## Contents

1  Installing Anaconda Enterprise ........................................... 3
2  Administering Anaconda Enterprise ................................... 59
3  Using Anaconda Enterprise ............................................. 153
4  Troubleshooting .............................................................. 243
5  Reference materials ......................................................... 249
Anaconda Enterprise 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 Anaconda Enterprise, 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
When you initially install Anaconda Enterprise, you can install it on one to five nodes. You are not bound to that initial configuration, however. After completing the installation, you can add or remove nodes on the cluster as needed, including GPUs.

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

1. **Verify Requirements**
2. **Install Cluster**
3. **Install Tools**
4. **Test Installation**

### 1.1 Installation requirements

For your Anaconda Enterprise installation to complete successfully, your systems must meet the requirements outlined below. The installation requirements for Anaconda Enterprise are the same whether you choose to install the platform on-premises, hosted VSphere, or on a cloud server. There are cloud-specific requirements related to performance, however, so ensure your chosen cloud platform meets the minimum specifications outlined here before you begin.

The installer performs pre-flight 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, before installation, you...
can run the command on your intended master and worker nodes after you download and extract the installer.

When you initially install Anaconda Enterprise, you can install the cluster on one to five nodes. You are not bound to that initial configuration, however. After completing the installation, you can add or remove nodes on the cluster as needed. For more information, see Adding and removing nodes.

A rule of thumb for determining how to size your system is 1 CPU, 1GB of RAM and 5 GB of disk space for each project session or deployment. For more information about sizing for a particular component, see the following minimum requirements:

- Hardware requirements
- Disk IOPS requirements
- Storage and memory requirements
- Operating system requirements
- Security requirements
- Kernel module requirements
- System control settings
- GPU requirements
- Network requirements
- TLS/SSL certificate requirements
- DNS requirements
- Browser requirements

To use Anaconda Enterprise with a cloud platform, refer to Cloud performance requirements for cloud-specific performance requirements.

To use Spark Hadoop data sources with Anaconda Enterprise, refer to Installing Livy server for Hadoop Spark access and Configuring Livy server for Hadoop Spark access.

To verify your systems meet the requirements, see Verifying system requirements.

Note: To gain a deeper understanding of the considerations around Anaconda Enterprise system requirements, you may visit our Understanding Anaconda Enterprise system requirements topic.

Hardware requirements

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

Note: We recommend having 1 master and 1 worker per cluster.
### 1.1. Installation requirements

#### Disk IOPS requirements

Master and worker nodes require a minimum of 3000 *concurrent* input/output operations per second (IOPS)–fewer than 3000 concurrent IOPS will fail. Cloud providers report concurrent disk IOPS.

Hard disk manufacturers report *sequential* IOPS, which are different than concurrent IOPS. On-premises installations require servers with disks that support a minimum of 50 sequential IOPS. We recommend using SSD or better.

---

**NOTES regarding the minimum disk space in /opt/anaconda:**

- This total includes project and package storage (including mirrored packages).
- Currently /opt and /opt/anaconda must be an ext4 or xfs filesystem, and cannot be an NFS mount-point. Subdirectories of /opt/anaconda may be mounted through NFS. See [Mounting an external file share](#) for more information.
- If you are installing Anaconda Enterprise on an xfs filesystem, it needs to support d_type to work properly. If your XFS filesystem has been formatted with the -n ftype=0 option, it won’t support d_type, and will therefore need to be recreated using a command similar to the following before installing Anaconda Enterprise:

```
mkfs.xfs -n ftype=1 /path/to/your/device
```

**NOTES regarding the minimum disk space in /var/lib/gravity:**

- This volume MUST be mounted on local storage. Core components of Kubernetes run from this directory, some of which are extremely intolerant of disk latency. Network-Attached Storage (NAS) and Storage Area Network (SAN) solutions are susceptible to latency, and are therefore not supported.
- This total includes additional space to accommodate upgrades, and is recommended to have available during installation as it can be difficult to add space after the fact.
- We **strongly recommend** 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.

To check the number of cores, run `nproc`.
Storage and memory requirements

Approximately 50GB of available free space on each node is required for the Anaconda Enterprise installer to temporarily decompress files to the `/tmp` directory during the installation process.

If adequate free space is not available in the `/tmp` directory, you can specify the location of the temporary directory to be used during installation by setting the `TMPDIR` environment variable to a different location.

**EXAMPLE:**
```
sudo TMPDIR=/tmp2 ./gravity install
```

**Note:** When using `sudo` to install, the temporary directory must be set explicitly in the command line to preserve `TMPDIR`. The master node and each worker node all require a temporary directory of the same size, and should each use the `TMPDIR` variable as needed.

To check your available disk space, use the built-in Linux `df` utility with the `-h` parameter for human readable format:
```
df -h /var/lib/gravity
df -h /opt/anaconda
df -h /tmp
# or
df -h $TMPDIR
```

To show the free memory size in GB, run:
```
free -g
```

Operating system requirements

- Anaconda Enterprise cannot be installed with heterogeneous versions in the same cluster. Before installing, verify that all cluster nodes are operating the same version of the OS.

Anaconda Enterprise currently supports the following Linux versions:
  - RHEL/CentOS 7.x, 8.x
  - Ubuntu 16.04
  - SUSE 12 SP2, 12 SP3 **Requirement:** Set `DefaultTasksMax=∞` in `/etc/systemd/system.conf`.

To find your operating system version run `cat /etc/*release*` or `lsb-release -a`.

- Optionally create a new directory and set `TMPDIR`. User 1000 (or the UID for the service account) needs to be able to write to this directory. This means they can read, write and execute on the `$TMPDIR`.

  For example, to give write access to UID 1000, run the following command:
Note: When installing Anaconda Enterprise on a system with multiple nodes, verify that the clock of each node is in sync with the others prior to starting the installation process, to avoid potential issues. We recommend using the Network Time Protocol (NTP) to synchronize computer system clocks automatically over a network. See instructions here.

Security requirements

- If you use an antivirus scanner, such as Auditd or Antivirus, ensure the scanner excludes the /var/lib/gravity folder from its security scans.
  
- Verify you have `sudo` access.
  
- 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 these settings immediately.

  Various tools may be used to configure firewalls and open required ports, including `iptables`, `firewall-cmd`, `susefirewall2`, and others.

For all CentOS and RHEL nodes:

- Ensure that SELinux is not in enforcing mode, by either disabling it or putting it in permissive mode in the `/etc/selinux/config` file.

After rebooting, run the following command to verify that SELinux is not being enforced:

```
~$ getenforce
```

The result should be either Disabled or Permissive.

Kernel module requirements

The Anaconda Enterprise installer checks to see if the following modules required for Kubernetes to function properly are present, and alerts you if any are not loaded:

<table>
<thead>
<tr>
<th>Linux Distribution</th>
<th>Version Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS 7.2</td>
<td>bridge, ebtable_filter, ebtables, iptable_filter, overlay</td>
</tr>
<tr>
<td>RedHat Linux 7.2</td>
<td>bridge, ebtable_filter, ebtables, iptable_filter</td>
</tr>
<tr>
<td>CentOS 7.3, 7.4, 7.5, 7.6, 7.7, 8.0</td>
<td>br_netfilter, ebtable_filter, ebtables, iptable_filter, overlay</td>
</tr>
<tr>
<td>RedHat Linux 7.3, 7.4, 7.5, 7.6, 7.7, 8.0</td>
<td>br_netfilter, ebtable_filter, ebtables, iptable_filter, overlay</td>
</tr>
<tr>
<td>Ubuntu 16.04</td>
<td>br_netfilter, ebtable_filter, ebtables, brnetfilter, iptable_filter, overlay</td>
</tr>
<tr>
<td>Suse 12 SP2, 12 SP3</td>
<td>br_netfilter, ebtable_filter, ebtables, brnetfilter, iptable_filter, overlay</td>
</tr>
</tbody>
</table>
Module name | Purpose
--- | ---
bridge | Required for Kubernetes iptables-based proxy to work correctly
br_netfilter | Required for Kubernetes iptables-based proxy to work correctly
overlay | Required to use overlay or overlay2 Docker storage driver
ebtable_filter | Required to allow a service to communicate back to itself via internal load balancing when necessary
ebtables | Required to allow a service to communicate back to itself via internal load balancing when necessary
iptable_filter | Required to make sure that the firewall rules that Kubernetes sets up function properly
iptable_nat | Required to make sure that the firewall rules that Kubernetes sets up function properly

To check if a particular module is loaded, run the following command:

```
lsmod | grep <module_name>
```

If the command doesn’t produce any result, the module is not loaded.

Run the following command to load the module:

```
sudo modprobe <module_name>
```

If your system does not load modules at boot, run the following—for each module—to ensure they are loaded upon reboot:

```
sudo echo -e '<module_name>' > /etc/modules-load.d/<module_name>.conf
```

**System control settings**

Anaconda Enterprise requires the following `sysctl` settings to function properly:

<table>
<thead>
<tr>
<th>System setting</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>net.bridge.bridge-nf-call-iptables</td>
<td>Works with bridge kernel module to ensure Kubernetes iptables-based proxy works correctly</td>
</tr>
<tr>
<td>net.bridge.bridge-nf-call-ip6tables</td>
<td>Works with bridge kernel module to ensure Kubernetes iptables-based proxy works correctly</td>
</tr>
<tr>
<td>fs.may_detach_mounts</td>
<td>Can cause conflicts with the docker daemon, and leave pods in stuck state if not enabled</td>
</tr>
<tr>
<td>net.ipv4.ip_forward</td>
<td>Required for internal load balancing between servers to work properly</td>
</tr>
<tr>
<td>fs.inotify.max_user_watches</td>
<td>Set to 1048576 to improve cluster longevity</td>
</tr>
</tbody>
</table>

Run the following commands to set system control settings:

```
sudo sysctl -w <system_setting>=1
```

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

```
sudo echo -e "<system_setting> = 1" > /etc/sysctl.d/10-<system_setting>.conf
```
Verifying system requirements

Anaconda Enterprise performs system checks during the install to verify CPU, RAM and other system requirements. The system checks can also be performed manually before the installation using the following commands from the installer directory, `~/anaconda-enterprise-<installer-version>`.

**Note:** You can perform this check after downloading and extracting the installer.

To perform system checks on a *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.

GPU requirements

To use GPUs with Anaconda Enterprise, you’ll need to install version 11.2 of the NVIDIA CUDA driver *on the host operating system of any GPU worker nodes*. You can install the drivers using the package manager or the Nvidia runfile or by using `rpm` (local) or `rpm` (network) for SLES, CentOS, and RHEL, and `deb` (local) or `deb` (network) for Ubuntu.

GPU deployments should use one of the following models:

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

We have not tested the other cards supported by this driver, however, we do expect this full list to work with your cluster, provided the proper installation steps are followed:

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 K40t, Tesla K20Xm, Tesla K20m, Tesla K20s, Tesla K20c, Tesla K10, Tesla K8

M-Class: M60, M40 24GB, M40, M6, M4

The DataCenter driver referenced can be found here: [https://www.nvidia.com/Download/driverResults.aspx/173142/en-us](https://www.nvidia.com/Download/driverResults.aspx/173142/en-us)
Network requirements

Anaconda Enterprise requires the following network ports to be **externally accessible**:

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>TCP</td>
<td>Anaconda Enterprise UI (plaintext)</td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>Anaconda Enterprise UI (encrypted)</td>
</tr>
<tr>
<td>32009</td>
<td>TCP</td>
<td>Operations Center Admin UI</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4242</td>
<td>TCP</td>
<td>Bandwidth checker utility</td>
</tr>
<tr>
<td>61009</td>
<td>TCP</td>
<td>Install wizard UI access required during cluster installation</td>
</tr>
<tr>
<td>61008, 61010, 61022-61024</td>
<td>TCP</td>
<td>Installer agent ports</td>
</tr>
</tbody>
</table>

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

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

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**:

```
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
```

Where `<node_ip>` specifies the internal IP address(es) used by all nodes in the cluster to connect to the AE5 master.
If you plan to use online package mirroring, you’ll need to **whitelist the following domains**: 

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

If any Anaconda Enterprise users will use the local graphical program Anaconda Navigator in online mode, they will need access to these sites, which may need to be whitelisted in your network’s firewall settings.

- https://conda.anaconda.org if any users will use conda-forge and other channels on Anaconda Cloud (anaconda.org)
- https://vscode-update.azurewebsites.net/ if any users will install Visual Studio Code
- google-public-dns-a.google.com (8.8.8.8:53) to check internet connectivity with Google Public DNS

### TLS/SSL certificate requirements

Anaconda Enterprise uses certificates to provide transport layer security for the cluster. To get you started, self-signed certificates are generated during the initial installation. You can configure the platform to use organizational TLS/SSL certificates after completing the installation.

You may 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, and
- a certificate private key.

See *Updating TLS/SSL certificates* for more information.

### DNS requirements

Web browsers use domain names and web origins to separate sites, so they cannot tamper with each other. Anaconda includes deployments from many users, and if these deployments had addresses on the same domain, such as https://anaconda.yourdomain.com/apps/001 and https://anaconda.yourdomain.com/apps/002, one app could access the cookies of the other, and JavaScript in one app could access the other app.
To prevent this potential security risk, Anaconda assigns deployments unique addresses such as https://uuid001.anaconda.yourdomain.com and https://uuid002.anaconda.yourdomain.com, where ‘yourdomain.com’ is replaced with your organization’s domain name, and uuid001 and uuid002 is replaced with dynamically generated universally unique identifiers (UUIDs), for example.

To facilitate this, Anaconda Enterprise 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 fully qualified domain name (FQDN) anaconda.yourdomain.com with a master node IP address of 12.34.56.78, the DNS entries would be as follows:

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

The wildcard subdomain’s DNS entry points to the Anaconda Enterprise master node.

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

Existing installations of dnsmasq will conflict with Anaconda Enterprise. If dnsmasq is installed on the master node or any worker nodes, you’ll need to remove it from all nodes before installing Anaconda Enterprise.

Run the following commands to ensure dnsmasq is stopped and disabled:

- To stop dnsmasq: `sudo systemctl stop dnsmasq`
- To disable dnsmasq: `sudo systemctl disable dnsmasq`
- To verify dnsmasq is disabled: `sudo systemctl status dnsmasq`

### Browser requirements

Anaconda Enterprise 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.

**Note:** JupyterLab and Jupyter Notebook don’t currently support Internet Explorer, so Anaconda Enterprise users will have to use another editor for their Notebook sessions if they choose to use that browser to access the AE platform.

### 1.2 Cloud performance requirements

The installation requirements for Anaconda Enterprise are the same whether you choose to install the platform on premises, hosted VSphere, or on a cloud server. The only cloud-specific requirement for running Anaconda Enterprise relates to performance, so ensure your chosen cloud platform meets these minimum specifications before you begin:
Amazon Web Services (AWS)
Due to etcd’s sensitivity to disk latency, only use EC2 instances with a minimum of 3000 IOPS. We recommend an instance type no smaller than m4.4xlarge for both master and worker nodes.

Microsoft Azure
To meet CPU and disk I/O requirements, the minimum size for the selected VM should be Standard D16s v3 (16 VCPUs, 64 GB memory).

Google Cloud Platform (GCP)
No requirements for installing Anaconda Enterprise are unique to Google Cloud Platform.

After you’ve verified that your system meets these performance requirements—as well as all system requirements—you are ready to installing the cluster.

1.3 Pre-install checklist

It’s essential that the systems in your environment where you will install Anaconda Enterprise meet ALL of the requirements outlined here. The installer performs some pre-flight checks, and only allows installation to continue on nodes that are configured correctly, but doesn’t verify all requirements are met.

We’ve created this pre-install checklist to help you verify that you’ve accounted for everything before you begin, and ensure your installation is successful. Consider printing out a copy and physically checking off items as you go.

Note: We’ve also packaged a pre-flight script that you can use to verify whether the systems on which you plan to install Anaconda Enterprise meet the minimum requirements to install successfully. Follow these instructions to install and run the script.

[ ] I’ve verified that all nodes in the cluster meet the minimum or recommended specifications for CPU, RAM and disk space.

[ ] I’ve verified that all nodes in the cluster meet the minimum IOPS required for reliable performance.

[ ] I’ve verified that there is 50GB of free space available in the /tmp directory (or another temporary directory to be used during installation) on each node in the cluster.

[ ] I’ve verified that all cluster nodes are operating the same version of the OS, and that the OS version is supported by Anaconda Enterprise.

[ ] I’ve used the Network Time Protocol (NTP) to synchronize computer system clocks, and I’ve verified that the clock of each node in the cluster is in sync with the others. (Instructions for using NTP are provided here.)

[ ] I’ve verified that I have sudo access on all systems, and the firewall is configured correctly.

[ ] I’ve verified that all required kernel modules are loaded.

[ ] I’ve verified that the system control settings are set correctly.
I’ve verified that any GPUs to be used with Anaconda Enterprise have a supported NVIDIA CUDA driver installed.

I’ve verified that the system meets all network port requirements, whether the specified ports need to be open internally, externally, or during installation only.

I’ve verified that any firewalls used for network security have been temporarily disabled, for the window of time when Anaconda Enterprise is being installed.

I’ve verified that the domains required for online package mirroring have been whitelisted, if applicable.

If I intend to replace the self-signed certificates generated during installation with others, I’ve gathered all the information and files for the TLS/SSL certificates I will use.

I’ve verified that the Anaconda Enterprise domain get resolved to the IP address of the master node, whether through an alias (A) record or canonical name (CNAME).

I’ve verified that any wildcard DNS entries for my organization’s domain names meet the DNS requirements outlined here.

I’ve verified that the /etc/resolv.conf file on all the nodes DOES NOT include the rotate option.

I’ve verified that any existing installations of Docker (and dockerd), dnsmasq, and lxd have been removed from all nodes, as they will conflict with Anaconda Enterprise.

I’ve verified that all web browsers to be used to access Anaconda Enterprise are supported by the platform.

## 1.4 Installing the cluster

After you have determined the initial topology for your Anaconda Enterprise cluster, and verified that your system meets all of the installation requirements, you’re ready to install the cluster.

**Before you begin:**

**Note:** If you haven’t already, consider using the pre-install checklist provided, to verify that you’ve accounted for everything before you begin.

- By default, Anaconda Enterprise installs using a service account with the user ID (UID) 1000. You can change the UID of the service account by using the --service-uid option or the GRAVITY_SERVICE_USER environment variable at installation time. To do so, you need to have first created a group for that user with the UID.

  For example, to use UID 1001, run the following commands on each node of the cluster:

  ```
  root$ groupadd mygroup -g 1001
  root$ useradd --no-create-home -u 1001 -g mygroup myuser
  ```

- The installer uses the TMPDIR directory that’s configured on the master node, so be sure the default directory contains sufficient space or create an alternate directory (with sufficient space) for the installer to use. If you choose to use an alternate directory, ensure it has the correct permissions enabled (drwxrwxrwx), and either add it to /etc/environment or explicitly specify the directory during installation.
Determine your install method

The method you use to install the cluster will vary, depending on your ability to access the target machine. If you have network access to the target machine, we recommend you install Anaconda Enterprise using a web browser. Otherwise, you’ll need to use a command line.

With both methods, you can create any number of nodes from one to five nodes. You can also add or remove nodes at any time after installation. For more information, see Adding and removing nodes.

If the cluster where you will install AE cannot connect to the internet, follow the instructions for Installing in an air-gapped environment.

Using a web browser (recommended)

1. On the master node, download and decompress the installer, replacing `<location_of_installer>` with the location of the installer, and `<version>` with your installer version:

   ```bash
   curl -O <location_of_installer>.tar.gz
   tar xvf anaconda-enterprise-<version>.tar.gz
   cd anaconda-enterprise-<version>
   ```

2. On the master node, run the pre-installation system checks as sudo or root user before proceeding with the installation:

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

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

   ```bash
   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.

4. After doing the pre-installation system checks, run the installer on the master node as sudo or root user:

   ```bash
   sudo ./gravity wizard
   ```

   **Note:** If you’re using a service account UID that’s different than the default 1000, append the command with the actual UID. For example, to use UID 1001, run `sudo ./gravity wizard --service-uid=1001`.

   If you’re using an alternate TMPDIR, pre-pend the command with the directory. For example, `sudo TMPDIR=/mytmp ./gravity wizard`

---

1.4. Installing the cluster 15
'gravity resume' to continue the operation. See https://gravitational.com/gravity/docs/cluster/#managing-an-ongoing-operation for details.

Tue Oct 29 14:22:22 UTC Connecting to installer
Tue Oct 29 14:22:27 UTC Connected to installer
Tue Oct 29 14:22:28 UTC Starting web UI install wizard
Tue Oct 29 14:22:28 UTC Open this URL in browser: https://172.31.67.113:61009/web/installer/new/gravitational.io/AnacondaEnterprise/5.4.0-36.gdf45da616?install_token=9954bf9f357b0eff8d2d2a4a48c8d9e6
Tue Oct 29 14:22:28 UTC Waiting for the operation to start

5. To start the browser-based install, copy the full URL that is generated into your browser. Ensure that you are connecting to the public network interface.

NOTES:

- If you’re using an alternate TMPDIR and DID NOT add it to /etc/environment, edit the copied URL to include the directory in the sudo bash command. For example, sudo TMPDIR=/mytmp bash.
- If you’re unable to connect to the URL due to security measures in place at your organization, select File > New Incognito Window to launch the installer.

6. The installer will install a self-signed TLS/SSL certificate, so you can click the link at the bottom of this warning message to proceed:

7. After reviewing the License Agreement, check I Agree To The Terms and click Accept.

8. Enter the name to use for your deployment in the Cluster Name field. The Bare Metal option is already selected, so you can click Continue.

9. Select the number of nodes—between one and five—that you want to install in the cluster. One node will act as the master node, and any remaining nodes will be worker nodes. See Fault tolerance for more information on
how to size your cluster.

10. **On each node** you plan to install Anaconda Enterprise, copy and run the command that’s provided as it applies to the master node and any worker nodes. As you run the command on each node, the host name of the node is listed below the nodes.

11. Use the **IP Address** drop-down to select the IP address for each node.

12. Accept the default directory for installing application data (**/opt/anaconda/**) or enter another location.

1.4. **Installing the cluster**
13. After all nodes are listed, click **Continue**. This process can take approximately 20 minutes to complete.

**Note:** To view the install logs, click the **EXECUTABLE LOGS** pulldown at the bottom of the panel.

When the installation is complete, the following screen is displayed:
14. Click **Continue & Finish Setup** to proceed to *Post-install configuration*.

**Note:** The installer running in the terminal will note that installation is complete and that you can stop the installer process. **Do not do so** until you have completed the *post-install configuration*.

---

**Using a command line**

If you cannot connect to the server from a browser—because you’re installing from a different network, for example—you can install Anaconda Enterprise using a command line.

On each node in the cluster, download and decompress the installer, replacing `<location_of_installer>` with the location of the installer, and `<version>` with your installer version:

```bash
curl -O <location_of_installer>.tar.gz
tar xvf anaconda-enterprise-<version>.tar.gz
cd anaconda-enterprise-<version>
```

*On the master node*, run the pre-installation system checks—as sudo or root user—before proceeding with the installation:

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

Create a file named `values.yaml` with the following values, replacing `HOSTNAME` with the fully-qualified domain name (FQDN) of the host server:

```yaml
apiVersion: v1
type: ConfigMap
metadata:
  name: anaconda-enterprise-install
data:
  values:
    hostname: HOSTNAME
    generateCerts: true
    keycloak:
      includeMasterRealm: true
```

After running the pre-installation system checks and creating the YAML file, run the following command *on the master node* as sudo or root user, where you replace:

- The **advertise-addr** IP address with the address you want to be visible to the other nodes
- **CLUSTERNAME** with a name, otherwise a random cluster name will be assigned

**Note:** Gravity requires the use of lowercase, alphanumeric characters for cluster names. Periods are acceptable for spacing, e.g. `test.cluster1`.

- `/path/to/values.yaml` with the path to the `values.yaml` file you created
For flavor, choose from the following options the one that represents the number and type of nodes you want to install in the cluster:

- **small**: installs a single-node cluster (one ae-master node). This is the default flavor.
- **medium**: installs three nodes (one ae-master node and two ae-worker nodes)
- **large**: installs five nodes (one ae-master node, two k8s-master nodes and two ae-worker nodes):

```bash
sudo ./gravity install --advertise-addr=192.168.1.1 --token=anaconda-enterprise --cluster=CLUSTERNAME --flavor=small --config /path/to/values.yaml
```

**NOTES:**

If you’re using a service account UID that’s different than the default 1000, append the command with the actual UID. For example, to use UID 1001, run:

```bash
sudo ./gravity install --advertise-addr=192.168.1.1 --token=anaconda-enterprise --cluster=CLUSTERNAME --flavor=small --config /path/to/values.yaml --service-uid=1001
```

**or**

```bash
sudo GRAVITY_SERVICE_USER=1001 ./gravity install --advertise-addr=192.168.1.1 --token=anaconda-enterprise --cluster=CLUSTERNAME --flavor=small --config /path/to/values.yaml
```

If you’re using an alternate TMPDIR, pre-pend the command with the directory. For example:

```bash
sudo TMPDIR=/mytmp ./gravity install --advertise-addr=192.168.1.1 --token=anaconda-enterprise --cluster=CLUSTERNAME --flavor=small --config /path/to/values.yaml
```

The command line displays the installer’s progress:

```
* [0/100] starting installer
* [0/100] preparing for installation... please wait
* [0/100] application: AnacondaEnterprise:5.2.x
* [0/100] starting non-interactive install
* [0/100] still waiting for 1 nodes of role "worker" to join
* [0/100] still waiting for 1 nodes of role "worker" to join
* [0/100] still waiting for 1 nodes of role "worker" to join
* [0/100] initializing the operation
* [20/100] configuring packages
* [50/100] installing software
```

If you’re installing AE on AWS, use the `--cloud-provider` option when installing the master. The installer automatically detects EC2 and uses the VPC-based flannel backend instead of VXLAN. To force the use of VXLAN, use the `--cloud-provider generic` option.

**On each worker node**, run the following command, replacing the advertise-addr IP address with the address you want to be visible to the other nodes:

```bash
sudo ./gravity join 192.168.1.1 --advertise-addr=192.168.1.2 --token=anaconda-enterprise --role=ae-worker
```

The command line displays the installer’s progress:

```
* [0/100] joining cluster
* [0/100] connecting to cluster
* [0/100] connected to installer at 192.168.1.1
```
This process takes approximately 20 minutes.

After you’ve finished installing Anaconda Enterprise, you’ll need to create a local user account and password to log into the Anaconda Enterprise Operations Center.

First, enter the Anaconda Enterprise environment on any of the master or worker nodes:

```
sudo gravity enter
```

Then, run the following command to create a local user account and password for the Anaconda Enterprise Operations Center, replacing `<your-email>` and `<your-password>` with the email address and password you want to use.

**Note:** Passwords must be at least six characters long.

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

### Installing in an air-gapped environment

If the cluster where you will install Anaconda Enterprise cannot connect to the internet, follow these instructions:

1. Download the installer tarball file to a jumpbox or USB key.
2. Move the installer tarball file to a designated head node in the cluster.
3. Untar the installer file and run `sudo ./gravity wizard` for browser-based installation or `sudo ./gravity install` for CLI-based installation.

Installation and post-install configuration steps are the same for air-gapped and internet-connected installations, so you can continue the installation process from this point, choosing your preferred method:

- **Browser installation** (must be on the same network as the target machines)
- **Unattended command-line interface (CLI) installation**

### Post-install configuration

After completing either installation path, complete the post-install configuration steps.

### 1.5 “Bring your own Kubernetes” installation guide
This topic provides guidance on installing clusters without the use of Gravity.

1.5.1 Install requirements

Note: To gain a deeper understanding of the considerations around Anaconda Enterprise system requirements, you may visit our Understanding Anaconda Enterprise system requirements topic.

Note: To gain a deeper understanding of the considerations around Anaconda Enterprise system requirements, you may visit our Understanding Anaconda Enterprise system requirements topic.

• The Ops Center is not a part of a Non-Gravitational based install, as this feature was provided by Gravity.
• An external internet connection is required. This allows you to pull docker images from repo.
• These instructions work for Kubernetes version 1.15 or lower.
Hardware requirements

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

- Persistent Volume will be 500GB
- SSD storage recommended for provisioner

**Note:** We recommend having 1 master and (at least) 1 worker per cluster, as well as using a 1:4 ratio for CPU cores / GB RAM.

<table>
<thead>
<tr>
<th>Master node</th>
<th>Minimum</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>4 cores</td>
<td>16 cores</td>
</tr>
<tr>
<td>RAM</td>
<td>16GB</td>
<td>64GB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Worker nodes</th>
<th>Minimum</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>4 cores</td>
<td>16 cores</td>
</tr>
<tr>
<td>RAM</td>
<td>16GB</td>
<td>64GB</td>
</tr>
</tbody>
</table>

To check the number of cores, run `nproc`.

GPU requirements

GPU deployments should use one of the following models:

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

TLS/SSL certificate requirements

Anaconda Enterprise uses certificates to provide transport layer security for the cluster. To get you started, self-signed certificates are generated during the initial installation. You can configure the platform to use organizational TLS/SSL certificates after completing the installation.

You may 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, and
- a certificate private key.

See [Updating TLS/SSL certificates](#) for more information.
DNS requirements

Web browsers use domain names and web origins to separate sites, so they cannot tamper with each other. Anaconda includes deployments from many users, and if these deployments had addresses on the same domain, such as https://anaconda.yourdomain.com/apps/001 and https://anaconda.yourdomain.com/apps/002, one app could access the cookies of the other, and JavaScript in one app could access the other app.

To prevent this potential security risk, Anaconda assigns deployments unique addresses such as https://uuid001.anaconda.yourdomain.com and https://uuid002.anaconda.yourdomain.com, where ‘yourdomain.com’ is replaced with your organization’s domain name, and uuid001 and uuid002 is replaced with dynamically generated universally unique identifiers (UUIDs), for example.

To facilitate this, Anaconda Enterprise 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 fully qualified domain name (FQDN) 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
```

The wildcard subdomain’s DNS entry points to the Anaconda Enterprise master node.

Browser requirements

Anaconda Enterprise 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.

Note: JupyterLab and Jupyter Notebook don’t currently support Internet Explorer, so Anaconda Enterprise users will have to use another editor for their Notebook sessions if they choose to use that browser to access the AE platform.

1.5.2 OpenShift container platform (OCP)

OCP-specific requirements

- OCP 4.2.36 or earlier.
- Storage Class to be used for Persistent Volume

For example:

Prerequisite steps

- 2 nodes (at minimum) needed - this is to label one as AE-Master, and one (or more) for AE-Worker. The nodes should be labeled:
- **AE-Master**
  * `oc label node <node> role=ae-master`
  * `oc label node <node> cluster_role=ae-master`
  * `oc label node <node> app_role=storage`

- **AE-WORKER**
  * `oc label node <node> role=ae-worker`
  * `oc label node <node> cluster_role=ae-worker`

- **DNS will need to be configured as:**
  - `<ae5>.apps.<OCP>`
  - `*.<ae5>.apps.<OCP>`

**Installation**

Anaconda will provide the OCP operator in `.tar.gz` format. If and when this is added to the OCP marketplace, you will be able to install from the the marketplace instead and skip this step, similar to Anaconda Team Edition.

To install the Anaconda Enterprise 5.5 operator, run the following commands, in order, using `oc`:

```
# Replace <OPERATOR-NAME> with your operator name
oc project openshift-operators
oc create -f deploy/crds/charts.helm.k8s.io_anacondaenterprises_crd.yaml
oc create -f deploy/service_account.yaml
oc create -f deploy/role.yaml
oc create -f deploy/olm-catalog/<OPERATOR_NAME>/manifests/<OPERATOR_NAME>.
  → clusterserviceversion.yaml
```

In the namespace you wish to install AE 5.5, navigate to the operator tab:
Manually enter your FQDN (line 45):

```yaml
other_variables:
  ssl_verify: false
dj:
  drivername: postgresql
  password: ***
  port: 5432
  connect_timeout: postgres
deploy:
  ui: false
  generate_ssl: true
  priority: false
```

Manually enter your storage provisioner (line 106):

```yaml
name: eks
region: eu-central-1
failure-domain: fdd
```

The install process can take anywhere from a few minutes to half an hour depending on the connection speed to the docker repo, from which images are being pulled. Once done, all pods should be in a running state.

SSL support

You will need to create two routes by hand to correctly handle SSL. These example routes are for `main` and `ui`, respectively.
**Note:** The Path cell is left empty.

<table>
<thead>
<tr>
<th>Create Route</th>
<th>DELETE</th>
<th>EDIT YAML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Path</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Service</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The Path cell contains a forward slash ("/").
Click **Create** after creating the routes.
The UI should now load correctly. Log in with default info and update SSL certs via the UI.

### GPU support

Assuming this has not already been done, perform the following:

1. Ensure `gpu=true` label is applied to all GPU nodes.
2. **Follow these guides:**
   - [https://www.openshift.com/blog/creating-a-gpu-enabled-node-with-openshift-4-2-in-amazon-ec2](https://www.openshift.com/blog/creating-a-gpu-enabled-node-with-openshift-4-2-in-amazon-ec2)
3. Create a GPU resource profile in the configmap.

The OpenShift container platform should now be installed.

### 1.5.3 Google Kubernetes engine (GKE)

#### GKE-specific requirements

- Kubernetes 1.15.12-gke.6002
- Storage Class to be used for Persistent Volume

#### Prerequisite steps

Label your nodes in the following manner:

- **AE-MASTER**
  - `kubectl label node <node> role=ae-master`
  - `kubectl label node <node> cluster_role=ae-master`
  - `kubectl label node <node> app_role=storage`

- **AE-WORKER**
  - `kubectl label node <node> role=ae-worker`
  - `kubectl label node <node> cluster_role=ae-worker`

Create your provisioner before platform install, for example:
Installation

Anaconda will provide the necessary helm charts for installing on GKE. `values.yaml` will need to be customized with an FQDN and provisioner. Once done, run the following:

```bash
helm install anaconda-enterprise ./installer/resources/helm-charts/Anaconda-Enterprise -f ./installer/resources/helm-charts/Anaconda-Enterprise/values.yaml
```

The install process can take anywhere from a few minutes to half an hour depending on the connection speed to the docker repo, from which images are being pulled. Once done, all pods should be in a running state.

Aim DNS at the Load Balancer external IP address being created:

Once all pods are running, log in to the UI and update the SSL certs.

GPU support

**Note:** Make sure GPU nodes have the label `gpu=true`.

Run the following command to install the GPU-specific drivers:

```bash
```

You will then need to create a GPU resource profile in the configmap.

Managing Anaconda Enterprise

You can make changes to the Anaconda Enterprise configmap by editing the deployed yaml file in the Anaconda Enterprise operator. This can be necessary for creating or making changes to resource profiles or other configurations.

To setup persistent storage, you will need to first create a Persistent Volume Claim to be consumed by the persistent storage. Ensure that this PVC is using a storage provisioner with the reclaim policy set to retain.
You have the option of using a different PVC for each mount, or one PVC with a different subpath for each mount. For example:

```
persistence:  
environments:  
pvc: env  
gallery:  
pvc: gal  
manager: anaconda-enterprise  
projects:  
pvc: pro
```

1. Enter the cluster Admin account credentials that you will use to log in to the Anaconda Enterprise Operations Center initially, and click Next. (You can change these, or authorize additional Operations Center Admins, as needed.)
Note: The installer will generate self-signed SSL certificates that you can use temporarily to get started. See ../admin/advanced/certs for information on how to change them later, if desired.

2. Enter the fully-qualified domain name (FQDN) where the cluster will be accessed and click **Finish Setup**.

3. Log in to the Anaconda Enterprise Operations Center using the cluster Admin credentials you provided in Step 1, and follow the instructions below to update the platform settings with the FQDN of the host server.

### 1.6.2 Command-line instructions

If you performed an unattended installation using the **command-line instructions**, follow the instructions below to generate self-signed SSL certificates that you can use temporarily to get started. See ../admin/advanced/certs for information on how to change them later, if desired.

**Note:** You need to have OpenJDK installed to be able to use the following method to generate self-signed SSL certificates.

1. **On the master node** for your Anaconda Enterprise installation, run the following commands to save your secrets file to a location where Anaconda Enterprise can access it, replacing `YOUR_FQDN` with the fully-qualified domain name of the cluster on which you installed Anaconda Enterprise:

   ```
   cd path/to/Anaconda/Enterprise/unpacked/installer
   cd DIY-SSL-CA
   bash create_noprompt.sh YOUR_FQDN
   cp out/DESIRED_FQDN/secret.yml /var/lib/gravity/planet/share/secret.yml
   ```

   Now `/var/lib/gravity/planet/share/secret.yml` is accessible as `/ext/share/secret.yml` within the Anaconda Enterprise environment, which can be accessed with the following command:

   ```
   sudo gravity enter
   ```

2. Replace the default secrets cert with the contents of your `secret.yml` file by running the following commands from within the Anaconda Enterprise environment:

   ```
   $ kubectl delete secrets anaconda-enterprise-certs
   secret "anaconda-enterprise-certs" deleted
   ```

   (continues on next page)
Note: If the post-install process doesn’t complete after using the CLI install, you can complete the process by running the following command within gravity.

To complete the post-install process:

```bash
$ kubectl create -f /ext/share/secret.yml
secret "anaconda-enterprise-certs" created
```

Now you are ready to follow the instructions below to test your installation.

### 1.6.3 Testing your installation

After you’ve finished installing Anaconda Enterprise, and completed the post-install configuration steps, you can do the following to verify that your installation succeeded:

1. Access the Anaconda Enterprise console by entering the URL of your AE server in a web browser: `https://anaconda.example.com`, replacing `anaconda.example.com` with the fully-qualified domain name of the host server.

2. Login with the default username and password `anaconda-enterprise/anaconda-enterprise`. After testing your installation, update the credentials for this default login. See Configuring user access for more information.

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

### 1.7 Updating TLS/SSL certificates

You can replace the self-signed certificates generated as part of the initial post-install configuration at any time.

Before you begin, follow the processes outlined below. Then you can update the Anaconda Enterprise platform to use your own certificates using the Anaconda Enterprise Admin Console or the command line.
Before you begin:

1. Ask all users to save their work, stop any sessions and deployments, and log out of the platform while you update the certificates.
2. Backup your current Anaconda Enterprise configuration following the backup process.
3. Gather all of the following information and files related to your certificates, so you have it available to copy and paste from in the procedure that follows:
   - Registered domain name for the server
   - SSL certificate for servername.domain.tld, named tls.crt
   - SSL private key for servername.domain.tld, named tls.key
   - Root SSL certificate (such as this default Root CA), named rootca.crt. A root certificate is optional but recommended.
   - Intermediate SSL certificate chain/bundle, named intermediate.pem (This certificate may also appear as the second entry in your fullchain.pem file.)
   - Wildcard domain name
   - Wildcard certificate for *.servername.domain.tld, named wildcard.crt.
   - Wildcard private key for *.servername.domain.tld, named wildcard.key.
4. After you've gathered all the information above, follow the steps below that correspond to whether you will use the Admin console or the command line to update the Anaconda Enterprise platform to use your certificates.

To update the platform using the Admin console:

1. Log in to Anaconda Enterprise, 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. Log in to the console using the Administrator credentials configured after installation.
3. Select Web Certificates from the left menu.
4. Copy and paste the certificate and key information from the files you gathered previously into the appropriate fields.

5. Click **Save** to update the platform with your changes.

---

**Note:** The default SSL certificate file names generated by the installer vary slightly between versions. *If you have upgraded from a previous version of Anaconda Enterprise,* you may need to update your configuration to make sure all services are referencing the correct SSL certificate filenames (see below).

<table>
<thead>
<tr>
<th>Previous</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>rootca.pem</td>
<td>rootca.crt</td>
</tr>
<tr>
<td>cert.pem</td>
<td>tls.crt</td>
</tr>
<tr>
<td>privkey.pem</td>
<td>tls.key</td>
</tr>
<tr>
<td>tls.crt</td>
<td>wildcard.crt</td>
</tr>
<tr>
<td>tls.key</td>
<td>wildcard.key</td>
</tr>
</tbody>
</table>

**Note:** The `keystore.jks` filename remains unchanged.

---

**To update the platform using the command line:**

*On the system where the certificate and private key reside:*

1. Install openjdk. For example, use the following command to install `java-1.8.0-openjdk` on CentOS 7.5:

   ```bash
   yum install java-1.8.0-openjdk -y
   ```

2. Run the following command to create the `keystore.jks` file that will be used by Java:

   ```bash
   openssl pkcs12 -passout pass:anaconda -export -in CERT.PEM -inkey KEY.PEM -out certificate.p12 -name auth
   keytool -importkeystore -deststorepass anaconda -destkeypass anaconda -destkeystore keystore.jks -srckeystore certificate.p12 -srcstoretype PKCS12 -srcstorepass anaconda -alias auth
   ```

   **Note:** If you’re using a certificate provided by Let’s Encrypt, use `FULLCHAIN.PEM` instead of `CERT.PEM`.

3. Create an updated Root CA to use with the system:

   ```bash
   cat ROOT.CA /etc/ssl/certs/ca-bundle.trust.crt > updated-trust-ca.crt
   ```
Note: If you’re using a certificate provided by Let’s Encrypt your can obtain the Root CA here. You must also prepend the `CHAIN.pem` to the Root CA.

Note: For RHEL-based systems, the path to the trusted CA is: `/etc/ssl/certs/ca-bundle.trust.crt`. For Ubuntu-based systems, the path to the system CA is `/etc/ssl/certs/ca-certificates.crt`.

4. Setup the basic structure of the `certificates.yaml` file, that you’ll be updating in the next several steps:

```yaml
cat > certificates.yaml <<EOL
apiVersion: v1
kind: Secret
metadata:
  name: anaconda-enterprise-certs
type: Opaque
data:
EOL
```

5. Add the main domain for the SSL certificate. For example `test.anaconda.com`:

```bash
printf "
  tls.crt: " >> certificates.yaml
base64 -i --wrap=0 CERT.PEM >> certificates.yaml
```

6. Add the private key for the certificate:

```bash
printf "
  tls.key: " >> certificates.yaml
base64 -i --wrap=0 KEY.PEM >> certificates.yaml
```

7. Add the SAN certificate to the file. For example `*.test.anaconda.com`:

```bash
printf "
  wildcard.crt: " >> certificates.yaml
base64 -i --wrap=0 CERT.PEM >> certificates.yaml
```

8. Add the private key for the SAN certificate:

```bash
printf "
  wildcard.key: " >> certificates.yaml
base64 -i --wrap=0 KEY.PEM >> certificates.yaml
```

9. Add the keystore you generated in Step 2:

```bash
printf "
  keystore.jks: " >> certificates.yaml
base64 -i --wrap=0 keystore.jks >> certificates.yaml
```

10. Add the updated Root CA that you created in Step 3:

```bash
printf "
  rootca.crt: " >> certificates.yaml
base64 -i --wrap=0 updated-trust-ca.crt >> certificates.yaml
```

11. Add a new line at the end of the file:

```bash
printf '\n' >> certificates.yaml
```

12. Copy the file to the share directory inside gravity:

```
1.7. Updating TLS/SSL certificates
```
13. Run the following commands to enter gravity and list your secrets:

```bash
gravity enter
kubectl get secrets
```

14. In the next step you’ll be removing and recreating a secret, so create a backup of the existing secrets first:

```bash
kubectl get secret anaconda-enterprise-certs -o yaml --export > anaconda_certs.backup
```

15. Remove the existing secret, and recreate it from the file you placed in the share directory (in Step 12):

```bash
kubectl delete secret anaconda-enterprise-certs
kubectl create -f /ext/share/certificates.yaml
```

16. Restart all pods to update Anaconda Enterprise to use your certificate:

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

### 1.7.1 Extracting TLS/SSL certificates

Run the following command for each certificate file you wish to extract, replacing `rootca.crt` below with the name of the specific file:

```bash
kubectl get secrets anaconda-enterprise-certs -o jsonpath="{.data['rootca.crt']}" | base64 -d > rootca.crt
```

The following certificate files are available:

- **rootca.crt**: the root certificate authority bundle
- **tls.crt**: the SSL certificate for individual services
- **tls.key**: the private key for the above certificate
- **wildcard.crt**: the SSL certificate for “wildcard” services, such as deployed apps and sessions
- **wildcard.key**: the private key for the above certificate
- **keystore.jks**: the Java Key Store containing these certificates used by some services

To copy the extracted root certificate and add it to the default RHEL/CentOS or Ubuntu trusted CA bundles, run the following commands:

```bash
# On Ubuntu
$ cp rootca.crt /usr/share/ca-certificates
$ update-ca-certificates

# RHEL/CentOS
$ cp rootca.crt /etc/pki/ca-trust/source/anchors/
$ update-ca-trust
```
1.7.2 Verifying TLS/SSL certificates

If you are using privately signed certificates, extract the rootca, then use openssl to verify the certificates and make sure the final Verify return code is 0:

```
# On Ubuntu
$ openssl s_client -connect anaconda.example.com:443 -CAfile /etc/ssl/certs/ca-certificates.crt
...  
    Verify return code: 0 (ok)
# On RHEL/CentOS
$ openssl s_client -connect anaconda.example.com:443 -CAfile /etc/pki/tls/certs/ca-bundle.crt
...  
    Verify return code: 0 (ok)
```

**Note:** The root CA for the self-signed certificates generated as part of the installation is contained in the certificate bundle at /etc/pki/tls/certs/ca-bundle.crt.

You can now install and use the Anaconda Enterprise CLI to configure the certificates for the platform repository.

1.8 Installing conda for packaging mirroring

To help improve performance and security, Anaconda Enterprise 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 Anaconda Enterprise installer contains a bootstrap executable that you can run to install conda.

**Prerequisites:**
- Install Anaconda Enterprise
- Complete the post-install configuration
- Update TLS/SSL certs

**To install conda:**

1. In a terminal window, navigate to the directory where you downloaded and extracted the Anaconda Enterprise installer, replacing `<version>` with your specific version number:

```
$ cd anaconda-enterprise-<version>\installer
```

2. Run the following command to verify whether the bzip2 package is installed:

```
which bunzip2
```

If the command returns a valid package, you can run the bootstrap executable. Otherwise use your package manager to install the binary, using either `yum install bzip2` or `apt-get install bzip2`.

3. Run the following command to run the bootstrap executable:
$ ./conda-bootstrap-<version>

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.

### 1.9 Installing the Anaconda Enterprise CLI

**Warning:** The following process for installing the Anaconda Enterprise CLI results in a broken conda env. Follow the workaround described here instead, until the issue is addressed in a future release of Anaconda Enterprise.

If you want to be able to create and share channels and packages from your Anaconda repository using conda commands, you need to download and install the Anaconda Enterprise CLI. If you updated the platform’s TLS/SSL certificate, you can also use the AE CLI to configure the TLS/SSL certs for the repository.

**Prerequisites:**

- Install Anaconda Enterprise
- Complete the post-install configuration
- Install conda

1. To install the CLI, run the following command, replacing `anaconda.example.com` with the fully-qualified domain name (FQDN) of your Anaconda Enterprise instance:

```
conda install -kc https://anaconda.example.com/repository/conda/anaconda-enterprise-cli cas-mirror git
```

**Note:** You’ll notice that running this command also installs `cas-mirror`, the package mirroring tool. For more information on package mirroring, see Configuring channels and packages.

2. After the list of package dependencies has been resolved, type `y` to proceed with the installation.

#### 1.9.1 Configuring the Anaconda Enterprise CLI

1. To add the url of the Anaconda repository to the set of available sites, run the following command with the fully-qualified domain name (FQDN) of your Anaconda Enterprise instance:
2. Run the following command to configure the instance of Anaconda repository you will be using as the default site:

```bash
anaconda-enterprise-cli config set default_site master
```

3. To see a consolidated view of the configuration, run the following command:

```bash
anaconda-enterprise-cli config view
```

The Anaconda Enterprise 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 it’s configured, modify the appropriate `cli.yml` file(s), based on your needs.

**Note:** Changing configuration settings at the user level overrides any system-level configuration.

If you *updated the platform’s TLS/SSL certificate*, you can use the AE CLI to configure the certificates for the repository using the following commands:

```bash
$ anaconda-enterprise-cli config set ssl_verify true
```

# On Ubuntu

```bash
$ anaconda-enterprise-cli config set sites.master.ssl_verify /etc/ssl/certs/ca-certificates.crt
```

# On RHEL/CentOS

```bash
$ anaconda-enterprise-cli config set sites.master.ssl_verify /etc/pki/tls/certs/ca-bundle.crt
```

### 1.9.2 Logging in to the Anaconda Enterprise CLI

1. Run this command to access the CLI:

```bash
anaconda-enterprise-cli login
```

2. Log in to the CLI using the same username and password that you use to log in the Anaconda Enterprise web interface:
Next Steps: You can now 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.

1.10 Installing Livy server for Hadoop Spark access

To support your organization’s data analysis operations, Anaconda Enterprise enables platform users to connect to remote Apache Hadoop or Spark clusters. Anaconda Enterprise uses Apache Livy to handle session management and communication to Apache Spark clusters, including different versions of Spark, independent clusters, and even different types of Hadoop distributions.

Livy provides all the authentication layers that Hadoop administrators are used to, including Kerberos. AE also authenticates to HDFS with Kerberos. **Kerberos Impersonation must be enabled.**

When Livy is installed, users can connect to a remote Spark cluster when creating projects by selecting the Spark template. They can either use the Python libraries available on the platform, or package a specific environment to target for the job. For more information, see *Hadoop / Spark.*

**Before you begin:**

Verify the connection requirements. The following table outlines the supported configurations for connecting to remote Hadoop and Spark clusters with Anaconda Enterprise.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop and HDFS</td>
<td>2.6.0+</td>
</tr>
<tr>
<td>Spark and Spark API</td>
<td>1.6+ and 2.X</td>
</tr>
<tr>
<td>Sparkmagic</td>
<td>0.12.7</td>
</tr>
<tr>
<td>Livy</td>
<td>0.5</td>
</tr>
<tr>
<td>Hive</td>
<td>1.1.0+</td>
</tr>
<tr>
<td>Impala</td>
<td>2.11+</td>
</tr>
</tbody>
</table>

**Note:** The Hive metastore may be Postgres or MySQL. The Livy server must run on an “edge node” or client in the Hadoop/Spark cluster. Verify that the `spark-submit` and/or the `spark repl` commands work on this machine.

**Installing Livy**

Follow the instructions below to install Livy into an existing Spark cluster, or download and install the official version of Livy.

**Note:** This example is specific to a Red Hat-based Linux distribution, with a Hadoop installation based on Cloudera CDH. To use other systems, you’ll need to look up the corresponding commands and locations.

1. Locate the directory that contains Anaconda Livy. Typically this will be `anaconda-enterprise-X.X.X-X-X/installer/anaconda-livy-0.5.0`, where `X.X-X-X` corresponds to the Anaconda Enterprise version.

2. Copy the entire directory that contains Anaconda Livy to an edge node on the Spark/Hadoop cluster.
After installing Livy server, you’ll need to configure it to work with Anaconda Enterprise. For example, you’ll need to enable impersonation, so users running Spark sessions are able to log in to each machine in the Spark cluster. For more information, see Configuring Livy server for Hadoop Spark access.

### 1.11 Upgrading Anaconda Enterprise

The process of moving from one version of Anaconda Enterprise to another varies slightly, depending on which version you are migrating to and from, so follow the instructions that correspond to your Anaconda Enterprise implementation. If you’re moving from an implementation of AE 4 to AE 5, 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 Anaconda Enterprise requires Administrators to migrate the package repository and project owners to migrate their notebooks.

Upgrading Anaconda Enterprise 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.

#### 1.11.1 Upgrading between versions of AE5

Due to the potential complexity of your custom configuration, please contact Anaconda Support before initiating the upgrade.

After you have determined the topology for your Anaconda Enterprise cluster, and verified that your system meets all of the installation requirements, you’re ready to upgrade the cluster.

**Before you begin:**

- Configure your A record in DNS for the master node with the actual domain name you will use for your Anaconda Enterprise installation.
- If you are using a firewall for network security, we recommend you temporarily disable it while you upgrade Anaconda Enterprise.
- When installing Anaconda Enterprise on a system with multiple nodes, verify that the clock of each node is in sync with the others prior to starting the installation process, to avoid potential issues. We recommend using the Network Time Protocol (NTP) to synchronize computer system clocks automatically over a network. See instructions here.
- Back up the `anaconda-enterprise-anaconda-platform.yml` file used to configure the platform, as config map settings such as external Git configuration are not automatically migrated to the new cluster as part of the upgrade process.
• Back up your custom `cas-mirror` and `anaconda-enterprise-cli` configurations (see Step 4 below), as `$HOME/cas-mirror` will be overwritten during the upgrade process. To avoid any compatibility issues, we recommend you upgrade your mirror tools as part of the upgrade process. Afterwards, simply copy over the configuration files you backed up to restore your custom configuration.

**Warning:** After the upgrade has begun, it won’t be possible to capture or back up data for any open sessions or deployments. We therefore recommend that you ask all users to save their work, stop any sessions and deployments, and log out of the platform during the upgrade window.

Anaconda Enterprise supports in-place upgrades. The upgrade process varies slightly, depending on your current version and which version you’re installing. To update an existing Anaconda Enterprise installation to a newer version, follow the process that corresponds to your particular scenario:

- **Upgrading from AE 5.4.x to 5.5.0**
- **Upgrading from AE 5.3.x to 5.5.0**
- **Upgrading from AE 5.3.0/5.3.1 to 5.4.x**
- **Upgrading from AE 5.2.x/5.3.0 to 5.3.1**

### Upgrading from AE 5.4.x to 5.5.0

1. Ensure that all AE users have closed any open sessions, stopped any deployed applications, and logged out of the platform. The `backup.sh` script that runs as part of the upgrade process will restart all pods, so if they don’t, they will lose any unsaved work.

2. On the master node running your current installation of AE, download and decompress the new installer, and then cd into the install directory, replacing `<location_of_installer>` with the location of the installer, and `<version>` with your installer version:

   ```bash
   curl -O <location_of_installer>.tar.gz
tar xvzf anaconda-enterprise-<version>.tar.gz
   ```

3. Run the following command as a pre-flight check:

   ```bash
   sudo ./gravity check
   ```

4. If the pre-flight check failed, make the applicable corrections as seen on our Installation requirements page.

5. Run the following command to start the upload and upgrade process:

   ```bash
   sudo ./upload
   sudo ./gravity upgrade
   ```

6. Depending on your implementation, the upgrade process may take an hour or more to complete. This is primarily due to the upload step. You can check the status of the upgrade process by running the following:

   ```bash
   watch sudo ./gravity status
   OR
   watch sudo ./gravity plan
   ```
7. Even after the upgrade process is completed, it will still take some time for the pods to come up. You can monitor those with the following command:

```
watch kubectl get pods
```

8. If you encounter errors while upgrading, you can check the status of the operation by running `sudo ./gravity plan`. You can then roll back any step in the upgrade process by running the `rollback` command against the name of the phase, as it’s listed in the `Phase` column:

```
sudo ./gravity rollback --phase=/<name-of-phase>
```

9. After addressing the error(s), you can resume the upgrade by running the following command:

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

After the upgrade process completes, follow the steps to verify that your upgrade was successful.

After you’ve confirmed that your upgrade was successful—and everything works as expected—you can run a script to remove images leftover from the previous installation and free up space. This will help prevent the cluster from running out of disk space on the master node.

**Upgrading from AE 5.3.x to 5.5.0**

Anaconda Enterprise 5.3.x and later support in-place upgrades, so you can follow these simple steps to update your 5.3.0 or 5.3.1 installation to the latest version.

1. Ensure that all AE users have closed any open sessions, stopped any deployed applications, and logged out of the platform. The `backup.sh` script that runs as part of the upgrade process will restart all pods, so if they don’t, they will lose any unsaved work.

2. On the master node running your current installation of AE, download and decompress the new installer, and then cd into the install directory, replacing `<location_of_installer>` with the location of the installer, and `<version>` with your installer version:

```
curl -O <location_of_installer>.tar.gz
tar xvzf anaconda-enterprise-<version>.tar.gz
```

3. Curl the shim download and untar the shim download. Then, cd into the shim:

```
curl -O <location_of_shim>.tar.gz
tar xvzf anaconda-enterprise-<version>-gravity-only.tar.gz
cd anaconda-enterprise-<version>-gravity-only
```

4. Run the following command as a pre-flight check:

```
sudo ./gravity check
```

5. If the pre-flight check failed, make the applicable corrections as seen on our Installation requirements page.

6. Run the following command to start the upload and upgrade process:

```
sudo ./upload
sudo ./gravity upgrade
```

7. Depending on your implementation, the upgrade process may take an hour or more to complete. This is primarily due to the upload step. You can check the status of the upgrade process by running the following:
watch sudo ./gravity status
OR
watch sudo ./gravity plan

8. Once the upgrade process has completed, change directory to the install directory and rerun the previous step:

cd ../anaconda-enterprise-<version>
sudo ./upgrade

9. Depending on your implementation, the upgrade process may take an hour or more to complete. This is primarily due to the upload step. You can check the status of the upgrade process by running the following:

watch sudo ./gravity status

10. Even after the upgrade process is completed, it will still take some time for the pods to come up. You can monitor those with the following command:

watch kubectl get pods

11. If you encounter errors while upgrading, you can check the status of the operation by running sudo ./gravity plan. You can then roll back any step in the upgrade process by running the rollback command against the name of the phase, as it’s listed in the Phase column:

sudo ./gravity rollback --phase=/<name-of-phase>

12. After addressing the error(s), you can resume the upgrade by running the following command:

sudo ./gravity upgrade --resume --force

After the upgrade process completes, follow the steps to verify that your upgrade was successful.

After you’ve confirmed that your upgrade was successful—and everything works as expected—you can run a script to remove images leftover from the previous installation and free up space. This will help prevent the cluster from running out of disk space on the master node.

**Upgrading from AE 5.3.0/5.3.1 to 5.4.x**

Anaconda Enterprise 5.3.0 and 5.3.1 support in-place upgrades, so you can follow these simple steps to update your 5.3.0 or 5.3.1 installation to the latest version.

1. Ensure that all AE users have closed any open sessions, stopped any deployed applications, and logged out of the platform. The backup.sh script that runs as part of the upgrade process will restart all pods, so if they don’t, they will lose any unsaved work.

2. On the master node running your current installation of AE, download and decompress the new installer, and then cd into the install directory, replacing <location_of_installer> with the location of the installer, and <version> with your installer version:

```bash
curl -O <location_of_installer>.tar.gz
tar xvzf anaconda-enterprise-<version>.tar.gz
```

3. Curl the shim download and untar the shim download. Then, cd into the shim:

```bash
curl -O <location_of_shim>.tar.gz
tar xvzf anaconda-enterprise-<version>-gravity-only.tar.gz
cd anaconda-enterprise-<version>-gravity-only
```
4. Run the following command to upload the installer to the AE environment:

```
sudo ./upload
```

5. When the upload process finishes, run the following command to start the upgrade process:

```
sudo ./gravity upgrade
```

6. Change directory to the install directory and rerun steps 4 and 5 above:

```
cd ../anaconda-enterprise-<version>
sudo ./upload
sudo ./gravity upgrade
```

7. Depending on your implementation, the upgrade process may take an hour or more to complete. You can check the status of the upgrade process by running `sudo ./gravity status`. If you encounter errors while upgrading, you can check the status of the operation by running `sudo ./gravity plan`. You can then roll back any step in the upgrade process by running the `rollback` command against the name of the phase, as it’s listed in the Phase column:

```
sudo ./gravity rollback --phase=/<name-of-phase>
```

After addressing the error(s), you can resume the upgrade by running the following command:

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

After the upgrade process completes, follow the steps to verify that your upgrade was successful.

After you’ve confirmed that your upgrade was successful—and everything works as expected—you can run a script to remove images leftover from the previous installation and free up space. This will help prevent the cluster from running out of disk space on the master node.

**Upgrading from AE 5.2.x/5.3.0 to 5.3.1**

Anaconda Enterprise 5.2.x and 5.3.0 support in-place upgrades, so you can follow these simple steps to update your 5.2.x or 5.3.0 installation to the latest version.

1. Ensure that all AE users have closed any open sessions, stopped any deployed applications, and logged out of the platform. The `backup.sh` script that runs as part of the upgrade process will restart all pods, so if they don’t, they will lose any unsaved work.

2. On the master node running your current installation of AE, download and decompress the new installer, replacing `<location_of_installer>` with the location of the installer, and `<version>` with your installer version:

```
curl -O <location_of_installer>.tar.gz
tar xzvf anaconda-enterprise-<version>.tar.gz
cd anaconda-enterprise<version>
```

3. Run the following command to upload the installer to the AE environment:

```
sudo ./upload
```

4. When the upload process finishes, run the following command to start the upgrade process:

```
sudo ./gravity upgrade
```
5. The upgrade process may take up to an hour to complete. You can check the status of the upgrade process by running `sudo ./gravity status`

If you encounter errors while upgrading, you can check the status of the operation by running `sudo ./gravity plan`. You can then roll back any step in the upgrade process by running the `rollback` command against the name of the phase, as it's listed in the `Phase` column:

```
sudo ./gravity rollback --phase=<name-of-phase>
```

After addressing the error(s), you can resume the upgrade by running the following command:

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

After the upgrade process completes, follow the steps to **verify that your upgrade was successful**.

After you’ve confirmed that your upgrade was successful—and everything works as expected—you can **run a script** to remove images leftover from the previous installation and free up space. This will help prevent the cluster from running out of disk space on the master node.

### Verify installation

After you’ve verified that all pods are running and updated the Anaconda Enterprise URLs, you can confirm that your upgrade was successful by doing the following:

1. Return to the Authentication Center and select **Users** in the **Manage** menu on the left.
2. Click **View all users** and verify that all user data has also been restored.
3. Access the Anaconda Enterprise user console by visiting this URL in your browser: `https://example.anaconda.com/`—replacing `example.anaconda.com` with the FQDN of your server—and logging in using the same credential you used in your previous installation.
4. Review the **Projects** list to verify that all project data has been restored.

**Note:** If you didn’t configure SSL certificates as part of the post-install configuration, do so now. See *Updating TLS/SSL certificates* for more information.

---

If you’re upgrading a cluster with external Git configured:

**Note:** The `git` section of the `anaconda-enterprise-anaconda-platform.yml` file used to configure Anaconda Enterprise 5.3.1 includes parameter changes. If you backed up your Anaconda Enterprise config map before upgrading, and copied it onto the newly-updated master node, you’ll need to **update your config map with the new information as described here.**
If you’re upgrading a Spark/Hadoop configuration:

After you successfully restore your Anaconda Enterprise data, run the following commands on the master node of the newly-installed Anaconda Enterprise server:

```
kubectl replace -f <path-to-anaconda-config-files-secrets.yaml>
```

To verify that your configuration upgraded correctly:

1. Log in to Anaconda Enterprise.
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. For example, run the `sc` command to connect to Sparkmagic and start Spark.

After you’ve confirmed that your upgrade was successful, we recommend you run the following command to remove all unused packages and images from previous versions of the application, and repopulate the registry to include only those images required by the current version of the application:

```
sudo gravity gc
```

The command’s progress is displayed in the terminal, so you can watch as it marks packages associated with the latest version as required, and deletes older versions.

If running the command generates an error, you can resume the command (after you fix the issue that caused the error) by running the following command:

```
sudo gravity gc --resume
```

### 1.11.2 Backing up and restoring AE

You may choose to back up AE regularly, based on your organization’s disaster recovery policies.

**Warning:** You can start the backup process while users are working; however, the backup process will not capture any open sessions or deployments. We therefore recommend that you ask all users to save their work, stop any sessions and deployments, and log out of the platform during the upgrade window if they’d like that content backed up.

This topic provides guidance on the following actions:
• Backing up
• Verify the installation
• Run the backup script
• Restore data from backup

Backing up

For the backup, there are just two bash scripts but they require `jq` so you would have to do a conda install so you get the dependencies. We recommend installing miniconda under `/opt/anaconda` to allow all AE admins to access it. This is helpful in case admins change.

Next, follow the steps in the section below if AE-master has internet access, or follow the steps under AE-master is airgapped if it’s airgapped.

AE-master has internet access

1. Install the package `ae5_backup_restore` from anaconda.org as shown below:
   
   ```
   conda create -n ae5-backup -c ae5-admin ae5_backup_restore
   conda activate ae5-backup
   ```

2. Proceed to Verify the installation below.

AE-master is airgapped

If you are air gapped, follow the below steps to download the installer and install:

1. Download and run the `ae5_backup_restore` installer.

2. If you already have miniconda installed, you can just install in an environment as follows. This will install the backup/restore scripts and the `jq` package in a conda environment:

   ```
   # These commands must be run from the master node
   chmod +x ./ae5_backup_restore-x.x.x-Linux-x86_64.sh
   ./ae5_backup_restore-x.x.x-Linux-x86_64.sh -p $CONDA_INSTALL_PREFIX/envs/ae5_
   →backup_restore
   ```

3. Proceed to Verify the installation below.

Verify the installation

Get usage help by running the following:

```ae_backup.sh -h```

Next, run the backup script.
Run the backup script

The following will create the `m_backup` directory if it does not already exist. Ensure you have `sudo` access before running, as the script needs it to copy files from `/opt/anaconda/storage`:

```
ae_backup.sh ./m_backup
```

The output will look like the following:

```
(backup-test) ubuntu@ip:/opt/anaconda$ ae_backup.sh ./m_backup
Backup AE5 to /opt/anaconda/m_backup
Backup secrets/cm/DB dump to /opt/anaconda/m_backup/ae5_config_db_202005201802.tar.gz
Backup data from /opt/anaconda/storage to /opt/anaconda/m_backup/ae5_data_202005201802.tar.gz;
excludes /opt/anaconda/storage/pgdata and /opt/anaconda/storage/object/anaconda-repository of conda packages
dump configmap (ns=default)
dump secrets of type=Opaque (ns=default)
dump custom secrets in (kube-system ns)
backup ingress - used only if swapping HOSTNAME on a running v5 system
dump DB from pod: anaconda-enterprise-postgres-7867898fbf-d519w
/opt/anaconda/m_backup /opt/anaconda
```

Note: Conda packages are not included in the backup. It is your responsibility to make a backup. These are stored in `/opt/anaconda/storage/object/anaconda-repository`.

Restore data from backup

If you backed up your Anaconda Enterprise installation, you can follow the steps below to restore the installation along with your config/DB.

Warning: This will delete user sessions from the database (DB).

1. Stop the DB. Rename the working `/opt/anaconda/storage` folder so the data remains intact. Create new `/opt/anaconda/storage` folder (ensuring permissions / owner are the same as the backup). Scale the DB back up and restart all application pods (anaconda-enterprise-ap):

```
kubectl scale --replicas=0 deploy anaconda-enterprise-postgres
# wait for postgres pod to terminate - kubectl get po /grep postgres
anaconda-enterprise-postgres-7867898fbf-d519w 1/1
   Terminating 10 7d
mv /opt/anaconda/storage /opt/anaconda/storage.working
# create new /opt/anaconda/storage folder (ensuring permissions / owner are the same as the backup)
kubectl scale --replicas=1 deploy anaconda-enterprise-postgres
```

(continues on next page)
kubectl get po | grep ap- | awk '{print $1}' | xargs kubectl delete pod

2. Once all the pods are back up, log in using default anaconda-enterprise user. Since your LDAP configs are all in the /opt/anaconda/storage.working location, LDAP login will not work. A new /opt/anaconda/storage is created that doesn’t contain any of your data/customizations.

3. Now let’s restore your config/DB as well as the data. ae_restore.sh uses sudo while copying the data back; ideally passwordless sudo should be setup for the account.

4. Continuing from the backup example, restore config/DB as well as the data:

```bash
$ ae_restore.sh m_backup/ae5_config_db_202005201802.tar.gz m_backup/ae5_data_202005201802.tar.gz
```

Restore secrets from m_backup/ae5_config_db_202005201802.tar.gz; replace ingress:

```bash
false
```

Restore AE5 data from m_backup/ae5_data_202005201802.tar.gz, to /opt/anaconda/storage. Backup current data to /opt/anaconda/storage.202005201913.

Are you sure? y

.. continue

replace secrets in kube-system ns

secret/anaconda-enterprise-certs replaced

secret/cluster-tls replaced

replace secrets in default ns

secret "anaconda-config-files" deleted

secret "anaconda-credentials-anaconda-enterprise-471gbhdt" deleted

secret "anaconda-credentials-user-creds-anaconda-enterprise-3ggji6dp" deleted

secret "anaconda-enterprise-certs" deleted

secret "anaconda-enterprise-keycloak" deleted

secret/anaconda-config-files created

secret/anaconda-credentials-anaconda-enterprise-471gbhdt created

secret/anaconda-credentials-user-creds-anaconda-enterprise-3ggji6dp created

secret/anaconda-enterprise-certs created

secret/anaconda-enterprise-keycloak created

extract and replace configmap in default namespace

configmap "anaconda-enterprise-anaconda-platform.yml" deleted

configmap "anaconda-enterprise-env-var-config" deleted

configmap "anaconda-enterprise-install" deleted

configmap "anaconda-enterprise-nginx-config" deleted

configmap "docs-nginx-config" deleted

configmap/anaconda-enterprise-anaconda-platform.yml created

configmap/anaconda-enterprise-env-var-config created

configmap/anaconda-enterprise-install created

configmap/anaconda-enterprise-nginx-config created

configmap/docs-nginx-config created

scale DB to 0 replicas and wait for it to terminate

deployment.extensions/anaconda-enterprise-postgres scaled

waiting for postgres pod to stop

waiting for postgres pod to stop

waiting for postgres pod to stop

waiting for postgres pod to stop

backup current storage

restore m_backup/ae5_data_202005201802.tar.gz

scale DB to 1 replica and wait for it to come up

(continues on next page)
deployment.extensions/anaconda-enterprise-postgres scaled
waiting for postgres pod to become ready
waiting for postgres pod to become ready
waiting for postgres pod to become ready
waiting for postgres pod to become ready
waiting for postgres pod to become ready
waiting for postgres pod to become ready
waiting for postgres pod to become ready
waiting for postgres pod to become ready
wait for postgres pod to become ready
restore full_postgres_backup.sql
Import DB into anaconda-enterprise-postgres-7867898fbf-x8hc from full_postgres_
backup.sql
ERROR: database "anaconda_auth" does not exist
ERROR: database "anaconda_auth_escrow" does not exist
ERROR: database "anaconda_deploy" does not exist
ERROR: database "anaconda_git" does not exist
ERROR: database "anaconda_operation_controller" does not exist
ERROR: database "anaconda_repository" does not exist
ERROR: database "anaconda_ui" does not exist
ERROR: database "anaconda_workspace" does not exist
ERROR: current user cannot be dropped
ERROR: role "postgres" already exists
clean up old sessions from the DB
DELETE 0
Unsure what to do w/ deployments so these may need to be restarted
restart all pods in default namespace
pod "anaconda-enterprise-ap-auth-77ccfb68b-jb9jp" deleted
pod "anaconda-enterprise-ap-auth-api-9f4474cc9-7btr" deleted
pod "anaconda-enterprise-ap-auth-escrow-f97b84696-84k57" deleted
pod "anaconda-enterprise-ap-deploy-59d45dbb5c-7xkhx" deleted
pod "anaconda-enterprise-ap-docs-6c4ccf486-f99zr" deleted
pod "anaconda-enterprise-ap-git-storage-cf855ddc9-f8xm" deleted
pod "anaconda-enterprise-ap-object-storage-cc7bf4b99-d8xm4" deleted
pod "anaconda-enterprise-ap-operation-controller-7ffcc6c657d-w4xlg" deleted
pod "anaconda-enterprise-ap-repository-685c685c76-s6xx8" deleted
pod "anaconda-enterprise-ap-storage-7b854f5f57-dp2ht" deleted
pod "anaconda-enterprise-ap-ui-7c8f8cfd49-75mgj" deleted
pod "anaconda-enterprise-ap-workspace-74ff4cdfc-hbtjc" deleted
pod "anaconda-enterprise-app-images-zttpm" deleted
pod "anaconda-enterprise-nginx-ingress-rc-fpght" deleted
pod "anaconda-enterprise-operation-images-k4t4l" deleted
pod "anaconda-enterprise-postgres-7867898fbf-x8hc" deleted
pod "anaconda-enterprise-redis-755489b7fd-8px5c" deleted

5. Wait for all pods to be in running state before testing. The ap-ui, ap-repository, ap-auth pods take some time to come back up.

Note: If the data file is not provided, then the DB is not restored; only the secrets/configmaps are restored.

6. Restore the anaconda-repository of conda packages from the backup created above:

```
sudo cp -pr /opt/anaconda/storage.working/object/anaconda-repository/* storage/
-object/anaconda-repository/
```
Warning: This version cleans out old sessions from the DB; but it doesn’t clean out deployments. You may need to manually terminate deployments since there will be no user sessions or deployments in the real backup/restore scenario.

The psql errors like `ERROR: database "anaconda_auth" does not exist` are an artifact of the testing. They result from moving the existing pgdata directory, and then letting it get recreated by scaling the DB pod; the tables do not exist in the way they would when you (re)install then restore. This error can be ignored.

### 1.11.3 Uninstalling AE

Before using the following instructions to uninstall Anaconda Enterprise, 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 Anaconda Enterprise 5.2.

To uninstall Anaconda Enterprise on a healthy cluster worker nodes, run:

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

To uninstall Anaconda Enterprise 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`).

### 1.12 Migrating from AE 4 to AE 5

The process of migrating from AE 4 to AE 5 involves the following tasks:

**For Administrators:**
- Export all packages and package info. from your AE 4 Repository.
- Import the packages into Anaconda Enterprise 5.

**For Notebook users:**
- Export each project environment to a `.yml` file.
- Convert each project into a format compatible with AE 5.
- Upload each project into AE 5.
Due to architectural changes between versions of the platform, there are some additional steps you may need to follow to migrate code between AE4 and AE5. These steps vary, based your current and new platform configurations.

1.12.1 Exporting packages

Anaconda Enterprise 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 AE 4 Repo and switch to the anaconda-server user.
2. To export your packages, run the following command on the server hosting your Anaconda Repository:

   `anaconda-server-admin export-site`

Running this command creates a directory structure containing all files and user information from your Anaconda Repository. For example:

```
site-dump/
  anaconda-user-1
     59921152446b5703f430383f--moto
     5992115f446b5703fa30383e--pysocks
     meta.json
  anaconda-organization
     5989fbd1446b575b99032652--future
     5989fc1f446b575b99032786--iso8601
     5989fc1f446b575b990327a8--simplejson
     5989fc26446b575b99032802--six
     5989fc31446b575b990328b0--xz
     5989fc35446b575b990328c6--zlib
     meta.json
  anaconda-user-2
     meta.json
```

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.json` 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.

1.12.2 Importing packages

You can choose whether to import packages into Anaconda Enterprise 5 by username or organization, or import all packages.

**Before you begin:**

- We recommend 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 Anaconda Enterprise command line interface (CLI) to import the packages you exported, so be sure to install the AE CLI if you haven’t already.
1. Log into the command line interface using the following command:

```bash
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:

```bash
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:

```bash
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 Anaconda Enterprise UI or CLI. See *Managing channels and packages* for more information.

## 1.12.3 Migrating AE 4 Notebook Projects

### Before you begin:

- If your project refers to channels in your on-premises repository or other channels in anaconda.org, ask your System Administrator to *mirror those channels* and make them available to you in AE 5.
- If your project uses non-conda packages, you’ll need to *upload those packages* to AE 5.
- 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.

### 1.12.4 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 Notebooks server.
2. Open a terminal window and activate conda environment 2.6 for your project.
3. Install *anaconda project* in the environment:

   ```bash
   conda install anaconda-project=0.6.0
   ```

   If you get a *not found* message, install it from anaconda.org:

   ```bash
   conda install -c anaconda anaconda-project=0.6.0
   ```

4. Export your environment to a file:

   ```bash
   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
   - 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:
1.12.5 Converting your project

To create a project that’s compatible with Anconda Enterprise 5, perform these steps:

1. Run the following command from an interactive shell:

   ```bash
   anaconda-project init
   ```

2. AE 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:

   ```bash
   anaconda-project remove-platforms win-64 osx-64
   ```

3. Run the following command to verify the platforms were removed:

   ```bash
   anaconda-project list-platforms
   ```


5. Run the following command to compress your project:

   ```bash
   anaconda-project archive FILENAME.tar.gz
   ```

**Note:** There is a 1GB file size limit for project files, and project names cannot contain spaces or special characters.

6. In Anaconda Enterprise Notebooks, from your project home page, open the Workbench. Locate your project file (e.g., `AENProject.tar.gz` in the image below) in the file list, right-click and select **Download**.

![AEN Workbench](image)

Now your project is ready to be uploaded into Anaconda Enterprise 5.

1.12.6 Uploading your project to AE 5

Log in to the Enterprise v5 interface and upload your project 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. Anaconda Enterprise projects are based on Git, so we recommend 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 we recommend using storage solutions designed for that type of data, and connecting to those data sources from within your Anaconda Enterprise sessions.

1.12.7 Migrating code

AE4 and AE5 are based on a different architecture, therefore some code inside your AE4 notebooks might not run as expected in AE5. AE4 sessions ran directly on the host filesystem, where the libraries, drivers, packages, and connectors required to run them were available. AE5 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, we recommend 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 AE5, see Connecting to the Hadoop and Spark ecosystem.

<table>
<thead>
<tr>
<th>Service/System</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Impala</td>
<td>impyla</td>
</tr>
<tr>
<td>Apache Hive</td>
<td>pyhive</td>
</tr>
<tr>
<td>Oracle</td>
<td>build conda package with their driver</td>
</tr>
</tbody>
</table>

If this is not possible, we recommended 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. AE5 sessions and deployments run in isolated containers, so users cannot use this method to access shared libraries.

Instead, we recommend you create a conda package for each library. This enables you to control access to each package library and version it—both essential to managing software at the enterprise level.

After you create the package, upload it to the internal AE5 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 Anaconda Enterprise 5. 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 Anaconda Enterprise repository, search anaconda.org or build your own conda package using conda-build then upload the conda package to the AE5 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.
There are several aspects of Anaconda Enterprise 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:
2.1 Managing cluster resources

After you’ve installed an Anaconda Enterprise cluster, you’ll want to continue to manage and monitor the cluster to ensure that it scales with your organization as needs change. These on-going management and monitoring tasks include the following:

- When you’ve outgrown your initial Anaconda Enterprise cluster installation, you can easily add new nodes—including GPUs. To make these nodes available to platform users, you’ll configure resource profiles.

- To help you manage your organization’s cluster resources more efficiently, Anaconda Enterprise enables you to track 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.

- To help you gain insights into user services and troubleshoot issues, Anaconda Enterprise 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 Anaconda Enterprise for information about what to do if a master node fails.

2.1.1 Adding and removing nodes

You can view, add, edit and delete server nodes from Anaconda Enterprise using the Admin Console’s Operations Center. If you would prefer to use a command line to join additional nodes to the AE master, follow the instructions provided below.

NOTES:

- Each installation can only support a single AE master node, as this node includes storage for the platform. DO NOT add an additional AE master node to your installation.

- As a best practice for etcd optimal cluster size, we recommend you add any additional Kubernetes master nodes in pairs, so that the total number (including the AE master) is an odd number.

- Anaconda Enterprise doesn’t 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**.

To manage the servers on your system:

1. Log in to Anaconda Enterprise, 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 Anaconda Enterprise:

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

   ![Add Node Button](image)

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

```
gravity status
```

The results should look similar to the following:

4. Copy and paste the join token for the cluster and the IP address for the AE 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 AE 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 AE 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:
6. To monitor the impact of the join operation on the cluster, run the `gravity status` command on the AE master.

The output will look similar to the following:

```
Cluster status: active
Application: AnacondaEnterprise, version 5.4.0-36.gdf45d6d16
Join token: 9954b79f357bbef78d2a40a4c8d9e6
Periodic updates: Not Configured
Remote support: Not Configured
Active operations:
  * operation_expand (27542137-ee9d-492c-8740-8f806d878fd6)
    started: Tue Oct 29 10:09 UTC (7 seconds ago)
    all packages on the joining node 2048 complete
Last completed operation:
  * operation_install (38c2b63a-4ff4-4b06-8096-9b6d439068c4)
    started: Tue Oct 29 14:25 UTC (1 hour ago)
    completed: Tue Oct 29 14:30 UTC (1 hour ago)
Cluster endpoints:
  * Authentication gateway:
    172.31.67.113:32089
  * Cluster management URL:
    https://172.31.67.113:32089
Cluster nodes: test-install.training.anacnado.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
      status: offline
```

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:
2.1.2 Setting resource limits for sessions and deployments

**Note:** You can separate system-level pods from user-level sessions and deployments as long as you have a multi-node setup (that is, a master node and at least one worker node). Contact support to complete this operation.

Each project editor session and deployment uses compute resources on the Anaconda Enterprise cluster. If Anaconda Enterprise users need to run applications which require more memory or compute power than provided by default, you can customize your installation to include these resources and allow users to access them while working within AE.

After the server resources are installed as nodes in the cluster, you create custom resource profiles to configure the number of cores and amount of memory/RAM available to users—so that it corresponds to your specific system configuration and the needs of your users.

For example, if your installation includes nodes with GPUs, add a GPU resource profile so users can use the GPUs to accelerate computation within their projects—essential for machine learning model training. For installation requirements, see *Installation requirements*.

Resource profiles apply to all nodes, users, editor sessions, and deployments in the cluster. So if your installation includes nodes with GPUs that you want to make available for users to accelerate computation within their projects, you’d create a GPU resource profile. Any resource profiles you configure are listed for users to select from when configuring a project and deploying a project. Anaconda Enterprise finds the node that matches their request.

To add a resource profile for a resource you have installed:

1. Log in to Anaconda Enterprise, 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.
6. Make a manual backup copy of this file before editing it, as any changes you make will impact how Anaconda Enterprise functions.
7. Scroll down to the **resource-profiles** section:
8. Add an additional resource following the format of the default specification. For example, to create a GPU resource profile, add the following to the `resource-profiles` section of the Config map:

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

By default, CPU sessions and deployments are also allowed to run on GPU nodes. *To reserve GPU nodes for only those sessions and deployments that require a GPU*—by preventing CPU sessions and deployments from accessing GPU nodes—comment out the following additional specification included after the `gpu-profile` entry:

![Code Snippet]

**Note:** Resource profiles are listed in alphabetical order—after any defaults—so if you want them to appear in a particular order in the drop-down list that users see, be sure to name them accordingly.

9. Click **Apply** to save your changes.

To update the Anaconda Enterprise server with your changes, you’ll need to do the following:

Restart the workspace and deploy services by running the following command:
Then check the project Settings and Deploy UI to verify that each resource profile you added or edited appears in the Resource Profile drop-down menu.

### 2.1.3 Monitoring cluster utilization

Anaconda Enterprise 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 Anaconda Enterprise, 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 Anaconda Enterprise 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.

### 2.1.4 Monitoring sessions and deployments

Anaconda Enterprise 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 Anaconda Enterprise, 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 AE5 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 Anaconda Enterprise 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:

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

To view CPU and memory utilization per pod run:

```bash
kubectl top pods --heapster-namespace=monitoring
```

### 2.1.5 Viewing system logs

To help you gain insights into user services and troubleshoot issues, Anaconda Enterprise 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 Anaconda Enterprise, 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-XXXXX**.

1. 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 Anaconda Enterprise 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.

### 2.1.6 Viewing activity logs

Anaconda Enterprise 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 Anaconda Enterprise, 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 Anaconda Enterprise provides a few options to make it more manageable:

1. Click the **Config** tab to configure the type of events you want Anaconda Enterprise 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.

### 2.1.7 Fault tolerance in Anaconda Enterprise

Anaconda Enterprise 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 Anaconda Enterprise is deployed to a cluster with *three or more nodes*, the core services are automatically configured into a fault tolerant mode—whether Anaconda Enterprise 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:

- Anaconda Enterprise 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 Anaconda Enterprise cluster is described in *Adding and removing nodes*.

**Storage and persistency layer**

Anaconda Enterprise 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 Anaconda Enterprise 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 Anaconda Enterprise services are resilient to master node failure.

To maintain operation of Enterprise 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 AE* for more information.

**To restore Anaconda Enterprise 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 AE*.

### 2.2 Configuring user access

As an Administrator, you’ll need to authorize users so they can use Anaconda Enterprise. 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 Anaconda Enterprise. Follow these instructions to grant Admins permission to manage AE users.

#### 2.2.1 Connecting to external identity providers

Anaconda Enterprise 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 Anaconda Enterprise, 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 ldap 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 `sAMAccountName` 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 Anaconda Enterprise 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 AE5 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 Anaconda Enterprise 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:
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.

**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 “AE5”, and users with the LDAP-synced role “Publisher”, permission to deploy apps, the deploy section would look like this:

```yaml
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
  hosts:
    - abc.demo.anaconda.com
  db:
    database: anaconda_deploy
  users: '*'
```

(continues on next page)
deployers:
  users: []
groups:
  - developers
  - AE5
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 Anaconda Enterprise.

**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 -name 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 Anaconda Enterprise Auth service:

1. Export the existing SSL certificates for your system by running the following commands:

   sudo gravity enter
ekubectl 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:

   echo " ldaps.jks: \$(base64 -i --wrap=0 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 Anaconda Enterprise 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:

```bash
sudo gravity enter
kubectl get pods | grep ap-auth | cut -d ' ' -f1 | xargs kubectl delete pods
```

### 2.2.2 Managing users

Managing access to Anaconda Enterprise 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.

This topic provides guidance on the following actions:

- Creating and configuring users
- Add a new master realm admin user/Reset password
- Enabling user registration
- Enabling required actions
- Impersonating users

---

**Creating and configuring users**

**To access the Authentication Center:**

1. Login to Anaconda Enterprise, 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 Anaconda Enterprise user, click Add user and specify a user name—and optionally provide values for the other fields—before clicking Save.

Warning: User names 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 Anaconda Enterprise 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 Anaconda Enterprise.

• 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 AE 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:

```
# Replace <USERNAME> with your username and <PASSWORD> with your password
/opt/jboss/keycloak/bin/add-user-keycloak.sh -u <USERNAME> -p <PASSWORD> -r master --roles=admin
```

3. Restart the server. Restarting the server will delete the container and any current state:

```
/opt/jboss/keycloak/bin/standalone.sh -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 Anaconda Enterprise:

**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.

![Impersonate command](image1.png)

Click **Impersonate** to display a list of applications the user has accessed on the platform, including editor sessions and deployments.

![Applications list](image2.png)

Click the **Anaconda Platform** link to interact with Anaconda Enterprise as the impersonated user.

![Application details](image3.png)
**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.

### 2.2.3 Managing roles and groups

Assigning access and permissions to individual users can be too fine-grained and cumbersome for organizations to manage, so Anaconda Enterprise enables you to assign access permissions to specific *roles*, then use *groups* to assign one or more roles to sets of users. Users inherit the attributes and role mappings assigned to each group they are members of—whether multiple or none.

The use of groups to assign permissions is entirely optional, so you can rely solely on roles to assign users permission to perform certain actions in Anaconda Enterprise.

- You’ll use the Admin Console’s Authentication Center to *create and manage roles and groups*. This includes creating new roles and groups, configuring defaults for each, and assigning roles to groups.
- You’ll use the Admin Console’s Operations Center to *configure permissions for any roles you create*, and optionally the default system roles provided by Anaconda Enterprise.

**Note:** When naming users and groups that you create, consider that Anaconda Enterprise users can add collaborators by user or group name when *sharing their projects* and *deployments*, as well as *packages and channels*.

To access the Authentication Center:

1. Login to Anaconda Enterprise, 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 configured after installation.

To manage roles:

Use roles to authorize individual or groups of users to perform specific actions within Anaconda Enterprise. Default roles allow you to automatically assign user role mappings when any user is newly created or imported (for example, through **LDAP**).

You’ll use the Authentication Center to configure new roles and specify default roles to be automatically added to all new user accounts.

1. In the Configure menu on the left, click **Roles** to display a list of roles configured for use with Anaconda Enterprise.

   To get you started, Anaconda Enterprise provides a set of “realm” roles. You can use these system roles as is, or as a basis for creating your own.

   ![Roles](image)

   **ae-admin**  Allows a user to access the Administrative console.
   
   **ae-creator**  Allows a user to create new projects.
   
   **ae-deployer**  Allows a user to create new deployments from projects.
   
   **ae-uploader**  Allows a user to upload packages.

2. To create a new role, click **Add Role** on the **Realm Roles** tab.
3. Enter a name and description of the role, and click **Save**.

   **Note:** Roles can be assigned to users automatically or require an explicit request. If a user has to explicitly request a realm role, enable the **Scope Param Required** switch. The role must then be specified using the **scope** parameter when requesting a token.

   The new role is now available to be used as a default role, or to be assigned to groups of users.

4. To configure default roles, click the **Default Roles** tab.

   When working with the **AnacondaPlatform** realm, you can configure default roles for **Anaconda Enterprise users** using the list of available and default **Realm Roles**.

**2.2. Configuring user access**
You’ll want to disable default roles (except for `offline_access` and `uma_auth`, as those are system config requirements) before mapping out specific roles, as not doing so may prevent your role mapping from working. The roles should look like the following when you’ve finished disabling those roles:

```
Roles
Realm Roles  Default Role
Available Roles:
- admin
- developer
- everyone
- user

Client Roles:
- offline_access
- uma_auth
```

To manage groups:

1. In the Manage menu on the left, click **Groups** to display a list of groups configured for use with Anaconda Enterprise.

   To get you started, Anaconda Enterprise provides a set of default groups, with different role mappings for each. You can use these defaults as is, or as a basis for creating your own. Default groups allow you to automatically assign group membership whenever a new user is created or imported.

2. Double-click the name of a group to view information about the group and modify it:
   - Use the **Role Mappings** tab to assign roles to the group from the list of available **Realm Roles** and **Client Roles**. See managing roles for information on how to create new roles. Permission to perform certain actions in Anaconda Enterprise are based on a user’s role, so you can grant permissions to a group of users by mapping the associated role(s) with the group. See the section below for the steps to configure permissions by role.
   - Use the **Members** tab to view all users who currently belong to the group. You add users to groups at the user level using the **Groups** tab for the user. See managing users for more information.
   - Use the **Permissions** tab to enable a set of fine grain permissions to use to define policies for allowing Admin users to manage the group. See the section below to understand how to configure permissions by role.
To configure permissions for roles:

1. Log in to Anaconda Enterprise, 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 to display the config map for Anaconda Enterprise.

**Note:** If *anaconda-platform.yml* is not displayed, be sure *anaconda-enterprise-anaconda-platform.yml* is selected in the **Config maps** drop-down list.

The following sections of the config map have permissions associated with them:

- **deploy:deployers**—used to configure which users can deploy projects
- **workspace:users**—used to configure which users can open project sessions
- **storage:creators**—used to configure which users can create projects
- **repository:uploaders**—used to configure which users can upload packages to the AE repository

5. **Save a copy of this file before making any changes** to *anaconda-platform.yml*. Any changes you make to the platform configuration will impact how Anaconda Enterprise functions, so you’ll want to have a backup if the need to restore a previous configuration arises.
6. Add each new role you create to the appropriate section—based on the permission you want to grant the role—and click **Apply** to save your changes.

For example, if you create a new role called `ae-managers`, and you want users with this role to be able to deploy applications, you need to add that role to the list of roles under `deploy:deployers` to map the permission to the role.

### 2.2.4 Managing System Administrators

Anaconda Enterprise distinguishes between System Administrators responsible for authorizing AE platform users, and System Administrators responsible for managing AE 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 AE 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 Anaconda Enterprise, 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 Anaconda Enterprise master node, replacing `<email>` and `<yourpass>` with the email address and password for the user:

```
sudo gravity enter
gridway --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
gridway --insecure user delete --email=<email> --ops-url=https://gravity-site.kube-
system.svc.cluster.local:3009
gridway --insecure user create --type=admin --email=<email> --password=<yourpass> --
ops-url=https://gravity-site.kube-system.svc.cluster.local:3009
```

### 2.2.5 Configuring session timeouts

As an Administrator, you can configure session timeouts for Anaconda Enterprise 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 Anaconda Enterprise, 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

<table>
<thead>
<tr>
<th>Configuration option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revoke Refresh Token</td>
<td>If enabled, limits refresh tokens to one-time use</td>
</tr>
<tr>
<td>SSO Session Idle</td>
<td>User will be logged out of session if inactive for this length of time</td>
</tr>
<tr>
<td>SSO Session Max</td>
<td>Maximum time a user session can remain active, regardless of activity</td>
</tr>
<tr>
<td>Offline Session Idle</td>
<td>Amount of time an offline session can be idle before the access token is revoked</td>
</tr>
<tr>
<td>Access Token Lifespan</td>
<td>Amount of time an access token will remain valid, before expiring</td>
</tr>
<tr>
<td>Access Token Lifespan For Implicit Flow</td>
<td>Timeout for access tokens created with Implicit Flow–no refresh token is provided</td>
</tr>
<tr>
<td>Client login timeout</td>
<td>Maximum time a client can take to complete the authorization process</td>
</tr>
<tr>
<td>Login timeout</td>
<td>Maximum time a user can take to authenticate before the process restarts</td>
</tr>
<tr>
<td>Login action timeout</td>
<td>Maximum time a user can spend on any one page in the authentication process</td>
</tr>
<tr>
<td>User-Initiated Action Lifespan</td>
<td>Maximum time before a user-initiated action (e.g., forgot password email) expires</td>
</tr>
<tr>
<td>Default Admin-Initiated Action Lifespan</td>
<td>Maximum time before an admin-initiated action (e.g., issue token to user) expires</td>
</tr>
<tr>
<td>Override User-Initiated Action Lifespan</td>
<td>Use to optionally configure different timeouts for each user-initiated action</td>
</tr>
</tbody>
</table>

7. Click **Save** to save your changes to the Anaconda Enterprise platform.

### 2.2.6 LDAP setup example

Configuring identity and access management is complex, and each enterprise has a different LDAP directory structure. While your implementation will be based on the specific structure and needs of your organization, the principles and processes outlined here will enable you to:

- **Reduce the number of users** that need to be mapped into Anaconda Enterprise (by mapping a functional
role—AE5 User—to an LDAP group). This also simplifies license management through a single group membership.

- **Reduce the number of groups** that are mapped into Anaconda Enterprise (by filtering groups to include only relevant functional roles and team memberships).

- **Automate the import of new groups** for team memberships based on filters.

- **Automate the provision of AE5 roles to users** based on group membership of functional roles.

Roles are used to determine the types of objects in Anaconda Enterprise that users with the role can access using the platform, such as packages or projects. This example is provided to help guide you through the process of mapping default Anaconda Enterprise roles to the following common functional business roles:

- Business Analyst
- Data Scientist
- Data Engineer
- DevOps
- Administrator

Follow the general processes outlined below for your specific implementation:

1. *Retrieving directory structures and user attributes*
2. *Setting up user federation*
3. *Testing your identity provider setup*
4. *Configuring group mappers*
5. *Mapping group roles*

### Retrieving directory structures and user attributes

The organizational structure of your enterprise is represented in LDAP by a directory structure or tree. You’ll need to request the bind user credentials from your Security Administrator.

While you can make assumptions about the directory structure based on the bind user credentials, it’s extremely difficult to setup an identity provider without the complete structure. For example, if the bind user credentials are `uid=binduser, cn=users, cn=accounts, dc=tools, dc=continuum, dc=io`, we can deduce that the root or base of the tree is `dc=tools, dc=continuum, dc=io`.

Tools are available to help you visualize your organization’s directory structure. For example, `phpldapadmin` generated the following view:
The rest of the bind user credentials become apparent after looking at the directory structure. In this example, we can see that users live under `cn=accounts > cn=users`, and groups live under `cn=accounts > cn=groups`.

Now that you know the directory structure, you can gather information about the user and group entries that you’ll need later.

You can use the `ldapsearch` tool—along with the bind user credentials—to learn details about an individual user based on their uid. Here’s a sample command for the user `gandalf`:

```bash
ldapsearch -D 'uid=binduser,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io' -W -H ldap://ipa.tools.continuum.io -b dc=tools,dc=continuum,dc=io "(uid=gandalf)"
```

Results will resemble the following:

```
# gandalf, users, compat, tools.continuum.io
dn: uid=gandalf,cn=users,cn=compat,dc=tools,dc=continuum,dc=io
objectClass: posixAccount
objectClass: ipaOverrideTarget
objectClass: top
gecos: gandalf the grey
cn: gandalf the grey
uidNumber: 1666600031
gidNumber: 1666600031
loginShell: /bin/sh
homeDirectory: /home/gandalf
ipaAnchorUUID:: OklQQTp0b29scy5jb250aW51dW0uaW86OTEyYTMwNjgtZDhmYy0xMWU4LTgzYTUtMTIyYTE3YWN1MzJh
uid: gandalf
# gandalf, users, accounts, tools.continuum.io
```

(continues on next page)

2.2. Configuring user access
Within these results, you’ll find the information you need to set up user federation for LDAP.

### Setting up LDAP user federation

You’ll use the Anaconda Enterprise Administrative Console’s Authentication Center to add LDAP as your identity provider:

1. Login to Anaconda Enterprise, 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 **User Federation**.

4. Select **ldap** from the Add provider selector to display the Add user federation provider **Required Settings**.

5. Configure the fields as follows: (Bold items are described in more detail below the table.)
### Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>ON</td>
</tr>
<tr>
<td>Console Display Name</td>
<td>ldap(tools.continuum.io)</td>
</tr>
<tr>
<td>Priority</td>
<td>0</td>
</tr>
<tr>
<td>Import Users</td>
<td>ON</td>
</tr>
<tr>
<td>Edit Mode</td>
<td>READ_ONLY</td>
</tr>
<tr>
<td>Sync Registration</td>
<td>OFF</td>
</tr>
<tr>
<td>Vendor</td>
<td>Red Hat Directory Server</td>
</tr>
<tr>
<td>Username LDAP attribute</td>
<td>uid</td>
</tr>
<tr>
<td>RDN LDAP attribute</td>
<td>uid</td>
</tr>
<tr>
<td>UUID LDAP attribute</td>
<td>uidNumber</td>
</tr>
<tr>
<td>User Object Classes</td>
<td>person,organizationalperson/inetorgperson</td>
</tr>
<tr>
<td>Connection URL</td>
<td>ldap://ipa.tools.continuum.io:389</td>
</tr>
<tr>
<td>Users DN</td>
<td>cn=users,cn=accounts,dc=tools,dc=continuum,dc=io</td>
</tr>
<tr>
<td>Authentication Type</td>
<td>simple</td>
</tr>
<tr>
<td>Bind DN</td>
<td>uid=binduser,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io</td>
</tr>
<tr>
<td>Custom User LDAP Filter</td>
<td>(&amp;(objectClass=person)(uid=*)(memberOf=cn=grp-ae5-user,cn=groups,cn=accounts,dc=tools,dc=continuum,dc=io))</td>
</tr>
<tr>
<td>Search Scope</td>
<td>One level</td>
</tr>
<tr>
<td>Validate Password Policy</td>
<td>OFF</td>
</tr>
<tr>
<td>User Truststore SPI</td>
<td>Only for ldaps</td>
</tr>
<tr>
<td>Connection Pooling</td>
<td>ON</td>
</tr>
<tr>
<td>Connection Timeout</td>
<td>Read Timeout</td>
</tr>
<tr>
<td>Pagination</td>
<td>ON</td>
</tr>
<tr>
<td>Allow Kerberos authentication</td>
<td>OFF</td>
</tr>
<tr>
<td>User Kerberos for Password Authentication</td>
<td>OFF</td>
</tr>
<tr>
<td>Batch Size</td>
<td>1000</td>
</tr>
<tr>
<td>Periodic Full Sync</td>
<td>OFF</td>
</tr>
<tr>
<td>Periodic Changed Users Sync</td>
<td>OFF</td>
</tr>
<tr>
<td>Cache Policy</td>
<td>DEFAULT</td>
</tr>
</tbody>
</table>

### Vendor

When you select a vendor from the drop-down list, default values for the most commonly used attributes will be prefilled. Be sure to select the correct one, and note that the default values may not match the way your organization has set up their application. Our example uses Red Hat Directory Server, which is based on Free IPA.

#### Username, RDN, UUID, User Object Classes, Users DN and Bind DN

Locate the values for these fields in the results of the `ldapsearch` command you ran previously. The following table outlines how the fields map to the relevant values from our gandalf user example:

<table>
<thead>
<tr>
<th>Field</th>
<th>LDAP Search Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>uid: gandalf</td>
<td>The unique ID used to identify the user.</td>
</tr>
<tr>
<td>RDN</td>
<td>uid: gandalf</td>
<td>Usually the same as the Username, but may default to something else depending on the vendor selected</td>
</tr>
<tr>
<td>UUID</td>
<td>uidNumber: 1666600031</td>
<td>Unique identifier</td>
</tr>
<tr>
<td>User Object Classes</td>
<td>objectClass: person objectClass: organizationalperson objectClass: inetorgperson</td>
<td>User object classes combined in a single field</td>
</tr>
<tr>
<td>Users DN</td>
<td>dn: uid=gandalf,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io</td>
<td>The dn less the uid entry</td>
</tr>
<tr>
<td>Bind DN</td>
<td>uid=binduser,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io</td>
<td>Usually provided by Security Admin</td>
</tr>
</tbody>
</table>

#### 2.2. Configuring user access
Custom User LDAP Filter

You can use a custom filter to restrict which users are returned from LDAP. In this case, we want only those persons (objectClass=person) with any uid (uid=*) that are a member of group grp-ae5-user (memberOf=cn=grp-ae5-user,cn=groups,cn=accounts,dc=tools,dc=continuum,dc=io). No other users will be able to log in, thereby preventing unauthorized access. This is also useful for managing licences, as users will have to be explicitly added to this group to be able to access the platform.

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.

Note: Avoid the temptation to add new groups into the Custom User LDAP Filter. LDAP search criteria are notorious for their complexity, and if it’s implemented incorrectly, all user access could be suspended or functionality disabled.

Testing your provider setup

Use the Test connection and Test authentication buttons to verify that the platform can connect to the provider with the credentials provided. You’ll need to resolve any errors before continuing.

By default, users will not be synced from LDAP until they log in. To test whether the Custom User LDAP Filter is working correctly, you can 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 the Required Settings, the provider is listed under User Federation:

Configuring group mappers

After you have successfully set up user federation, set up a group mapper for your identify provider using the Mappers tab. For example, you can create one called ldap-group-mapper and configure it based on the results generated by the ldapsearch command. In this case, we ran the command against a known group to retrieve additional information needed:

```bash
ldapsearch -D 'uid=binduser,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io' -W -H ldap://ipa.tools.continuum.io -b dc=tools,dc=continuum,dc=io "(cn=grp-ae5-user)"
```

With the following results:

```
# grp-ae5-user, groups, compat, tools.continuum.io
dn: cn=grp-ae5-user,cn=groups,cn=compat,dc=tools,dc=continuum,dc=io
objectClass: posixGroup
objectClass: ipaOverrideTarget
objectClass: ipaexternalgroup
objectClass: top
gidNumber: 1666600026
```

(continues on next page)
memberUid: czhang
cmemberUid: dlawrence
memberUid: edill
memberUid: escissorhands
memberUid: gcavanaugh
memberUid: jsandhu
memberUid: rbarthelmie
memberUid: vghadban
memberUid: gandalf

ipaAnchorUUID:: OklQQTip0b29scy5jb250aW51dW0uaW86NGFhOTQ4NzYtZDg4YS0xMWU4LWE2ZD
tcMTIyYTE3YWNlMzJh
cn: grp-ae5-user

# grp-ae5-user, groups, accounts, tools.continuum.io
dn: cn=grp-ae5-user,cn=groups,cn=accounts,dc=tools,dc=continuum,dc=io
objectClass: top
objectClass: groupofnames
objectClass: nestedgroup
objectClass: ipausergroup
objectClass: ipaobject
objectClass: posixgroup
cn: grp-ae5-user
ipaUniqueID: 4aa94876-d88a-11e8-a6d7-122a17ace32a
gidNumber: 1666600026
member: uid=czhang,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io
member: uid=dlawrence,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io
member: uid=edill,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io
member: uid=escissorhands,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io
member: uid=gcavanaugh,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io
member: uid=jsandhu,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io
member: uid=rbarthelmie,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io
member: uid=vghadban,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io
member: uid=gandalf,cn=users,cn=accounts,dc=tools,dc=continuum,dc=io

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2

2.2. Configuring user access

101
<table>
<thead>
<tr>
<th>Field</th>
<th>LDAP Search Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name *</td>
<td>ldap-group-mapper</td>
</tr>
<tr>
<td>Mapper Type</td>
<td>group-ldap-mapper</td>
</tr>
<tr>
<td>LDAP Groups DN</td>
<td>cn=groups, cn=accounts, dc=tools, dc=continuum, dc=io</td>
</tr>
<tr>
<td>Group Name LDAP Attribute</td>
<td>cn</td>
</tr>
<tr>
<td>Group Object Classes</td>
<td>groupOfNames</td>
</tr>
<tr>
<td>Preserve Group Inheritance</td>
<td>ON</td>
</tr>
<tr>
<td>Ignore Missing Groups</td>
<td>OFF</td>
</tr>
<tr>
<td>Membership LDAP Attribute</td>
<td>member</td>
</tr>
<tr>
<td>Membership Attribute Type</td>
<td>DN</td>
</tr>
<tr>
<td>Membership User LDAP Attribute</td>
<td>uid</td>
</tr>
<tr>
<td>LDAP Filter</td>
<td>(cn=grp-ae5*)</td>
</tr>
<tr>
<td>Mode</td>
<td>READ_ONLY</td>
</tr>
<tr>
<td>User Groups Retrieve Strategy</td>
<td>LOAD_GROUPS_BY_MEMBER_ATTRIBUTE</td>
</tr>
<tr>
<td>Member-Of LDAP Attribute</td>
<td>memberOf</td>
</tr>
<tr>
<td>Mapped Group Attributes</td>
<td></td>
</tr>
<tr>
<td>Drop non-existing groups during sync</td>
<td>OFF</td>
</tr>
</tbody>
</table>

**Note:** Avoid the temptation to add new groups into the LDAP Filter in the Group Mapper. LDAP search criteria are notorious for their complexity, and if it's implemented incorrectly all user access could be suspended or functionality disabled.

**LDAP Groups DN**

Derived from the `ldapsearch` field: `dn: cn=grp-ae5-user,**cn=groups,cn=accounts,dc=tools,dc=continuum,dc=io**

**Group Name LDAP Attribute**

Derived from the `ldapsearch` field: `cn: grp-ae5-user`

**Group Object Classes**

A default should have been selected. In this case it is `objectClass: groupOfNames`.

**LDAP Filter**

All relevant groups—whether they are based on functional role or team membership—have been set up with the prefix `grp-ae5-`. This prefix is used to filter the relevant groups from the User Federation provider, preventing any unnecessary groups from being pulled into the AE platform.

For example, the user Gandalf is a member of the following groups:

```
memberOf: cn=ipausers,cn=groups,cn=accounts,dc=tools,dc=continuum,dc=io
memberOf: cn=grp-ae5-user,cn=groups,cn=accounts,dc=tools,dc=continuum,dc=io
memberOf: cn=grp-ae5-wizards,cn=groups,cn=accounts,dc=tools,dc=continuum,dc=io
memberOf: cn=grp-lord-of-the-rings,cn=groups,cn=accounts,dc=tools,dc=continuum
```

If you perform a group synchronisation, only the groups in **bold** will be imported. Additionally, when Gandalf logs in, only the `grp-ae5-wizards` group from his profile will be imported. You can test this by deleting the `grp-ae5-wizards` group, then login as the user gandalf. His team membership group `grp-ae5-wizards` will be visible in the Auth Center, but the group `grp-lord-of-the-rings` will be filtered out and therefore not imported.
Mapping group roles

As a final step, you can map Anaconda Enterprise roles to the LDAP groups that are imported into the platform.

In this example, we’ll assign functional role groups the default roles that will allow them to interact with the platform in a way that makes sense for the business. You can also create custom roles, if needed.

<table>
<thead>
<tr>
<th>LDAP Group</th>
<th>ae-admin</th>
<th>ae-creator</th>
<th>ae-deployer</th>
<th>of-line</th>
<th>uma-access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grp-ae5-biz-analyst</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Business Analysts can access the system. They cannot create projects or grant others access to the system.</td>
</tr>
<tr>
<td>grp-ae5-data-scientist</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Data Scientists can create and share projects, but cannot deploy them.</td>
</tr>
<tr>
<td>grp-ae5-data-engineer</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Data Engineers can additionally deploy projects, as well as grant access to others.</td>
</tr>
<tr>
<td>grp-ae5-devops</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>DevOps can deploy projects and upload packages, but cannot create projects.</td>
</tr>
<tr>
<td>grp-ae5-sec-admin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This group should be used to administer user access within the system. Therefore, no roles should be defined in the AnacondaPlatform realm. If required, roles can be defined and access granted in the Auth Center Master realm.</td>
</tr>
<tr>
<td>grp-ae5-sysadmin</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>By default, the ae-admin role is a superuser for all other roles.</td>
</tr>
<tr>
<td>grp-ae5-sysacct</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The roles for system accounts are yet to be defined. These could be used for automated CI/CD tasks.</td>
</tr>
<tr>
<td>grp-ae5-user</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This is used as a coarse-grained control for access to AE5, so no roles are defined.</td>
</tr>
<tr>
<td>grp-ae5-wizards</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This is a team membership role, so no AE roles are defined for it.</td>
</tr>
</tbody>
</table>

**Note:** Functional role groups should be setup once and left alone.

Use the Role Mappings tab to assign the appropriate role(s) to each group:
2.2.7 Google IAM setup example

In addition to providing out-of-the-box support for LDAP, Active Directory, SAML and Kerberos, Anaconda Enterprise 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.
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](image)

4. Click **What credentials do I need?** to create the appropriate credentials for the platform.

5. Enter a meaningful name, such as **Anaconda Enterprise**, 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 Anaconda Enterprise server instance.

7. **Open the Anaconda Enterprise 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 **Anaconda Enterprise** (or something else meaningful to platform users).

• Optionally, upload a logo to help users recognize **Anaconda Enterprise**.

• 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 Enterprise**.

• Optionally provide authorized links to your organization’s privacy policy and terms of service.

9. Click **Create** to display the OAuth client credentials that you’ll need to copy and paste into **Anaconda Enterprise**, 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 Anaconda Enterprise, you need to use the Anaconda Enterprise Administrative Console’s Authentication Center to configure Google as your external identity provider:

1. Login to Anaconda Enterprise, 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`.

5. 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 Enterprise 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 AE users. As an Administrator, you can then configure their group assignments and role mappings. For more information, see *Managing roles and groups*.

### 2.3 Configuring channels and packages

Anaconda Enterprise enables you to distribute software through the use of **channels** and **packages**. Channels represent locations of repositories where Anaconda Enterprise 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:** Anaconda Enterprise 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 Anaconda Enterprise to point conda to an on-premises repository, or use a proxy for conda packages.

2.3.1 Accessing remote package repositories

As an Administrator, you can configure Anaconda Enterprise to use packages from an online package repository such as anaconda and r.

You can then mirror channels & packages into your organization’s internal AE 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 Anaconda Enterprise configuration—anaconda-enterprise-anaconda-platform.yml—to include the appropriate channels, as follows:

```yaml
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_FQDN>/repository/conda
```

To update Anaconda Enterprise with your changes to the configuration, restart its services:

```bash
sudo gravity enter
kubectl get pods | grep 'ap-' | cut -d ' ' -f1 | xargs kubectl delete pods
```
2.3.2 Mirroring channels and packages

Anaconda Enterprise 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 Anaconda Enterprise’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 Anaconda Enterprise
- Install conda
- Install the Anaconda Enterprise 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 Anaconda Enterprise and Sharing channels.

Configuration options:

- Configuring SSL verification
- Mirroring in a proxied environment
- Mirroring specific platforms and versions
- Using blacklisting to exclude packages
- Using whitelisting to include packages
- Combining multiple configuration arguments

Log into Anaconda Enterprise as an existing user using the following command:
$ anaconda-enterprise-cli login
Username: anaconda-enterprise
Password:
Logged anaconda-enterprise in!

Note: If Anaconda Enterprise 5 is installed in a proxied environment, see *Mirroring in a proxied environment* for information on setting the NO_PROXY variable.

## Mirroring the Anaconda repository

We recommend the following process as a best practice for mirroring the Anaconda 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:**

```yaml
dest_channel: main
categories:
  - https://repo.anaconda.com/pkgs/main
platforms:
  - linux-64
  - noarch
```

**Example free.yaml file:**

```yaml
dest_channel: free
categories:
  - 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:

```bash
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 blacklisted.

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.

### 2.3. Configuring channels and 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:

```bash
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 Anaconda Enterprise. (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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>The local user under which the PyPI packages are imported. Default: pypi.</td>
</tr>
<tr>
<td>pkg_list</td>
<td>A list of packages to mirror. Only packages listed are mirrored. If this is</td>
</tr>
<tr>
<td></td>
<td>set, blacklist and whitelist settings are ignored. Default: [].</td>
</tr>
<tr>
<td>whitelist</td>
<td>A list of packages to mirror. Only packages listed are mirrored. If the list</td>
</tr>
<tr>
<td></td>
<td>is empty, all packages are checked. Default: [].</td>
</tr>
<tr>
<td>blacklist</td>
<td>A list of packages to skip. The packages listed are ignored. Default: [].</td>
</tr>
<tr>
<td>latest_only</td>
<td>Only download the latest versions of the packages. Default: false.</td>
</tr>
<tr>
<td>remote_url</td>
<td>The URL of the PyPI mirror. /pypi is appended to build the XML RPC API URL,</td>
</tr>
<tr>
<td></td>
<td>/simple for the simple index and /pypi/{package}/{version}/json for the JSON</td>
</tr>
<tr>
<td></td>
<td>API. Default: <a href="https://pypi.python.org/">https://pypi.python.org/</a>.</td>
</tr>
<tr>
<td>xml_rpc_api_url</td>
<td>A custom value for XML RPC URL. If this value is present, it takes precedence</td>
</tr>
<tr>
<td></td>
<td>over the URL built using remote_url. Default: null.</td>
</tr>
<tr>
<td>simple_index_url</td>
<td>A custom value for the simple index URL. If this value is present, it takes</td>
</tr>
<tr>
<td></td>
<td>precedence over the URL built using remote_url. Default: null.</td>
</tr>
<tr>
<td>use_xml</td>
<td>Whether to use the XML RPC API as specified by PEP381. If this is set to</td>
</tr>
<tr>
<td></td>
<td>true, the XML RPC API is used to determine which packages to check. Other-</td>
</tr>
<tr>
<td></td>
<td>wise the scripts falls back to the simple index. If the XML RPC fails, the</td>
</tr>
<tr>
<td></td>
<td>simple index is used. Default: true.</td>
</tr>
<tr>
<td>use_serial</td>
<td>Whether to use the serial number provided by the XML RPC API. Only packages</td>
</tr>
<tr>
<td></td>
<td>updated since the last serial saved are checked. If this is set to false,</td>
</tr>
<tr>
<td></td>
<td>all PyPI packages are checked for updates. Default: true.</td>
</tr>
<tr>
<td>create_org</td>
<td>Create the mirror user as an organization instead of a regular user account.</td>
</tr>
<tr>
<td></td>
<td>All superusers are added to the “Owners” group of the organization. Default:</td>
</tr>
<tr>
<td></td>
<td>false.</td>
</tr>
</tbody>
</table>

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:

```yaml
whitelist:
  - 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:

```ini
[global]
index-url=https://anaconda.example.com/repoery/pypi/pypi/simple/
```

replacing `https://anaconda.example.com` with the actual URL to your Anaconda Enterprise.

To configure Anaconda Enterprise 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,

```
pip install --index-url https://anaconda.example.com/repoery/pypi/pypi/simple/ ...
```

replacing `https://anaconda.example.com` with the actual URL to your Anaconda Enterprise 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.


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 whitelist, blacklist or license_blacklist 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/
```

(continues on next page)
# 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`:

```bash
# 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 `<path 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 Anaconda Enterprise installation under the username `anaconda`.

## Configuring Anaconda Enterprise

After creating the mirror, *edit your Anaconda Enterprise configuration* to add this new mirrored channel to the default Anaconda Enterprise 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 Anaconda Enterprise.

**Note:** The `ap-workspace` pod must be restarted for the configuration change to take effect on new project editor sessions.

To update the Anaconda Enterprise 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:
   ```bash
   kubectl get pods
   ```

2. Restart the workspace services by running the following command:
   ```bash
   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 blacklisting to exclude packages.
- Using whitelisting 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 Anaconda Enterprise 5 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 blacklisting 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 blacklist, whitelist or license_blacklist 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:**
```
python_versions:
    - 3.3
```

Mirrors only Anaconda packages built for Python 3.3.

License blacklist mirroring

The mirroring script supports license blacklisting for the following license families:

AGPL
GPL2
GPL3
LGPL
BSD
MIT
Apache
PSF
Public-Domain
Proprietary
Other

**EXAMPLE:**
```
license_blacklist:
    - 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 blacklisted.

**Blacklist mirroring**

The blacklist allows access to all packages EXCEPT those explicitly listed. If the `license_blacklist` and `blacklist` arguments are combined, `license_blacklist` is evaluated first, and `blacklist` is a supplemental modifier.

**EXAMPLE:**

```
blacklist:
    - bzip2
    - tk
    - openssl
```

This example mirrors the entire repository EXCEPT the bzip2, Tk, and OpenSSL packages.

**Whitelist mirroring**

The `whitelist` argument adds or includes packages that would be otherwise excluded by the `blacklist` and/or `license_blacklist` functions.

**EXAMPLE:**

```
license_blacklist:
    - GPL2
    - GPL3
whitelist:
    - 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_blacklist` and `blacklist` arguments are combined, `license_blacklist` is evaluated first, and `blacklist` 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 blacklisted. It does mirror all other packages in the repository:

```
license_blacklist:
  - GPL2
blacklist:
  - pyqt
```

If the `blacklist` and `whitelist` arguments are both employed, the blacklist is evaluated first, with the whitelist functioning as a modifier.

**EXAMPLE:** This example mirrors all packages in the repository except `astropy` and `pygments`. Despite being listed on the blacklist, `accelerate` is mirrored because it is listed on the whitelist.

```
blacklist:
  - accelerate
  - astropy
  - pygments
whitelist:
  - accelerate
```

### 2.3.3 Managing channels and packages

Anaconda Enterprise 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**: The anaconda-enterprise channel is used for internal purposes only, and should not be modified.

---

**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 *Managing 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
```

2.3.4 Pointing conda to an on-premises repository

Anaconda Enterprise 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:
Replacing `<your-server.domain.com>` with the fully-qualified domain name (FQDN) of your installation of Anaconda Enterprise.

See this section of the conda docs for more information.

2.3.5 Using a proxy for conda packages

You can configure Anaconda Enterprise to use a proxy for conda packages, if your organization’s network security policy requires it. To do so, you’ll need to do the following:

- Obtain your proxy variables
- Verify the proxy works
- Configure Anaconda Enterprise

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 into Anaconda Enterprise.
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 Anaconda Enterprise

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
```

2.4 Generating custom Anaconda installers

As an Anaconda Enterprise 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.

2.4.1 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.

Anaconda Enterprise 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.
2.4.2 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:

    Anaconda Enterprise 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 Cloudera CDH cluster to make it available to users:

Note: If you are using CDH 5.x, you’ll need to manually download the parcel, move it to the Cloudera Manager node, then configure Cloudera Manager for a local parcel repository. This is because CDH 5.x does not work with TLS 1.2 that Anaconda Enterprise uses to serve the parcel, so you’ll see a protocol version error if you attempt to use AE as a remote parcel repository with CDH 5.x.

If you are using CDH 6.x with parcels, you can configure Anaconda Enterprise as a remote parcel repository, or you can manually download the parcel and configure a local parcel repository.

1. In the Installers list, click the parcel name to view its details—including the logs generated during the creation process.
2. Depending on the version of CDH you are using (see NOTE above), either copy the path to the parcel or download the parcel installer.

3. From the Cloudera Manager Admin Console, click the Parcels indicator in the top navigation bar.

4. Click the Configuration button on the top right of the Parcels page to display the Parcel Settings.

5. If you downloaded the parcel from AE in Step 2 above, copy it to the Local Parcel Repository Path you’ve configured for Cloudera Manager.

   –or–

   To configure AE as a remote parcel repository, add the URL you copied in Step 2 above to the Remote Parcel Repository URLs section and click Save Changes.

   If automatic downloading and distribution are not enabled, go to the Parcels page and select Distribute to install the parcel across your CDH cluster. The custom-generated Anaconda parcel is now ready to use with Spark or other distributed frameworks on your Cloudera CDH cluster.

   For more information, see these instructions from Cloudera.

### 2.5 Advanced platform settings

After installing Anaconda Enterprise, 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 AE 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 Anaconda Enterprise 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.

2.5.1 Editing platform settings

You configure the Anaconda Enterprise 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 Anaconda Enterprise installation.

You can modify the default configuration using the Operations Center UI, or a command shell. Any changes you make will impact how Anaconda Enterprise 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:

1. Log in to Anaconda Enterprise, 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 want to configure global environment variables, please ignore the other entries in the Config maps and Namespace drop-downs. They impact the underlying Kubernetes system, so making changes to them may 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. We recommend 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
   ```

2.5. Advanced platform settings
2.5.2 Changing the database password

You can change the password for the Anaconda Enterprise 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 Anaconda Enterprise 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 Anaconda Enterprise Config map.
4. In the GLOBAL CONFIGURATION section of the configuration file, locate the db section and enter the password you just set:

   ![Configuration map](image)

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
2.5.3 Changing the platform redirect URLs

You’ll use the Anaconda Enterprise 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 Anaconda Enterprise server, or the following symbols:

   `Valid Redirect URIs` would look something like this: `https://server-name.domain.com/*`.

6. Click Save to update the server with your changes.

2.5.4 Configuring Livy server for Hadoop Spark access

After installing Livy server, there are main 3 aspects you need to configure on Apache Livy server for Anaconda Enterprise users to be able to access Hadoop Spark within Anaconda Enterprise:

- 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 Anaconda Enterprise, 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:

```
export JAVA_HOME=/usr/java/jdk1.8.0_121-cloudera/jre/
export SPARK_HOME=/opt/cloudera/parcels/CDH/lib/spark/
export SPARK_CONF_DIR=$SPARK_HOME/conf
export HADOOP_HOME=/etc/hadoop/
export 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:

```ini
livy.server.port = 8998
# What spark master Livy sessions should use: yarn or yarn-cluster
livy.spark.master = yarn
# What spark deploy mode Livy sessions should use: client or cluster
livy.spark.deployMode = cluster

# Kerberos settings
livy.server.auth.type = kerberos
livy.impersonation.enabled = true

# livy.server.launch.kerberos.principal = livy/$HOSTNAME@ANACONDA.COM
# livy.server.launch.kerberos.keytab = /etc/security/livy.keytab
# livy.server.auth.kerberos.principal = HTTP/$HOSTNAME@ANACONDA.COM
# livy.server.auth.kerberos.keytab = /etc/security/httplivy.keytab

# livy.server.access_control.enabled = true
# livy.server.access_control.users = livy,hdfs,zeppelin
# livy.superusers = livy,hdfs,zeppelin
```

After configuring Livy server, you’ll need to restart it:

```
./bin/anaconda-livy-server stop
./bin/anaconda-livy-server start
```

Consider using a process control mechanism to restart Livy server, to ensure that it’s reliably restarted in the event of a failure.

### 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 2 keytabs for Apache Livy using `kadmin.local`.

**IMPORTANT:** The keytab principals for Livy must match the hostname that the Livy server is deployed on, or you’ll see the following exception: `GSSEException: 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/ip-172-31-3-131.ec2.internal
WARNING: no policy specified for livy/ip-172-31-3-131.ec2.internal@ANACONDA.COM; --defaulting to no policy
Enter password for principal "livy/ip-172-31-3-131.ec2.internal@ANACONDA.COM":
Re-enter password for principal "livy/ip-172-31-3-131.ec2.internal@ANACONDA.COM":

kadmin.local:  xst -k livy/ip-172-31-3-131.ec2.internal.keytab livy/ip-172-31-3-131.ec2.internal@ANACONDA.COM
...

kadmin.local:  addprinc HTTP/ip-172-31-3-131.ec2.internal
```

(continues on next page)
WARNING: no policy specified for HTTP/ip-172-31-3-131.ec2.internal@ANACONDA.COM; →defaulting to no policy

Enter password for principal "HTTP/ip-172-31-3-131.ec2.internal@ANACONDA.COM":
Re-enter password for principal "HTTP/ip-172-31-3-131.ec2.internal@ANACONDA.COM":
→ec2.internal@ANACONDA.COM

This will generate two files: livy-ip-172-31-3-131.ec2.internal.keytab and HTTP-ip-172-31-3-131.ec2.internal.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:

```
livy.server.auth.type = kerberos
livy.impersonation.enabled = false  # see notes below

# principals and keytabs to exactly match those generated before
livy.server.launch.kerberos.principal = livy/ip-172-31-3-131@ANACONDA.COM
livy.server.launch.kerberos.keytab = /home/centos/conf/livy-ip-172-31-3-131.keytab
livy.server.auth.kerberos.principal = HTTP/ip-172-31-3-131@ANACONDA.COM
livy.server.auth.kerberos.keytab = /home/centos/conf/HTTP-ip-172-31-3-131.keytab

# this may not be required when delegating auth to kerberos
livy.server.access-control.enabled = true
livy.server.access-control.allowed-users = livy,zeppelin,testuser
livy.superusers = livy,zeppelin,testuser
```

NOTES:

- The hostname and domain are not the same—verify that they match your Kerberos configuration.

- `livy.server.access-control.enabled = true` is only required if you're going to also whitelist the allowed users with the `livy.server.access-control.allowed-users <user>` key.

### Configuring project access

After you’ve installed Livy and configured cluster access, some additional configuration is required before Anaconda Enterprise users will be able to connect to a remote Hadoop Spark cluster from within their 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 Anaconda Enterprise.
Use the following command to upload the yaml file to the server:

```
sudo kubectl replace -f anaconda-config-files-secret.yaml
```

To update the Anaconda Enterprise 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.

---

**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:

   ```
   keytool -genkey -alias <host> -keyalg RSA -keysize 1024 -dname CN=<host>,OU=hw,
   O=hw,L=paloalto,ST=ca,C=us -keypass <keyPassword> -keystore <keystore_file> -
   storepass <storePassword>
   ```

2. Create a certificate:

   ```
   keytool -export -alias <host> -keystore <keystore_file> -rfc -file <cert_file> -
   storepass <StorePassword>
   ```

3. Create a truststore file:

   ```
   keytool -import -noprompt -alias <host> -file <cert_file> -keystore <truststore_file> -
   storepass <truststorePassword>
   ```

4. Update `livy.conf` with the keystore details. For example:

   ```
   livy.keystore = /home/centos/livy-0.5.0-incubating-bin/keystore.jks
   livy.keystore.password = anaconda
   livy.key-password = anaconda
   ```

---

2.5. Advanced platform settings  

135
5. Update ~/.sparkmagic/config.json. For example:

```
"kernel_python_credentials" : {
    "username": "",
    "password": "",
    "url": "https://35.172.121.109:8998",
    "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.

**Warning:** 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}
headers = {'Content-Type': 'application/json'}
r = requests.post(livy_url, data=json.dumps(data), headers=headers, auth=HTTPKerberosAuth(mutual_authentication=REQUIRED, sanitize MutualErrorResponse=False), verify=False)
r.json()
```

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 MutualErrorResponse=False), verify=False)
r.json()
```

Then submit the following statement:

```
session_url = "https://<livy host>:<livy port>/sessions/0/statements"
data ={"code": "sc.parallelize(1 to 10).count()"}
headers = {'Content-Type': 'application/json'}
r = requests.get(session_url, headers=headers, auth=HTTPKerberosAuth(mutual_authentication=REQUIRED, sanitize MutualErrorResponse=False), verify=False)
r.json()
```
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:

   ```bash
   ```

2. Use the following command to create the keystore.jks file:

   ```bash
   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 Anaconda Enterprise installation:

   ```bash
   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:

   ```bash
   keytool -importcert -keystore keystore.jks -storepass <password> -alias rootCA -file rootca.crt
   ```

5. Add the keystore.jks file to the livy.conf file. For example:

   ```bash
   livy.keystore = /home/centos/livy-0.5.0-incubating-bin/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):

   ```bash
   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 AE server, do the following:

1. Install the ca-certificates package:

   ```bash
   yum install ca-certificates
   ```

2. Enable dynamic CA configuration:

   ```bash
   update-ca-trust force-enable
   ```

3. Add your rootca.crt as a new file:

   ```bash
   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:

```python
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.

### 2.5.5 Connecting to an external version control repository

If your organization already uses a shared repository for version control, you can configure Anaconda Enterprise to use that repository instead of the internal Git server. To associate an external repository with Anaconda Enterprise, you simply need to provide the information required to connect to it.

After you do so, platform users will be able to access the repository within their sessions and deployments without having to leave the platform. Anaconda Enterprise creates a repository for each project that’s created by platform users.

Anaconda Enterprise supports integration with the following external repositories:

<table>
<thead>
<tr>
<th>External repository</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitbucket Enterprise</td>
<td>5.9.1, 5.12.1, 5.15.0, 5.16.5, 6.2.0, 6.8.0</td>
</tr>
<tr>
<td>Bitbucket Cloud</td>
<td>bitbucket.org</td>
</tr>
<tr>
<td>GitHub Enterprise</td>
<td>2.15, 2.16, and 2.17</td>
</tr>
<tr>
<td>GitHub Cloud</td>
<td>github.com</td>
</tr>
<tr>
<td>GitLab Enterprise</td>
<td>10.4.2, 10.7.1, 11.10.0, and 12.1.6</td>
</tr>
<tr>
<td>GitLab Cloud</td>
<td>gitlab.com</td>
</tr>
</tbody>
</table>

**Warning:** If you are going to use an external repository for version control, we strongly recommend you set it up *before users start creating projects in Anaconda Enterprise*. If your organization changes Git hosting services, and you therefore need to migrate projects from one version control repository to another, we recommend you *follow the process outlined here*.

**NOTES:**

- Neither Bitbucket.com or GitLab.com support versioning of archive downloads and app deployments. In other words, the latest revision will always be downloaded or deployed.
- To provide permission granularity and maintain parity with your Git hosting solution, Anaconda Enterprise 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 personal access token *before they create their first project* in Anaconda Enterprise. We recommend you *advise users to create an ever-lasting token*, so they can retain permanent
access to their files from within Anaconda Enterprise. The **specific auth token permissions required for each repository** are outlined here.

**Before you begin, gather the following information:**

- 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

---

**To associate a specific version control repository with a project:**

1. Log in to Anaconda Enterprise, 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:** Please ignore the other entries in the **Config maps** and **Namespace** drop-downs. They impact the underlying Kubernetes system, so making changes to them may have unintended consequences or cause the platform to behave unpredictably.

6. Locate the **git** section of the configuration file. The default behavior is to use the internal Anaconda Enterprise repository for version control (see default settings pictured below).

7. To override this default setting, **uncomment** the **Example external repo configuration** section of the Config map, and replace the placeholder settings with the correct values for your organization’s repository:

---

2.5. **Advanced platform settings** 139
Where:

name = A descriptive name for the service your organization uses.

type = The type of version control repository your organization uses: github-v3-api (GitHub Enterprise and Cloud), bitbucket-v1-api (Bitbucket Server), bitbucket-v2-api (Bitbucket Cloud), or gitlab-v4-api (GitLab Cloud and GitLab server).

NOTE: The values for this parameter have changed from AE 5.3.0.

url = The URL of the API (e.g., https://api.github.com/, https://api.bitbucket.org, or https://gitlab.com).

credential-url = The URL to authenticate against for repository operations such as cloning and pushing.

NOTE: This parameter replaces the credential-hostname parameter used in AE 5.3.0.

repository = Must be '{owner}-{id}' encased in single quotes.

organization = The name of your Github organization, Bitbucket team, or GitLab group. (Bitbucket does not support dashes in team names.)

username = The username associated with the Administrator account at Github, Bitbucket, or GitLab. This account must have full Admin permissions.

auth-token = The Github personal access token, Bitbucket app password, or GitLab access token for the Administrator account associated with the username. (You must enable 2FA to get personal access tokens in GitLab.)

8. Comment out the Internal repo configuration section of the Config map that follows, as it relates to the Anaconda Enterprise internal Git server settings that you are overriding:

9. Click Apply to save your changes to the Config map.
To update the Anaconda Enterprise server with your changes, restart services by running these commands on the master node:

```
sudo gravity enter
cubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

To verify that Anaconda Enterprise users can access the version control repository you added, create a project. See Working with projects for more information.

### 2.5.6 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, we recommend you follow this high-level process:

1. **Perform pre-migration setup.**
2. **Run the project migration script.**
3. **Perform post migration cleanup.**

#### Prequisites:

- **Update the Anaconda Enterprise 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 Anaconda Enterprise cluster.
- You’ll also need the Postgres database password, origin Git host token/password, and destination Git host token/password.

#### Pre-migration setup

1. If you haven’t already done so, install the version of conda provided with the Anaconda Enterprise installer on the master node:

   ```bash
   bash anaconda-enterprise-5.3.1-56.gf54c3abad/installer/conda-bootstrap-4.5.12
   ```

2. After conda is finished installing, login to the terminal again.
3. Install git, using the command that’s appropriate for your environment:
   - On RHEL/CentOS: `yum install git`
   - On Ubuntu/Debian: `apt install git`
4. Use the following command to create the conda environment:

   ```conda create --name migrate --file anaconda-enterprise-5.3.1-56.gf54c3abad/environment.txt```

5. Use the following command to activate the conda environment:
conda activate migrate

6. 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:

```yaml
# anaconda-enterprise-anaconda-platform.yml

git:
  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: 8080
  https: true
  key: /etc/secrets/certs/tls.key
  certificate: /etc/secrets/certs/tls.crt
  db:
    database: anaconda_git
  directory: /opt/app/git-data # directory where git server will store its data
  username: anaconda # DS username that the git server should run under
  reverse-proxy-auth: false
  lfs-secret: /etc/secrets/lfs-secret # LFS authentication token secret. Should be uniquely generated for each installation.
  secret-key: E39992F3XAM3H3G3mgmQ2613p7Z0yg6SvW8pMTB4 # git server secret key. Should be uniquely generated for each installation.
  max-commit-file-size: 50000000
```

7. Run the following command to restart the associated pod on the master node:

```
kubectl delete pod -l 'app=ap-git-storage'
```

8. Create a user mappings file that maps Anaconda Enterprise user IDs to Git user IDs. This is a colon-separated text file where the first field is the AE user name, and the second field is the corresponding Git user name. For example:

```
ea-e-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**

**Note:** 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.

**Warning:** 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 AE5 installation tarball. It can be used in the following ways:
    [--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
    --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
    --force-migrate       Forces migration by replacing local and destination
                           repositories
    --scratch-dir SCRATCH_DIR
                           The scratch directory for cloning project repositories
    --postgres-host POSTGRES_HOST
                           Hostname of AE5 Postgres DB
    --postgres-user POSTGRES_USER
                           Username of AE5 postgres DB
    --postgres-passwd POSTGRES_PASSWD
                           Password of AE5 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
                           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
For example, the tool can be used in the following way:

```
```

To ensure tokens are not visible in bash history, they can be omitted and can be entered via stdin when running the script.

**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
```

**Warning:** If you do not re-enable reverse proxy authentication, Anaconda Enterprise will not work.

To verify that the new repository is being used by Anaconda Enterprise, *edit an existing project* and *commit your changes to it*.

### 2.5.7 Mounting an external file share

Anaconda Enterprise 5 provides a mechanism to mount shared volumes into sessions, deployments, and jobs, using the `volumes:` section of the platform configuration YAML file. In AE 5.4 and earlier, this was limited strictly to NFS volumes.

With AE 5.5 and the *Bring your own Kubernetes* 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 Anaconda Enterprise ConfigMap and restarting the workspace and deploy pods. For that reason, we recommend 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 Anaconda Enterprise in the `volumes:` section of the ConfigMap, as seen in the following example:

```yaml
volumes:
  /absolute/path1:
    subPath: "
    readOnly: false
    groupID: 1234
    pvc: efs-volume1
  relative/path2:
    subPath: "
    readOnly: true
    groupID: 1001
    nfs:
      server: 10.123.23.45
      path: /exported/volume
      readOnly: true
```

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`. AE5 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 AE 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.

2.5. Advanced platform settings
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 AE5 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, AE 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 Gravitational installs. Bring your own Kubernetes installs do not have the Operations Center and may need to use a different method of managing the ConfigMap.

1. Log in to Anaconda Enterprise, 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 we recommend 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 Anaconda Enterprise 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 pods
```

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 Anaconda Enterprise 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 Anaconda Enterprise goes offline, editor sessions, applications and deployment jobs that use it will not start. If downtime occurs, we recommend 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 AE5 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 uuid 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}
```

(continues on next page)
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 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 we recommend 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.

2.5.8 Disabling sudo for yum

By default, sudo access for yum is enabled on the Anaconda Enterprise platform. You can easily disable it, however, if your organization requires it.

1. Log in to Anaconda Enterprise, 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: We recommend 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 Anaconda Enterprise 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.

### 2.5.9 Specifying alternate wildcard domains

By default, Anaconda Enterprise 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 Anaconda Enterprise, 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 Anaconda Enterprise 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 Anaconda Enterprise server with your changes, restart services by running these commands **on the master node**:

```bash
sudo gravity enter
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

### 2.5.10 Setting global config variables

Anaconda Enterprise 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 Anaconda Enterprise CLI, as you can add any variable supported by conda configuration.

1. Log in to Anaconda Enterprise, 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 Anaconda Enterprise to `use a proxy for conda packages`, you might add entries that resemble the following:

```yaml
HTTP_PROXY: proxy.url.com:3128
NO_PROXY: anaconda-test.url.com
```
8. Click **Apply** to save your changes.

9. To update Anaconda Enterprise 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
```
3.1 Working with projects

Anaconda Enterprise makes it easy for you to create and share interactive data visualizations, live notebooks or machine learning models built using popular libraries such Python, R, Bokeh and Shiny.

AE 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. For more information, see Developing a project.

Project components are all compressed into a .tar.bz2, .tar.gz or .zip file to make the project portable—so it’s easier to store and share with others.

To get you started, Anaconda Enterprise provides several sample projects, including the following:

- Anaconda Distribution for Python 2.7, 3.5 and 3.6
- Minimal Python templates for versions 2.7, 3.5, 3.6, and 3.7
- R notebooks & R Shiny apps
- Matplotlib and HvPlots written in Jupyter Notebooks
- Panel and HoloViz tutorials
- Dashboards for Gapminder data set, oil and gas exploration, NYC taxi data, and attractor equations
- TensorFlow apps for Flask, Tornado and MNIST trained data
- Tutorial on the Intake data catalog package
- Tutorials for database access and time series modeling

You can access them by clicking Sample Projects from the Projects view. To use a sample project as a starting point, you can copy it to your project list.

To work with a project, click on it or select View details from its menu in the list view. Then use the menu on the left as follows:
• Click **Session** to open the project in the default editor. This is Jupyter Notebook, unless you’ve specified a different editor under **Settings**.

• Click **Deployments** to view deployments initiated from this project.

• Click **Schedules** to view and schedule deployments of the project.

• Click **Runs** to view a list of all project deployments that have run based on a schedule.

• Click **Share** to share the project with selected collaborators.

• Click **Audit Trail** to view a list of all actions performed on the project.

• Click **Settings** to change the project name or default editor—Jupyter Notebook—for the project. For example, if you prefer to work with Apache Zeppelin or JupyterLab, choose it as your default editor.

You can also select a resource profile that meets or exceeds your requirements for the project, or delete the project. With admin configurations, your projects (sessions/deployments) can now run separately from the master node.

---

**Warning:** Deleting a project is irreversible, and therefore can only be done if it is not shared.

---

**To make changes to the project files,** click **Open session**.

**Note:** If the system gets overloaded and there are issues copying, opening, or saving changes to a project, the platform will visually notify you by displaying it in red—in addition to generating a text notification. We recommend you check the notifications in the **Audit Trail** for additional information about the error, or delete the project and try again.

**To work with the contents offline,** you can download the compressed file and then upload it to work with it within AE.

You can also create new—or upload existing—projects to add them to the server.
To update the project repository with your changes, you commit your changes to the project.

Note: To maintain performance, there is a 1GB file size limit for project files you upload. Anaconda Enterprise projects are versioned using Git, so we recommend 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 we recommend using storage solutions designed for that type of data, and connecting to those data sources from within your Anaconda Enterprise sessions.

If your organization would prefer to use its own supported external version control repository, your Administrator can configure Anaconda Enterprise to use that repository instead of the internal GitHub server. After they do so, you will be prompted for your personal access token before you create your first project in Anaconda Enterprise. We recommend you create an ever-lasting access token, so you can retain permanent access to your files from within Anaconda Enterprise. See Configuring your user settings for the permissions that must be set for your auth token, and the steps to configure connectivity to your version control repository.

3.1.1 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.

### 3.1.2 Developing a project

To enable Anaconda Enterprise 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.

**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:
If you prefer to use the UI to configure your project settings, you’ll need to change the default editor from Jupyter Notebook to JupyterLab. Do this in the project’s Settings and restart the editor session.

Adding packages to a project

Anaconda Enterprise 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 PACKAGES 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`
Warning: 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 Anaconda Enterprise 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:

```
packages:
  - six>=1.4.0
  - gunicorn==19.1.0
  - pip:
    - python-mimeparse
    - falcon==1.0.0
```

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 an Anaconda Enterprise 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 may use either of these methods to specify the environment for a project:

- In a JupyterLab editing session, click the **Project** tab on the far left and click the plus sign to the right of the **ENVIRONMENTS** field. Choose whether you want to **Prepare all environments** or **Add environments**.

  Select an environment and then select **Run**, **Check** or **Edit**. Running an environment opens a terminal window with that environment active.

  When creating an environment, you may choose to inherit from an existing environment, and choose the environment’s supported platforms, its channels, and its packages.

  --or--

- You can use the terminal and command line. 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:

  ```
  anaconda-project remove-env-spec anaconda50_py36
  ```

**Warning:** 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:

   ```bash
   jupyter
   ```

   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 Anaconda Enterprise 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:

```yaml
commands:
  default:
    notebook: your-notebook.ipynb
```

For a project with a Bokeh (version 0.12) app defined in a main.py file:

```yaml
commands:
  default:
    bokeh_app: .
supports_http_options: True
```

For a Panel dashboard (panel must be installed in your project):

```yaml
commands:
  default:
    unix: panel serve script-or-notebook-file
    supports_http_options: True
```

For a generic script or web framework, including Python or R:

```yaml
commands:
  default:
    unix: bash run.sh
    supports_http_options: true

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:

3.1. Working with projects
2. Run the following command, replacing `anaconda44_py35` with the name of your environment, if it’s different:

```bash
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:

```bash
anaconda-project run <command-name>
```

If there are any errors preventing a successful deployment, they will be displayed in the terminal.
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
```

3.1.3 Saving and committing changes in a project

Saving changes to files within an editor or project is different from committing those changes to the Anaconda Enterprise 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.

**Warning:** Files names containing unicode characters—special characters, punctuation, symbols—can’t be committed to the server, so avoid them when naming files.

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.

**Note:** If the size of your stored git files totals more than 1GB, your system may become bogged down and inoperative. As such, we recommend keeping file sizes under 50MB. If a file of greater size is required for your work, please contact your administrator.

Binary files are difficult for version control systems to manage, so we recommend using storage solutions designed for that type of data, and connecting to those data sources from within your Anaconda Enterprise 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. We recommend 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.

### 3.1.4 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:** Anaconda Enterprise 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.

Anaconda Enterprise 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 AE, 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.

Anaconda Enterprise 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 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.

3.1. Working with projects
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.

### 3.2 Using project templates

In addition to sample projects, Anaconda Enterprise 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:
3.2.1 Python templates

Anaconda Enterprise 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 Anaconda Enterprise: 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 Working with projects.

3.2.2 R templates

The R template contains the R Essentials bundle of approximately 80 packages: r-base version 3.4.2, plus the most commonly used R packages for data science, including caret, dplyr, ggplot2, glmnet, irkernel, rbokeh, shiny, and tidyverse.

You can add other R packages as described in Developing a project. You’ll need to be able to connect to the appropriate repository to do so. Otherwise, your Administrator may need to mirror the channels and packages for you to be able to access them.

R notebooks can be edited with any of the editors provided with Anaconda Enterprise: Jupyter Notebooks, JupyterLab, or Apache Zeppelin. To change the default editor for your R 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 Working with projects.

3.2.3 Hadoop / Spark

If your Anaconda Enterprise 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 Anaconda Enterprise 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, we recommend 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 Anaconda Enterprise with Spark requires Livy and Sparkmagic. The Hadoop/Spark project template includes Sparkmagic, but your Administrator must have configured Anaconda Enterprise to work with a Livy server.
Supported versions

The following combinations of the multiple tools are supported:

- Python 2 and Python 3, Apache Livy 0.5, Apache Spark 2.1, Oracle Java 1.8
- Python 2, Apache Livy 0.5, Apache Spark 1.6, Oracle Java 1.8

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 Anaconda Enterprise, you can connect to a remote Spark cluster using Apache Livy with any of the available clients, including Jupyter notebooks with Sparkmagic. Anaconda Enterprise 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
• Python 3
• R
• Spark
• SparkR
• Python 2

To work with Livy and Python, use PySpark. Do not use PySpark3.
To work with Livy and R, use R with the sparklyr package. Do not use the kernel SparkR.
To work with Livy and Scala, use Spark.

You can use Spark with Anaconda Enterprise 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 %%local.

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:

```bash
%%configure -f
{"conf": {"spark.driver.python": "/opt/anaconda2/bin/python", "spark.executor.python": "/opt/anaconda2/bin/python"}}
```

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:

```bash
%%configure -f
{"conf": {"spark.driver.python": "/opt/anaconda3/bin/python", "spark.executor.python": "/opt/anaconda3/bin/python"}}
```

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**

Anaconda Enterprise Administrators can generate custom parcels for Cloudera CDH or custom management packs for Hortonworks HDP to distribute customized versions of Anaconda across a Hadoop/Spark cluster using Cloudera Manager for CDH 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:

```bash
%%configure -f
{"conf": {"spark.yarn.appMasterEnv.PYSPARK_PYTHON": "/opt/anaconda/bin/python", "spark.yarn.appMasterEnv.PYSPARK_DRIVER_PYTHON": "/opt/anaconda/bin/python", "spark.yarn.executorEnv.PYSPARK_PYTHON": "/opt/anaconda/bin/python", "spark.pyspark.python": "/opt/anaconda/bin/python", "spark.pyspark.driver.python": "/opt/anaconda/bin/python"}}
```

**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, we recommend 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:

```yaml
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.

**Warning:** 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 *Installing Livy server for Hadoop Spark access* and *Configuring Livy server for Hadoop Spark access* for information on installing and configuring Livy.
Python

Example code showing Python with a Spark kernel:

```python
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

- Hadoop 2.6.0, Python 2 or 3

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:
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:

```bash
$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:

```python
from hdfs import InsecureClient

client = InsecureClient('http://<Namenode>:50070')
client.list("/")
```

Python with HDFS with Kerberos:

```python
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 as JDBC, ODBC and Thrift. Anaconda recommends Thrift with Python and JDBC with R.

**Dependencies**

- pyhive
- RJDBC
Supported versions

- Hive 1.1.0, JDK 1.8, Python 2 or Python 3

Kernels

- [anaconda50_hadoop] Python 3

Drivers

Using JDBC 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.

Cloudera EXAMPLE:

- Hive JDBC Connector 2.5.4 - Download
- Hive JDBC Connection 2.5.4 - Documentation

We recommend downloading the respective JDBC drivers and committing them to the project so that they are always available when the project starts.

Once the drivers are located in the project, Anaconda recommends using the RJDBC library to connect to Hive. Sample code for this is shown below.

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:

```python
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:

```python
from pyhive import hive
conn = hive.connect('<Hive Server 2>', port=10000, auth='KERBEROS', kerberos_service_name='hive')
```

(continues on next page)
cursor.execute('SHOW TABLES')
cursor.fetchall()

# This prints: [('iris'), ('t1')]
cursor.execute('SELECT * FROM iris')
cursor.fetchall()

# This prints the output of that table

Note: The output will be different, depending on the tables available on the cluster.

R

Anaconda recommends the JDBC method to connect to Hive from R.

Using JDBC allows for multiple types of authentication including Kerberos. The only difference between the types is that different flags are passed to the URI connection string on JDBC. Please follow the official documentation of the driver you picked and for the authentication you have in place.

Hive using RJDBC:

library("RJDBC")

hive_classpath <- list.files("<PATH TO JDBC DRIVER>", pattern="jar$", full.names=T)
drv <- JDBC(driverClass = "com.cloudera.hive.jdbc4.HS2Driver", classPath = hive_classpath, identifier.quote='')
url <- "jdbc:hive2://<HIVE SERVER 2 HOST>:10000/default;SSL=1;AuthMech=1;KrbRealm=<KRB REALM>;KrbHostFQDN=<KRB HOST>;KrbServiceName=hive"

conn <- dbConnect(drv, url)
dbGetQuery(conn, "SHOW TABLES")
dbDisconnect(conn)

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 JDBC, ODBC and Thrift. Anaconda recommends Thrift with Python and JDBC with R.
Dependencies

- impyla
- implyr
- RJDBC

Supported versions

- Impala 2.12.0, JDK 1.8, Python 2 or Python 3

Kernels

- Python 2

Drivers

Using JDBC requires downloading a driver for the specific version of Impala that you are using. This driver is also specific to the vendor you are using.

Cloudera EXAMPLE:

- Impala JDBC Connection 2.5.43 - Download
- Impala JDBC Connection 2.5.43 - Documentation

We recommend downloading the respective JDBC drivers and committing them to the project so that they are always available when the project starts.

Once the drivers are located in the project, Anaconda recommends using the RJDBC library to connect to both Hive and Impala. Sample code for this is shown below.

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:

```python
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:

```
from impala.dbapi import connect
cconn = connect(host='<Impala Daemon>', port=21050, auth_mechanism='GSSAPI', kerberos_→service_name='impala')
cursor = conn.cursor()
cursor.execute('SHOW TABLES')
results = cursor.fetchall()
results
# This prints: [('iris',),]
cursor.execute('SELECT * FROM iris')
cursor.fetchall()# This prints the output of that table
```

**Note:** The output will be different, depending on the tables available on the cluster.

**R**

Anaconda recommends the JDBC method to connect to Impala from R.

Using JDBC allows for multiple types of authentication including Kerberos. The only difference between the types is
that different flags are passed to the URI connection string on JDBC. Please follow the official documentation of the
driver you picked and for the authentication you have in place.

Anaconda recommends Implyr to manipulate tables from Impala. This library provides a dplyr interface for Impala
tables that is familiar to R users. Implyr uses RJBDC for connection.

Impala using RJDBC and Implyr:

```
library(implyr)
library(RJDBC)

impala_classpath <- list.files(path = "<PATH TO JDBC DRIVER>", pattern = "\.jar$", full.names = TRUE)
drv <- JDBC(driverClass = "com.cloudera.hive.jdbc4.HS2Driver", classPath = hive_→classpath, identifier.quote=""
url <- "jdbc:impala://<IMPALA DAEMON HOST>:10000/default;SSL=1;AuthMech=1;KrbRealm=→<KRB REALM>;KrbHostFQDN=<KRB HOST>;KrbServiceName=impala"

# Use implyr to create a dplyr interface
impala <- src_impala(drv, url)
# This will show all the available tables
```

(continues on next page)
3.3 Working with packages

Anaconda Enterprise 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: Anaconda Enterprise supports the use of both conda and pip packages in its repository. To create and share channels and packages from your Anaconda Repository using conda commands, first install anaconda-enterprise-cli and log in to your AE instance.

Creating a package requires familiarity with the conda package manager and command line interface (CLI), so not all AE users will create packages and channels.

Many Anaconda Enterprise 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.

3.3.1 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:
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.

### 3.3.2 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 thereby 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:

```bash
anaconda-enterprise-cli channels create <channelname>
```

**Note:** The channel name `<channelname>` you enter must not already exist.

Now you can upload a package to the channel by entering the following:

```bash
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:

```bash
anaconda-enterprise-cli config set default_channel <channelname>
```

To display your current default channel:

---

**3.3. Working with packages** 181
After setting the default channel, upload to your default channel:

```bash
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.

### 3.3.3 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 Anaconda Enterprise. 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.
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.

**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**.

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:

```bash
anaconda-enterprise-cli channels list
```

Share a channel with a specific user using the `share` command:

```bash
anaconda-enterprise-cli channels share --user username --level r <channelname>
```

You can also share a channel with an existing group created by your Administrator:

```bash
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:

```bash
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`:

```bash
anaconda-enterprise-cli channels share --help
```

### 3.3.4 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:

```yaml
channels:
  - <anaconda_dot_org_username>
  - http://some.custom/channel
  - file:///some/local/directory
  - defaults
```

For more information, see this section of the conda docs.

### 3.4 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, select the top **Channels** menu, then click the **Installers** link in the top right corner.

To download an installer, simply 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.

### 3.5 Working with data

#### 3.5.1 Loading data into your project

Anaconda Enterprise 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 Anaconda Enterprise 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:

```python
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.
3.5.2 Exploring project data

With Anaconda Enterprise, 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):

<table>
<thead>
<tr>
<th>customer_id</th>
<th>title</th>
<th>industry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>data scientist</td>
<td>retail</td>
</tr>
<tr>
<td>2</td>
<td>data scientist</td>
<td>academia</td>
</tr>
<tr>
<td>3</td>
<td>compiler optimizer</td>
<td>academia</td>
</tr>
<tr>
<td>4</td>
<td>data scientist</td>
<td>finance</td>
</tr>
<tr>
<td>5</td>
<td>compiler optimizer</td>
<td>academia</td>
</tr>
<tr>
<td>6</td>
<td>data scientist</td>
<td>academia</td>
</tr>
<tr>
<td>7</td>
<td>compiler optimizer</td>
<td>academia</td>
</tr>
<tr>
<td>8</td>
<td>data scientist</td>
<td>retail</td>
</tr>
<tr>
<td>9</td>
<td>compiler optimizer</td>
<td>finance</td>
</tr>
</tbody>
</table>

1. Begin by importing libraries, and reading data into a Pandas DataFrame:

```python
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:

```python
print(irisdf.columns)
```

Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'], dtype='object')

3. Summary statistics include minimum, maximum, mean, median, percentiles, and more:

```python
print('length:', len(irisdf))  # length of data set
print('shape:', irisdf.shape)  # length * width
print('size:', irisdf.size)    # length * width
print('min: ', irisdf['sepal_width'].min())
print('max: ', irisdf['sepal_width'].max())
print('mean: ', irisdf['sepal_width'].mean())
print('median: ', irisdf['sepal_width'].median())
print('50th percentile: ', irisdf['sepal_width'].quantile(0.5))  # 50th percentile, also known as median
print('5th percentile: ', irisdf['sepal_width'].quantile(0.05))
print('10th percentile: ', irisdf['sepal_width'].quantile(0.1))
print('95th percentile: ', irisdf['sepal_width'].quantile(0.95))
```
4. 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.

```python
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`.

```python
print(irisdf.dtypes)
print()
irisdf['class'] = irisdf['class'].astype('category')
print(irisdf.dtypes)
```

<table>
<thead>
<tr>
<th>sepal_length</th>
<th>float64</th>
</tr>
</thead>
<tbody>
<tr>
<td>sepal_width</td>
<td>float64</td>
</tr>
<tr>
<td>petal_length</td>
<td>float64</td>
</tr>
<tr>
<td>petal_width</td>
<td>float64</td>
</tr>
<tr>
<td>class</td>
<td>object</td>
</tr>
<tr>
<td>dtype: object</td>
<td></td>
</tr>
<tr>
<td>sepal_length</td>
<td>float64</td>
</tr>
<tr>
<td>sepal_width</td>
<td>float64</td>
</tr>
</tbody>
</table>

(continues on next page)
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.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

timedelta = pd.DataFrame(np.random.randn(1000, 4), index=pd.date_range('1/1/2015', periods=1000), columns=list('ABCD'))
timedelta = timedelta.cumsum()
timedelta.plot()
```

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:

```python
plt.hist(iris['sepal_length'])
plt.show()
```
Bar charts

The following sample code produces a bar chart of the industries of customers in the customer data set.

```python
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

<table>
<thead>
<tr>
<th>row</th>
<th>customer_id</th>
<th>title</th>
<th>industry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>data scientist</td>
<td>retail</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>data scientist</td>
<td>academia</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>compiler optimizer</td>
<td>academia</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>data scientist</td>
<td>finance</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>compiler optimizer</td>
<td>academia</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>data scientist</td>
<td>academia</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>compiler optimizer</td>
<td>academia</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>data scientist</td>
<td>retail</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>compiler optimizer</td>
<td>finance</td>
</tr>
</tbody>
</table>

To sort by industry and show the results:

customerdf.sort_values(by=['industry'])
To sort by industry and then title:

```
customerdf.sort_values(by=['industry', 'title'])
```

<table>
<thead>
<tr>
<th>row</th>
<th>customer_id</th>
<th>title</th>
<th>industry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>data scientist</td>
<td>academia</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>compiler optimizer</td>
<td>academia</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>compiler optimizer</td>
<td>academia</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>data scientist</td>
<td>academia</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>compiler optimizer</td>
<td>academia</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>data scientist</td>
<td>finance</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>compiler optimizer</td>
<td>finance</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>data scientist</td>
<td>retail</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>data scientist</td>
<td>retail</td>
</tr>
</tbody>
</table>

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().
```

```python
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())
```

### 3.5. Working with data
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:

```python
pd.cut(irisdf['sepal_length'], 4).head()
```

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
</tr>
<tr>
<td>1</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
</tr>
<tr>
<td>2</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
</tr>
<tr>
<td>3</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
</tr>
<tr>
<td>4</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
<td>(4.296, 5.2]</td>
</tr>
</tbody>
</table>

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] ]

Or make a custom bin array to divide the sepal length measurements into integer-sized bins from 4 through 8:

```python
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/.

3.5.3 Data preparation

Anaconda Enterprise supports data preparation using numeric libraries such as NumPy, SciPy, and Pandas.

These examples use this small data file vendors.csv:

<table>
<thead>
<tr>
<th>Vendor Number</th>
<th>Vendor Name</th>
<th>Month</th>
<th>Day</th>
<th>Year</th>
<th>Active</th>
<th>Open Orders</th>
<th>2015</th>
<th>2016</th>
<th>Percent Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;104.0&quot;</td>
<td>ACME Inc</td>
<td>2</td>
<td>15</td>
<td>2014</td>
<td>&quot;Y&quot;</td>
<td>200</td>
<td>&quot;$45,000.00&quot;</td>
<td>&quot;$54000.00&quot;</td>
<td>20.00%</td>
</tr>
<tr>
<td>205</td>
<td>Apogee LTD</td>
<td>8</td>
<td>12</td>
<td>2015</td>
<td>&quot;Y&quot;</td>
<td>150</td>
<td>&quot;$29,000.00&quot;</td>
<td>&quot;$30,450.00&quot;</td>
<td>5.00%</td>
</tr>
<tr>
<td>143</td>
<td>Zenith Co</td>
<td>4</td>
<td>5</td>
<td>2014</td>
<td>&quot;Y&quot;</td>
<td>290</td>
<td>&quot;$18,000.00&quot;</td>
<td>&quot;$23400.00&quot;</td>
<td>30.00%</td>
</tr>
<tr>
<td>166</td>
<td>Hollerith Propulsion</td>
<td>9, 25, 2015</td>
<td>&quot;Y&quot;</td>
<td>180</td>
<td>&quot;Y&quot;</td>
<td>180</td>
<td>&quot;$48,000.00&quot;</td>
<td>&quot;$48960.00&quot;</td>
<td>2.00%</td>
</tr>
<tr>
<td>180</td>
<td>Airtek Industrial</td>
<td>8, 2</td>
<td>2, 2014</td>
<td>&quot;N&quot;</td>
<td>Closed</td>
<td>&quot;$23,000.00&quot;</td>
<td>&quot;$17250.00&quot;</td>
<td>-25.00%</td>
<td></td>
</tr>
</tbody>
</table>

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:

```python
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:

```python
df['2015']+df['2016']
```

```
0   $45,000.00$54000.00
1   $29,000.00$30,450.00
2   $18,000.00$23,400.00
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
Month          5 non-null int64
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.

```python
df['2015'].astype('float')
```

ValueError: could not convert string to float: '$23,000.00'

```python
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.

```python
df['Active'].astype('bool')
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>True</td>
</tr>
<tr>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>True</td>
</tr>
<tr>
<td>4</td>
<td>True</td>
</tr>
</tbody>
</table>
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.

```python
def currency_to_float(a):
    return float(a.replace(',', '').replace('$', ''))
```

Test the function on the 2015 column with the `apply()` function:

```python
df['2015'].apply(currency_to_float)
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>45000.0</td>
</tr>
<tr>
<td>1</td>
<td>29000.0</td>
</tr>
<tr>
<td>2</td>
<td>18000.0</td>
</tr>
<tr>
<td>3</td>
<td>48000.0</td>
</tr>
<tr>
<td>4</td>
<td>23000.0</td>
</tr>
</tbody>
</table>
Name: 2015, dtype: float64

Convert the 2015 and 2016 columns and show the dtypes:

```python
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:

```python
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

```python
df['Percent Growth'] = df['Percent Growth'].apply(percent_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 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:

```python
np.where(df['Active'] == "Y", True, False)
```

array([ True, True, True, True, False])

```python
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 
(continues on next page)
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')
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>200.0</td>
</tr>
<tr>
<td>1</td>
<td>150.0</td>
</tr>
<tr>
<td>2</td>
<td>290.0</td>
</tr>
<tr>
<td>3</td>
<td>180.0</td>
</tr>
<tr>
<td>4</td>
<td>NaN</td>
</tr>
</tbody>
</table>

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)
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>200.0</td>
</tr>
<tr>
<td>1</td>
<td>150.0</td>
</tr>
<tr>
<td>2</td>
<td>290.0</td>
</tr>
<tr>
<td>3</td>
<td>180.0</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
</tr>
</tbody>
</table>

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']])
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2014-02-15</td>
</tr>
<tr>
<td>1</td>
<td>2015-08-12</td>
</tr>
<tr>
<td>2</td>
<td>2014-04-05</td>
</tr>
<tr>
<td>3</td>
<td>2015-09-25</td>
</tr>
<tr>
<td>4</td>
<td>2014-08-02</td>
</tr>
</tbody>
</table>

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
```

<table>
<thead>
<tr>
<th>Vendor Number</th>
<th>Vendor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>int64</td>
<td>object</td>
</tr>
</tbody>
</table>

(continues on next page)
Month int64
Day int64
Year int64
Active bool
Open Orders float64
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.

```python
df2 = pd.read_csv('vendors.csv',
                 dtype={'Vendor Number': 'int'},
                 converters={'2015': currency_to_float,
                             '2016': currency_to_float,
                             'Percent Growth': percent_to_float})
df2['Active'] = np.where(df2['Active'] == 'Y', True, False)
df2['Open Orders'] = pd.to_numeric(df2['Open Orders'], errors='coerce').fillna(0)
df2['First Purchase Date'] = pd.to_datetime(df2[['Month', 'Day', 'Year']])
df2
```

<table>
<thead>
<tr>
<th>Vendor Number</th>
<th>Vendor Name</th>
<th>Month</th>
<th>Day</th>
<th>Year</th>
<th>Active</th>
<th>Open Orders</th>
<th>2015</th>
<th>2016</th>
<th>Percent Growth</th>
<th>First Purchase Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>104 ACME Inc</td>
<td>2</td>
<td>15</td>
<td>2014</td>
<td>True</td>
<td>200.0</td>
<td>45000.0</td>
<td>54000.0</td>
<td>0.20</td>
<td>2014-02-15</td>
</tr>
<tr>
<td>1</td>
<td>205 Apogee LTD</td>
<td>8</td>
<td>12</td>
<td>2015</td>
<td>True</td>
<td>150.0</td>
<td>29000.0</td>
<td>30450.0</td>
<td>0.05</td>
<td>2015-08-12</td>
</tr>
<tr>
<td>2</td>
<td>143 Zenith Co</td>
<td>4</td>
<td>5</td>
<td>2014</td>
<td>True</td>
<td>290.0</td>
<td>18000.0</td>
<td>23400.0</td>
<td>0.30</td>
<td>2014-04-05</td>
</tr>
<tr>
<td>3</td>
<td>166 Hollerith Propulsion</td>
<td>9</td>
<td>25</td>
<td>2015</td>
<td>True</td>
<td>180.0</td>
<td>48000.0</td>
<td>48960.0</td>
<td>0.02</td>
<td>2015-09-25</td>
</tr>
<tr>
<td>4</td>
<td>180 Airtek Industrial</td>
<td>8</td>
<td>2</td>
<td>2014</td>
<td>False</td>
<td>0.0</td>
<td>23000.0</td>
<td>17250.0</td>
<td>-0.25</td>
<td>2014-08-02</td>
</tr>
</tbody>
</table>

```python
df2.dtypes
```

<table>
<thead>
<tr>
<th>Vendor Number</th>
<th>Vendor Name</th>
<th>Month</th>
<th>Day</th>
<th>Year</th>
<th>Active</th>
<th>Open Orders</th>
</tr>
</thead>
<tbody>
<tr>
<td>int64</td>
<td>object</td>
<td>int64</td>
<td>int64</td>
<td>int64</td>
<td>bool</td>
<td>float64</td>
</tr>
</tbody>
</table>
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:

```python
left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                    'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3']})
right = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})
pd.merge(left, right, on='key')
```

<table>
<thead>
<tr>
<th>key</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>K0</td>
<td>A0</td>
<td>B0</td>
<td>C0</td>
</tr>
<tr>
<td>1</td>
<td>K1</td>
<td>A1</td>
<td>B1</td>
<td>C1</td>
</tr>
<tr>
<td>2</td>
<td>K2</td>
<td>A2</td>
<td>B2</td>
<td>C2</td>
</tr>
<tr>
<td>3</td>
<td>K3</td>
<td>A3</td>
<td>B3</td>
<td>C3</td>
</tr>
</tbody>
</table>

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:

```python
left = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                     'key2': ['K0', 'K1', 'K0', 'K1'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})
right = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                      'key2': ['K0', 'K0', 'K0', 'K0'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})
pd.merge(left, right, on=['key1', 'key2'])
```

<table>
<thead>
<tr>
<th>key1</th>
<th>key2</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>K0</td>
<td>K0</td>
<td>A0</td>
<td>B0</td>
<td>C0</td>
</tr>
<tr>
<td>1</td>
<td>K1</td>
<td>K0</td>
<td>A2</td>
<td>B2</td>
<td>C1</td>
</tr>
<tr>
<td>2</td>
<td>K1</td>
<td>K0</td>
<td>A2</td>
<td>B2</td>
<td>C2</td>
</tr>
</tbody>
</table>

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

When merging or joining tables, the `key` parameter is used to select columns or indexes that will be used as the basis for matching rows.

#### pd.merge(left, right, how='left', on=['key1', 'key2'])

<table>
<thead>
<tr>
<th>key1</th>
<th>key2</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>K0</td>
<td>K0</td>
<td>A0</td>
<td>B0</td>
<td>C0</td>
<td>D0</td>
</tr>
<tr>
<td>K0</td>
<td>K1</td>
<td>A1</td>
<td>B1</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>K1</td>
<td>K0</td>
<td>A2</td>
<td>B2</td>
<td>C1</td>
<td>D1</td>
</tr>
<tr>
<td>K1</td>
<td>K0</td>
<td>A2</td>
<td>B2</td>
<td>C2</td>
<td>D2</td>
</tr>
<tr>
<td>K2</td>
<td>K1</td>
<td>A3</td>
<td>B3</td>
<td>NaN</td>
<td>NaN</td>
</tr>
</tbody>
</table>

#### pd.merge(left, right, how='right', on=['key1', 'key2'])

<table>
<thead>
<tr>
<th>key1</th>
<th>key2</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>K0</td>
<td>K0</td>
<td>A0</td>
<td>B0</td>
<td>C0</td>
<td>D0</td>
</tr>
<tr>
<td>K0</td>
<td>K1</td>
<td>A2</td>
<td>B2</td>
<td>C1</td>
<td>D1</td>
</tr>
<tr>
<td>K0</td>
<td>K1</td>
<td>A2</td>
<td>B2</td>
<td>C2</td>
<td>D2</td>
</tr>
<tr>
<td>K2</td>
<td>K0</td>
<td>NaN</td>
<td>NaN</td>
<td>C3</td>
<td>D3</td>
</tr>
</tbody>
</table>

#### pd.merge(left, right, how='outer', on=['key1', 'key2'])

<table>
<thead>
<tr>
<th>key1</th>
<th>key2</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>K0</td>
<td>K0</td>
<td>A0</td>
<td>B0</td>
<td>C0</td>
<td>D0</td>
</tr>
<tr>
<td>K0</td>
<td>K1</td>
<td>A1</td>
<td>B1</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>K1</td>
<td>K0</td>
<td>A2</td>
<td>B2</td>
<td>C1</td>
<td>D1</td>
</tr>
<tr>
<td>K1</td>
<td>K0</td>
<td>A2</td>
<td>B2</td>
<td>C2</td>
<td>D2</td>
</tr>
<tr>
<td>K2</td>
<td>K1</td>
<td>A3</td>
<td>B3</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>K2</td>
<td>K0</td>
<td>NaN</td>
<td>NaN</td>
<td>C3</td>
<td>D3</td>
</tr>
</tbody>
</table>

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.

#### Example

```python
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')
```

<table>
<thead>
<tr>
<th>A_x</th>
<th>B</th>
<th>A_y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

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.

```python
left = pd.DataFrame({'A': ['A0', 'A1', 'A2'],
                    'B': ['B0', 'B1', 'B2']},
                   index=['K0', 'K1', 'K2'])
right = pd.DataFrame({'C': ['C0', 'C2', 'C3'],
                      'D': ['D0', 'D2', 'D3']},
                      index=['K0', 'K1', 'K2'])
pd.merge(left, right, how='inner')
```
left.join(right)

A  B  C  D
K0 A0 B0 C0 D0
K1 A1 B1 NaN NaN
K2 A2 B2 C2 D2

left.join(right, how='outer')

A  B  C  D
K0 A0 B0 C0 D0
K1 A1 B1 NaN NaN
K2 A2 B2 C2 D2
K3 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:

left = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'key': ['K0', 'K1', 'K0', 'K1']})
right = pd.DataFrame({'C': ['C0', 'C1'],
                      'D': ['D0', 'D1'],
                      index=['K0', 'K1']})
left.join(right, on='key')

A  B  key  C  D
0 A0 B0 K0 C0 D0
1 A1 B1 K1 C1 D1
2 A2 B2 K0 C0 D0
3 A3 B3 K1 C1 D1

You can join on multiple keys if the passed DataFrame has a MultiIndex:

left = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'key1': ['K0', 'K0', 'K1', 'K2'],
                    'key2': ['K0', 'K1', 'K0', 'K1']})
index = pd.MultiIndex.from_tuples([('K0', 'K0'), ('K1', 'K0'),
                                   ('K2', 'K0'), ('K2', 'K1')])

(continues on next page)
right = pd.DataFrame({
    'C': ['C0', 'C1', 'C2', 'C3'],
    'D': ['D0', 'D1', 'D2', 'D3'],
},
index=index)
right

left.join(right, on=['key1', 'key2'])

Note that this defaulted to a left join, but other types are also available:

left.join(right, on=['key1', 'key2'], how='inner')

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')

Join has similar arguments lsuffix and rsuffix.

left = left.set_index('k')
right = right.set_index('k')
left.join(right, lsuffix='_l', rsuffix='_r')
You can join a list or tuple of DataFrames on their indexes:

```python
right2 = pd.DataFrame({'v': [7, 8, 9]}, index=['K1', 'K1', 'K2'])
left.join([right, right2])
```

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:

```python
df1 = pd.DataFrame([[np.nan, 3., 5.],
                    [-4.6, np.nan, np.nan],
                    [np.nan, 7., np.nan]])
df2 = pd.DataFrame([[42.6, np.nan, -8.2],
                    [-5., 1.6, 4]],
                    index=[1, 2])
df1.combine_first(df2)
```

The method `update()` overwrites values in a frame with values from another frame:

```python
df1.update(df2)
df1
```

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:
import pandas as pd
import numpy as np

df = pd.DataFrame({'VendorNumber': [104, 205, 143, 166, 180],
                  'VendorName': ['ACME Inc', 'Apogee LTD', 'Zenith Co', 'Hollerith Propulsion', 'Airtek Industrial'],
                  'Active': [True, True, True, True, False],
                  'OpenOrders': [200, 150, 290, 180, 0],
                  'Purchases2015': [45000.0, 29000.0, 18000.0, 48000.0, 23000.0],
                  'Purchases2016': [54000.0, 30450.0, 23400.0, 48960.0, 17250.0],
                  'PercentGrowth': [0.20, 0.05, 0.30, 0.02, -0.25],
                  'FirstPurchaseDate': ['2014-02-15', '2015-08-12', '2014-04-05', '2015-09-25', '2014-08-02']})
df['FirstPurchaseDate'] = df['FirstPurchaseDate'].astype('datetime64[ns]')
df

To filter only certain rows from a DataFrame, call the query method with a boolean expression based on the column names.

df.query('OpenOrders>160')

Filtering can be done with indices instead of queries:

df[(df.OpenOrders < 190) & (df.Active == True)]

3.5. Working with data
3.5.4 Using statistics

Anaconda Enterprise 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:

- 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.

```python
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:

```python
#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:

```python
#Description of the dataset
print(boston.DESCR)
```

Boston House Prices dataset
============================

Notes
------
Data Set Characteristics:
:Number of Instances: 506
Number of Attributes: 13 numeric/categorical predictive

Median Value (attribute 14) is usually the target

Attribute Information (in order):
- CRIM  per capita crime rate by town
- ZN  proportion of residential land zoned for lots over 25,000 sq.ft.
- 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
- B  1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- 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 has been used in many machine learning papers that address regression problems.

References:
- many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)

Show the first five records of the dataset:

```python
# Check the first five records
df.head()
```

<table>
<thead>
<tr>
<th>row</th>
<th>CRIM</th>
<th>ZN</th>
<th>INDUS</th>
<th>CHAS</th>
<th>NOX</th>
<th>RM</th>
<th>AGE</th>
<th>DIS</th>
<th>RAD</th>
<th>TAX</th>
<th>PTRATIO</th>
<th>B</th>
<th>LSTAT</th>
<th>MEDV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(continues on next page)
Show summary statistics for each variable: count, mean, standard deviation, minimum, 25th 50th and 75th percentiles, and maximum.

```
#Descriptions of each variable
df.describe()
```

<table>
<thead>
<tr>
<th>stat</th>
<th>CRIM</th>
<th>ZN</th>
<th>INDUS</th>
<th>CHAS</th>
<th>NOX</th>
<th>RM</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIS</td>
<td>RAD</td>
<td>TAX</td>
<td>PTRATIO</td>
<td>B</td>
<td>LSTAT</td>
<td>target</td>
<td></td>
</tr>
<tr>
<td>count</td>
<td>506.000000</td>
<td>506.000000</td>
<td>506.000000</td>
<td>506.000000</td>
<td>506.000000</td>
<td>506.000000</td>
<td>506.000000</td>
</tr>
<tr>
<td>mean</td>
<td>3.593761</td>
<td>11.363636</td>
<td>11.136779</td>
<td>0.069170</td>
<td>0.554695</td>
<td>6.284634</td>
<td>68.574901 3.795043</td>
</tr>
<tr>
<td>std</td>
<td>8.596783</td>
<td>23.322453</td>
<td>6.860353</td>
<td>0.253994</td>
<td>0.115878</td>
<td>0.702617</td>
<td>28.148861 2.1085710</td>
</tr>
<tr>
<td>min</td>
<td>0.006320</td>
<td>0.000000</td>
<td>0.460000</td>
<td>0.000000</td>
<td>0.385000</td>
<td>3.561000</td>
<td>2.900000 1.730000 5.000000</td>
</tr>
<tr>
<td>25%</td>
<td>0.082045</td>
<td>0.000000</td>
<td>5.190000</td>
<td>0.000000</td>
<td>0.449000</td>
<td>5.885500</td>
<td>45.025000 2.000000</td>
</tr>
<tr>
<td>50%</td>
<td>0.256510</td>
<td>0.000000</td>
<td>9.690000</td>
<td>0.000000</td>
<td>0.538000</td>
<td>6.208500</td>
<td>77.500000 3.795043</td>
</tr>
<tr>
<td>75%</td>
<td>3.647423</td>
<td>12.500000</td>
<td>18.100000</td>
<td>0.000000</td>
<td>0.624000</td>
<td>6.623500</td>
<td>94.075000 5.188425</td>
</tr>
<tr>
<td>max</td>
<td>88.976200</td>
<td>100.000000</td>
<td>27.740000</td>
<td>1.000000</td>
<td>0.871000</td>
<td>8.780000</td>
<td>100.000000 12.000000</td>
</tr>
<tr>
<td>126500</td>
<td>24.000000</td>
<td>711.000000</td>
<td>22.000000</td>
<td>396.900000</td>
<td>37.970000</td>
<td>50.000000</td>
<td></td>
</tr>
</tbody>
</table>

**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
```
Format with asterisks

Format the correlation matrix by rounding the numbers to two decimal places and adding asterisks to denote statistical significance:

```python
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))
```

(continues on next page)
# 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)

Heatmap

Heatmap of the correlation matrix:
sns.heatmap(corr,
            xticklabels=corr.columns,
            yticklabels=corr.columns)
Pairwise distributions with seaborn

```python
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.

```python
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.

```python
# Define features as X, target as y.
X = df.drop('target', axis='columns')
y = df['target']

# Splitting the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

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.

```python
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# The intercept
print('Intercept: \n', regressor.intercept_)

# The coefficients
print('Coefficients: \n', pd.Series(regressor.coef_, index=X.columns, name='coefficients'))
```

```
Intercept: 36.98045533762056
Coefficients:
    CRIM    -0.116870
       2N     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
        B     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.

```python
# Predicting the Test set results
y_pred = regressor.predict(X_test)

# Visualising the Test set results
# code adapted from https://joomik.github.io/Housing/
fig, ax = plt.subplots()
ax.scatter(y_test, y_pred, 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()
```

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:

```python
# 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**

```python
model = sm.ols('target ~ AGE + B + CHAS + CRIM + DIS + INDUS + LSTAT + NOX + PTRATIO + RAD + RM + TAX + ZN', df)
result = model.fit()
result.summary()
```

### OLS Regression Results

<table>
<thead>
<tr>
<th>Dep. Variable:</th>
<th>target</th>
<th>R-squared:</th>
<th>0.741</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model:</td>
<td>OLS</td>
<td>Adj. R-squared:</td>
<td>0.734</td>
</tr>
<tr>
<td>Method:</td>
<td>Least Squares</td>
<td>F-statistic:</td>
<td>108.1</td>
</tr>
<tr>
<td>Date:</td>
<td>Thu, 23 Aug 2018</td>
<td>Prob (F-statistic):</td>
<td>6.95e-135</td>
</tr>
<tr>
<td>Time:</td>
<td>07:29:16</td>
<td>Log-Likelihood:</td>
<td>-1498.8</td>
</tr>
<tr>
<td>Df Residuals:</td>
<td>492</td>
<td>BIC:</td>
<td>3085.</td>
</tr>
<tr>
<td>Df Model:</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Covariance Type:</td>
<td>nonrobust</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| coef   | std err | t     | P>|t| | [0.025 | 0.975] |
|--------|---------|-------|------|--------|--------|
| Intercept | 36.4911 | 5.104 | 7.149 | 0.000  | 26.462  | 46.520 |
| AGE    | 0.0008  | 0.013 | 0.057 | 0.955  | -0.025  | 0.027  |
| B      | 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 |

(continues on next page)
### Principal component analysis

The initial dataset has a number of feature or predictor variables and one target variable to predict.

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:

```python
# 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:

```python
# Project data to 3 dimensions
pca = PCA(n_components=3)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents, 
                          columns = ['principal component 1', 'principal component 2', 'principal component 3'])
principalDf.head()
```

<table>
<thead>
<tr>
<th>row</th>
<th>principal component 1</th>
<th>principal component 2</th>
<th>principal component 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-2.097842</td>
<td>0.777102</td>
<td>0.335076</td>
</tr>
<tr>
<td>1</td>
<td>-1.456412</td>
<td>0.588088</td>
<td>-0.701340</td>
</tr>
<tr>
<td>2</td>
<td>-2.074152</td>
<td>0.602185</td>
<td>0.161234</td>
</tr>
<tr>
<td>3</td>
<td>-2.611332</td>
<td>-0.005981</td>
<td>-0.101940</td>
</tr>
<tr>
<td>4</td>
<td>-2.457972</td>
<td>0.098860</td>
<td>-0.077893</td>
</tr>
</tbody>
</table>
```

Show a 2D graph of this data:
Show a 3D graph of this data:

```python
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
           principalDf['principal component 3'])
plt.show()
```

Measure how much of the variance is explained by each of the three components:

```python
# Variance explained by each component
explained_variance = pca.explained_variance_ratio_
explained_variance
```

```python
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:

```python
y_test_linear = y_test
y_pred_linear = y_pred
X = principalDf
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

Plot the predictions from the linear regression in green again, and the new predictions in blue:

```python
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:

```python
print("Linear regression mean squared error: \$%.2f\$ % mean_squared_error(y_test_linear, y_pred_linear))
print("PCA mean squared error: \$%.2f\$ % mean_squared_error(y_test, y_pred))
```

Linear regression mean squared error: 29.79
PCA mean squared error: 43.49
3.6 Working with deployments

When you deploy a project, Anaconda Enterprise finds and builds all of the software dependencies—the libraries on which the project depends in order to run—and encapsulates them, so they are completely self-contained and easy to share with others. This is called a deployment.

Whether you deploy a notebook, Bokeh application or REST API, everything needed to deploy and run the project is included. You can then share your deployment with others so they can interact with it.

**Note:** You can create multiple deployments from a single project. Each deployment can be a different version, and can be shared with different users.

After logging in to Anaconda Enterprise, click **Deployments** to view a list of all of the deployments you have created—or that others have shared with you. Simply click on a deployment to open the deployed Notebook or application and interact with it.

Anaconda Enterprise maintains a log of all deployments created by all users in the Administrator’s Authentication Center.

3.6.1 Deploying a project

When you are ready to use your interactive visualization, live notebook or machine learning model, you deploy the associated project. You can also deploy someone else’s project if you have been added as a collaborator on the project. See **Collaborating on projects** for more information.

When you deploy a project, Anaconda Enterprise finds and builds the software dependencies—all of the libraries required for it to run—and encapsulates them so they are completely self-contained. This allows you to easily share it with others.

You configure how a project is deployed by adding the appropriate command to run the project in the configuration file `anaconda-project.yml`. You can also accept the default command, like the following example Bokeh app:

See **Configuring project settings** for more information about adding deployment commands to project.
To deploy a project:

1. Select it in the Projects list and click Deploy.

2. 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.

3. If there are multiple versions of the project, select the version you want to deploy.

4. Select the command to use to deploy the project. 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.

5. Enter the URL where you want the deployment to be hosted in the Static URL field.

   Note: This is the URL you’ll use to call the deployment from within a web application, and therefore it must be unique. Disable the Static URL toggle if you want Anaconda Enterprise to automatically generate a URL for the deployment.

6. Choose whether you want to keep the deployment Private—and therefore accessible to authenticated platform users, only—or make it Public, and therefore available to non-authenticated users. After it’s deployed, you can share the deployment with others.

7. Click Deploy. Anaconda Enterprise displays the status of the deployment, then lists it in the project’s Deployments. Private deployments are displayed with a lock next to their name, to indicate their secure status.

   Note: It may take a few minutes to obtain and build all the dependencies for the project deployment.
To view or interact with a deployment, click its name in the list.

You can also schedule a project to be deployed on a regular basis or at a specific time.

### 3.6.2 Deploying a REST API

Anaconda Enterprise enables you to deploy your machine learning or predictive models as a REST API endpoint so others can query and consume results from them. REST APIs are web server endpoints, or callable URLs, which provide results based on a query, allowing developers to create applications that programmatically query and consume them via other user interfaces or applications.

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.

REST API endpoints deployed with Anaconda Enterprise 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 Anaconda Enterprise.

The process of deploying a REST API involves the following steps:

- **Create a project** to encapsulate all of the components necessary to use or run your model.

- **Deploy the project** with the `rest_api` command (shown in Step 4 below) to build the software dependencies—all of the libraries required for it to run—and encapsulate them so they are completely self-contained.
• Share the deployment, or the URL of the endpoint, and generate a unique token so that they can connect to the deployment and use it from within notebooks, APIs or other applications.

Using the API wrapper

As an alternative to using the REST API wrapper provided with Anaconda Enterprise, you can construct an API endpoint using any web framework and serve the endpoint on port 8086 within your deployment, to make it available as a secure REST API endpoint in Anaconda Enterprise.

Follow this process to wrap your code with an API:

1. Open the Jupyter Notebook and add this code to be able to handle HTTP requests. Define a global REQUEST JSON string that will be replaced on each invocation of the API.

   ```python
   import json
   REQUEST = json.dumps({
       'path' : {},
       'args' : {},
       'body': {}
   })
   ```

2. Import the Anaconda Enterprise publish function.

   ```python
   from anaconda.enterprise import publish
   @publish(methods=['GET', 'POST'])
   def function():
       ...
       return json.dumps(...)  
   ```

3. Add the deployment command and an appropriate environment variable to the anaconda-project.yml file:

   ```yaml
   commands:
   deploy-api:
       rest_api: {notebook}.ipynb
       supports_http_options: true
       default: true
   
   variables:
   KG_FORCE_KERNEL_NAME:
       default: python3
   ```

4. Ensure the anaconda-enterprise channel is listed under channels: and the anaconda-enterprise-web-publisher package is listed under packages:. For example:

   ```yaml
   packages:
   - python=3.6
   - pandas
   - dask
   - matplotlib
   - scikit-learn
   - requests
   - anaconda-enterprise-web-publisher
   
   channels:
   - defaults
   - anaconda-enterprise
   ```
5. Use the following command to test the API within your notebook session (without deploying it):

```
anaconda-project run deploy-api
```

6. Now if you visit http://localhost:8888/{function} from within a notebook session you will see the results of your function.

From within a notebook session, execute the following command:

```
curl localhost:8888/{function}
```

7. Click the **Deploy** icon in the toolbar to deploy the project as an API.

   This deploys the notebook as an API which you can then query.

8. To query externally, create a token and find the url to the running project.

   Example using **curl**:

   ```
   export TOKEN="<generated-token-goes-here>"  # save long string of text in variable
curl -L -H "Authorization: Bearer $TOKEN" <url-of-project>
   ```

   The `-L` option tells curl to follow redirects. The `-H` adds a header. In this case `-H` adds the token required to authorize the client to visit that URL.

   If you deploy the project as described above you can add the `-X POST` option to curl to access that function.

### 3.6.3 Deploying a Flask application

The process of deploying a Flask application (website and REST APIs) on Anaconda Enterprise 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.

   ```python
   from flask import Flask, Blueprint

   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.run()
   ```
Running behind an HTTPS proxy

Anaconda Enterprise 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 Anaconda Enterprise.

The simplest way to do this is with the `ProxyFix` function from `werkzeug`. More information about proxies is provided here.

```python
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:

```yaml
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`:

```python
import sys
from argparse import ArgumentParser

# ... 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')

    # ... the Flask application blueprint
```

(continues on next page)
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.

Here is the complete code for the Hello World application.

```python
import sys
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()
```

(continues on next page)
3.6.4 Sharing deployments

After you have deployed a project, you can share the deployment with others. You can share a deployment publicly, with other Anaconda Enterprise users, or both.

Any collaborators you add to your deployment will see your deployment in their Deployments list when they log in to AE.

**Note:** Your Anaconda Enterprise 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 Anaconda Enterprise 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 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.

### 3.6.5 Scheduling deployments

If you want to **deploy a project** on a regular basis, Anaconda Enterprise 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 we recommend 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.

![Create a Schedule](image)

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, we recommend 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.

![Schedule page]

10. Click on a schedule in the list to view and edit its details.

![Schedule 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.
3.6.6 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.

### 3.7 Using GPUs in sessions and deployments

Anaconda Enterprise 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*. 

---

3.7. Using GPUs in sessions and deployments 233
3.8 Configuring your user settings

Anaconda Enterprise 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 Anaconda Enterprise 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 Anaconda Enterprise, select Password.
- To enable two-factor authentication for your account, select Authenticator.
- To view a history of your sessions using Anaconda Enterprise, select Sessions. You can also log out of all sessions in one click here.
- To view a list of AE 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.

3.8.1 Configuring access to version control

If your Administrator has configured Anaconda Enterprise 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. We recommend you create an ever-lasting token, so you can retain permanent access to your files from within Anaconda Enterprise.

Your auth token must also have the following permissions:

<table>
<thead>
<tr>
<th>External Repository</th>
<th>Permissions Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitbucket Enterprise</td>
<td>Admin access for Projects and Repositories</td>
</tr>
<tr>
<td>GitHub Enterprise</td>
<td>repo:status, repo_deployment, public_repo, repo:invite, and delete_repo</td>
</tr>
<tr>
<td>GitLab Enterprise</td>
<td>Check the api access check box when creating your access token.</td>
</tr>
</tbody>
</table>

Note: You’ll be prompted to configure your personal access token when you attempt to create your first project in Anaconda Enterprise, 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 to access the repository within your sessions and deployments without having to leave the platform. Anaconda Enterprise creates a repository for each project that you create.

### 3.8.2 Storing secrets

Anaconda Enterprise 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 for the item then 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.

3.8. Configuring your user settings
3.9 Visualizations and dashboards

Anaconda Enterprise 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, Anaconda Enterprise provides sample projects of Bokeh applications for clustering and cross filtering data. There are also several examples of AE5 projects that use PyViz 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, panel, pyct, and bokeh:

   ```bash
   anaconda-project add-packages hvplot 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:

```python
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.Pane(p).servable()
```

**Note:** In this example, the data is being read from the Internet. Alternatively, you could [download](http://bit.ly/tips-csv) 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 Anaconda Enterprise will use when you deploy the notebook:

   ```yaml
   commands:
   scatter-plot:
     unix: panel serve tips.ipynb
     supports_http_options: True
   
   5. Save and commit your changes.

6. Now you’re ready to deploy the project.
Chapter 3. Using Anaconda Enterprise
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.
3.10 Machine learning and deep learning

Anaconda Enterprise facilitates machine learning and deep learning by enabling you to develop models, train them, and deploy them. You can also use AE to query and score models that have been deployed as a REST API.

To help get you started, Anaconda Enterprise includes several sample notebooks for common repetitive tasks. You can access them from the gallery of Sample Projects available from Projects. See Working with projects for more information.

We’ve also provided a walkthrough of the process for creating an interactive data visualization.

3.10.1 Developing models

Anaconda Enterprise 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 Anaconda Enterprise 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 Kernal 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.

3.10.2 Training models

Anaconda Enterprise 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 Anaconda Enterprise 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.

### 3.10.3 Deploying models as endpoints

Anaconda Enterprise 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 Anaconda Enterprise 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.

### 3.10.4 Querying and scoring models

Anaconda Enterprise 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 AE, 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 necessarily 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 Anaconda Enterprise.
Anaconda Enterprise provides detailed logs and monitoring information related to the Kubernetes services and containers it uses. You can use the 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.

4.1 The Anaconda Enterprise cluster

As an Operations Center Admin, you can use the Operations Center to configure and monitor the platform.

To access the Operations Center:

1. Log in to Anaconda Enterprise, 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 Anaconda Enterprise 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:

```bash
kubectl describe node 172.31.25.175
```
5. To view the resource utilization per pod, run this command:

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

![Image of Kubernetes dashboard](image)

6. To view the current status of all pods in the cluster, run `kubectl get pods`.

![Image of Kubernetes pods](image)

The following table summarizes common pod states:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running</td>
<td>The pod has been bound to a node, and at least one container is running.</td>
</tr>
<tr>
<td>Pending</td>
<td>The pod is waiting for one or more container images to be created.</td>
</tr>
<tr>
<td>Terminating</td>
<td>The pod is in the process of being terminated.</td>
</tr>
<tr>
<td>Error</td>
<td>An error has occurred with the pod.</td>
</tr>
<tr>
<td>Init:CrashLoopBackoff</td>
<td>The pod failed to start, and will make another attempt in a few minutes.</td>
</tr>
</tbody>
</table>

7. To view information for a particular pod, run the `kubectl describe pod` command against the pod (listed under NAME). For example:

```
kubectl describe pod anaconda-session-89747d7fdb154b89b182d5ea25b2e59-7f497db55w19g
```

4.1. The Anaconda Enterprise cluster
You can also use the Operations Center Logs to gain insights into pod behavior and troubleshoot issues. See logging for more information.

## 4.2 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. Login to Anaconda Enterprise, 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. 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 Anaconda Enterprise as the user.
4.3 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, Anaconda Enterprise 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 template 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.

**Warning:** 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.
The following information is provided for your reference, to help you understand some of the core terminology used in Anaconda Enterprise, 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/.

## 5.1 Glossary

### Anaconda

Sometimes used as shorthand for the Anaconda Distribution, Anaconda, Inc. is the company behind Anaconda Distribution, conda, conda-build and Anaconda Enterprise.

### Anaconda Cloud

A cloud package repository hosting service at https://www.anaconda.org. With a free account, you can publish packages you create to be used publicly.

### Anaconda Distribution

Open source repository of hundreds of popular data science packages, along with the conda package and virtual environment manager for Windows, Linux, and MacOS. Conda makes it quick and easy to install, run, and upgrade complex data science and machine learning environments like scikit-learn, TensorFlow, and SciPy.
Anaconda Enterprise
A software platform for developing, governing, and automating data science and AI pipelines from laptop to production. Enterprise enables collaboration between teams of thousands of data scientists running large-scale model deployments on high-performance production clusters.

Anaconda Navigator
A desktop Graphical User Interface (GUI) included in Anaconda Distribution that allows you to easily use and manage IDEs, conda packages, environments, channels, and notebooks without the need to use the Command Line Interface (CLI).

Anaconda project
An encapsulation of your data science assets to make them easily portable. Projects may include files, environment variables, runnable commands, services, packages, channels, environment specifications, scripts, and notebooks. Each project also includes an anaconda-project.yml configuration file to automate setup, so you can easily run and share it with others. You can create and configure projects from the Enterprise web interface or command line interface.

Channel
A location in the repository where Anaconda Enterprise looks for packages. Enterprise Administrators and users can define channels, determine which packages are available in a channel, and restrict access to specific users or groups.

Commit
To make a set of local changes permanent by copying them to the remote server. Anaconda Enterprise checks to see if your work will conflict with any commits that your colleagues have made on the same project, so the files will not be overwritten unless you so choose to do so.

Conda
An open source package and environment manager that makes it quick and easy to install, run, and upgrade complex data science and machine learning environments like scikit-learn, TensorFlow, and SciPy. Thousands of Python and R packages can be installed with conda on Windows, MacOS X, Linux and IBM Power.

**Conda-build**

A tool used to build conda packages from recipes.

**Conda environment**

A superset of Python virtual environments, conda environments make it easy to create projects with different versions of Python and avoid issues related to dependencies and version requirements. A conda environment maintains its own files, directories, and paths so that you can work with specific versions of libraries and/or Python itself without affecting other Python projects.

**Conda package**

A binary tarball file containing system-level libraries, Python and R modules, executable programs, or other components. Conda tracks dependencies between specific packages and platforms, making it simple to create operating system-specific environments using different combinations of packages.

**Conda recipe**

Instructions used to tell conda-build how to build a package.

**Deployment**

A deployed Anaconda project containing a Notebook, web app, dashboard or machine learning model (exposed via an API). When you deploy a project, Anaconda Enterprise builds a container with all the required dependencies and runtime components—the libraries on which the project depends in order to run—and launches it with the security and access permissions defined by the user. This allows you to easily run and share the application with others.
Interactive data application

Visualizations with sliders, drop-downs and other widgets that allow users to interact with them. Interactive data applications can drive new computations, update plots and connect to other programmatic functionality.

Interactive development environment (IDE)

A suite of software tools that combines everything a developer needs to write and test software. It typically includes a code editor, a compiler or interpreter, and a debugger that the developer accesses through a single Graphical User Interface (GUI). An IDE may be installed locally, or it may be included as part of one or more existing and compatible applications accessed through a web browser.

Jupyter

A popular open source IDE for building interactive Notebooks by the Jupyter Foundation.

JupyterHub

An open source system for hosting multiple Jupyter Notebooks in a centralized location.

JupyterLab

Jupyter Foundation’s successor IDE to Jupyter, with flexible building blocks for interactive and collaborative computing. For Jupyter Notebook users, the interface for JupyterLab is familiar and still contains the notebook, file browser, text editor, terminal, and outputs.

Jupyter Notebook

The default browser-based IDE available in Anaconda Enterprise. It combines the notebook, file browser, text editor, terminal and outputs.
JupyterLab and Jupyter Notebooks are web-based IDE applications that allow you to create and share documents that contain live code in R or Python, equations, visualizations, and explanatory text.

**Package**

Software files and information about the software—such as its name, description, and specific version—bundled into a file that can be installed and managed by a package manager. Packages can be encapsulated into Anaconda projects for easy portability.

**Project template**

Contains all the base files and components to support a particular programming environment. For example, a Python Spark project template contains everything you need to write Python code that connects to Spark clusters. When creating a new project, you can select a template that contains a set of packages and their dependencies.

**Repository**

Any storage location from which software or software assets may be retrieved and installed on a local computer.

**REST API**

A common way to operationalize a machine learning model is through a REST API. A REST API is a web server endpoint, or callable URL, which provides results based on a query. REST APIs allow developers to create applications that incorporate machine learning and prediction, without having to write models themselves.

**Session**

An open project, running in an editor or IDE.

**Spark**

A distributed SQL database and project of the Apache Foundation. While Spark has historically been tightly associated with Apache Hadoop and run on Hadoop clusters, recently the Spark project has sought to separate itself from Hadoop
by releasing support for Spark on Kubernetes. The core data structure in Spark is the RDD (Resilient Distributed Dataset)—a collection of data types, distributed in redundant fashion across many systems. To improve performance, RDDs are cached in memory by default, but can also be written to disk for persistence. Spark Ignite is a project to offer Spark RDDs that can be shared in-memory across applications.

5.2 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 Anaconda Enterprise, consider reviewing the known issues documented here to find workarounds.

5.2.1 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`.

### 5.2.2 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:**
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

5.2.3 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. We recommend 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.

5.2.4 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.

5.2.5 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.

5.2.6 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.

5.2.7 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.
5.2.8 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.

5.2.9 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 JupyterLab). 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.

5.2.10 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
5.2.11 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

---

### 5.2.12 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

---

### 5.2.13 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
5.2.14 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

5.2.15 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

---

### 5.2.16 Anaconda Enterprise 5.0.6

Released: November 9, 2017

### 5.2.17 Anaconda Enterprise 5.0.5

Released: November 7, 2017

### 5.2.18 Anaconda Enterprise 5.0.4

Released: September 12, 2017

### 5.2.19 Anaconda Enterprise 5.0.3

Released: August 31, 2017 (General Availability Release)

### 5.2.20 Anaconda Enterprise 5.0.2

Released: August 15, 2017 (Early Adopter Release)
5.2.21 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 notebooks, 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.

5.3 Known issues

We are aware of the following issues using Anaconda Enterprise. If you’re experiencing other unexpected behavior, consider checking our Support Knowledge Base.

5.3.1 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.

5.3.2 Process for installing the Anaconda Enterprise CLI doesn’t work

The process of installing the Anaconda Enterprise CLI downgrades packages that are essential to the AE CLI, resulting in a conda env that won’t work with the tool.

**Workaround**

Follow this process to create a working conda environment, and activate it:

```
conda create -n cli-test -c https://anaconda.example.com/repository/conda/anaconda-enterprise anaconda-enterprise-cli git python=3.6 cas-mirror
conda activate cli-test
```

To access help for using the Anaconda Enterprise CLI, run `anaconda-enterprise-cli --help`.

5.3.3 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 Anaconda Enterprise. For more information on PyViz compatibility, see https://github.com/pyviz/pyviz_comms#compatibility.

**Workaround**

Open the project in Jupyter Notebook.
5.3.4 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.

5.3.5 Unexpected metadata in a package breaks AE 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 AE channel, or update your conda-forge mirror to pull in the latest packages.

5.3.6 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 Anaconda Enterprise 5.3.1, you can also set global config variables via a config map, as an alternative to using the AE CLI.

5.3.7 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 AE is installed in the default namespace.

5.3.8 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.
5.3.9 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:

```
sudo echo -e "fs.inotify.max_user_watches = 1048576" > /etc/sysctl.d/10-fs.inotify.
max_user_watches.conf
```

5.3.10 Invalid issuer URL causes library to get stuck in a sync loop

When using the Anaconda Enterprise 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:
   
   ```
   # systemctl list-units | grep gravity
   gravity__gravitational.io__planet-master__0.1.87-1714.service loaded
   → active running
   Auto-generated service for the gravitational.io/planet-master:0.1.87-1714
   → package
   gravity__gravitational.io__teleport__2.3.5.service loaded
   → active running
   Auto-generated service for the gravitational.io/teleport:2.3.5 package
   ```

2. Shut down the teleport service:
   ```
   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
```
5.3.11 GPU affinity setting reverts to default during upgrade

When upgrading Anaconda Enterprise 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.

5.3.12 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.

Anaconda Enterprise 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 Anaconda Enterprise` 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:
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:

```
helm template --name anaconda-enterprise /var/lib/gravity/local/packages/unpacked/
  --gravitational.io/AnacondaEnterprise/5.X.X/resources/Anaconda-Enterprise/ -x /var/
  --lib/gravity/local/packages/unpacked/gravitational.io/AnacondaEnterprise/5.X.X/
  --resources/Anaconda-Enterprise/templates/wagonwheel.yaml > wagon.yaml
```

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:

```
kubectl create -f /var/lib/gravity/site/packages/unpacked/gravitational.io/
  AnacondaEnterprise/5.X.X/resources/wagon.yaml -n kube-system
```

**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:

```
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**.
5.3.13 Kernel parameters may be overwritten and cause networking errors

If networking starts to fail in Anaconda Enterprise, it may be because a kernel parameter related to networking was inadvertently overwritten.

**Workaround**

On the master node running AE, 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.

5.3.14 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

5.3.15 AE auth pod throws OutOfMemory Error

If you see an exception similar to the following, Anaconda Enterprise 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]
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/
     -configuration/
      ```

(continues on next page)
Affected versions

5.2.1

5.3.16 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

5.2.2

5.3.17 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
5.3.18 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

5.3.19 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

5.3.20 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

5.3.21 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
5.3.22 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

5.3.23 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

5.3.24 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 Anaconda Enterprise 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

5.3.25 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
5.3.26 Disk pressure errors on AWS

If your Anaconda Enterprise 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:
   ```bash
   kubectl get node
   ```
2. Identify which node is experiencing issues, then run the following command against it, to view the log for that node:
   ```bash
   kubectl describe node <master-node-name>
   ```

If there is disk pressure, the log will display an error message similar to the following:

![Error Message Example]

**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 AE](#) 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 AE 5 master.
3. To find your new disk’s name run `fdisk -l`. 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 -l` 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/nvme1n1p1 /opt/aetmp`

9. Shut down the Kubernetes system. 

Find the gravity services: `systemctl list-units | grep gravity`

You will see output like this:

```
# systemctl list-units | grep gravity
gravity__gravitational.io__planet-master__0.1.87-1714.service          loaded
  → active running
    Auto-generated service for the gravitational.io/planet-master:0.1.87-1714
  → package
gravity__gravitational.io__teleport__2.3.5.service                loaded
  → active running
    Auto-generated service for the gravitational.io/teleport:2.3.5 package
```

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 0 0
```

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:

```
systemctl start gravity__gravitational.io__planet-master__0.1.87-1714.service
systemctl start gravity__gravitational.io__teleport__2.3.5.service
```

### 5.3.27 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 AE* for more information.

### 5.3.28 General diagnostic and troubleshooting steps

**Entering Anaconda Enterprise environment**
To enter the Anaconda Enterprise environment and gain access to `kubectl` and other commands within Anaconda Enterprise, 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 Anaconda Enterprise environment. If so, there are two *shared mounts* to pass data back and forth between the two environments:

- **host**: `/opt/anaconda/` -> **AE environment**: `/opt/anaconda/`
- **host**: `/var/lib/gravity/planet/share` -> **AE 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 Anaconda Enterprise 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
```

### 5.3. Known issues
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`.

5.4 Frequently asked questions

5.4.1 General

When was the general availability release of Anaconda Enterprise v5?

Our GA release was August 31, 2017 (version 5.0.3). Our most recent version was released April 14, 2021 (version 5.5.0).

Which notebooks or editors does Anaconda Enterprise support?

Anaconda Enterprise supports the use of Jupyter Notebooks and JupyterLab, which are the most popular integrated data science environments for working with Python and R notebooks. In version 5.2.2 we added support for Apache Zeppelin, 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.

Can I deploy multiple data science applications to Anaconda Enterprise?

Yes, you can deploy multiple data science applications and languages across an Anaconda Enterprise 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. Anaconda Enterprise handles all of the resource allocation and application scheduling for you.

Does Anaconda Enterprise support high availability deployments?

Partially. Some of the Anaconda Enterprise services and user-deployed apps will be automatically configured when installed to three or more nodes. Anaconda Enterprise 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 Anaconda Enterprise.

Which identity management and authentication protocols does Anaconda Enterprise support?

Anaconda Enterprise comes with out-of-the-box support for the following:

- LDAP / AD
- SAML
- Kerberos

For more information, see Connecting to external identity providers.

Does Anaconda Enterprise support two-factor authentication (including one-time passwords)?
Yes, Anaconda Enterprise 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 Anaconda Enterprise by navigating to the authentication center and clicking on Authentication and then OTP Policy.

### 5.4.2 System requirements

**What operating systems are supported for Anaconda Enterprise?**

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 Anaconda Enterprise nodes?**

Please see [system requirements](#).

**Which browsers are supported for Anaconda Enterprise?**

Please see [browser requirements](#).

**Does Anaconda Enterprise come with a version control system?**

Yes, Anaconda Enterprise includes an internal Git server, which allows users to save and commit versions of their projects.

**Can Anaconda Enterprise integrate with my own Git server?**

Yes, as described in [Connecting to an external version control repository](#).

### 5.4.3 Installation

**How do I install Anaconda Enterprise?**

The Anaconda Enterprise installer is a single tarball that includes Docker, Kubernetes, system dependencies, and all of the components and images necessary to run Anaconda Enterprise. The system administrator runs one command on each node.

**Can Anaconda Enterprise be installed on-premises?**

Yes, including airgapped environments.

**Can Anaconda Enterprise be installed on cloud environments?**

Yes, including Amazon AWS, Microsoft Azure, and Google Cloud Platform.

**Does Anaconda Enterprise support air gapped (off-line) environments?**

Yes, the Anaconda Enterprise installer includes Docker, Kubernetes, system dependencies, and all of the components and images necessary to run Anaconda Enterprise 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 Anaconda Enterprise?**

No. The installation of Anaconda Enterprise is supported only by using the single-file installer. The Anaconda Enterprise installer includes Docker, Kubernetes, system dependencies, and all of the components and images necessary for Anaconda Enterprise.

**Can I install Anaconda Enterprise on my own instance of Kubernetes?**

5.4. Frequently asked questions
Yes, please refer to our Kubernetes installation guide.

Can I get the AE installer packaged as a virtual machine (VM), Amazon Machine Image (AMI) or other installation package?

No. The installation of Anaconda Enterprise is supported only by using the single-file installer.

Which ports are externally accessible from Anaconda Enterprise?

Please see network requirements.

Can I use Anaconda Enterprise to connect to my Hadoop/Spark cluster?

Yes. Anaconda Enterprise supports connectivity from notebooks to local or remote Spark clusters by using the Sparkmagic client and a Livy REST API server. Anaconda Enterprise 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 Anaconda Enterprise. A data scientist can use these Anaconda libraries from a notebook as part of a Spark job.

On how many nodes can I install Anaconda Enterprise?

You can install Anaconda Enterprise 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 Anaconda Enterprise cluster at any time.

One node serves as the master node and writes storage to disk, and the other nodes serve as worker nodes. Anaconda Enterprise 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 Anaconda Enterprise.

   Saving this file as `/var/lib/gravity/planet/share/secrets.yaml` on the Anaconda Enterprise master node makes it accessible as `/ext/share/secrets.yaml` within the Anaconda Enterprise 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 Anaconda Enterprise environment and run these commands:
5.4.4 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 Anaconda Enterprise 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.

Anaconda Enterprise 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, AE 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-gpu)
- **PyTorch** (pytorch)
- **MXNet** (mxnet-gpu)

For boosted decision tree models:

- **XGBoost** (py-xgboost-gpu)

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 Anaconda Enterprise in a cluster configuration. Each installation should have an odd number of master nodes, and we recommend at least one worker node. The master node runs all Anaconda Enterprise 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 Installation requirements for the baseline hardware requirements for Anaconda Enterprise.

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 an Anaconda Enterprise 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 notebook sessions and deployments that are expected. Adding nodes to an Anaconda Enterprise 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, Anaconda Enterprise 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, enterprise 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 Anaconda Enterprise 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, Anaconda Enterprise 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.

5.4.5 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, PyCharm, 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 Anaconda Enterprise when deployed.

**Does Anaconda Enterprise include Docker images for my data science projects?**

Anaconda Enterprise 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 AE5 my projects no longer work**

If you’ve upgraded to AE 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 AE version 5.3.1 where the package list has been removed from the env_specs and the required packages added to the global list.

```yaml
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
- psycopg2
- holoviews

platforms:
- linux-64
- osx-64
- win-64

env_specs:
  anaconda50_py36:
    packages: []
    channels: []
```

---

5.4. Frequently asked questions 283
5.4.6 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 Anaconda Enterprise available?
Anaconda Project is free and open-source. Anaconda Enterprise is a commercial product.

Where can I find example projects for Anaconda Enterprise?
Sample projects are included as part of the Anaconda Enterprise 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 Anaconda Enterprise account or download the sample projects to their local machine.

Does Anaconda Enterprise support batch scoring with REST APIs?
Yes, Anaconda Enterprise 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 Anaconda Enterprise 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. Anaconda Enterprise 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.

5.4.7 Help and training

Do you offer support for Anaconda Enterprise?
Yes, we offer full support with Anaconda Enterprise.

Do you offer training for Anaconda Enterprise?
Yes, we offer product training for collaborative, end-to-end data science workflows with Anaconda Enterprise.

Do you have a question not answered here?
Please contact us for more information.
5.5 Understanding Anaconda Enterprise system requirements

Anaconda Enterprise is a DS/AI/ML development and deployment platform built on top of the industry standard Kubernetes container orchestration system. Anaconda Enterprise leverages Kubernetes to create and manage sessions, deployments, and jobs. During normal operation, users and administrators are insulated from this underlying complexity. Anaconda Enterprise 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 Anaconda Enterprise 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.

The installation process itself actually 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 Anaconda Enterprise container images into the internal Docker registry
- Use a standard Helm process to install the Anaconda Enterprise application
- Run final Anaconda-specific application configuration tasks

Enumerating these steps helps to illustrate just how “thin” the steps specific to Anaconda Enterprise truly are. The bulk of our implementation effort involves the construction of a stable, performant Kubernetes cluster.

5.5.1 Hardware considerations

CPU and Memory: node considerations

In our preferred implementation, the primary node—where the central Kubernetes services and Anaconda Enterprise system containers run—does not host user workloads. Therefore, our standard recommendation of 16 cores and 64 GB 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. Our system requirements of 16 cores and 64GB RAM per worker node ensures that at least a nominal number of user sessions can be run on each node. Nodes with more cores and RAM are marginally better than smaller nodes, if only because it allows the aggregate user workload to be accommodated with less total hardware.

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 Anaconda Enterprise 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 AE5 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 an Anaconda Enterprise 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 Anaconda Enterprise. 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 May 2021, the official Kubernetes documentation about GPU scheduling marks it as an “experimental” feature.

In our experience, customers *can* be successful deploying GPUs in Anaconda Enterprise. Anaconda Enterprise 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 Anaconda Enterprise is a challenge, but one that many customers have nevertheless found worthwhile.

**5.5.2 Operating system**
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 Anaconda Enterprise functionality. Please make sure that additional firewall configurations are disabled or confirmed to be compatible with Anaconda Enterprise. 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 AE 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 afoot of Anaconda Enterprise is when the application requires exceptions to configurations enforced by these tools. It is essential that these 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 Anaconda Enterprise 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 Anaconda Enterprise.
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. AE has its own scripts that can be used to make a backup of the application on a regular basis.

5.5.3 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.

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 AE5 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. For a cloud-based installation, consider a managed NFS offering such as Azure’s NFS Files.

In theory, it is possible to export a directory from the master node via NFS. If an independent file sharing option is available, we recommend 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 AE. 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 we recommend at least studying the cost of a larger disk instead of simply boosting IOPS.

5.5.4 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 Anaconda Enterprise itself.

Private networking

For very understandable reasons, customers usually need to place Anaconda Enterprise 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

Anaconda Enterprise 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.
5.5.5 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 Anaconda Enterprise.
- 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.

5.5.6 BYOK8s (Bring Your Own Kubernetes)

With the release of version 5.5 of Anaconda Enterprise, we introduced our first limited offering that allows our Gravity/Planet Kubernetes stack to be replaced with a customer-supplied Kubernetes offering. We will soon be evolving this document to include recommendations that are specific to that offering.

At a high level, however, 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.
- Anaconda Enterprise 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 AE5 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 AE5 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.

Deprecation of Ops Center

If you have administered a Gravity-based AE5 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.

**Operator**

Currently the Openshift operator is not on OperatorHub. Anaconda will provide the operator which will need to be installed manually per our *Installation* instructions.