are generous with disk space, you are less likely to have issues.
- With some providers (e.g., Azure), the *only* mechanism for increasing performance is to increase the disk size.
- Providers like AWS offer *managed IOPS*, allowing you to provision size and IOPS separately. This is a reasonable approach, and *may* enable lower costs, but Anaconda recommends at least studying the cost of a larger disk instead of simply boosting IOPS.

#### 6.4.4 Network

It is vitally important that the nodes of the cluster have unfettered access to each other. Whenever network performance is impacted by hardware or operating system issues, the Kubernetes cluster will be unstable, and thus so will Workbench itself.

### Private networking

For very understandable reasons, customers usually need to place Workbench behind a firewall or VPN. It is important that this firewall does not interrupt communication between nodes, however. If possible, use *private networking* to connect the nodes to each other so that they may communicate over more direct connections even as the public-facing access to the cluster is restricted.

#### Load balancing

Workbench does not currently support being placed behind an *SSL termination* load balancer. Our experience is that it will function properly behind an *SSL passthrough* load balancer, however.

#### **Proxies**

Proxies may be required to access external data stores, repositories, and so forth. However, they must not be required for the nodes to speak with each other, and proxies must not be enabled at the OS/system level.

# WAN accelerators (IDS, packet caching, etc.)

Network acceleration technology should be disabled. Kubernetes needs to manage its own traffic shaping.

#### Shared volume (NFS) access

As is commonly understood, losing access to an external NFS share can cause disk waits and other significant issues on Unix machines. This is true for Kubernetes clusters as well. The platform can be expected to behave unreliably until access to any attached NFS volumes is restored. Interruptions to access for the managed persistence volume in particular will be severely disruptive.

# 6.4.5 Cloud vs. On-premise

Most of our customers know in advance whether or not they will be deploying onto on-premise hardware or on a major cloud provider (AWS, Google, Azure). Others have the option to choose either option, and look to us for advice on which to prefer.

In our experience, cloud installations are smoother and more reliable for a number of reasons:

- It is easier to ensure that the hardware requirements are met. For each of the major cloud providers, we can recommend specific instance types that are known to provide good performance for Workbench.
- There tends to be less additional software installed on cloud hardware, reducing the likelihood of unexpected behavior caused by interactions with the Gravity stack.
- The provisioning process is faster, as is the process of adding additional nodes or disk when required.
- We have found it significantly easier to ensure a compatible GPU configuration in the cloud. On-premise GPU
  nodes often require BIOS modifications or other configuration changes to successfully deploy.

That is not to say that cloud installations are always perfectly smooth. Indeed, all of the guidance in this document applies to both cloud and on-premise installations, and we have included cloud-specific amplifications above.

# 6.4.6 Bring Your Own Kubernetes

At a high level, many of the recommendations offered above have been developed with the assumption of an Anaconda-supplied, Gravity/Planet-based Kubernetes stack. In contrast, our BYOK8s customers will be able to leverage existing Kubernetes resources—either an on-premise Kubernetes cluster already configured to support multiple tenants, or a managed Kubernetes offering such as EKS (AWS), AKS (Azure), GKE (Google). In these scenarios, many of the above concerns are not relevant:

 Concerns about disk performance for /var/lib/gravity are tied to the need to ensure a performant Kubernetes stack.

- Operating system requirements will likely be settled either by the Kubernetes administrator or the managed Kubernetes provider.
- Workbench will likely *not have access* to the Kubernetes control plane; instead, its own application containers will be running on worker nodes alongside user workloads.

In short, most of the system requirements we have historically offered for Workbench center around *ensuring a reliable* and *performant Kubernetes cluster*. Most of our requirements, therefore, are *superseded* by the requirements imposed by your cluster.

Assuming the existence of a stable Kubernetes cluster, therefore, here is a list of some of the remaining "requirements" that remain. These include some special caveats that we have accumulated from experience with customers who may be new to the use of Kubernetes to deploy resource-intensive data science workloads.

## **Docker image sizes**

Our Docker images are larger than many Kubernetes administrators are accustomed to. In particular, the Docker image on which users run their sessions, deployments, and jobs is nearly 20GB uncompressed. This is probably the most difficult requirement for some Kubernetes administrators to swallow. Here are a couple of points to emphasize when discussing this with your administrators.

First: this does *not* imply that every session, deployment, and job will consume 20GB of disk space. Docker images are *shared* across all containers that utilize them. Therefore, the disk space consumed by the image is *amortized* across all of its uses on a given node

Second: the primary reason for this disk consumption is the set of pre-baked, global data science environments contained in this image. Future versions of Workbench will have the option to remove those environments or move them to shared storage; however, the image size is likely to never drop below 5GB.

In our experience, the response to our image sizes among Kubernetes administrators is somewhat bimodal: some react strongly negatively to it, while others have already seen images of comparable size.

#### **Resource profiles**

In our experience, Kubernetes administrators who are not accustomed to serving data science workloads will be surprised by our requirements. For many microservice workloads, CPU limits of less than a single core, and memory limits of less than 1GB, will be very common. Data science workloads require several times this much per session.

On the other hand, our standard oversubscription recommendation of 4:1—that is, the ratio between our memory/CPU *limits* and *requests* values—is a somewhat standard choice. Higher levels of oversubscription will result in sporadic performance issues for your users.

We reiterate here what we emphasized in the CPU and Memory section above: do not compromise the CPU and memory allocations for your users.

#### **Storage**

The /opt/anaconda/storage volume does not have the same strict performance requirements that /var/lib/gravity has on a Gravity installation. However, we definitely encourage the use of a "premium" performance tier for this volume if possible, as well as for the managed persistence volume.

A high-performance storage tier should be chosen for the managed persistence volume as well. Remember, users will be interacting with that volume to create Python environments and run data science workloads. Performance limitations on this volume will directly impact the user experience.

#### **Security**

In Openshift (OCP), containers by default will not run as root, and will use the Restricted Security Context Constraint (SCC). However, to use certain features such as authenticated NFS, we may need to allow pods to use the "anyuid" SCC.

#### Replacing the Ops Center

If you have administered a Gravity-based Workbench installation, you are accustomed to using the Ops Center for cluster configuration / management / monitoring. This was a feature unique to Gravitational installs, as it was provided by the Gravity site pod. In a BYOK8s environment, you will need to use the built-in management / configuration / monitoring tools provided by your k8s platform.

#### **Autoscaling**

We do not yet support autoscaling, but we are investigating it for feasibility. It is important to note however that scaling *down* Anaconda Enterprise—that is, reducing the number of nodes consumed—is not likely to be feasible in an automatic fashion. This is because downscaling requires moving workload from the nodes being decommissioned onto the remaining nodes. For user sessions, that is not something you should do without warning or planning, as doing so can interrupt active work.

# 6.5 Gravity update policy

Updated November 2023

Data Science & AI Workbench includes support for customer-supplied Kubernetes clusters in both cloud-hosted and onpremise environments. This is the preferred method of installation host for Workbench. Here, we explain the rationale behind this decision and provide guidance for our existing Gravity-based customers.

# 6.5.1 Background

Gravity is an open-source application delivery system, developed by Teleport (formerly Gravitational). Gravity creates installers that bundle application assets with Planet, a containerized Kubernetes stack. Gravity has enabled us to deliver Workbench to customers for installation on bare metal or virtual machine clusters with no existing Kubernetes support.

The benefits of the Gravity approach come with a number of practical challenges. The most important of these challenges stems from recent changes to Gravity's support model. Until recently, Teleport offered paid commercial support, providing us with fast access to technical experts when needed. In 2021, they chose to sunset that offering completely. At the end of June of 2023, Teleport suspended further support for Gravity.

This change has important practical consequences:

- Reliance solely on community support limits our velocity and ability to diagnose low-level performance or functionality issues.
- There are no longer upstream developers to deliver bug and security fixes for the platform. As a result, we are unable to resolve CVEs related to Kubernetes or the underlying virtualization layer.
- The latest production version of Gravity ships with Kubernetes 1.17.9, with versions 1.19.15 and 1.21.5 are in pre-release only. In contrast, the latest upstream Kubernetes release as of the writing of this document is 1.28.3.

In addition to these logistical challenges, our experience is that Gravity performance is very sensitive to the precise underlying operating system configuration. Our document *Understanding Workbench system requirements* represents a compendium of the challenges our customers have experienced.

We have observed that many of these configuration issues are a natural consequence of standard, legitimate IT policies that are not designed with Kubernetes in mind. Transferring ownership of Kubernetes uptime to your IT department, therefore, should help ensure a stable, performant platform for users. It also allows them to make security decisions about the Kubernetes stack that properly balance application stability with security risk.

# 6.5.2 Gravity build roadmap

We recognize that, for many of our customers, the benefits of Gravity outweigh these practical concerns. Their IT departments simply may not be prepared to formally support Kubernetes infrastructure. For these reasons, *Anaconda intends to continue to create Gravity installers* for Workbench. Here are our plans moving forward:

- The production build of Workbench 5.7.0 is built on top of Gravity 7.0.39, which utilizes Kubernetes 1.17.9. This installer has received our full QA cycle, including tests of upgrades from an existing Gravity environment.
- In some environments, in-place upgrades that update the major version of Gravity can fail. For this reason, Anaconda has developed a uninstall/reinstall cycle that preserves existing data. This alternative upgrade approach has received our full QA cycle.

# 6.5.3 Migration policy

When a customer is ready to migrate from Gravity to an in-house Kubernetes platform, Anaconda is committed to working with them to ensure that this process proceeds smoothly. To that end, Anaconda can support the following migration workflow:

- In a standard pre-implementation meeting, we review our Kubernetes system requirements with your cluster administrators.
- We assist in the installation of a new instance of Workbench on a customer-supplied Kubernetes cluster—running in parallel with an existing, Gravity-based cluster.
- We use our standard DR & Sync tooling to transfer a snapshot of the current cluster's content to the new cluster, so that users can exercise the new environment.
- Once the customer is satisfied that the new cluster is ready to be promoted to production, we transfer a final snapshot, including the hostname and SSL certificate, and update the DNS records to point to the new cluster.
- Once the cutover is complete, the customer is free to retire the old cluster.

#### Some logistical notes:

- We cannot provide direct assistance with the specification and provisioning of your new Kubernetes cluster. However, our *documentation* offers a set of templates and provisioning details for the major Kubernetes offerings, both cloud and on-premise, that your administrators are free to build from. Our installation process leverages Helm, making it compatible with all major Kubernetes platforms.
- We consider it essential that both clusters are running simultaneously for at least a brief validation period. For this reason, we do not support an in-place "upgrade" of a production Gravity-based cluster to an alternative Kubernetes platform.
- It is reasonable to consider initially under-provisioning the destination Kubernetes cluster. That is, the initial allocation of worker nodes to the new cluster can be smaller, in anticipation that the nodes from the Gravity cluster will be converted to additional workers once validation is complete. In this scenario, it would be necessary only to ensure that the new cluster has enough resources to complete the validation process before the cutover.

Many Kubernetes administrators are accustomed to hosting workloads that are less resource intensive than a data science development session or machine learning model. In particular, our Docker image sizes and recommended resource profiles are likely to surprise them. We encourage you to review the BYOK8s section of our document *Understanding Workbench system requirements* with your Kubernetes team prior to firming up a migration plan. They should also review the *BYOK8s installation requirements* and work through the *pre-installation checklist* on that page.

Finally, let us address two concerns about cost.

- There will be *no license charge* levied for a parallel installation, as long as the intent is to fully migrate workloads to the new cluster, and decommission the original, once the installation is verified.
- When migrating from Gravity to an alternative Kubernetes cluster, particularly one with multiple tenants, you may find it necessary to allow Workbench to run on more worker nodes to support the same workload. There will be *no additional per-node charges* levied as a result of this migration. Upon renewal, we will cap your per-node fees.