## Contents

1 Anaconda Enterprise Innovator Program 3
Anaconda Enterprise is an enterprise-ready, secure and scalable data science platform that empowers teams to govern data science assets, collaborate and deploy data science projects.
The Anaconda Enterprise Innovator program is our early market access program. The next generation Anaconda Enterprise platform is our 5.0 release which is scheduled for June 2017. This program includes two early access stages:

- **5.0 Public Cloud Sandbox** - Mar 2017 through Sep 2017
- **5.0 On Premises Sandbox** - June 2017 through Sep 2017

In the first stage of the Innovator Program, Enterprise includes new capabilities:

- Easily deploy your projects into interactive data applications, live notebooks and machine learning models with APIs.
- Share those applications with colleagues and collaborators.
- Manage your data science assets: notebooks, packages, environments and projects in an integrated data science experience.

The public cloud sandbox provides example projects with notebooks, plots and interactive web apps that allow you to experience the new deployment capabilities. You can also upload your own projects and deploy them in the public sandbox.

NOTE: Do not upload anything private, as all sandbox uploads are public.

To begin working with Enterprise, see *Getting started*.

### 1.1 User guide

To learn about the key concepts underlying Anaconda Enterprise, see *Concepts*.

To quickly become familiar with Enterprise, see *Getting started*.

To learn by doing, see *Tutorials*.

Enterprise brings together different types of users: data scientists, business analysts, viewers and IT administrators.

The user guide tasks are organized based on these roles.
1.1.1 Concepts

Applications

An application (“app”) is a deployed project. You can deploy a project as an interactive visualization, a live notebook or a machine learning model with an API application. You can also deploy a project as multiple apps. For example, one project may contain both a notebook and an interactive data visualization.

You configure how a project is deployed by adding a run command in the configuration file `anaconda-project.yml` and selecting the appropriate deployment command.

Channels

The locations of the repositories where Anaconda Enterprise looks for packages. Channels may point to a Cloud repository or a private location on a remote or local repository that you or your organization created. The `conda channel` command has a default set of channels to search beginning with `https://repo.continuum.io/pkgs/`. You may override the default channels, for example to maintain a private or internal channel. In `conda` commands and in the `.condarc` file, these default channels are referred to by the channel name defaults.

Deployments

Software deployment is all of the activities that make a software application available for use. When you deploy an app, Anaconda Enterprise finds and builds all of the software dependencies—the programs on which the app depends in order to run—and encapsulates them so they are completely self-contained. This allows you to easily share the application with others.

Interactive data applications

Interactive data applications are visualizations with sliders, drop-downs and other widgets that allow users to interact with them. Widgets can drive new computations, update plots and connect to other programmatic functionality.

With Enterprise, you can create and deploy interactive data applications built using popular libraries such as Bokeh and Shiny.

For more information, see the Shiny example `r_shiny_distribution` and the Bokeh example `weather_statistics` in the anaconda-enterprise channel on Anaconda.org.

Live notebooks

Data scientists use live notebooks (or “notebooks”) to present their work and share it with others. With Enterprise, you can immediately deploy your notebooks with just a few clicks, so that other users can run them and get updated results on demand.

Jupyter Notebooks is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses for Jupyter Notebooks include:

- Data cleaning and transformation.
- Numerical simulation.
- Statistical modeling.
For more information, see Jupyter.

Enterprise also supports R and Python notebooks. For more information, see the Python notebook example `stocks_live_notebook` and the R notebook example `stocks_live_notebook` in the anaconda-enterprise channel.

**Packages**

Software files and information about the software, such as its name, the specific version and a description, that are bundled into a file that can be installed and managed by a package manager.

**Projects**

A project is a portable encapsulation of your data science assets that includes all the necessary configuration to automate its setup and deployment: packages, file downloads, environment variables and runnable commands. Data scientists use projects to encapsulate data science projects and make them easily portable. A project is usually compressed into a `.tar.bz2` file for sharing and storage.

Anaconda Enterprise uses the Anaconda Project specification format.

An Anaconda Project is a folder that contains a configuration file named `anaconda-project.yml` together with scripts, notebooks and other files. The configuration file can include commands, variables, services, downloads, packages, channels and environment specifications.

Anaconda Project automates setup steps, so you can deploy projects just by entering `anaconda-project run`.

**REST APIs**

A common way to operationalize machine learning models is through REST APIs. This allows developers to make their applications intelligent without having to write models themselves.

RESTful endpoints are a great way to bridge the gap between the data scientists writing machine learning models and the developers writing end-to-end applications. With Enterprise REST API deployment, applications can request predictions as a service.

For more information, see the `quote_api` example in the anaconda-enterprise channel.

**User scopes**

Anaconda Enterprise brings together many types of users. Some users simply need to view files, so we call these users *viewers*. Those who want to run applications that are already deployed are called *business analysts*. Users who write and deploy applications are called *data scientists*. Finally, those who install, configure and administer Anaconda Enterprise, usually IT administrators, are called *administrators*.

**Web applications**

Python is the leading open source data science language, but it is also widely used for web development. With Python, developers and data scientists can write intelligent web applications using the same language, which improves collaboration. Popular Python web frameworks include Flask and Django.

For more information, see the Flask example `image_classifier_flask` in the anaconda-enterprise channel of Anaconda.org.
Workspace

A workspace is a location where you can view all of your data science assets: notebooks, packages, environments and projects. The workspace contains both your assets and the assets of organizations that you belong to on Anaconda Cloud.

All assets that you upload to Anaconda Cloud with the command line or Anaconda Navigator are visible through the Enterprise workspace.

1.1.2 Getting started

1. Sign up for the Anaconda Enterprise Innovator Program.
   
   After sign up, you receive an email with instructions on how to register and log in.

2. Click the link in your Innovator Program registration email, or in your browser, type the address to Enterprise.

3. On the Enterprise registration page, create your Enterprise account:

   ![Registration Form](image)

4. On the Enterprise Home page, you can view the list of projects available to you:

   Projects that have already been deployed as apps are located in the top navigation bar in Deployments.

5. Familiarize yourself with Enterprise key concepts.

6. To learn by doing, go through the tutorials.
1.1.3 General tasks

All users perform the following tasks:

Creating an Account

1. **Sign up** for the Anaconda Enterprise Innovator Program.
   
   After sign up, you receive an email with instructions on how to register and log in.

2. Click the link in your Innovator Program registration email, or in your browser, type the address to Enterprise.

3. On the Enterprise registration page, create your Enterprise account:
Changing your account settings

1. In the top-right corner of the Anaconda Enterprise screen, click your user name.
2. In the menu that appears, select Settings:

3. On the Settings screen, click the options on the left to access different account settings:
   - Account–Change your email address, first name or last name.
   - Password–Change your password.
   - Authenticator–Enable 2-step verification with FreeOTP or Google Authenticator.
   - Sessions–View a list of when sessions started, last access, IP address, and log out of all sessions.
   - Applications–View a list of running applications, permissions and granted personal info.
   - Log–View a log of all account actions with date, event, IP, client and details.
Logging in

1. In the Anaconda Enterprise login page, enter your registered user name and password.

2. You will see the following Enterprise **Home** page:
Logging out

1. In the top-right corner of the Anaconda Enterprise screen, click your user name.
2. In the menu that appears, select Sign Out.

1.1.4 Managing projects

Creating a project

1. To create a new project, click the Add button in the top left and select New Project.

   ![Image of creating a project](user-guide/projects/../../images/ae50-createproject1.png)

2. In the section Provide a unique name for your project field, type a descriptive name for your project and click Next.
3. Select the project type, whether Python or R language:
• Anaconda 2: Latest Anaconda Distribution with Python=2.7
• Anaconda 3: Latest Anaconda Distribution with Python=3.6
• R: R-essentials

4. Review the project name, type and then create your project.

5. When the project is created it appears in your Projects list. Click View to see its project page.

6. To edit the project click Return to JupyterLab in the top right corner.

Creating a notebook

To create a notebook, follow the steps used to create a project.

Creating a bokeh application

To create a bokeh app, follow the steps used to create a project.
### Importing libraries

You can import library packages to your projects by using Anaconda Project in a terminal window:

```
anaconda-project add-packages package1 package2
```

**NOTE:** Replace package1 and package2 with the names of the packages that you want to include. You can specify as many packages as you want.

**EXAMPLE:** To add the packages Bokeh and pandas:

```
anaconda-project add-packages bokeh=0.12 pandas
```

This will add the packages to the packages section of your `anaconda-project.yml` file. You can check to confirm by opening this file in an editor.

### Editing a project in JupyterLab

JupyterLab is an integrated data science development environment.

Learn more about Jupyter.

To edit a project in JupyterLab, go to a project page view, and click the top right corner button Open in JupyterLab.

Your project opens in the JupyterLab editing environment.
1. You can edit the project by creating a notebook, console, text editor or from a terminal in JupyterLab. You can choose the kernel for notebook and console.

2. You can see the files created, running sessions, commands, launcher, cell tools, open tabs from the left side vertical menu.

Use the Anaconda Project toolbox in the left side panel to manage your environments, packages and deployment commands.

You can also manage environments, packages and deployment commands from the command line using the Terminal widget in JupyterLab.

More information on command line project editing is available in the documentation for Anaconda Project.

**Uploading a project**

After creating and saving a data science project, you can upload the project to Cloud. The project then appears in your Project list, and you can deploy it as a live application.

To upload a data science project to Cloud:

1. In a terminal window, navigate to the directory where you created and defined your data science project:
cd data-science-project

2. If you are not already logged in to Cloud, log in.

3. To upload your project, in a terminal window run the `anaconda-project upload` command:

```
anaconda-project upload
Using Anaconda API: https://api.anaconda.org
Project is at
https://anaconda.org/anaconda-enterprise/project/data-science-project
Upload successful.
```

NOTE: For the Anaconda Enterprise Innovator Program, projects must be uploaded as public packages to be deployed with Enterprise. After the Innovator Program, Enterprise will support direct uploading of projects that can be set as public or private and shared between users.

4. After you have uploaded your project, you can view it in Cloud at:

   https://anaconda.org/YOUR-USERNAME/projects

   NOTE: Replace YOUR-USERNAME with your Cloud user name.

   - In the Cloud interface, you can view project information, revisions, settings and delete your projects.
   - Your project is also listed in the Enterprise interface in the Workspace and Projects views.

**Uploading data science project revisions to Cloud**

If you make changes to the project on your local machine, upload your project revisions in a terminal window using the `anaconda-project upload` command:

```
anaconda-project upload
Using Anaconda API: https://api.anaconda.org
Project is at
```

(continues on next page)
Cloud stores revisions of your project under the same name. You can view the different revisions of your project by viewing it on Cloud.

**Specifying project dependencies**

After you have created a project, you can define a customized list of conda packages and specific versions for your application to use.

To customize your data science project and specify the applications that you want to encapsulate, use the `anaconda-project` command in a terminal window, or use any text editor to edit another `anaconda-project.yml` file.

**EXAMPLE:** To add a Jupyter Notebook to the project that depends on `numpy`, `pandas` and `bokeh`, run the following commands:

```bash
anaconda-project add-command --type notebook default data-science-notebook.ipynb
anaconda-project add-packages numpy pandas bokeh
```

Or, you can edit the `anaconda-project.yml` file directly. Either way, the resulting data science project definition file contains the following information:

```yaml
name: data-science-notebook
commands:
  default:
    notebook: data-science-notebook.ipynb
packages:
  - python=3.5
  - jupyter
  - numpy
  - pandas=0.19.2
  - bokeh=0.12.4
env_specs:
  default:
    channels: []
    packages: []
```

**EXAMPLE2:** You can generate similar data science project definition files for projects that include scripts/models with REST APIs, as shown here:

```yaml
name: quote_api
description: A simple script with an API.
commands:
  default:
    unix: python $(PROJECT_DIR)/quote.py
    windows: python %PROJECT_DIR%\quote.py
    supports_http_options: true
packages:
```

(continues on next page)
EXAMPLE 3: You can also generate data science project definition files for an interactive Bokeh application, as shown here:

```yaml
name: stocks_app
description: An example Bokeh application.
commands:
  default:
    bokeh_app: .
downloads:
  QUANTQUOTE: http://quantquote.com/files/quantquote_daily_sp500_83986.zip
packages:
  - bokeh=0.12.4
  - pandas=0.19.2
env_specs:
  default:
    channels: []
    packages: []
```

After you have saved your `anaconda-project.yml` file, next see *Uploading a project*.

### Sharing a Project

In Anaconda Enterprise, you can select users, groups or roles to share an application with. You can also view or remove access from the users, roles and groups that you previously shared an application with.

To share an app or remove access to it:

1. Run the app that you want to share.
2. From the top navigation bar, click **Deployments**.
3. Select the project you want to share.
4. In the left navigation bar, click **Share**.
5. In the Share App pane, search for and select the users, groups, or roles that you want to share your application with.
6. Click the Apply button.

The selected users can see your deployed project when they log in to Enterprise.

To remove access, under Shared with, click the Delete icon next to the user, group, or role you want to remove, and click the Apply button.

**Save project version**

[TODO]

**Tag a deployable project revision**

Before you start, your project must have a deployment command.

For self-service notebooks, the command section in your anaconda-project.yml will look something like this:

```yaml
commands:
  default:
    notebook: pandas.ipynb
```

In the above example, the deployable command is the name of the notebook to run.

After you have completed your data science work and are ready to save a deployable revision, click the Save icon in the top right corner:

NOTE: When you save your file, it will overwrite the copy on the server, including any changes made by your collaborators.

For example, suppose there are two collaborators Julia and Sarah, and three files `main.py`, `utils.py`, `setup.py`. They have both opened the project in JupyterLab and are editing. Julia makes changes to `main.py` and `utils.py` and saves. Sarah makes changes to `utils.py` and `setup.py` and saves after Julia. Then `main.py` will be Julia’s version, and `utils.py` and `setup.py` will be Sarah’s version.

Furthermore, Julia and Sarah will keep only seeing the state of the files they have on their own machines, which will not match the files on the server with their collaborator’s changes. To see the new state, they must close JupyterLab
by pressing the stop button and then open it again. When they do, Sarah will see Julia’s updates to `main.py`. Julia will see Sarah’s updates to `utils.py` and `setup.py`, and will see that Julia’s changes to `utils.py` have been overwritten.

Watch for more Anaconda Enterprise features coming soon to provide more information about file conflicts and more options to merge edits from multiple collaborators.

After saving your file, click Next.

If you want this saved version to be deployable, check the box.

Click Save.

When you deploy that project you will see the available commands and the saved revision in the dropdown menus:

Click Deploy.

Your deployed self-service notebook will soon be up and running in your Deployments tab.
1.1. User guide
1.1.5 Using channels and packages

Use the Anaconda command line interface (CLI) to create and share channels. First you create a channel, then you can upload a package to the channel.

You should be comfortable using a CLI to use this section.

Creating a channel

Use the Anaconda command line interface (CLI) to create and share channels.

1. Open a terminal window, navigate to your Anaconda directory, then run:

   ```bash
   anaconda-project run cli_dev && source activate envs/cli_dev
   source activate envs/cli_dev
   ```

2. Log into the Anaconda platform with the same username and password you use in the web interface:

   ```bash
   anaconda-platform-cli login
   Username: your-username
   Password: your-password
   ```

3. Create the channel

   In your terminal, run:
   ```bash
   anaconda-platform-cli channels create username/channelname
   ```

   NOTE: The channel name `username/channelname` must not already exist.
   NOTE: The slash character (“/”) is allowed in channel names.
   This feature is similar to labels on Anaconda Cloud.

Manage channels

You can get a list of all the channels on the platform with the `channels list` subcommand:

```bash
anaconda-platform-cli channels list
```
Learn more about channels

Run `anaconda-platform-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-platform-cli channels share --help
```

After you’ve created your channel, next see Uploading A Package.

Uploading A Package

After logging in and creating a channel, you can upload a package to the channel with:

```
anaconda-platform-cli upload path/to/pkgs/notebookname.bz2 --channel username/channelname
```

NOTE: Replace `path/to/pkgs` with the actual path to the package you want to upload, and `username/channelname` with the actual username and channel name.

Setting a default channel

You can set a default channel and skip the `--channel` option in each command:

```
anaconda-platform-cli config set default_channel username/channelname
```

To display the current default channel:

```
$ anaconda-platform-cli config get default_channel
'username/channelname'
```

NOTE: There is no `default_channel` in a fresh install until you set it.

After setting the default channel, you can upload to your default with this command:

```
anaconda-platform-cli upload path/to/pkgs/notebookname.bz2
```

NOTE: Replace `path/to/pkgs` with the actual path to the package you want to upload.

Viewing uploaded packages

To see the packages you have uploaded, log onto the web interface and view the Packages list.

1.1.6 Managing deployments

Deploying a project

When you deploy a data science project, an application (“app”) is created.

1. In the projects list, click the name of the project that you want to deploy.
2. In the left navigation bar, click Deploy.
3. In the Deploy Project pane, enter the name of the application to be created.
4. From the drop-down box, select the command that you want the application to run.
5. Click the Deploy button.

The build may take a few minutes to complete as it obtains and builds all the dependencies needed to run the app. You will see periodic status updates.

After the app is built, it starts running on the deployment server.

NOTE: You can create multiple applications from a single data science project. Each running application can be a different version, and can be shared with a different set of users.

**Deploy a notebook**

To deploy a notebook, select a notebook project from the Projects list and deploy it the same as above. After it finishes the build, in the Deployments list, open your newly deployed notebook app by clicking its name.

**Deploy a Bokeh application**

To deploy a Bokeh app, select a project containing Bokeh from the Projects list and deploy it the same as above. After it finishes the build, in the Deployments list, open your newly deployed Bokeh app by clicking its name.

**Deploy a REST API**

To deploy a REST API app, select a project containing a REST API from the Projects list and deploy it the same as above.
After it finishes the build, in the Deployments list, open your newly deployed REST API app by clicking its name.

**Sharing a deployment**

In Anaconda Enterprise, you can select users, groups or roles with whom you want to share access to a deployed application. You can also view or remove access from the users, roles and groups with whom you previously shared access to an application.

To share a deployed app or remove access to it:

1. From the top navigation bar, click **Deployments**.
2. Select the project you want to share.
3. In the left navigation bar, click **Share**.
4. In the Share App pane, search for and select the users, groups, or roles that you want to share your application with.
5. Click the Apply button.

The selected users can see your deployed project when they log in to Enterprise.

To remove access, under **Shared with**, click the Delete icon next to the user, group, or role you want to remove, and click the Apply button.

**Manage deployment tokens**

Tokens allow you to control access to an app.

To manage deployment tokens for an app that has been created and deployed in Anaconda Enterprise:

1. From the top navigation bar, click **Deployments**.
2. In the Deployments list, click the name of the app you want to manage.
3. On the left side-menu, click **Advanced**.
4. Use the **Generate** button to generate a new token, and copy it to the clipboard with the Copy link or by copying it with mouse or keyboard shortcuts like any other text. Use this token to connect to the deployed app from Notebooks, APIs, and other running code.

**View logs**

Anaconda Enterprise keeps an extensive log of everything that occurs to your deployed applications.

To view a deployed application’s log in Anaconda Enterprise:

1. From the top navigation bar, click **Deployments** to view a list of all deployed apps.

2. In the left column, click Logs, then click the Fetch Logs button.

   If you want a copy of the log, highlight the log text and copy it with normal keyboard or mouse shortcuts such as Control-C or Command-C. You can paste the text of the log into any text editor with normal keyboard or mouse shortcuts such as Control-V or Command-V.

**Terminating an app**

Terminating an app:
Deployment Name

View

Share

Logs

Advanced

Token URL has been copied to your clipboard.

Deployment URL

https://master.enterprise.demo.continuum.io:31543/

Generate Tokens

A token lets you connect to app_name via Notebooks, APIs and other robots. Tokens are powerful and should be protected like passwords.

Manage Tokens

Token

eyJhbGciOSUJw1fNhbi8qO3w6U2hwa2r3SMa8G1J355SD0VX9V56mMr9dH
B2NHY1fS9Y2V5d0fG9S9f1WE9p9P2PzZUfU13S0FtK80y4AfG6G3cM8OQJMQQ
1Yr4Zs9f2w2aXfXs3L2QWty7KtYfHRE99NYyTTM3971M1MkjxmZnOLCJ9EHAO9JAAS
U236ZCANixhWF0jSnND41079JODJ6ZLJ2p2E38OJ0dHwzc0tL2IncF1018bn

Generate

Terminate Deployment

Stop the deployment deployment_name and free all the resources.

Deployed from:

Project Name

Projects

Filter by Name and Owner

<table>
<thead>
<tr>
<th>Name</th>
<th>Owner</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_clustering</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>data_crossfilter</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>gapminder_visualization</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>image_classifier_flask</td>
<td>anaconda-enterprise</td>
<td>25 days ago</td>
</tr>
<tr>
<td>markowitz_notebook</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>quote_api</td>
<td>anaconda-enterprise</td>
<td>a month ago</td>
</tr>
<tr>
<td>r_predictive_notebook</td>
<td>anaconda-enterprise</td>
<td>4 months ago</td>
</tr>
<tr>
<td>r_shiny_distribution</td>
<td>anaconda-enterprise</td>
<td>a month ago</td>
</tr>
<tr>
<td>stocks</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>streaming_ohl_data</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>surface3d_plot</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>weather_statistics</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
</tbody>
</table>
• Stops the app from running.
• Removes the app from the deployment server.
• Makes the app unavailable to you and any users you had shared it with.
• Frees all of its resources.

Terminating an app does not affect the original project that the application was created from.

To terminate an app:

1. On any page where the app is running, from the left navigation bar, click **Terminate**.
2. Confirm that you want to stop the app in the Terminate App pane by typing the name of the app to confirm, or copy and paste the app name shown at the top of the pane.

3. Click the Terminate button.
   
   The application stops running.

### 1.1.7 Tutorials

You can create projects and upload them to Anaconda Enterprise in two ways:

- Use **Anaconda Navigator**, the graphical user interface (GUI) for Anaconda®.
- Use **Anaconda Project** at the command line.

These tutorials walk you through these tasks and deploying both types of projects.

#### Using Anaconda Navigator to create a data science project

This tutorial walks you thorough the easiest way of creating and uploading a data science project, which is using Anaconda Enterprise with **Anaconda Navigator**.

Navigator is a desktop graphical user interface included in Anaconda that allows you to create and edit projects, launch applications and easily manage conda packages, environments and channels without using command line commands. Navigator is automatically installed when you install Anaconda.
After completing this tutorial, you will be able to:

- Create a data science project with Navigator.
- Upload your project to Anaconda Cloud. Then you can use Anaconda Enterprise to deploy your project.

NOTE: If you prefer to use a command line interface (CLI), please see the tutorial Using Anaconda Project to create a data science project.

**Before you start**

Verify that you have Navigator 1.5 or higher:

- Windows–In the File menu in Navigator, select About.
- macOS–In the Navigator menu in Navigator, select About.

**Creating a data science project with Navigator**

Each Anaconda project can include Jupyter notebooks, web apps, Bokeh apps and any other Python scripts or command-line commands.

We will create a very simple project that runs only one Python script, the classic “Hello, World!” script that simply prints Hello, World! and then ends.

1. Open a new text file.
2. In the first line, enter or paste:

```python
print("Hello, World!")
```

3. Save the new file in your home directory with the name `hello.py`.
4. Open Navigator and click the Projects (beta) tab:

5. To create a new project, click the Create button.
6. In the dialog box that appears, enter the project name `hello`, and then click the Create button:

The new project now appears on the Files tab:

7. Navigate to your home directory where you saved the file `hello.py`.
8. Right-click the file, and then select Add to project:
The file `hello.py` appears in the project directory:

9. To edit the blank project file to include the name of your new file, click the **Edit** tab, and then find this line:

   ```
   commands: {}
   ```

10. Edit or paste the text so that line now reads:

    ```
    commands: {'hello': {'windows': 'python hello.py', 'env_spec': 'default'}}
    ```

11. Check your work, and then click the Save button:
1.1. User guide
Uploading a project to Cloud

To upload your new project, click the Upload to Anaconda Cloud button:

Your data science project is uploaded to your Cloud account so that you can share it with colleagues. You can also see the uploaded project in Enterprise.

Using Anaconda Project to create a data science project

This tutorial walks you through creating and uploading a data science project using Anaconda Enterprise with the Anaconda Project command line interface (CLI).

After completing this tutorial, you will be able to:

- Create and initialize a data science project
- Customize a data science project
- Run a data science project on your local machine
- Log in to Anaconda Cloud
- Upload a data science project to Cloud
- Upload data science project revisions to Cloud
Before you start

Check that you have the right version of Anaconda by running the command `conda list anaconda`.

If the output shows a version older than 4.3.1, update Anaconda by running `conda update anaconda`.

Creating and initializing a data science project

This procedure creates a data science project definition file named `anaconda-project.yml`. This file specifies the scripts, notebooks, project dependencies and other project-specific information.

The project directory also has a `.projectignore` file that contains an optional list of files to ignore when the project is uploaded.

To create a data science project:

1. In a terminal window, create a new directory for all of your data science project files:

   ```bash
   mkdir data-science-project
   ```

2. Navigate to your newly created directory:

   ```bash
   cd data-science-project
   ```

3. From within this directory, initialize your data science project by running:
Customizing a data science project

By default, a data science project specifies anaconda in its runtime environment, but you can also define a customized list of conda packages and specific versions for your application to use.

To customize your data science project and specify the applications that you want to encapsulate, use the anaconda-project command or use any text editor to edit another anaconda-project.yml file.

EXAMPLE: To add a Jupyter Notebook to the project that depends on numpy, pandas and bokeh, run the following commands:

```
anaconda-project add-command --type notebook default data-science-notebook.ipynb
anaconda-project add-packages numpy pandas bokeh
```

The resulting data science project definition file, anaconda-project.yml, contains the following information:

```
name: data-science-notebook

commands:
  default:
    notebook: data-science-notebook.ipynb

packages:
  - python=3.5
  - jupyter
  - numpy
  - pandas=0.19.2
  - bokeh=0.12.4

env_specs:
  default:
    channels: []
    packages: []
```

You can generate similar data science project definition files for projects that include scripts/models with REST APIs, as shown here:

```
name: quote_api
description: A simple script with an API.

commands:
  default:
    unix: python ${PROJECT_DIR}/quote.py
    windows: python %PROJECT_DIR%\quote.py
    supports_http_options: true

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

env_specs:
```

(continues on next page)
You can also generate data science project definition files for an interactive Bokeh application, as shown here:

```
name: stocks_app
description: An example Bokeh application.

commands:
  default:
    bokeh_app: .

downloads:
  QUANTQUOTE: http://quantquote.com/files/quantquote_daily_sp500_83986.zip

packages:
  - bokeh=0.12.4
  - pandas=0.19.2

env_specs:
  default:
    channels: []
    packages: []
```

### Running a data science project on your local machine

To make sure your project is defined correctly, test and run your application locally before uploading it to Cloud.

To test the project on your local machine, run:

```
anaconda-project run
```

### Logging in to Anaconda Cloud

To log into Cloud:

1. Register for a free account if you do not already have one.
2. Enter the command `anaconda login`.
3. When prompted, enter your user name and password.

Your terminal session will look like this:

```
anaconda login
Using Anaconda API: https://api.anaconda.org
Username: anaconda-enterprise
anaconda-enterprise\'s Password:
login successful
```

NOTE: Replace `anaconda-enterprise` with your user name.
Uploading a data science project to Cloud

After creating a data science project and defining its dependencies, you can upload the project to Cloud. The project then appears in your Enterprise Workspace and Project views, and you can deploy it as a live application.

To upload a data science project to Cloud:

1. In a terminal window, Navigate to the directory where you created and defined your data science project:
   
   ```
   cd data-science-project
   ```

2. If you are not already logged in to Cloud, log in.

3. To upload your project, run the `anaconda-project upload` command:
   
   ```
   anaconda-project upload
   Using Anaconda API: https://api.anaconda.org
   Project is at
   https://anaconda.org/anaconda-enterprise/project/data-science-project
   Upload successful.
   ```

   NOTE: For the Anaconda Enterprise Innovator Program, projects must be uploaded as public packages to be deployed with Enterprise. After the Innovator Program, Enterprise will support direct uploading of projects that can be set as public or private and shared between users.

4. After you have uploaded your project, you can view it in Cloud at:
   
   ```
   https://anaconda.org/YOUR-USERNAME/projects
   ```

   NOTE: Replace `YOUR-USERNAME` with your Cloud user name.
   
   • In the Cloud interface, you can view project information, revisions, settings and delete your projects.
   • Your project is also listed in the Enterprise interface in the Workspace and Projects views.

Uploading data science project revisions to Cloud

If you make changes to the project on your local machine, upload your project revisions using the `anaconda-project upload` command:

```
anaconda-project upload
Using Anaconda API: https://api.anaconda.org
Project is at
https://anaconda.org/anaconda-enterprise/project/data-science-project
Upload successful.
```

Cloud stores revisions of your project under the same name. You can view the different revisions of your project by viewing it on Cloud.

Deploying a Jupyter Notebook project

This tutorial walks you through deploying a data science project and using the resulting app. The process is the same whether you have created the project using Anaconda Navigator, or with a command line interface, then uploaded your project to Anaconda Cloud.

You should have completed the above steps before you begin this tutorial, or you can use a sample project that has already been uploaded to Anaconda Cloud.
This tutorial uses the sample project, Markowitz Notebook—an investment portfolio analysis that demonstrates a Jupyter Notebook running both R language and Python. The notebook also contains both pandas and Bokeh packages. It outputs four different interactive graphs. A step-by-step description is available inside the notebook.

To deploy the project:

1. In Anaconda Enterprise, click the Name column heading to sort the list alphabetically.
2. Click the project name, “markowitz_notebook” to open it.
3. In the left navigation bar, click Deploy.
4. In the Deploy Project pane, review the default deployment options. If desired, you can enter a new name for your app and provide other details:

5. Click the Deploy button.

   NOTE: It may take a few minutes for the project to deploy, as it is building a new app with all of the called packages and dependencies. A screen similar to the following is displayed as the app is building:

TIP: To watch what the program is doing in the background as it builds the app, go to Deployments tab, click the app name, click the the Log tab, then click the Fetch Logs button.

To use the Markowitz Notebook:

1. In the Jupyter Notebook top navigation bar, in the Cell menu, select Run All to run all of the included code blocks and generate the output:
Deploy project

Deploy Project: markowitz_notebook

App Name
markowitz_notebook

Revision
r1 (uploaded 2 months ago)

Command
default

You can edit who this deployment is shared with later.

STARTING 

Mark analysis of functional genomics data using Pandas and Bokeh

This notebook utilizes the opensources R package for analyzing data that is obtained in R’s binary data format, so we’re using R itself and the nycg package to import it into Python. If you don’t have either of those packages, just make sure that use_R is False; the notebook will load a pre-converted version of the data instead.

In [1]: use_R = False
save_Data = False
2. Scroll down to see the generated plots and graphs:

3. Interact with the app to change the data being sampled. Interactive selections in this notebook include pan, box zoom, resize and reset. The plots and graphs immediately adjust as you interact with them.

Deploying a data science project as an interactive application

This tutorial walks you through deploying a data science project as an interactive application and sharing the app with others.

Before you begin, you should have already created a project either by using Anaconda Navigator, or with the command line interface (CLI), and uploaded it to Anaconda Cloud using the previous tutorials.

1. In Anaconda Enterprise, click the Projects tab.
2. In the project list, click the example project “weather_statistics” to open it:

3. In the left navigation bar, click Deploy.
4. In the Deploy Project pane, review the default deployment options:
Projects

<table>
<thead>
<tr>
<th>Name</th>
<th>Owner</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>weather_statistics</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>surface3d_plot</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>streaming_ohlc_data</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>stocks</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>r_shiny_distribution</td>
<td>anaconda-enterprise</td>
<td>1 month ago</td>
</tr>
<tr>
<td>r_predictive_notebook</td>
<td>anaconda-enterprise</td>
<td>4 months ago</td>
</tr>
<tr>
<td>quote_api</td>
<td>anaconda-enterprise</td>
<td>1 month ago</td>
</tr>
<tr>
<td>markowitz_notebook</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>image_classifier_flask</td>
<td>anaconda-enterprise</td>
<td>25 days ago</td>
</tr>
<tr>
<td>gapminder_visualization</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>data_crossfilter</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
<tr>
<td>data_clustering</td>
<td>anaconda-enterprise</td>
<td>2 months ago</td>
</tr>
</tbody>
</table>

Weather Statistics

Deploy project

Deploy Project: weather_statistics

App Name
weather_statistics

Revision
r1 (uploaded 2 months ago)

Command
default

You can edit who this deployment is shared with later.

Deploy
5. At the bottom of the pane, click the Deploy button.

Your project is deployed.

To share the app:

1. In the top navigation bar, click **Deployments**.

2. Click the Deployment Name column heading to sort the list alphabetically.

3. Click the app name “weather_statistics,” which is the app that you just built when you deployed the weather_statistics project:

   The app starts running.

4. In the left navigation bar, click **Share**.

5. In the Share App pane, search for and select the users, groups, or roles that you want to share your application with:

6. Click the Apply button.

   The selected users can see your deployed project when they log in to Enterprise.
1.1.8 Cheat sheets - coming soon

Coming soon!

1.1.9 Troubleshooting - coming soon

Coming soon!

1.2 Admin guide

This is the admin guide for IT administrators of Anaconda Enterprise.

1.2.1 Concepts

Coming soon!

1.2.2 Installation

Installation Requirements

System Requirements

Anaconda Enterprise can be installed on one to four nodes during the initial installation. After the initial installation, you can add or remove nodes from the Anaconda Enterprise cluster any time.

- **Master node:**
  - 16 CPUs
  - 32 GB RAM
  - 500 GB disk (includes projects and repository)

- **Worker nodes:**
  - 4 CPUs
  - 16 GB RAM
- 100 GB disk

A rule of thumb for each project session or deployment is 1 CPU, 1 GB of RAM and 5 GB of disk space.

**Operating system requirements**

**Linux versions:**
- RHEL/CentOS 7.2+
- Debian 8+
- Ubuntu 16.04+

**Browser requirements**

- Edge 14+
- Internet Explorer 11+
- Chrome 39+
- Firefox 49+
- Safari 10+

**Network Requirements**

The following network ports for Anaconda Enterprise are externally accessible:

- 30071 - Documentation service
- 30080 - Authentication service
- 30081 - Deployment service
- 30082 - Authentication API service
- 30085 - Git storage service
- 30087 - Object storage service
- 30088 - Git Proxy service
- 30089 - Repository service
- 30090 - UI service
- 30091 - Authentication escrow service
- 30095 - Storage service
- 30095 - Spaces service
- 32009 - Operations center
- 30000-32767 - User-deployed applications
Security Requirements

For CentOS and RHEL:

- Disable SELinux in all nodes.
- Before installing in each node, flush `iptables` by running `iptables -F`.

Prerequisites

Configure DNS Records

Configure A records in DNS for the master node.

`anaconda.example.com`

`*.anaconda.example.com` (wildcard DNS support coming soon!)

NOTE: Replace `anaconda.example.com` with the actual domain name you will use for your Anaconda Enterprise installation.

Docker

The machines on which you are installing must not already have Docker installed. Remove Docker or use a clean machine.

Installing on Amazon EC2

The one thing to note is that etcd is very sensitive to disk latency.

Use a configuration where etcd is located into its own EBS volume. This terraform configuration file has the recommended drive configuration: https://gist.github.com/danielfrg/8db7f72d79694da60edb4f5b8d17e1e4#file-gravity-tf-L35

To use that terraform configuration file you need to change the keyname and the AMI to the target you want to test.

- CentOS 7.2 (ami-692df57f)
- RHEL 7.2 (ami-9e2f0988)

If you don’t want to use the terraform configuration file, we recommend that you install Anaconda Enterprise on machines (master AND worker) with IOPS of a minimum of 3000 IOPS on Amazon EC2.

NOTE: Fewer IOPS than 3000 will fail.

For Instance Type we recommend the the `c4.2xlarge` instance types or larger.

Installation Instructions

There are two methods to install Anaconda Enterprise: installation with a browser (must be on the same network as the target machines), or unattended command-line interface (CLI) installation.

For either method you can create any number of nodes from one to four nodes, and you can add or remove nodes at any time after installation.
Option 1. Browser Installation

On the master node, create a new directory for the installer, navigate to the new directory, then download and decompress the installer:

```bash
mkdir anaconda-enterprise
cd anaconda-enterprise
curl -O <location of installer>.tar
tar xvf <installer_name>.tar
```

NOTE: The tarball file is 10GB, it may take a while to download.

On the master node, run the installer as SUDO:

```bash
sudo ./install
```

NOTE: All commands are run as SUDO.

Select the network interface to bind to the installer UI and confirm:

```bash
sudo ./install
* [0/100] starting installer
You have following interfaces:

[interfaces]
------------
1. 172.31.27.121

    Select an interface for installer to listen on.
*IMPORTANT*: target servers must be able to connect to this IP

select interface number: [1-1]:
1
confirm the config:

* IP address: 172.31.27.121
confirm (yes/no):
yes

* [0/100] preparing for installation... please wait
* [0/100] application: AnacondaEnterprise:5.0.1-343-g9c60a89
* [0/100] starting web UI install wizard

---------------------

OPEN THIS IN BROWSER: https://172.31.27.121:61009/web/installer/new/gravitational.io/AnacondaEnterprise/5.0.1-343-g9c60a89?install_token=7b0415d5cc848f1d315efe2b92b099cf

---------------------

Connect to the URL in your browser, ensuring that you are connecting to the public network interface:

`https://54.86.16.230:61009/web/installer/new/gravitational.io/AnacondaEnterprise/5.0.1-343-g9c60a89?install_token=7b0415d5cc848f1d315efe2b92b099cf`

The installer will install a self-signed TLS/SSL certificate, so you can proceed beyond the initial security warning:
If you agree to the License Agreement terms, click the Accept button.

Enter the Cluster Name. The Bare Metal option is already selected. Click Next.
Select the number of nodes between one and four that you want to install to, where one node is the master node and the remaining nodes are worker nodes.

Run the following command on each node on which you want to install Anaconda Enterprise, copying the appropriate master node or worker node command:

```
curl -s --tlsv1.2 -k "https://172.31.27.121:61009/t/01eedae7b9965c958336b5d31f334c3/worker" | sudo bash
```

As you run the command on each node, you’ll see each node listed in the installer.
Click on Start Installation.

The installer will proceed through the installation stages. This process requires about 15 minutes.

After the installation is complete, you will see the following screen.

Then, click on the Continue button.

On the next screen, create the initial admin account credentials that you’ll use to login to the Anaconda Enterprise Operations Center.

Finally, click on the Finish button.

At this point, you are logged into the Anaconda Enterprise Operations Center.
To access the Anaconda Enterprise Operations Center in the future, enter the following URL in your browser.

'https://anaconda.example.com:32009'

NOTE: Replace anaconda.example.com with your actual domain name.

To continue after browser installation, go to the Configuration page.

Option 2. Unattended Command-line (CLI) Installation

Unattended CLI installation is used if you cannot connect to the nodes with a browser. This includes installation from a separate network.

On each node in the cluster, create a new directory for the installer, navigate to the new directory, then download and decompress the installer:

```
mkdir anaconda-enterprise
cd anaconda-enterprise
curl -O <location of installer>.tar
tar xvf <installer_name>.tar
```

On the master node, run the following command, referring to details below this code block:

```
sudo ./gravity install --advertise-addr=192.168.1.1 --token=anaconda-enterprise --flavor=small
  [0/100] starting installer
  [0/100] preparing for installation... please wait
  [0/100] application: AnacondaEnterprise:5.0.1-1557-gbe6f22b
```

(continues on next page)
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

NOTE: For the above “Advertised Address” option advertise-addr, replace the IP address with the address you want to be visible to the other nodes.

For the Flavor option flavor, choose one of:

- ‘tiny’: One node (one master node - DEFAULT)
- ‘small’: Two nodes (one master node and one worker node)
- ‘medium’: Three nodes (one master node and two worker nodes)
- ‘large’: Four nodes (one master node and three worker nodes)

On each worker node, run the following command:

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

* [0/100] joining cluster
* [0/100] connecting to cluster
* [0/100] connected to installer at 172.31.0.236
* [0/100] initializing the operation
* [20/100] configuring packages
* [50/100] installing software

NOTE: For the above “Advertised Address” option advertise-addr, replace the IP address with the address you want to be visible to the other nodes.

The installer will proceed through its stages. This process requires about 20 minutes.

After installing Anaconda Enterprise as part of an unattended installation, create a local user account and password to log into the Anaconda Enterprise Operations Center.

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

```bash
sudo gravity enter
```

Then, run the following command to create a local user account and password for the Anaconda Enterprise Operations Center:

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

NOTE: Replace <email> with the email address, and <yourpass> with the password you want to use.

Now you can access the Anaconda Enterprise Operations Center in your browser by entering the following URL:

```bash
https://anaconda.example.com:32009
```

After logging in, you will be viewing the Anaconda Enterprise Operations Center.

To continue, go to the Configuration page.
### 1.2.3 Configuration

1. **Configure TLS/SSL**
   
   Locate the TLS/SSL certificates and keys and the Root Certificate Authority for the keys. These are usually located in the directory, /etc/pki/CA/certs and /etc/pki/CA/private. Refer to the instructions from your certificate authority.

2. **Configure Root CA**
   
   In order to pull from other sources such as Anaconda Cloud to download packages, you need to add certificates (certs) for those sources to the root certificate authority (rootca) that is used to verify SSL connections.

   To add certificates, concatenate those rootca files into one rootca.pem. The Anaconda.org cert can be downloaded here: https://gist.github.com/danielfrg/1fe49084c885ab7d49203292305a2b03

   **NOTE:** The method to obtain certificates will vary depending on which package source you use. Refer to the issuer’s instructions.

3. **Generate a Java Keystore (JKS):**

   ```bash
   $ openssl pkcs12 -export -in cert.pem -inkey privkey.pem -out pkcs.p12 -name auth
   Enter Export Password: anaconda
   Verifying - Enter Export Password: anaconda
   $ keytool -importkeystore -deststorepass anaconda -destkeypass anaconda -destkeystore keystore.jks -srckeystore pkcs.p12 -srcstoretype PKCS12 -srcstorepass anaconda -alias auth
   ```

4. **Change secret yaml file**

   Delete the `certs` secret. Paste the TLS/SSL certs base64 encoded into the `certs` secret yaml file. Apply those new values by recreating the `certs` secret:

   ```bash
   base64 -i --wrap=0 cert.pem
   base64 -i --wrap=0 keystore.jks
   ```

   (continues on next page)
base64 -i --wrap=0 privkey.pem
base64 -i --wrap=0 rootca.pem  # This is the big concatenated rootca that ...

sudo gravity enter
# Change secrets.yaml file
kubectl delete secrets certs
kubectl apply -f secrets.yaml

In most cases, you will be using rootca.pem, cert.pem and privkey.pem, along with the keystore.

Once the encoded keys are provided in the secrets file, you can reference these TLS/SSL certificates in the
platform configuration settings.

5. Configure FQDN

To access Anaconda Enterprise, you’ll need to edit the platform configuration settings to configure the fully
qualified domain name (FQDN).

You can access the platform configuration in the Anaconda Enterprise Operations Center by visiting this URL
in your browser: https://anaconda.example.com:32009

NOTE: Always replace anaconda.example.com with the domain name you are using.

Or, you can edit the platform configuration file via CLI by running these commands on any Anaconda Enterprise
node:

```
sudo gravity enter
kubectl edit cm
```

In the configuration file, find and replace all occurrences of anaconda.example.com with the FQDN of
the master node.

Finally, delete all ap-* pods to trigger container restarts with the new configuration:

```
sudo gravity enter
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```
6. Authentication Redirects and Allowed URLs

Now edit the redirect URLs in Anaconda Enterprise.

Access the authentication server on the Anaconda Enterprise cluster by visiting this URL in your browser: https://anaconda.example.com:30080

Access the Anaconda Enterprise Operations Center and login with the default username and password: admin/admin. You’ll be asked to change the default password when you log in.

Check to be sure that you are on the AnacondaPlatform realm, then in the left menu, click on the Clients configuration tab.

Replace all URLs in the following client settings with the FQDN of the Anaconda Enterprise server:

- **anaconda-deploy-proxy** - Valid Redirect URIs: https://anaconda.example.com:*
- **anaconda-platform** - Valid Redirect URIs: https://anaconda.example.com:30090/*
- **anaconda-platform** - Base URL: https://anaconda.example.com:30090
- **anaconda-platform** - Web Origins: https://anaconda.example.com:30090
- **anaconda-spaces** - Valid Redirect URIs: https://anaconda.example.com:*

You can also add/remove users from the Anaconda Enterprise authentication server. The authentication server admin user exists under the master realm, and all other Anaconda Enterprise users exist under the AnacondaPlatform realm.

7. Access Anaconda Enterprise in your browser

Access Anaconda Enterprise by visiting this URL in your browser: https://anaconda.example.com:30090

You can test your install by:

- Creating a new project and clicking “Open in JupyterLab”
- Deploying a project
- Generating a token from a deployment

Additional Resources

Telekube documentation: http://gravitational.com/docs/overview/
Troubleshooting

Entering Telekube environment

To enter the Telekube environment and gain access to kubectl and other commands within Anaconda Enterprise, use the command:

```
sudo gravity enter
```

Debugging

AWS Traffic needs to handle the public IPs/port. You should either use a canonical security group with the proper ports opened or manually add the specific ports listed above in Network Requirements.

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.

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.

Upgrading Anaconda Enterprise

Upgrade Anaconda Enterprise by downloading a new installer file and running this command in the installer:

```
sudo ./upgrade
```

Uninstalling Anaconda Enterprise

If you have errors during the installation, you can uninstall Telekube and try again by running these commands on each cluster node:

```
gravity leave
gravity system uninstall
```

1.2.4 User Management

Anaconda Enterprise uses Keycloak for Identity and Access Management.
Server Administration Guide

a. Overview
   i. Features
   ii. How Does Security Work?
   iii. Core Concepts and Terms

b. Server Initialization

c. Admin Console
   i. The Master Realm
   ii. Creating a New Realm
   iii. Realm SSL Mode
   iv. Clearing Server Caches
   v. Email Settings
   vi. Themes and Internationalization

d. User Management
   i. Viewing Users
   ii. Creating New Users
   iii. User Attributes
   iv. Credentials
   v. Required Actions
   vi. Impersonation
   vii. User Registration
      A. Recaptcha Support

e. Login Page Settings
   i. Forgot Password
   ii. Remember Me

f. Authentication
   i. Password Policies
   ii. OTP Policies
   iii. Authentication Flows
   iv. Kerberos
   v. X509 Client Certificate Authentication

g. SSO Protocols
   i. OpenID Connect
   ii. SAML
   iii. OIDC vs. SAML
   iv. Docker
h. Managing Clients
   i. OIDC Clients
      A. Confidential Client Credentials
      B. Service Accounts
   ii. SAML Clients
      A. IDP Initiated Login
      B. SAML Entity Descriptors
   iii. Client Links
   iv. Token and Assertion Mappings
   v. Generating Client Adapter Config
   vi. Client Templates
i. Roles
   i. Realm Roles
   ii. Client Roles
   iii. Composite Roles
   iv. User Role Mappings
      A. Default Roles
   v. Client Scope
j. Groups
   i. Groups Vs. Roles
   ii. Default Groups
k. Admin Console Access Control and Permissions
   i. Master Realm
   ii. Dedicated Realm Admin Consoles
   iii. Fine Grain Admin Permissions
l. Realm Keys
m. Identity Brokering
   i. Brokering Overview
   ii. Default Provider
   iii. General Configuration
   iv. Social Login
      A. Google
      B. Facebook
      C. Twitter
      D. Github
      E. Linked-In
F. Microsoft
G. Stack Overflow
H. Openshift
v. OIDC Providers
vi. SAML Providers
vii. Client Suggested Identity Provider
viii. Mapping Claims and Assertions
ix. Available User Session Data
x. First Login Flow
xi. Retrieving External IDP Tokens

n. User Session Management
   i. Administering Sessions
   ii. Revocation Policies
   iii. Session and Token Timeouts
   iv. Offline Access

o. User Storage Federation
   i. LDAP/AD Integration
   ii. SSSD and FreeIPA/IdM Integration
   iii. Custom Providers

p. Auditing and Events
   i. Login Events
   ii. Admin Events

q. Export and Import

r. User Account Service

s. Threat Model Mitigation
   i. Password Guess, Brute Force Attacks
   ii. Clickjacking
   iii. SSL/HTTPS Requirement
   iv. CSRF
   v. Unspecific Redirect URIs
   vi. Compromised Access and Refresh tokens
   vii. Compromised Access Codes
   viii. Open Redirectors
   ix. Password database compromised
   x. Limiting Scope
   xi. SQL Injection Attacks

1.2. Admin guide
Overview

Keycloak is a single sign on solution for web apps and RESTful web services. The goal of Keycloak is to make security simple so that it is easy for application developers to secure the apps and services they have deployed in their organization. Security features that developers normally have to write for themselves are provided out of the box and are easily tailorable to the individual requirements of your organization. Keycloak provides customizable user interfaces for login, registration, administration, and account management. You can also use Keycloak as an integration platform to hook it into existing LDAP and Active Directory servers. You can also delegate authentication to third party identity providers like Facebook and Google+.

Features

- Single-Sign On and Single-Sign Out for browser applications.
- OpenID Connect support.
- OAuth 2.0 support.
- SAML support.
- Identity Brokering - Authenticate with external OpenID Connect or SAML Identity Providers.
- Social Login - Enable login with Google, GitHub, Facebook, Twitter, and other social networks.
- User Federation - Sync users from LDAP and Active Directory servers.
- Kerberos bridge - Automatically authenticate users that are logged-in to a Kerberos server.
- Admin Console for central management of users, roles, role mappings, clients and configuration.
- Account Management console that allows users to centrally manage their account.
- Theme support - Customize all user facing pages to integrate with your applications and branding.
- Two-factor Authentication - Support for TOTP/HOTP via Google Authenticator or FreeOTP.
- Login flows - optional user self-registration, recover password, verify email, require password update, etc.
- Session management - Admins and users themselves can view and manage user sessions.
- Token mappers - Map user attributes, roles, etc. how you want into tokens and statements.
- Not-before revocation policies per realm, application and user.
- CORS support - Client adapters have built-in support for CORS.
- Service Provider Interfaces (SPI) - A number of SPIs to enable customizing various aspects of the server. Authentication flows, user federation providers, protocol mappers and many more.
- Client adapters for JavaScript applications, WildFly, JBoss EAP, Fuse, Tomcat, Jetty, Spring, etc.
- Client adapters for JavaScript applications, JBoss EAP, Fuse, etc.
- Supports any platform/language that has an OpenID Connect Resource Provider library or SAML 2.0 Service Provider library.
How Does Security Work?

Keycloak is a separate server that you manage on your network. Applications are configured to point to and be secured by this server. Keycloak uses open protocol standards like Open ID Connect or SAML 2.0 to secure your applications. Browser applications redirect a user’s browser from the application to the Keycloak authentication server where they enter their credentials. This is important because users are completely isolated from applications and applications never see a user’s credentials. Applications instead are given an identity token or assertion that is cryptographically signed. These tokens can have identity information like username, address, email, and other profile data. They can also hold permission data so that applications can make authorization decisions. These tokens can also be used to make secure invocations on REST-based services.

Core Concepts and Terms

There are some key concepts and terms you should be aware of before attempting to use Keycloak to secure your web applications and REST services.

users Users are entities that are able to log into your system. They can have attributes associated with themselves like email, username, address, phone number, and birth day. They can be assigned group membership and have specific roles assigned to them.

authentication The process of identifying and validating a user.

authorization The process of granting access to a user.

credentials Credentials are pieces of data that Keycloak uses to verify the identity of a user. Some examples are passwords, one-time-passwords, digital certificates, or even fingerprints.

roles Roles identify a type or category of user. Admin, user, manager, and employee are all typical roles that may exist in an organization. Applications often assign access and permissions to specific roles rather than individual users as dealing with users can be too fine grained and hard to manage.

user role mapping A user role mapping defines a mapping between a role and a user. A user can be associated with zero or more roles. This role mapping information can be encapsulated into tokens and assertions so that applications can decide access permissions on various resources they manage.

composite roles A composite role is a role that can be associated with other roles. For example a superuser composite role could be associated with the sales-admin and order-entry-admin roles. If a user is mapped to the superuser role they also inherit the sales-admin and order-entry-admin roles.

groups Groups manage groups of users. Attributes can be defined for a group. You can map roles to a group as well. Users that become members of a group inherit the attributes and role mappings that group defines.

realms A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.

clients Clients are entities that can request Keycloak to authenticate a user. Most often, clients are applications and services that want to use Keycloak to secure themselves and provide a single sign-on solution. Clients can also be entities that just want to request identity information or an access token so that they can securely invoke other services on the network that are secured by Keycloak.

client adapters Client adapters are plugins that you install into your application environment to be able to communicate and be secured by Keycloak. Keycloak has a number of adapters for different platforms that you can download. There are also third-party adapters you can get for environments that we don’t cover.

consent Consent is when you as an admin want a user to give permission to a client before that client can participate in the authentication process. After a user provides their credentials, Keycloak will pop up a screen identifying the client requesting a login and what identity information is requested of the user. User can decide whether or not to grant the request.
client templates When a client is registered you need to enter configuration information about that client. It is often useful to store a template to make creating new clients easier. Keycloak provides the concept of a client template for this.

client role Clients can define roles that are specific to them. This is basically a role namespace dedicated to the client.

identity token A token that provides identity information about the user. Part of the OpenID Connect specification.

access token A token that can be provided as part of an HTTP request that grants access to the service being invoked on. This is part of the OpenID Connect and OAuth 2.0 specification.

assertion Information about a user. This usually pertains to an XML blob that is included in a SAML authentication response that provided identity metadata about an authenticated user.

service account Each client has a built-in service account which allows it to obtain an access token.

direct grant A way for a client to obtain an access token on behalf of a user via a REST invocation.

protocol mappers For each client you can tailor what claims and assertions are stored in the OIDC token or SAML assertion. You do this per client by creating and configuring protocol mappers.

session When a user logs in, a session is created to manage the login session. A session contains information like when the user logged in and what applications have participated within single-sign on during that session. Both admins and users can view session information.

user federation provider Keycloak can store and manage users. Often, companies already have LDAP or Active Directory services that store user and credential information. You can point Keycloak to validate credentials from those external stores and pull in identity information.

identity provider An identity provider (IDP) is a service that can authenticate a user. Keycloak is an IDP.

identity provider federation Keycloak can be configured to delegate authentication to one or more IDPs. Social login via Facebook or Google+ is an example of identity provider federation. You can also hook Keycloak to delegate authentication to any other OpenID Connect or SAML 2.0 IDP.

identity provider mappers When doing IDP federation you can map incoming tokens and assertions to user and session attributes. This helps you propagate identity information from the external IDP to your client requesting authentication.

required actions Required actions are actions a user must perform during the authentication process. A user will not be able to complete the authentication process until these actions are complete. For example, an admin may schedule users to reset their passwords every month. An update password required action would be set for all these users.

authentication flows Authentication flows are work flows a user must perform when interacting with certain aspects of the system. A login flow can define what credential types are required. A registration flow defines what profile information a user must enter and whether something like reCAPTCHA must be used to filter out bots. Credential reset flow defines what actions a user must do before they can reset their password.

events Events are audit streams that admins can view and hook into.

themes Every screen provided by Keycloak is backed by a theme. Themes define HTML templates and stylesheets which you can override as needed.

Server Initialization

After performing all the installation and configuration tasks defined in Server Installation and Configuration, you
will need to create an initial admin account. Keycloak does not have any configured admin account out of the box. This account will allow you to create an admin that can log into the master realm’s administration console so that you can start creating realms, users and registering applications to be secured by Keycloak.

If your server is accessible from localhost, you can boot it up and create this admin user by going to the http://localhost:8080/auth URL.

Welcome Page.

Simply specify the username and password you want for this initial admin.

If you cannot access the server via a localhost address, or just want to provision Keycloak from the command line you can do this with the ./bin/add-user-keycloak script.

add-user-keycloak script.
The parameters are a little different depending if you are using the standalone operation mode or domain operation mode. For standalone mode, here is how you use the script.

```sh
$ .../bin/add-user-keycloak.sh -r master -u <username> -p <password>
```

```bat
> ...in\add-user-keycloak.bat -r master -u <username> -p <password>
```

For domain mode, you have to point the script to one of your server hosts using the `-sc` switch.

```sh
$ .../bin/add-user-keycloak.sh --sc domain/servers/server-one/configuration -r master -u <username> -p <password>
```

```bat
> ...in\add-user-keycloak.bat --sc domain/servers/server-one/configuration -r master -u <username> -p <password>
```

**Admin Console**

The bulk of your administrative tasks will be done through the Keycloak Admin Console. You can go to the console url directly at [http://localhost:8080/auth/admin/](http://localhost:8080/auth/admin/)

*Login Page.*
Enter the username and password you created on the Welcome Page or the `add-user-keycloak` script in the bin directory. This will bring you to the Keycloak Admin Console.

*Admin Console.*
The left drop down menu allows you to pick a realm you want to manage or to create a new one. The right drop down menu allows you to view your user account or logout. If you are curious about a certain feature, button, or field within the Admin Console, simply hover your mouse over any question mark ? icon. This will pop up tooltip text to describe the area of the console you are interested in. The image above shows the tooltip in action.

**User Management**

This section describes the administration functions for managing users.

**Login Page Settings**

There are several nice built-in login page features you can enable if you need the functionality.

**Authentication**

There are a few features you should be aware of when configuring authentication for your realm. Many organizations have strict password and OTP policies that you can enforce via settings in the Admin Console. You may or may not want to require different credential types for authentication. You may want to give users the option to login via Kerberos or disable or enable various built-in credential types. This chapter covers all of these topics.
**SSO Protocols**

The chapter gives a brief overview of the authentication protocols and how the Keycloak authentication server and the applications it secures interact with these protocols.

**Managing Clients**

Clients are entities that can request authentication of a user. Clients come in two forms. The first type of client is an application that wants to participate in single-sign-on. These clients just want Keycloak to provide security for them. The other type of client is one that is requesting an access token so that it can invoke other services on behalf of the authenticated user. This section discusses various aspects around configuring clients and various ways to do it.

**Roles**

Roles identify a type or category of user. **Admin, user, manager, and employee** are all typical roles that may exist in an organization. Applications often assign access and permissions to specific roles rather than individual users as dealing with users can be too fine grained and hard to manage. For example, the Admin Console has specific roles which give permission to users to access parts of the Admin Console UI and perform certain actions. There is a global namespace for roles and each client also has its own dedicated namespace where roles can be defined.

**Groups**

Groups in Keycloak allow you to manage a common set of attributes and role mappings for a set of users. Users can be members of zero or more groups. Users inherit the attributes and role mappings assigned to each group. To manage groups go to the **Groups** left menu item.
Groups are hierarchical. A group can have many subgroups, but a group can only have one parent. Subgroups inherit the attributes and role mappings from the parent. This applies to the user as well. So, if you have a parent group and a child group and a user that only belongs to the child group, the user inherits the attributes and role mappings of both the parent and child. In this example, we have a top level Sales group and a child North America subgroup. To add a group, click on the parent you want to add a new child to and click New button. Select the Groups icon in the tree to make a top-level group. Entering in a group name in the Create Group screen and hitting Save will bring you to the individual group management page.
The Attributes and Role Mappings tab work exactly as the tabs with similar names under a user. Any attributes and role mappings you define will be inherited by the groups and users that are members of this group.

To add a user to a group you need to go all the way back to the user detail page and click on the Groups tab there.

*User Groups.*
Select a group from the Available Groups tree and hit the join button to add the user to a group. Vice versa to remove a group. Here we've added the user Jim to the North America sales group. If you go back to the detail page for that group and select the Membership tab, Jim is now displayed there.

*Group Membership.*
Admin Console Access Control and Permissions

Each realm created on the Keycloak has a dedicated Admin Console from which that realm can be managed. The master realm is a special realm that allows admins to manage more than one realm on the system. You can also define fine-grained access to users in different realms to manage the server. This chapter goes over all the scenarios for this.

Realm Keys

The authentication protocols that are used by Keycloak require cryptographic signatures and sometimes encryption. Keycloak uses asymmetric key pairs, a private and public key, to accomplish this.

Keycloak has a single active keypair at a time, but can have several passive keys as well. The active keypair is used to create new signatures, while the passive keypairs can be used to verify previous signatures. This makes it possible to regularly rotate the keys without any downtime or interruption to users.

When a realm is created a key pair and a self-signed certificate is automatically generated.

To view the active keys for a realm select the realm in the admin console click on Realm settings then Keys. This will show the currently active keys for the realm. Keycloak currently only supports RSA signatures so there is only one active keypair. In the future as more signature algorithms are added there will be more active keypairs.

To view all available keys select All. This will show all active, passive and disabled keys. A keypair can have the status Active, but still not be selected as the currently active keypair for the realm. The selected active pair which is used for signatures is selected based on the first key provider sorted by priority that is able to provide an active keypair.
Rotating keys

It’s recommended to regularly rotate keys. To do so you should start by creating new keys with a higher priority than the existing active keys. Or create new keys with the same priority and making the previous keys passive.

Once new keys are available all new tokens and cookies will be signed with the new keys. When a user authenticates to an application the SSO cookie is updated with the new signature. When OpenID Connect tokens are refreshed new tokens are signed with the new keys. This means that over time all cookies and tokens will use the new keys and after a while the old keys can be removed.

How long you wait to delete old keys is a tradeoff between security and making sure all cookies and tokens are updated. In general it should be acceptable to drop old keys after a few weeks. Users that have not been active in the period between the new keys where added and the old keys removed will have to re-authenticate.

This also applies to offline tokens. To make sure they are updated the applications need to refresh the tokens before the old keys are removed.

As a guideline it may be a good idea to create new keys every 3-6 months and delete old keys 1-2 months after the new keys where created.

Adding a generated keypair

To add a new generated keypair select Providers and choose rsa-generated from the dropdown. You can change the priority to make sure the new keypair becomes the active keypair. You can also change the keysize if you want smaller or larger keys (default is 2048, supported values are 1024, 2048 and 4096).

Click Save to add the new keys. This will generated a new keypair including a self-signed certificate.

Changing the priority for a provider will not cause the keys to be re-generated, but if you want to change the keysize you can edit the provider and new keys will be generated.

Adding an existing keypair and certificate

To add a keypair and certificate obtained elsewhere select Providers and choose rsa from the dropdown. You can change the priority to make sure the new keypair becomes the active keypair.

Click on Select file for Private RSA Key to upload your private key. The file should be encoded in PEM format. You don’t need to upload the public key as it is automatically extracted from the private key.

If you have a signed certificate for the keys click on Select file next to X509 Certificate. If you don’t have one you can skip this and a self-signed certificate will be generated.

Loading keys from a Java Keystore

To add a keypair and certificate stored in a Java Keystore file on the host select Providers and choose java-keystore from the dropdown. You can change the priority to make sure the new keypair becomes the active keypair.

Fill in the values for Keystore, Keystore Password, Key Alias and Key Password and click on Save.

Making keys passive

Locate the keypair in Active or All then click on the provider in the Provider column. This will take you to the configuration screen for the key provider for the keys. Click on Active to turn it OFF, then click on Save. The keys
will no longer be active and can only be used for verifying signatures.

**Disabling keys**

Locate the keypair in **Active** or **All** then click on the provider in the **Provider** column. This will take you to the configuration screen for the key provider for the keys. Click on **Enabled** to turn it **OFF**, then click on **Save**. The keys will no longer be enabled.

Alternatively, you can delete the provider from the **Providers** table.

**Compromised keys**

Keycloak has the signing keys stored just locally and they are never shared with the client applications, users or other entities. However if you think that your realm signing key was compromised, you should first generate new keypair as described above and then immediately remove the compromised keypair.

Then to ensure that client applications won’t accept the tokens signed by the compromised key, you should update and push not-before policy for the realm, which is doable from the admin console. Pushing new policy will ensure that client applications won’t accept the existing tokens signed by the compromised key, but also the client application will be forced to download new keypair from the Keycloak, hence the tokens signed by the compromised key won’t be valid anymore. Note that your REST and confidential clients must have set **Admin URL**, so that Keycloak is able to send them the request about pushed not-before policy.

**Identity Brokering**

An Identity Broker is an intermediary service that connects multiple service providers with different identity providers. As an intermediary service, the identity broker is responsible for creating a trust relationship with an external identity provider in order to use its identities to access internal services exposed by service providers.

From a user perspective, an identity broker provides a user-centric and centralized way to manage identities across different security domains or realms. An existing account can be linked with one or more identities from different identity providers or even created based on the identity information obtained from them.

An identity provider is usually based on a specific protocol that is used to authenticate and communicate authentication and authorization information to their users. It can be a social provider such as Facebook, Google or Twitter. It can be a business partner whose users need to access your services. Or it an be a cloud-based identity service that you want to integrate with.

Usually, identity providers are based on the following protocols:

- SAML v2.0
- OpenID Connect v1.0
- OAuth v2.0

In the next sections we’ll see how to configure and use Keycloak as an identity broker, covering some important aspects such as:

- Social Authentication
- OpenID Connect v1.0 Brokering
- SAML v2.0 Brokering
- Identity Federation
User Session Management

When a user logs into a realm, Keycloak maintains a user session for them and remembers each and every client they have visited within the session. There are a lot of administrative functions that realm admins can perform on these user sessions. They can view login stats for the entire realm and dive down into each client to see who is logged in and where. Admins can logout a user or set of users from the Admin Console. They can revoke tokens and set up all the token and session timeouts there too.

User Storage Federation

Many companies have existing user databases that hold information about users and their passwords or other credentials. In may cases, it is just not possible to migrate off of those existing stores to a pure Keycloak deployment. Keycloak can federate existing external user databases. Out of the box we have support for LDAP and Active Directory. You can also code your own extension for any custom user databases you might have using our User Storage SPI.

The way it works is that when a user logs in, Keycloak will look into its own internal user store to find the user. If it can’t find it there it will iterate over every User Storage provider you have configured for the realm until it finds a match. Data from the external store is mapped into a common user model that is consumed by the Keycloak runtime. This common user model can then be mapped to OIDC token claims and SAML assertion attributes.

External user databases rarely have every piece of data need to support all the features that Keycloak has. In this case, the User Storage Provider can opt to store some things locally in the Keycloak user store. Some providers even import the user locally and sync periodically with the external store. All this depends on the capabilities of the provider and how its configured. For example, your external user store may not support OTP. Depending on the provider, this OTP support can be handled and stored by Keycloak.

Adding a Provider

To add a storage provider go to the User Federation left menu item in the Admin Console.

User Federation.
On the right side, there is an Add Provider list box. Choose the provider type you want to add and you will be brought to the configuration page of that provider.

**Auditing and Events**

Keycloak provides a rich set of auditing capabilities. Every single login action can be recorded and stored in the database and reviewed in the Admin Console. All admin actions can also be recorded and reviewed. There is also a Listener SPI with which plugins can listen for these events and perform some action. Built-in listeners include a simple log file and the ability to send an email if an event occurs.

**Export and Import**

Keycloak has the ability to export and import the entire database. This can be especially useful if you want to migrate your whole Keycloak database from one environment to another or migrate to a different database (for example from MySQL to Oracle). Export and import is triggered at server boot time and its parameters are passed in via Java system properties. It is important to note that because import and export happens at server startup, no other actions should be taken on the server or the database while this happens.

You can export/import your database either to:

- Directory on local filesystem
- Single JSON file on your filesystem

When importing using the directory strategy, note that the files need to follow the naming convention specified below. If you are importing files which were previously exported, the files already follow this convention.
• `{realm_name}`-realm.json, such as “acme-roadrunner-affairs-realm.json” for the realm named “acme-roadrunner-affairs”
• `{realm_name}`-users-{index}.json, such as “acme-roadrunner-affairs-users-0.json” for the first users file of the realm named “acme-roadrunner-affairs”

If you export to a directory, you can also specify the number of users that will be stored in each JSON file.

Note
If you have a bigger amount of users in your database (500 or more), it’s highly recommended to export into directory rather than to single file. Exporting into single file may lead to the very big file. Also the directory provider is using separate transaction for each “page” (file with users), which leads to much better performance. Default count of users per file (and transaction) is 50, which showed us best performance, but you have possibility to override (See below). Exporting to single file is using one transaction per whole export and one per whole import, which results in bad performance with large amount of users.

To export into unencrypted directory you can use:

```bash
bin/standalone.sh -Dkeycloak.migration.action=export
-Dkeycloak.migration.provider=dir -Dkeycloak.migration.dir=<DIR TO EXPORT TO>
```

And similarly for import just use `-Dkeycloak.migration.action=import` instead of export. To export into single JSON file you can use:

```bash
bin/standalone.sh -Dkeycloak.migration.action=export
-Dkeycloak.migration.provider=singleFile -Dkeycloak.migration.file=<FILE TO EXPORT TO>
```

Here’s an example of importing:

```bash
bin/standalone.sh -Dkeycloak.migration.action=import
-Dkeycloak.migration.provider=singleFile -Dkeycloak.migration.file=<FILE TO IMPORT>
-Dkeycloak.migration.strategy=OVERWRITE_EXISTING
```

Other available options are:

- `-Dkeycloak.migration.realmName` This property is used if you want to export just one specified realm instead of all. If not specified, then all realms will be exported.

- `-Dkeycloak.migration.usersExportStrategy` This property is used to specify where users are exported. Possible values are:
  - DIFFERENT_FILES - Users will be exported into different files according to the maximum number of users per file. This is default value.
  - SKIP - Exporting of users will be skipped completely.
  - REALM_FILE - All users will be exported to same file with the realm settings. (The result will be a file like “foo-realm.json” with both realm data and users.)
  - SAME_FILE - All users will be exported to same file but different from the realm file. (The result will be a file like “foo-realm.json” with realm data and “foo-users.json” with users.)

- `-Dkeycloak.migration.usersPerFile` This property is used to specify the number of users per file (and also per DB transaction). It’s 50 by default. It’s used only if usersExportStrategy is DIFFERENT_FILES

- `-Dkeycloak.migration.strategy` This property is used during import. It can be used to specify how to proceed if a realm with same name already exists in the database where you are going to import data. Possible values are:
  - IGNORE_EXISTING - Ignore importing if a realm of this name already exists.
• OVERWRITE_EXISTING - Remove existing realm and import it again with new data from the JSON file. If you want to fully migrate one environment to another and ensure that the new environment will contain the same data as the old one, you can specify this.

When importing realm files that weren’t exported before, the option `keycloak.import` can be used. If more than one realm file needs to be imported, a comma separated list of file names can be specified. This is more appropriate than the cases before, as this will happen only after the master realm has been initialized. Examples:

- `-Dkeycloak.import=/tmp/realm1.json`
- `-Dkeycloak.import=/tmp/realm1.json,/tmp/realm2.json`

**Admin console export/import**

Import of most resources can be performed from the admin console as well as export of most resources. Export of users is not supported.

Note: Attributes containing secrets or private information will be masked in export file.

The files created during a “startup” export can also be used to import from the admin UI. This way, you can export from one realm and import to another realm. Or, you can export from one server and import to another. Note: The admin console export/import allows just one realm per file.

**Warning**

The admin console import allows you to “overwrite” resources if you choose. Use this feature with caution, especially on a production system.

**Warning**

The admin console export allows you to export clients, groups, and roles. If there is a great number of any of these assets in your realm, the operation may take some time to complete. During that time server may not be responsive to user requests. Use this feature with caution, especially on a production system.

**User Account Service**

Keycloak has a built-in User Account Service which every user has access to. This service allows users to manage their account, change their credentials, update their profile, and view their login sessions. The URL to this service is `<server-root>/auth/realms/\{realm-name\}/account`.

*Account Service.*
The initial page is the user’s profile, which is the **Account** left menu item. This is where they specify basic data about themselves. This screen can be extended to allow the user to manage additional attributes. See the **Server Developer Guide** for more details.

The **Password** left menu item allows the user to change their password.

*Password Update.*
The Authenticator menu item allows the user to set up OTP if they desire. This will only show up if OTP is a valid authentication mechanism for your realm. Users are given directions to install FreeOTP or Google Authenticator on their mobile device to be their OTP generator. The QR code you see in the screen shot can be scanned into the FreeOTP or Google Authenticator mobile application for nice and easy setup.

*OTP Authenticator.*
The **Federated Identity** menu item allows the user to link their account with an *identity broker* (this is usually used to link social provider accounts together). This will show the list of external identity providers you have configured for your realm.

*Federated Identity.*
The Sessions menu item allows the user to view and manage which devices are logged in and from where. They can perform logout of these sessions from this screen too.

Sessions.
The **Applications** menu item shows users which applications they have access to.

*Applications.*
Themeable

Like all UIs in Keycloak, the User Account Service is completely themeable and internationalizable. See the Server Developer Guide for more details.

Threat Model Mitigation

This chapter discusses possible security vulnerabilities any authentication server could have and how Keycloak mitigates those vulnerabilities. A good list of potential vulnerabilities and what security implementations should do to mitigate them can be found in the OAuth 2.0 Threat Model document put out by the IETF. Many of those vulnerabilities are discussed here.

Admin CLI

NOTE: Admin CLI is a Technology Preview feature and is not fully supported.

In previous chapters we have described how to use the Keycloak Admin Console to perform administrative tasks. All those tasks can also be performed from command line by using Admin CLI command line tool.
Installing Admin CLI

Admin CLI is packaged inside Keycloak Server distribution. You can find execution scripts inside bin directory. The Linux script is called kcadm.sh, and the one for Windows is called kcadm.bat.

In order to setup the client to be used from any location on the filesystem you may want to add Keycloak server directory to your PATH.

On Linux:

```
$ export PATH=$PATH:$KEYCLOAK_HOME/bin
$ kcadm.sh
```

On Windows:

```
c:\> set PATH=%PATH%;%KEYCLOAK_HOME%\bin
c:\> kcadm
```

**Note**

To avoid unnecessary repetition the rest of this document will only give Windows examples in places where difference in command line is more than just in `kcadm` command name.

Using Admin CLI

Usually a user will first start an authenticated session by providing credentials, then perform some CRUD operations.

For example on Linux:

```
$ kcadm.sh config credentials --server http://localhost:8080/auth --realm demo --user admin --client admin
$ kcadm.sh create realms -s realm=demorealm -s enabled=true -o
$ CID=$(kcadm.sh create clients -r demorealm -s clientId=my_client -s 'redirectUris=[ "http://localhost:8980/myapp/*"]' -i)
$ kcadm.sh get clients/$CID/installation/providers/keycloak-oidc-keycloak-json
```

Or on Windows:

```
c:\> kcadm config credentials --server http://localhost:8080/auth --realm demo --user admin --client admin
c:\> kcadm create realms -s realm=demorealm -s enabled=true -o
c:\> kadm create clients -r demorealm -s clientId=my_client -s "redirectUris=["http://localhost:8980/myapp/*"]" -i > clientid.txt
c:\> set /p CID=<clientid.txt
c:\> kcadm get clients/%CID%/installation/providers/keycloak-oidc-keycloak-json
```

In a production environment Keycloak has to be accessed with https: to avoid exposing tokens to network sniffers. If server’s certificate is not issued by one of the trusted CAs that are included in Java’s default certificate truststore, then you will need to prepare a truststore.jks file, and instruct Admin CLI to use it.

For example on Linux:

```
$ kadm.sh config truststore --trustpass $PASSWORD ~/.keycloak/truststore.jks
```

Or on Windows:
Authenticating

Admin CLI works by making HTTP requests to Admin REST endpoints. Access to them is protected and requires authentication.

When logging in with Admin CLI you specify a server endpoint url, and a realm. Then you specify a username, or alternatively you can only specify a client id, which will result in special service account being used. In the first case, a password for the specified user has to be used at login. In the latter case there is no user password - only client secret or a Signed JWT is used.

The account that logs in needs to have proper permissions in order to be able to invoke Admin REST API operations. Specifically, realm-admin role of realm-management client is required for user to administer the realm within which the user is defined.

There are two primary mechanisms to authenticate. One is by using `kcadm config credentials` to start an authenticated session:

```
$ kadm.sh config credentials --server http://localhost:8080/auth --realm master --user admin --password admin
```

This approach maintains an authenticated session between kadm command invocations by saving the obtained access token, and associated refresh token, possibly other secrets as well in a private configuration file. By default this file is called kadm.config and is located under user’s home directory - it’s full pathname is $HOME/.keycloak/kcadm.config (on Windows it’s %HOMEPATH%\keycloak\kcadm.config). The file can be named something else by using -c, --config option.

See next chapter for more info on configuration file.

Another approach is to authenticate with each command invocation for the duration of that invocation only. This approach results in more load on the server, and more time spent with round-trips obtaining tokens, but has a benefit of not needing to save any tokens between invocations, thus nothing is saved to disk.

For example, when performing an operation we specify all the information required for authentication:

```
$ kadm.sh get realms --no-config --server http://localhost:8080/auth --realm master --user admin --password admin
```

See built-in help for more information on using Admin CLI.

For example:

```
$ kadm.sh help
```

See kadm.sh config credentials --help for more information about starting an authenticated session.

Working with alternative configurations

By default, Admin CLI automatically maintains a configuration file at a default location .keycloak/kcadm.config under user’s home directory.

You can use --config option at any time to point to a different file / location. This way you can maintain multiple authenticated sessions in parallel. It is safest to perform operations tied to a single config file from a single thread.
Make sure to not make a config file visible to other users on the system as it contains access tokens, and secrets that should be kept private.

You may want to avoid storing any secrets at all inside a config file for the price of less convenience and having to do more token requests. In that case you can use `--no-config` option with all your commands. You will have to specify all authentication info with each `kcadm` invocation.

**Basic operations, and resource URIs**

Admin CLI allows you to perform CRUD operations against Admin REST API endpoints in quite a generic way, with additional commands that simplify performing certain actions.

Main usage pattern is the following:

```
$ kcadm.sh create ENDPOINT [ARGUMENTS]
$ kcadm.sh get ENDPOINT [ARGUMENTS]
$ kcadm.sh update ENDPOINT [ARGUMENTS]
$ kcadm.sh delete ENDPOINT [ARGUMENTS]
```

Where operations `create`, `get`, `update`, and `delete` are mapped to HTTP verbs POST, GET, PUT, and DELETE, respectively. ENDPOINT is a target resource URI, and can either be absolute - starting with `http:` or `https:`, or relative - used to compose an absolute URL of the following format:

```
SERVER_URI/admin/realms/REALM/ENDPOINT
```

For example, if the server we authenticate against is `http://localhost:8080/auth`, and realm is `master`, then using `users` as ENDPOINT will result in the following resource URL: `http://localhost:8080/auth/admin/realms/master/users`.

If we set ENDPOINT to `clients` the effective resource URI would be: `http://localhost:8080/auth/admin/realms/master/clients`.

There is `realms` endpoint which is treated slightly differently since it is the container for realms. It resolves simply to:

```
SERVER_URI/admin/realms
```

There is also `serverinfo` which is treated the same way since it is independent of realms.

When authenticating as a user with realm-admin powers you may need to perform operations on multiple different realms. In that case you can specify `-r`, `--target-realm` option to tell explicitly which realm the operation should be executed against. Instead of using `REALM` as specified via `--realm` option of `kcadm.sh config credentials`, the TARGET_REALM will be used:

```
SERVER_URI/admin/realms/TARGET_REALM/ENDPOINT
```

For example:

```
$ kcadm.sh config credentials --server http://localhost:8080/auth --realm master --user admin --password admin
$ kcadm.sh create users -s username=testuser -s enabled=true -r demorealm
```

In this example we first start a session authenticated as `admin` user in `master` realm. Then we perform a POST call against the following resource URL:

```
http://localhost:8080/auth/admin/realms/demorealm/users
```
Realm operations

Creating a new realm A new realm can be created by specifying individual attributes on command line. They will be converted into a JSON document and sent to the server:

```
$ kcadm.sh create realms -s realm=demorealm -s enabled=true
```

Realm is not enabled by default. By enabling it, it can be used for authentication immediately.

A description for a new object can be in JSON format as well:

```
$ kcadm.sh create realms -f demorealm.json
```

JSON document with realm attributes can be sent directly from file or piped to standard input.

For example on Linux:

```
$ kcadm.sh create realms -f - << EOF
  { "realm": "demorealm", "enabled": true }
EOF
```

Or on Windows:

```
c:\> echo { "realm": "demorealm", "enabled": true } | kcadm create realms -f -
```

Listing existing realms The following will return a list of all the realms:

```
$ kcadm.sh get realms
```

Note, that the list of realms returned is additionally filtered on the server to only return realms the user has permissions for.

Often that is too much information as we may only be interested in realm name, or - for example - if it is enabled or not. You can specify the attributes to return by using `--fields` option:

```
$ kcadm.sh get realms --fields realm,enabled
```

You may even display the result as comma separated values:

```
$ kcadm.sh get realms --fields realm,enabled --format csv --noquotes
```

Getting a specific realm As is common for REST web services, in order to get an individual item of a collection, append an id to collection URI:

```
$ kcadm.sh get realms/master
```

Updating a realm There are several options when updating any resource. You can first get current state of resource, and save it into a file, then edit that file, and send it to server for update. For example:

```
$ kcadm.sh get realms/demorealm > demorealm.json
$ vi demorealm.json
$ kcadm.sh update realms/demorealm -f demorealm.json
```

This way the resource on the server will be updated with all the attributes in the sent JSON document.

Another option is to perform the update on-the-fly using `-s`, `--set` options to set new values:

```
$ kcadm.sh update realms/demorealm -s enabled=false
```
That would only update enabled attribute to false.

Deleting a realm  It’s very simple to delete a realm:

```
$ kcadm.sh delete realms/demorealm
```

Turning on all login page options for the realm  Set the attributes controlling specific capabilities to true. For example:

```
$ kcadm.sh update realms/demorealm -s registrationAllowed=true -s registrationEmailAsUsername=true -s rememberMe=true -s verifyEmail=true -s resetPasswordAllowed=true -s editUsernameAllowed=true
```

Listing the realm keys  It’s very simple to list the realm keys for a specific realm:

```
$ kcadm.sh get keys -r demorealm
```

Generating new realm keys  To add a new RSA generated keypair, first get id of the target realm. For example, to get id for a realm whose realm attribute is demorealm:

```
$ kcadm.sh get realms/demorealm --fields id --format csv --noquotes
```

Then add a new key provider with higher priority than any of the existing providers as revealed by kadm.sh get keys -r demorealm:

For example on Linux:

```
$ kcadm.sh create components -r demorealm -s name=rsa-generated -s providerId=rsa-generated -s providerType=org.keycloak.keys.KeyProvider -s parentID=959844c1-d149-41d7-8359-6a527fca0b0 -s 'config.priority=["101"]' -s 'config.enabled=["true"]' -s 'config.active=["true"]' -s 'config.keySize=["2048"]'
```

Or on Windows:

```
c:\> kcadm create components -r demorealm -s name=rsa-generated -s providerId=rsa-generated -s providerType=org.keycloak.keys.KeyProvider -s parentID=959844c1-d149-41d7-8359-6a527fca0b0 -s "config.priority=["101"]" -s "config.enabled=["true"]" -s "config.active=["true"]" -s "config.keySize=["2048"]"
```

Attribute parentId should be set to the value of target realm’s id.

The newly added key should now become the active key as revealed by kadm.sh get keys -r demorealm.

Adding new realm keys from Java Key Store file  To add a new keypair already prepared as a JKS file on the server, add a new key provider as follows:

For example on Linux:

```
$ kcadm.sh create components -r demorealm -s name=java-keystore -s providerId=java-keystore -s providerType=org.keycloak.keys.KeyProvider -s parentID=959844c1-d149-41d7-8359-6a527fca0b0 -s 'config.priority=["101"]' -s 'config.enabled=["true"]' -s 'config.active=["true"]' -s 'config.keystore=\[/opt/keycloak/keystore.jks\]' -s 'config.keystorePassword=["secret"]' -s 'config.keyPassword=["secret"]' -s 'config.alias=["localhost"]'
```

Or on Windows:

```
c:\> kcadm create components -r demorealm -s name=java-keystore -s providerId=java-keystore -s providerType=org.keycloak.keys.KeyProvider -s parentID=959844c1-d149-41d7-8359-6a527fca0b0 -s "config.priority=["101"]" -s "config.enabled=["true"]" -s "config.active=["true"]" -s "config.keystore=\[/opt/keycloak/keystore.jks\]" -s "config.keystorePassword=["secret"]" -s "config.keyPassword=["secret"]" -s "config.alias=["localhost"]"
```
And change attribute values for keystore, keystorePassword, keyPassword, and alias to match your specific keystore.

Attribute parentId should be set to the value of target realm’s id.

**Making key passive or disabling it** Identify the key you wish to make passive:

```
$ kcadm.sh get keys -r demorealm
```

Use providerId attribute of the key to construct an endpoint uri - components/PROVIDER_ID.

Then perform an update. For example on Linux:

```
$ kcadm.sh update components/PROVIDER_ID -r demorealm -s 'config.active=["false"]'
```

Or on Windows:

```
c:> kcadm update components/PROVIDER_ID -r demorealm -s "config.active=[\"false\"]"
```

Analogously, other key attributes can be updated.

To disable the key set new enabled value, for example: 'config.enabled=["false"]'

To change key’s priority set new priority value, for example: 'config.priority=["110"]'

**Deleting an old key** Make sure that the key you are deleting has been passive for some time, and then disabled for some time in order to prevent any existing tokens held by applications and users from abruptly failing to work.

Identify the key you wish to make passive:

```
$ kcadm.sh get keys -r demorealm
```

Use the providerId of that key to perform a delete. For example:

```
$ kcadm.sh delete components/PROVIDER_ID -r demorealm
```

**Configuring event logging for a realm** Use update against events/config endpoint.

Attribute eventsListeners sets the list of EventListenerProviderFactory id’s specifying all the event listeners receiving events. Separately from that there are attributes that control a built-in event storage which allows querying of past events via Admin REST API. There is separate control over logging of service calls - 'eventsEnabled, and auditing events triggered during Admin Console or Admin REST API - adminEventsEnabled. You may want to limit the time when old events expire so that your database doesn’t get filled up - eventsExpiration is set to time-to-live expressed in seconds.

For example, this is how you set a built-in event listener that will receive all the events and log them through jboss-logging (error events are logged as WARN, others as DEBUG, using a logger called org.keycloak.events):

On Linux:

```
$ kcadm.sh update events/config -r demorealm -s 'eventsListeners=["jboss-logging"]'
```

Or on Windows:

```
c:> kcadm update events/config -r demorealm -s "eventsListeners=[\"jboss-logging\"]"
```

This is how you turn on storage of all available ERROR events - not auditing events - for 2 days so they can be retrieved via Admin REST:

```
$ kcadm.sh update events/config -r demorealm -s 'eventsExpiration=172800'
```

1.2. Admin guide
On Linux:

```bash
$ kcadm.sh update events/config -r demorealm -s eventsEnabled=true -s
  -"enabledEventTypes=["LOGIN_ERROR","REGISTER_ERROR","LOGOUT_ERROR","CODE_TO_TOKEN_ERROR",
  "CLIENT_LOGIN_ERROR","FEDERATED_IDENTITY_LINK_ERROR","REMOVE_FEDERATED_IDENTITY_ERROR",
  "UPDATE_EMAIL_ERROR","UPDATE_PROFILE_ERROR","UPDATE_PASSWORD_ERROR","UPDATE_TOTP_ERROR",
  "VERIFY_EMAIL_ERROR","REMOVE_TOTP_ERROR","SEND_VERIFY_EMAIL_ERROR","SEND_RESET_PASSWORD_ERROR",
  "SEND_IDENTITY_PROVIDER_LINK_ERROR","RESET_PASSWORD_ERROR","IDENTITY_PROVIDER_FIRST_LOGIN_ERROR",
  "IDENTITY_PROVIDER_POST_LOGIN_ERROR","CUSTOM_REQUIRED_ACTION_ERROR","EXECUTE_ACTIONS_ERROR",
  "CLIENT_REGISTER_ERROR","CLIENT_UPDATE_ERROR","CLIENT_DELETE_ERROR"]' -s eventsExpiration=172800
```

Or on Windows:

```bash
c:\> kcadm update events/config -r demorealm -s eventsEnabled=true -s
  -"enabledEventTypes=["LOGIN_ERROR","REGISTER_ERROR","LOGOUT_ERROR","CODE_TO_TOKEN_ERROR",
  "CLIENT_LOGIN_ERROR","FEDERATED_IDENTITY_LINK_ERROR","REMOVE_FEDERATED_IDENTITY_ERROR",
  "UPDATE_EMAIL_ERROR","UPDATE_PROFILE_ERROR","UPDATE_PASSWORD_ERROR","UPDATE_TOTP_ERROR",
  "VERIFY_EMAIL_ERROR","REMOVE_TOTP_ERROR","SEND_VERIFY_EMAIL_ERROR","SEND_RESET_PASSWORD_ERROR",
  "SEND_IDENTITY_PROVIDER_LINK_ERROR","RESET_PASSWORD_ERROR","IDENTITY_PROVIDER_FIRST_LOGIN_ERROR",
  "IDENTITY_PROVIDER_POST_LOGIN_ERROR","CUSTOM_REQUIRED_ACTION_ERROR","EXECUTE_ACTIONS_ERROR",
  "CLIENT_REGISTER_ERROR","CLIENT_UPDATE_ERROR","CLIENT_DELETE_ERROR"]' -s eventsExpiration=172800
```

This is how you reset stored event types to all available event types - setting to empty list is the same as enumerating all:

```bash
$ kcadm.sh update events/config -r demorealm -s enabledEventTypes=[]
```

And this is how you turn on auditing events:

```bash
$ kadm.sh update events/config -r demorealm -s adminEventsEnabled=true -s
  -adminEventsDetailsEnabled=true
```

Here is how you get the last 100 events - they are ordered from newest to oldest:

```bash
$ kadm.sh get events --offset 0 --limit 100
```

Here is how you delete all saved events:

```bash
$ kadm delete events
```

**Flushing the caches** Use create operation, and one of the following endpoints: clear-realm-cache, clear-user-cache, clear-keys-cache.

Set realm to the same value as target realm.

For example:

```bash
$ kadm.sh create clear-realm-cache -r demorealm -s realm=demorealm
```

```bash
$ kadm.sh create clear-user-cache -r demorealm -s realm=demorealm
```

```bash
$ kadm.sh create clear-keys-cache -r demorealm -s realm=demorealm
```
Role operations

Creating a realm role  To create a realm role use `roles` endpoint:

```
$ kcadm.sh create roles -r demorealm -s name=user -s 'description=Regular user with limited set of permissions'
```

Creating a client role  To create a client role identify the client first - use `get` to list available clients:

```
$ kcadm.sh get clients -r demorealm --fields id,clientId
```

Then create a new role by using client’s id attribute to construct an endpoint uri `clients/ID/roles`.

For example:

```
$ kcadm.sh create clients/a95b6af3-0bdc-4878-ae2e-6d61a4eca9a0/roles -r demorealm -s name=editor -s 'description=Editor can edit, and publish any article'
```

Listing realm roles  To list existing realm roles use `get` command:

```
$ kcadm.sh get roles -r demorealm
```

You can also use `get-roles` command:

```
$ kcadm.sh get-roles -r demorealm
```

Listing client roles  Use special `get-roles` command, passing it either `clientId` (via `--cclientid` option) or id (via `--cid` option) to identify the client, and list defined roles:

For example:

```
$ kcadm.sh get-roles -r demorealm --cclientid realm-management
```

Getting a specific realm role  Use `get` command, and role `name` to construct an endpoint uri for a specific realm role `roles/ROLE_NAME`

For example:

```
$ kcadm.sh get roles/user -r demorealm
```

Where `user` is the name of existing role.

Alternatively, use special `get-roles` command, passing it role `name` (via `--rolename` option) or id (via `--roleid` option):

For example:

```
$ kcadm.sh get-roles -r demorealm --rolename user
```

Getting a specific client role  Use special `get-roles` command, passing it either `clientId` (via `--cclientid` option) or id (via `--cid` option) to identify the client, and passing it either role `name` (via `--rolename` option) or id (via `--roleid`) to identify a specific client role:

For example:

```
$ kcadm.sh get-roles -r demorealm --cclientid realm-management --rolename manage_clients
```

Updating a realm role  Use `update` operation with the same endpoint uri as for getting a specific realm role. For example:
Updating a client role  Use update operation with the same endpoint uri as for getting a specific client role. For example:

```
$ kcadm.sh update clients/a95b6af3-0bdc-4878-ae2e-6d61a4eca9a0/roles/editor -r demorealm -s 'description=User that can edit, and publish articles'
```

Deleting a realm role  Use delete operation with the same endpoint uri as for getting a specific realm role. For example:

```
$ kcadm.sh delete roles/user -r demorealm
```

Deleting a client role  Use delete operation with the same endpoint uri as for getting a specific client role. For example:

```
$ kcadm.sh delete clients/a95b6af3-0bdc-4878-ae2e-6d61a4eca9a0/roles/editor -r demorealm
```

Listing assigned, available and effective realm roles for a composite role

There is a dedicated `get-roles` command to simplify listing of both realm and client roles. It is an extension of `get` command thus it behaves like `get` command with additional semantics for listing roles.

To list **assigned** realm roles for the composite role you can specify the target composite role by either name (via `--rname` option) or id (via `--rid` option).

For example:

```
$ kcadm.sh get-roles -r demorealm --rname testrole
```

To list **effective** realm roles, use additional `--effective` option.

For example:

```
$ kcadm.sh get-roles -r demorealm --rname testrole --effective
```

To list realm roles that can still be added to the composite role, use `--available` option instead.

For example:

```
$ kcadm.sh get-roles -r demorealm --rname testrole --available
```

Listing assigned, available, and effective client roles for a composite role

You can again use `get-roles` command to simplify listing of roles.

To list **assigned** client roles for the composite role you can specify the target composite role by either name (via `--rname` option) or id (via `--rid` option), and client by either `clientId` (via `--cclientid` option) or id (via `--cid` option).

For example:

```
$ kcadm.sh get-roles -r demorealm --rname testrole --cclientid realm-management
```

To list **effective** realm roles, use additional `--effective` option.

For example:
To list realm roles that can still be added to the target composite role, use the \texttt{--available} option instead.

For example:

\begin{verbatim}
$ kcadm.sh get-roles -r demorealm --rname testrole --cclientid realm-management --available
\end{verbatim}

### Adding realm roles to a composite role

There is a dedicated \texttt{add-roles} command that can be used for adding both realm roles and client roles.

For example, to add the \texttt{user} role to composite role \texttt{testrole}:

\begin{verbatim}
$ kcadm.sh add-roles --rname testrole --rolename user -r demorealm
\end{verbatim}

### Removing realm roles from a composite role

There is a dedicated \texttt{remove-roles} command that can be used to remove both realm roles and client roles.

For example, to remove the \texttt{user} role from target composite role \texttt{testrole}:

\begin{verbatim}
$ kcadm.sh remove-roles --rname testrole --rolename user -r demorealm
\end{verbatim}

### Adding client roles to a composite role

There is a dedicated \texttt{add-roles} operation that can be used for adding both realm roles and client roles.

For example, to add to \texttt{testrole} composite role two roles defined on client \texttt{realm-management} - \texttt{create-client role} and \texttt{view-users role}:

\begin{verbatim}
$ kcadm.sh add-roles -r demorealm --rname testrole --cclientid realm-management --rolename create-client --rolename view-users
\end{verbatim}

### Removing client roles from a composite role

There is a dedicated \texttt{remove-roles} operation that can be used for removing both realm roles and client roles.

For example, to remove from \texttt{testrole} composite role two roles defined on client \texttt{realm management} - \texttt{create-client role} and \texttt{view-users role}:

\begin{verbatim}
$ kcadm.sh remove-roles -r demorealm --rname testrole --cclientid realm-management --rolename create-client --rolename view-users
\end{verbatim}

### Client operations

#### Creating a client

A new client can be created by using the \texttt{create} command against the clients endpoint. For example:

\begin{verbatim}
$ kcadm.sh create clients -r demorealm -s clientId=myapp -s enabled=true
\end{verbatim}

#### Listing clients

It’s very easy to list existing clients. For example:

\begin{verbatim}
$ kcadm.sh get clients -r demorealm --fields id,clientId
\end{verbatim}

Here we filter the output to only list \texttt{id}, and \texttt{clientId} attributes.

#### Getting a specific client

Use client’s \texttt{id} to construct an endpoint uri targeting specific client - \texttt{clients/ID}. For example:

\begin{verbatim}
$ kcadm.sh get clients -r demorealm --fields id,clientId
\end{verbatim}
Getting adapter configuration file (keycloak.json) for specific client  Use client’s id to construct an endpoint uri targeting specific client - clients/ID/installation/providers/keycloak-oidc-keycloak-json.

For example:

```
$ kcadm.sh get clients/c7b8547f-e748-4333-95d0-410b76b3f4a3/installation/providers/keycloak-oidc-keycloak-json -r demorealm
```

Getting Wildfly subsystem adapter configuration for specific client  Use client’s id to construct an endpoint uri targeting specific client - clients/ID/installation/providers/keycloak-oidc-jboss-subsystem.

For example:

```
$ kcadm.sh get clients/c7b8547f-e748-4333-95d0-410b76b3f4a3/installation/providers/keycloak-oidc-jboss-subsystem -r demorealm
```

Updating a client  Use update operation with the same endpoint uri as for getting a specific client. For example on Linux:

```
```

Or on Windows:

```
```

Deleting a client  Use delete operation with the same endpoint uri as for getting a specific client. For example:

```
$ kcadm.sh delete clients/c7b8547f-e748-4333-95d0-410b76b3f4a3 -r demorealm
```

User operations

Creating a user  A new user can be created using the create command against the users endpoint. For example:

```
$ kcadm.sh create users -r demorealm -s username=testuser -s enabled=true
```

Listing users  Use users endpoint to list users. Number of users may be large, and you may want to limit how many are returned:

```
$ kcadm.sh get users -r demorealm --offset 0 --limit 1000
```

It’s also possible to filter users by username, firstName, lastName, or email. For example:

```
$ kcadm.sh get users -r demorealm -q email=google.com
$ kcadm.sh get users -r demorealm -q username=testuser
```

Note that filtering doesn’t use exact matching. For example, the above would match the value of username attribute against *testuser* pattern.
You can also filter across multiple attributes by specifying multiple \(-q\) options, which would return only users that match condition for all the attributes.

**Getting a specific user**  Use `user id` to compose an endpoint uri matching a specific user - `users/USER_ID`.

For example:

```
$ kcadm.sh get users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm
```

**Updating a user**  Use `update` operation with the same endpoint uri as for getting a specific user. For example on Linux:

```
$ kcadm.sh update users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm -s "requiredActions=["VERIFY_EMAIL","UPDATE_PROFILE","CONFIGURE_TOTP","UPDATE_PASSWORD"]"
```

Or on Windows:

```
c:\> kcadm update users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm -s "requiredActions=["VERIFY_EMAIL","UPDATE_PROFILE","CONFIGURE_TOTP","UPDATE_PASSWORD"]"
```

**Deleting a user**  Use `delete` operation with the same endpoint uri as for getting a specific user. For example:

```
$ kcadm.sh delete users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm
```

**Resetting user’s password**  There is a dedicated `set-password` command specifically to reset user’s password. For example:

```
$ kcadm.sh set-password -r demorealm --username testuser --password NEWPASSWORD --temporary
```

That will set a temporary password for the user, which they will have to change the next time they login.

You can use `--userid` if you want to specify the user by using `id` attribute.

The same can be achieved using the `update` operation against an endpoint constructed from one for getting a specific user - `users/USER_ID/reset-password`.

For example:

```
$ kcadm.sh update users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2/reset-password -r demorealm -s type=password -s value=NEWPASSWORD -s temporary=true -n
```

The last parameter \((-n)\) forces a so called *no-merge* update which performs a PUT only, without first doing a GET to retrieve current state of the resource. In this case it is necessary since `reset-password` endpoint doesn’t support GET.

**Listing assigned, available, and effective realm roles for a user**  There is a dedicated `get-roles` command to simplify listing of both realm and client roles. It is an extension of `get` command thus it behaves like `get` command with additional semantics for listing roles.

To list *assigned* realm roles for the user you can specify the target user by either `username` or `id`. For example:

```
$ kcadm.sh get-roles -r demorealm --username testuser
```

To list *effective* realm roles, use additional `--effective` option. For example:

```
$ kcadm.sh get-roles -r demorealm --username testuser --effective
```
$ kcadm.sh get-roles -r demorealm --username testuser --effective

To list realm roles that can still be added to the user, use --available option instead.
For example:

$ kcadm.sh get-roles -r demorealm --username testuser --available

Listing assigned, available, and effective client roles for a user  You can again use get-roles command to simplify listing of roles.
To list assigned client roles for the user you can specify the target user by either username (via –username option) or id (via –uid option), and client by either clientId (via –cclientid option) or id (via –cid option).
For example:

$ kcadm.sh get-roles -r demorealm --username testuser --cclientid realm-management

To list effective realm roles, use additional --effective option.
For example:

$ kcadm.sh get-roles -r demorealm --username testuser --cclientid realm-management --effective

To list realm roles that can still be added to the user, use --available option instead.
For example:

$ kcadm.sh get-roles -r demorealm --username testuser --cclientid realm-management --available

Adding realm roles to a user  There is a dedicated add-roles command that can be used for adding both realm roles and client roles.
For example, to add user role to user testuser:

$ kcadm.sh add-roles --username testuser --rolename user -r demorealm

Removing realm roles from a user  There is a dedicated remove-roles command that can be used to remove both realm roles and client roles.
For example, to remove user role from user testuser:

$ kcadm.sh remove-roles --username testuser --rolename user -r demorealm

Adding client roles to a user  There is a dedicated add-roles operation that can be used for adding both realm roles and client roles.
For example, to add to user testuser two roles defined on client realm management -create-client role and view-users role:

$ kcadm.sh add-roles -r demorealm --username testuser --cclientid realm-management --rolename create-client --rolename view-users

Removing client roles from a user  There is a dedicated remove-roles operation that can be used for removing both realm roles and client roles.
For example, to remove from user testuser two roles defined on client realm management -create-client role and view-users role:
Listing user's sessions  First identify user's id then use it to compose an endpoint uri - users/ID/sessions. Now use get to retrieve a list of user's sessions.

For example:

```bash
$ kcadm.get users/6da5ab89-3397-4205-afaa-e201ff638f9e/sessions
```

Logging out user from specific session  To invalidate a session you only need session's id. You can get it by listing user's sessions.

Use session's id to compose an endpoint uri - sessions/ID.

The use delete to invalidate it. For example:

```bash
$ kcadm.sh delete sessions/d0eaa7cc-8c5d-489d-811a-69d3c4ec84d1
```

Logging out user from all sessions  You need user's id to construct an endpoint uri - users/ID/logout.

Use create to send logout-from-all-sessions request:

```bash
$ kcadm.sh create users/6da5ab89-3397-4205-afaa-e201ff638f9e/logout -r demorealm -s realm=demorealm -s user=6da5ab89-3397-4205-afaa-e201ff638f9e
```

Group operations

Creating a group  Use create operation, and groups endpoint to create a new group:

```bash
$ kcadm.sh create groups -r demorealm -s name=Group
```

Listing groups  Use get operation, and groups endpoint to list groups:

```bash
$ kcadm.sh get groups -r demorealm
```

Getting a specific group  Use group’s id to construct an endpoint uri - groups/GROUP_ID:

For example:

```bash
$ kcadm.sh get groups/51204821-0580-46db-8f2d-27106c6b5ded -r demorealm
```

Updating a group  Use update operation with the same endpoint uri as for getting a specific group. For example:

```bash
$ kcadm.sh update groups/51204821-0580-46db-8f2d-27106c6b5ded -r demorealm #email=["group@example.com"]' -r demorealm
```

Deleting a group  Use delete operation with the same endpoint uri as for getting a specific group. For example:

```bash
$ kcadm.sh delete groups/51204821-0580-46db-8f2d-27106c6b5ded -r demorealm
```

Creating a sub-group  Find id of the parent group - by listing groups for example. Use that id to construct an endpoint uri - groups/GROUP_ID/children:

For example:
Moving a group under another group  Find id of existing parent group, and of existing child group. Use parent group’s id to construct and endpoint uri - groups/PARENT_GROUP_ID/children.

Make create operation against this endpoint, and pass child group id as JSON body. For example:

```
$ kcadm.sh create groups/51204821-0580-46db-8f2d-27106c6b5ded/children -r demorealm -
   -s name=SubGroup
```

Get groups for specific user  To get user’s membership in groups, use user’s id to compose a resource URI - users/USER_ID/groups

For example:

```
$ kcadm.sh get users/b544f379-5fc4-49e5-8a8d-5cfb71f46f53/groups -r demorealm
```

Adding user to a group  To join user to a group use update operation against a resource uri composed from user’s id, and group’s id - users/USER_ID/groups/GROUP_ID.

For example:

```
$ kcadm.sh update users/b544f379-5fc4-49e5-8a8d-5cfb71f46f53/groups/ce01117a-7426-
   -4670-a29a-5c118056f20 -r demorealm -s realm=demorealm -s userId=b544f379-5fc4-49e5-
   -8a8d-5cfb71f46f53 -s groupId=ce01117a-7426-4670-a29a-5c118056f20 -n
```

Removing user from a group  To remove user from a group use delete operation against the same resource uri as used for adding user to a group - users/USER_ID/groups/GROUP_ID.

For example:

```
$ kcadm.sh delete users/b544f379-5fc4-49e5-8a8d-5cfb71f46f53/groups/ce01117a-7426-
   -4670-a29a-5c118056f20 -r demorealm
```

Listing assigned, available, and effective realm roles for a group  There is a dedicated get-roles command to simplify listing of roles. It is an extension of get command thus it behaves like get command with additional semantics for listing roles.

To list assigned realm roles for the group you can specify the target group by name (via --gname option), path (via --gpath option), or id (via --gid option).

For example:

```
$ kcadm.sh get-roles -r demorealm --gname Group
```

To list effective realm roles, use additional --effective option.

For example:

```
$ kcadm.sh get-roles -r demorealm --gname Group --effective
```

To list realm roles that can still be added to the group, use --available option instead.

For example:

```
$ kcadm.sh get-roles -r demorealm --gname Group --available
```

Listing assigned, available, and effective client roles for a group  A dedicated get-roles command can be used to list for both realm roles and client roles.
To list assigned client roles for the user you can specify the target group by either name (via --gname option) or id (via --gid option), and client by either clientId (via --cclientid option) or id (via --id option).

For example:

```bash
$ kcadm.sh get-roles -r demorealm --gname Group --cclientid realm-management
```

To list effective realm roles, use additional --effective option.

For example:

```bash
$ kcadm.sh get-roles -r demorealm --gname Group --cclientid realm-management --effective
```

To list realm roles that can still be added to the group, use --available option instead.

For example:

```bash
$ kcadm.sh get-roles -r demorealm --gname Group --cclientid realm-management --available
```

### Identity Providers operations

**Listing available identity providers** Use serverinfo endpoint to list available identity providers. For example:

```bash
$ kcadm.sh get serverinfo -r demorealm --fields 'identityProviders(*)'
```

Note that serverinfo endpoint is handled similarly to realms endpoint in that it is not resolved into resource URI as relative to target realm.

**Listing configured identity providers** Use identity-provider/instances endpoint. For example:

```bash
$ kcadm.sh get identity-provider/instances -r demorealm --fields alias,providerId, enabled
```

**Getting a specific configured identity provider** To get a specific identity provider use an alias attribute of identity provider to construct an endpoint uri - identity-provider/instances/ALIAS.

For example:

```bash
$ kcadm.sh get identity-provider/instances/facebook -r demorealm
```

**Removing a specific configured identity provider** Use delete operation with the same endpoint uri as for getting a specific configured identity provider. For example:

```bash
$ kcadm.sh delete identity-provider/instances/facebook -r demorealm
```

**Configuring a Keycloak OpenID Connect identity provider** For Keycloak OpenID Connect use keycloak-oidc as providerId when creating a new identity provider instance.

Provide config attributes authorizationUrl, tokenUrl, clientId, and clientSecret.

For example:

```bash
```

(continues on next page)
### Configuring an OpenID Connect identity provider
You configure the generic OpenID Connect provider the same way as Keycloak OpenID Connect provider, except that you set `providerId` attribute value to `oidc`.

### Configuring a SAML 2 identity provider
Use `saml` as `providerId` when creating a new identity provider instance. Provide `config` attributes - `singleSignOnServiceUrl`, `nameIDPolicyFormat`, and `signatureAlgorithm`.

For example:

```bash
```

### Configuring a Facebook identity provider
Use `facebook` as `providerId` when creating a new identity provider instance. Provide `config` attributes - `clientId` and `clientSecret` as obtained from Facebook Developers application configuration page for your application.

```bash
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=facebook -s providerId=facebook -s enabled=true -s 'config.useJwksUrl="true"' -s config.clientId=FACEBOOK_CLIENT_ID -s config.clientSecret=FACEBOOK_CLIENT_SECRET
```

### Configuring a Google identity provider
Use `google` as `providerId` when creating a new identity provider instance. Provide `config` attributes - `clientId` and `clientSecret` as obtained from Google Developers application configuration page for your application.

```bash
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=google -s providerId=google -s enabled=true -s 'config.useJwksUrl="true"' -s config.clientId=GOOGLE_CLIENT_ID -s config.clientSecret=GOOGLE_CLIENT_SECRET
```

### Configuring a Twitter identity provider
Use `twitter` as `providerId` when creating a new identity provider instance. Provide `config` attributes - `clientId` and `clientSecret` as obtained from Twitter Application Management application configuration page for your application.

```bash
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=twitter -s providerId=twitter -s enabled=true -s 'config.useJwksUrl="true"' -s config.clientId=TWITTER_API_KEY -s config.clientSecret=TWITTER_API_SECRET
```

### Configuring a GitHub identity provider
Use `github` as `providerId` when creating a new identity provider instance. Provide `config` attributes - `clientId` and `clientSecret` as obtained from GitHub Developer Application Settings page for your application.

```bash
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=github -s providerId=github -s enabled=true -s 'config.useJwksUrl="true"' -s config.clientId=GITHUB_CLIENT_ID -s config.clientSecret=GITHUB_CLIENT_SECRET
```

### Configuring a LinkedIn identity provider
Use `linkedin` as `providerId` when creating a new identity provider instance. Provide `config` attributes - `clientId` and `clientSecret` as obtained from LinkedIn Developer Console application page for your application.

```bash
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=linkedin -s providerId=linkedin -s enabled=true -s 'config.useJwksUrl="true"' -s config.clientId=LINKEDIN_CLIENT_ID -s config.clientSecret=LINKEDIN_CLIENT_SECRET
```
Configuring a Microsoft Live identity provider Use `microsoft` as `providerId` when creating a new identity provider instance. Provide `config` attributes `- clientId` and `clientSecret` as obtained from Microsoft Application Registration Portal page for your application.

```bash
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=microsoft -s providerId=microsoft -s enabled=true -s 'config.useJwksUrl="true"' -s config.clientId=MICROSOFT_APP_ID -s config.clientSecret=MICROSOFT_PASSWORD
```

Configuring a StackOverflow identity provider Use `stackoverflow` as `providerId` when creating a new identity provider instance. Provide `config` attributes `- clientId`, `clientSecret` and `key` as obtained from Stack Apps OAuth page for your application.

```bash
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=stackoverflow -s providerId=stackoverflow -s enabled=true -s 'config.useJwksUrl="true"' -s config.clientId=STACKAPPS_CLIENT_ID -s config.clientSecret=STACKAPPS_CLIENT_SECRET -s config.key=STACKAPPS_KEY
```

Storage Providers operations

Configuring a Kerberos storage provider Use `create` against `user-federation/instances` endpoint. Specify `kerberos` as a value of `providerName` attribute.

For example:

```bash
$ kcadm.sh create user-federation/instances -r demorealm -s providerName=kerberos -s priority=0 -s config.debug=false -s config.allowPasswordAuthentication=true -s 'config.editMode="UNSYNCED"' -s config.updateProfileFirstLogin=true -s config.allowKerberosAuthentication=true -s 'config.kerberosRealm="KEYCLOAK.ORG"' -s 'config.keyTab="http.keytab"' -s 'config.serverPrincipal="HTTP/localhost@KEYCLOAK.ORG"'
```

Configuring an LDAP user storage provider Use `create` against `components` endpoint. Specify `ldap` as a value of `providerId` attribute, and `org.keycloak.storage.UserStorageProvider` as value of `providerType` attribute. Provide `realmId` as value of `parentId` attribute.

For example, to create a Kerberos integrated LDAP provider:

```bash
$ kcadm.sh create components -r demorealm -s name=kerberos-ldap-provider -s providerId=ldap -s providerType=org.keycloak.storage.UserStorageProvider -s parentId=3d9c572b-8f33-483f-98a6-8bb421667867 -s 'config.priority=["1"]' -s 'config.fullSyncPeriod=["-1"]' -s 'config.changedSyncPeriod=["-1"]' -s 'config.cachePolicy=["DEFAULT"]' -s config.evictionDay[]= -s config.evictionHour[]= -s config.evictionMinute[]= -s config.maxLifespan=["1000"] -s 'config.maxSizeForSync=["1000"]' -s 'config.editMode=["WRITABLE"]' -s 'config.syncRegistrations=[false]"' -s 'config.vendor=["other"]' -s 'config.usernameLDAPAttribute=["uid"]' -s 'config.rdnLDAPAttribute=["uid"]' -s 'config.uuidLDAPAttribute=["entryUUID"]' -s 'config.fullSyncPeriod=["-1"]' -s 'config.changedSyncPeriod=["-1"]' -s 'config.cachePolicy=["DEFAULT"]' -s 'config.evictionDay[]= -s config.evictionHour[]= -s config.evictionMinute[]= -s config.maxLifespan=["1000"] -s 'config.fullSyncPeriod=["-1"]' -s 'config.changedSyncPeriod=["-1"]' -s 'config.cachePolicy=["DEFAULT"]' -s 'config.evictionDay[]= -s config.evictionHour[]= -s config.evictionMinute[]= -s config.maxLifespan=["1000"] -s 'config.fullSyncPeriod=["-1"]' -s 'config.changedSyncPeriod=["-1"]' -s 'config.cachePolicy=["DEFAULT"]' -s 'config.evictionDay[]= -s config.evictionHour[]= -s config.evictionMinute[]= -s config.maxLifespan=["1000"] -s 'config.fullSyncPeriod=["-1"]' -s 'config.changedSyncPeriod=["-1"]' -s 'config.cachePolicy=["DEFAULT"]' -s 'config.evictionDay[]= -s config.evictionHour[]= -s config.evictionMinute[]= -s config.maxLifespan=["1000"] -s 'config.fullSyncPeriod=["-1"]' -s 'config.changedSyncPeriod=["-1"]' -s 'config.cachePolicy=["DEFAULT"]' -s 'config.evictionDay[]= -s config.evictionHour[]= -s config.evictionMinute[]= -s config.maxLifespan=["1000"] -s 'config.fullSyncPeriod=["-1"]' -s 'config.changedSyncPeriod=["-1"]' -s 'config.cachePolicy=["DEFAULT"]' -s 'config.evictionDay[]= -s config.evictionHour[]= -s config.evictionMinute[]= -s config.maxLifespan=["1000"] -s 'config.fullSyncPeriod=["-1"]' -s 'config.changedSyncPeriod=["-1"]'
```
Removing a user storage provider instance  Use storage provider instance’s `id` attribute to compose an endpoint uri - `components/ID`.

Perform `delete` operation against this endpoint. For example:

```bash
$ kcadm.sh delete components/3d9c572b-8f33-483f-98a6-8bb421667867 -r demorealm
```

Triggering synchronization of all users for specific user storage provider

Use storage provider’s `id` attribute to compose an endpoint uri - `user-storage/ID_OF_USER_STORAGE_INSTANCE/sync` Add `action=triggerFullSync` query parameter and use `create`.

For example:

```bash
$ kcadm.sh create user-storage/b7c63d02-b62a-4fc1-977c-947d6a09e1ea/sync?action=triggerFullSync
```

Triggering synchronization of changed users for specific user storage provider

Use storage provider’s `id` attribute to compose an endpoint uri - `user-storage/ID_OF_USER_STORAGE_INSTANCE/sync` Add `action=triggerChangedUsersSync` query parameter and use `create`.

For example:

```bash
$ kcadm.sh create user-storage/b7c63d02-b62a-4fc1-977c-947d6a09e1ea/sync?action=triggerChangedUsersSync
```

Test LDAP user storage connectivity  Perform `get` operation against `testLDAPConnection` endpoint. Provide query parameters `bindCredential`, `bindDn`, `connectionUrl`, and `useTruststoreSpi`, and set `action` query parameter to `testConnection`.

For example:

```bash
$ kcadm.sh get testLDAPConnection -q action=testConnection -q bindCredential=secret -q bindDn=uid=admin,ou=system -q connectionUrl=ldap://localhost:10389 -q useTruststoreSpi=ldapsOnly
```

Test LDAP user storage authentication  Perform `get` operation against `testLDAPConnection` endpoint. Provide query parameters `bindCredential`, `bindDn`, `connectionUrl`, and `useTruststoreSpi`, and set `action` query parameter to `testAuthentication`.

For example:

```bash
$ kcadm.sh get testLDAPConnection -q action=testAuthentication -q bindCredential=secret -q bindDn=uid=admin,ou=system -q connectionUrl=ldap://localhost:10389 -q useTruststoreSpi=ldapsOnly
```

Adding mappers

Adding a hardcoded role LDAP mapper  Use `create` against `components` endpoint. Set `providerType` attribute to `org.keycloak.storage.ldap.mappers.LDAPStorageMapper`. Set `parentId` attribute to `id` of LDAP provider instance. Set `providerId` attribute to `hardcoded-ldap-role-mapper`. Make sure to provide a value of `role config` parameter.

For example:
Adding a MS Active Directory mapper Use create against components endpoint. Set providerType attribute to org.keycloak.storage.ldap.mappers.LDAPStorageMapper. Set parentId attribute to id of LDAP provider instance. Set providerId attribute to msad-user-account-control-mapper.

For example:

```bash
$ kcadm.sh create components -r demorealm -s name=msad-user-account-control-mapper -s providerId=msad-user-account-control-mapper -s providerType=org.keycloak.storage.ldap.mappers.LDAPStorageMapper -s parentId=b7c63d02-b62a-4fc1-977c-947d6a09e1ea -s 'config.role=["realm-management.create-client"]'
```

Adding a user attribute LDAP mapper Use create against components endpoint. Set providerType attribute to org.keycloak.storage.ldap.mappers.LDAPStorageMapper. Set parentId attribute to id of LDAP provider instance. Set providerId attribute to user-attribute-ldap-mapper.

For example:

```bash
$ kcadm.sh create components -r demorealm -s name=user-attribute-ldap-mapper -s providerId=user-attribute-ldap-mapper -s providerType=org.keycloak.storage.ldap.mappers.LDAPStorageMapper -s parentId=b7c63d02-b62a-4fc1-977c-947d6a09e1ea -s 'config."user.model.attribute"=["email"]' -s 'config."ldap.attribute"=["mail"]' -s 'config."read.only"=["false"]' -s 'config."always.read.value.from.ldap"=["false"]'
```

Adding a group LDAP mapper Use create against components endpoint. Set providerType attribute to org.keycloak.storage.ldap.mappers.LDAPStorageMapper. Set parentId attribute to id of LDAP provider instance. Set providerId attribute to group-ldap-mapper.

For example:

```bash
$ kcadm.sh create components -r demorealm -s name=group-ldap-mapper -s providerId=group-ldap-mapper -s providerType=org.keycloak.storage.ldap.mappers.LDAPStorageMapper -s parentId=b7c63d02-b62a-4fc1-977c-947d6a09e1ea -s 'config."groups.dn"=[]' -s 'config."group.name.ldap.attribute"=["cn"]' -s 'config."group.object.classes"=["groupOfNames"]' -s 'config."preserve.group.inheritance"=["true"]' -s 'config."membership.ldap.attribute"=["member"]' -s 'config."membership.attribute."groups.dn"=["true"]' -s 'config."membership.attribute."groups.object.classes"=["groupOfNames"]' -s 'config."groups.dn"="true"' -s 'config."groups.object.classes"="true"' -s 'config."grant.group.inheritance"=["true"]' -s 'config."membership.attribute."groups.dn"=["true"]' -s 'config."membership.attribute."groups.object.classes"=["groupOfNames"]' -s 'config."groups.dn"="true"' -s 'config."groups.object.classes"="true"' -s 'config."grant.group.inheritance"=["true"]' -s 'config."membership.attribute."groups.dn"=["true"]' -s 'config."membership.attribute."groups.object.classes"=["groupOfNames"]'
```

Adding a full name LDAP mapper Use create against components endpoint. Set providerType attribute to org.keycloak.storage.ldap.mappers.LDAPStorageMapper. Set parentId attribute to id of LDAP provider instance. Set providerId attribute to full-name-ldap-mapper.

For example:

```bash
$ kcadm.sh create components -r demorealm -s name=full-name-ldap-mapper -s providerId=full-name-ldap-mapper -s providerType=org.keycloak.storage.ldap.mappers.LDAPStorageMapper -s parentId=b7c63d02-b62a-4fc1-977c-947d6a09e1ea -s 'config."ldap.full.name.attribute"=["cn"]' -s 'config."read.only"=["false"]' -s 'config."write.only"=["true"]'
```

(continues on next page)
Authentication operations

Setting a password policy  Set realm's passwordPolicy attribute to an enumeration expression including specific policy provider id, and an optional configuration:

For example, to set password policy to 20000 hash iterations, requiring at least one special character, at least one uppercase character, at least one digit character, not be equal to user's username, and be at least 8 characters long you would use the following:

```bash
$ kcadm.sh update realms/demorealm -s 'passwordPolicy="hashIterations and specialChars and upperCase and digits and notUsername and length"'
```

If you want want to use values different from defaults, pass configuration in brackets.

For example, to set password policy to 25000 hash iterations, requiring at least two special characters, at least two uppercase characters, at least two lowercase characters, at least two digits, be at least nine characters long, not be equal to user's username, and not repeat for at least four changes back:

```bash
$ kcadm.sh update realms/demorealm -s 'passwordPolicy="hashIterations(25000) and specialChars(2) and upperCase(2) and lowerCase(2) and digits(2) and length(9) and notUsername and passwordHistory(4)"'
```

Getting the current password policy  Get current realm configuration and filter out everything but passwordPolicy attribute.

For example, to display passwordPolicy attribute for demorealm:

```bash
$ kcadm.sh get realms/demorealm --fields passwordPolicy
```

Listing authentication flows  Use get operation against authentication/flows endpoint. For example:

```bash
$ kcadm.sh get authentication/flows -r demorealm
```

Migration from older versions

To upgrade to a new version of Keycloak first download and install the new version of Keycloak. Once the new version is installed migrate the config files, database, keycloak-server.json, providers, themes and applications to the new applications.

This chapter contains some general migration details which are applicable to all versions. There are instructions for migration that is only applicable to a specific release. If you are upgrading from a very old version you need to go through all intermediate version specific migration.

It’s highly recommended that you backup your database prior to upgrading Keycloak.

Migration from a candidate release (CR) to a Final release is not supported. We do however recommend that you test migration for a CR so we can resolve any potential issues before the Final is released.

Migration of config files using migration scripts

As part of your migration, you should use your old versions of config files standalone.xml, standalone-ha.xml, and/or domain.xml. These files typically contain configuration that is unique to your own environment. So,
the first thing to do as part of a migration is to copy those files to the new Keycloak server installation, replacing the default versions.

If migrating from Keycloak version 2.1.0 or older, you should also copy `keycloak-server.json` to `standalone/configuration` and/or `domain/configuration`.

There will be configuration in those config files that pertains to Keycloak, which will need to be upgraded. For that, you should run one or more of the appropriate upgrade scripts. They are `migrate-standalone.cli`, `migrate-standalone-ha.cli`, `migrate-domain-standalone.cli` and `migrate-domain-clustered.cli`.

The server should not be running when you execute a migration script.

```bash
$ .../bin/jboss-cli.sh --file=migrate-standalone.cli
```

Note that for upgrading domain.xml, there are two migration scripts, `migrate-domain-standalone.cli` and `migrate-domain-clustered.cli`. These scripts migrate separate profiles within your domain.xml that were originally shipped with Keycloak. If you have changed the name of the profile there is a variable near the top of the script that you will need to change.

If you are migrating `keycloak-server.json`, this will also be migrated as needed. If you prefer, you can migrate `keycloak-server.json` beforehand using the instructions in the next section.

One thing to note is that the migration scripts only work for Keycloak versions 1.8.1 forward. If migrating an older version you will need to manually upgrade your config files to at least be 1.8.1 compliant.

Lastly, you may want to examine the contents of the scripts before running. They show exactly what will be changed for each version. They also have values at the top of the script that you may need to change based on your environment.

**Migrate and convert keycloak-server.json**

If you ran one of the migration scripts from the previous section then you have probably already migrated your keycloak-server.json. This section is kept here for reference. If you prefer, you can follow the instructions in this section before running the migration scripts above.

You should copy `standalone/configuration/keycloak-server.json` from the old version to make sure any configuration changes you’ve done are added to the new installation. The version specific section below will list any changes done to this file that you have to do when upgrading from one version to another.

Keycloak is moving away from the use of keycloak-server.json. For this release, the server will still work if this file is in `standalone/configuration/keycloak-server.json`, but it is highly recommended that you convert to using `standalone.xml`, `standalone-ha.xml`, or `domain.xml` for configuration. We may soon remove support for keycloak-server.json.

To convert your keycloak-server.json, you will use a jboss-cli operation called `migrate-json`. It is recommended that you run this operation while the server is not running.

The `migrate-json` operation assumes you are migrating with an xml configuration file from an old version. For example, you should not try to migrate your keycloak-server.json to a standalone.xml file that already contains `<spi>` and `<theme>` tags under the keycloak subsystem. Before migration, your keycloak subsystem should look like the one below:

```xml
<subsystem xmlns="urn:jboss:domain:keycloak-server:1.1">  
  <web-context>auth</web-context>
</subsystem>
```

The jboss-cli tool is discussed in detail in Server Installation and Configuration.
migrate-json in Standalone Mode

For standalone, you will issue the `migrate-json` operation in embed mode without the server running.

```
$ .../bin/jboss-cli.sh
[disconnected /] embed-server --server-config=standalone.xml
[standalone@embedded /] /subsystem=keycloak-server/:migrate-json
```

The `migrate-json` operation will look for your `keycloak-server.json` file in the `standalone/configuration` directory. You also have the option of using the `file` argument as shown in the domain mode example below.

migrate-json in Domain Mode

For a domain, you will stop the Keycloak server and issue the `migrate-json` operation against the running domain controller. If you choose not to stop the Keycloak server, the operation will still work, but your changes will not take effect until the Keycloak server is restarted.

Domain mode migration requires that you use the `file` parameter to upload your `keycloak-server.json` from a local directory. The example below shows connecting to localhost. You will need to substitute the address of your domain controller.

```
$ .../bin/jboss-cli.sh -c --controller=localhost:9990
[domain@localhost:9990 /] cd profile=auth-server-clustered
[domain@localhost:9990 profile=auth-server-clustered] cd subsystem=keycloak-server
[domain@localhost:9990 subsystem=keycloak-server] :migrate-json(file="./keycloak-server.json")
```

You will need to repeat the `migrate-json` operation for each profile containing a `keycloak-server` subsystem.

Migrate database

Keycloak can automatically migrate the database schema, or you can choose to do it manually.

Relational database

To enable automatic upgrading of the database schema set the `migrationStrategy` property to `update` for the default `connectionsJpa` provider:

```
<spi name="connectionsJpa">
  <provider name="default" enabled="true">
    <properties>
      ...
      <property name="migrationStrategy" value="update"/>
    </properties>
  </provider>
</spi>
```

```
/subsystem=keycloak-server/spi=connectionsJpa/provider=default/:map-put(name=properties,key=migrationStrategy,value=update)
```

When you start the server with this setting your database will automatically be migrated if the database schema has changed in the new version.
To manually migrate the database set the `migrationStrategy` to `manual`. When you start the server with this configuration it will check if the database needs migration. If changes are needed the required changes are written to an SQL file that you can review and manually run against the database.

There’s also the option to disable migration by setting the `migrationStrategy` to `validate`. With this configuration the database will be checked at startup and if it is not migrated the server will exit.

**Mongo**

Mongo doesn’t have a schema, but there may still be things like collections and indexes that are added to new releases. To enable automatic creation of these set the `migrationStrategy` property to `update` for the default `connectionsMongo` provider:

```xml
<spi name="connectionsMongo">
  <provider name="default" enabled="true">
    <properties>
      ...
      <property name="migrationStrategy" value="update"/>
    </properties>
  </provider>
</spi>
```

The `Mongo` provider does not have the option to manually apply the required changes.

There’s also the option to disable migration by setting the `migrationStrategy` to `validate`. With this configuration the database will be checked at startup and if it is not migrated the server will exit.

**Migrate providers**

If you have implemented any SPI providers you need to copy them to the new server. The version specific section below will mention if any of the SPI’s have changed. If they have you may have to update your code accordingly.

**Migrate themes**

If you have created a custom theme you need to copy them to the new server. The version specific section below will mention if changes have been made to themes. If there is you may have to update your themes accordingly.

**Migrate application**

If you deploy applications directly to the Keycloak server you should copy them to the new server. For any applications including those not deployed directly to the Keycloak server you should upgrade the adapter. The version specific section below will mention if any changes are required to applications.
Version specific migration

Migrating to 3.2.0

New Password Hashing algorithms

We’ve added two new password hashing algorithms (pbkdf2-sha256 and pbkdf2-sha512). New realms will use the pbkdf2-sha256 hashing algorithm with 27500 hashing iterations. Since pbkdf2-sha256 is slightly faster than pbkdf2 the iterations was increased to 27500 from 20000.

Existing realms are upgraded if the password policy contains the default value for hashing algorithm (not specified) and iteration (20000). If you have changed the hashing iterations you need to manually change to pbkdf2-sha256 if you’d like to use the more secure hashing algorithm.

ID Token requires scope=openid

OpenID Connect specification requires that parameter scope with value openid is used in initial authorization request. So in Keycloak 2.1.0 we changed our adapters to use scope=openid in the redirect URI to Keycloak. Now we changed the server part too and ID token will be sent to the application just if scope=openid is really used. If it’s not used, then ID token will be skipped and just Access token and Refresh token will be sent to the application. This also allows that you can omit the generation of ID Token on purpose - for example for space or performance purposes.

Direct grants (OAuth2 Resource Owner Password Credentials Grant) and Service accounts login (OAuth2 Client credentials grant) also don’t use ID Token by default now. You need to explicitly add scope=openid parameter to have ID Token included.

Authentication sessions and Action tokens

We are working on support for multiple datacenters. As the initial step, we introduced authentication session and action tokens. Authentication session replaces Client session, which was used in previous versions. Action tokens are currently used especially for the scenarios, where the authenticator or requiredActionProvider requires sending email to the user and requires user to click on the link in email.

Here are concrete changes related to this, which may affect you for the migration.

First change related to this is introducing new infinispan caches authenticationSessions and actionTokens in standalone.xml or standalone-ha.xml. If you use our migration CLI, you don’t need to care much as your standalone(-ha).xml files will be migrated automatically.

Second change is changing of some signatures in methods of authentication SPI. This may affect you if you use custom Authenticator or RequiredActionProvider. Classes AuthenticationFlowContext and RequiredActionContext now allow to retrieve authentication session instead of client session.

We also added some initial support for sticky sessions. You may want to change your loadbalancer/proxy server and configure it if you want to suffer from it and have better performance. The route is added to the new AUTH_SESSION_ID cookie. More info in the clustering documentation.

Another change is, that token.getClientSession() was removed. This may affect you for example if you’re using Client Initiated Identity Broker Linking feature.

The ScriptBasedAuthenticator changed the binding name from clientSession to authenticationSession, so you would need to update your scripts if you’re using this authenticator.

Finally we added some new timeouts to the admin console. This allows you for example to specify different timeouts for the email actions triggered by admin and by user himself.
Migrating to 2.5.1

Migration of old offline tokens

If you migrate from version 2.2.0 or older and you used offline tokens, then your offline tokens didn’t have KID in the token header. We added KID to the token header in 2.3.0 together with the ability to have multiple realm keys, so Keycloak is able to find the correct key based on the token KID.

For the offline tokens without KID, Keycloak 2.5.1 will always use the active realm key to find the proper key for the token verification. In other words, migration of old offline tokens will work. So for example, your user requested offline token in 1.9.8, then you migrate from 1.9.8 to 2.5.1 and then your user will be still able to refresh his old offline token in 2.5.1 version.

But there is limitation, that once you change the realm active key, the users won’t be able to refresh old offline tokens anymore. So you shouldn’t change the active realm key until all your users with offline tokens refreshed their tokens. Obviously newly refreshed tokens will have KID in the header, so after all users exchange their old offline tokens, you are free to change the active realm key.

Migrating to 2.5.0

Changes to the infinispan caches

The realms cache defined in the infinispan subsystem in standalone.xml or standalone-ha.xml configuration file, now has the eviction with the 10000 records by default. This is the same default like the users cache.

Also the authorization cache now doesn’t have any eviction on it by default.

Migrating to 2.4.0

Server SPI split into Server SPI and Sever SPI Private

The keycloak-server-spi module has been split into keycloak-server-spi and keycloak-server-spi-private. APIs within keycloak-server-spi will not change between minor releases, while we reserve the right and may quite likely change APIs in keycloak-server-spi-private between minor releases.

Key encryption algorithm in SAML assertions

Key in SAML assertions and documents are now encrypted using RSA-OAEP encryption scheme. If you want to use encrypted assertions, make sure that service providers understand this encryption scheme. In the unlikely case that SP would not be able to handle the new scheme, Keycloak can be made to use legacy RSA-v1.5 encryption scheme when started with system property keycloak.saml.key_trans.rsa_v1.5 set to true.

Infinispan caches realms and users are always local

Even if you use Keycloak in cluster, the caches realms and users defined in infinispan subsystem in standalone-ha.xml are always local caches now. There is separate cache work, which handles sending invalidation messages between cluster nodes and informing whole cluster what records in underlying realms and users caches should be invalidated.
Migrating to 2.3.0

Default max results on paginated endpoints

All Admin REST API endpoints that support pagination now have a default max results set to 100. If you want to return more than 100 entries you need to explicitly specify that with \texttt{?max=<RESULTS>}.

\texttt{realm-public-key} adapter property not recommended

In 2.3.0 release we added support for Public Key Rotation. When admin rotates the realm keys in Keycloak admin console, the Client Adapter will be able to recognize it and automatically download new public key from Keycloak. However this automatic download of new keys is done just if you don’t have \texttt{realm-public-key} option in your adapter with the hardcoded public key. For this reason, we don’t recommend to use \texttt{realm-public-key} option in adapter configuration anymore.

Note this option is still supported, but it may be useful just if you really want to have hardcoded public key in your adapter configuration and never download the public key from Keycloak. In theory, one reason for this can be to avoid man-in-the-middle attack if you have untrusted network between adapter and Keycloak, however in that case, it is much better option to use HTTPS, which will secure all the requests between adapter and Keycloak.

Added infinispan cache keys

In this release, we added new cache keys to the infinispan subsystem, which is defined in standalone.xml or standalone-ha.xml configuration file. It has also some eviction and expiration defined. This cache is internally used for caching the external public keys of the entities trusted by the server (Identity providers or clients, which uses authentication with signed JWT).

Migrating to 2.2.0

databaseSchema property deprecated

The \texttt{databaseSchema} property for both JPA and Mongo is now deprecated and has been replaced by \texttt{initializeEmpty} and \texttt{migrationStrategy}. \texttt{initializeEmpty} can be set to \texttt{true} or \texttt{false} and controls if an empty database should be initialized. \texttt{migrationStrategy} can be set to update, validate and manual. \texttt{manual} is only supported for relational databases and will write an SQL file with the required changes to the database schema. Please note that for Oracle database, the created SQL file contains \texttt{SET DEFINE OFF} command understood by Oracle SQL clients. Should the script be consumed by any other client, please replace the lines with equivalent command of the tool of your choice that disables variable expansion or remove it completely if such functionality is not applicable.

Changes in Client’s Valid Redirect URIs

The following scenarios are affected:

- When a Valid Redirect URI with query component is saved in a Client (e.g. \texttt{http://localhost/auth? foo=bar}), \texttt{redirect_uri} in authorization request must exactly match this URI (or other registered URI in this Client).
- When a Valid Redirect URI without a query component is saved in a Client, \texttt{redirect_uri} must exactly match as well.
• Wildcards in registered Valid Redirect URIs are no longer supported when query component is present in this URI, so the redirect_uri needs to exactly match this saved URI as well.
• Fragments in registered Valid Redirect URIs (like `http://localhost/auth#fragment`) are no longer allowed.

**Authenticate by default removed from Identity Providers**

Identity providers no longer has an option to set it as a default authentication provider. Instead go to Authentication, select the **Browser** flow and configure the **Identity Provider Redirector**. It has an option to set the default identity provider.

**Migrating to 2.0.0**

**Upgrading from 1.0.0.Final no longer supported**

Upgrading from 1.0.0.Final is no longer supported. To upgrade to this version upgrade to 1.9.8.Final prior to upgrading to 2.0.0.

**Migrating to 1.9.5**

**Default password hashing interval increased to 20K**

The default password hashing interval for new realms has been increased to 20K (from 1 previously). This change will have an impact on performance when users authenticate. For example with the old default (1) it takes less than 1 ms to hash a password, but with the new default (20K) the same operation can take 50-100 ms.

**Migrating to 1.9.3**

**Add User script renamed**

The script to add admin users to Keycloak has been renamed to `add-user-keycloak`.

**Migrating to 1.9.2**

**Adapter option auth-server-url-for-backend-requests removed**

We’ve removed the option auth-server-url-for-backend-requests as there were issues in some scenarios when it was used. In more details, it was possible to access the Keycloak server from 2 different contexts (internal and external), which was causing issues in token validations etc.

If you still want to use the optimization of network, which this option provided (avoid the application to send backchannel requests through loadbalancer but send them to local Keycloak server directly) you may need to handle it at hosts configuration (DNS) level.
Migrating to 1.9.0

Themes and providers directory moved

We’ve moved the themes and providers directories from standalone/configuration/themes and standalone/configuration/providers to themes and providers respectively. If you have added custom themes and providers you need to move them to the new location. You also need to update keycloak-server.json as it’s changed due to this.

Adapter Subsystems only bring in dependencies if Keycloak is on

Previously, if you had installed our SAML or OIDC Keycloak subsystem adapters into Wildfly or JBoss EAP, we would automatically include Keycloak client jars into EVERY application irregardless if you were using Keycloak or not. These libraries are now only added to your deployment if you have Keycloak authentication turned on for that adapter (via the subsystem, or auth-method in web.xml).

Client Registration service endpoints moved

The Client Registration service endpoints have been moved from {realm}/clients to {realm}/clients-registrations.

Session state parameter in authentication response renamed

In the OpenID Connect authentication response we used to return the session state as session-state this is not correct according to the specification and has been renamed to session_state.

Deprecated OpenID Connect endpoints

In 1.2 we deprecated a number of endpoints that where not consistent with the OpenID Connect specifications, these have now been removed. This also applies to the validate token endpoints that was replaced with the new introspect endpoint in 1.8.

Updates to theme templates

Feedback in template.ftl has been moved and format has changed slightly.

Module and Source Code Re-org

Most of our modules and source code have been consolidated into two maven modules: keycloak-server-spi and keycloak-services. SPI interfaces are in server-spi, implementations are in keycloak-services. All JPA dependent modules have been consolidated under keycloak-model-jpa. Same goes with mongo and Infinispan under modules keycloak-model-mongo and keycloak-model-infinispan.
For adapters, session id changed after login

To plug a security attack vector, for platforms that support it (Tomcat 8, Undertow/Wildfly, Jetty 9), the Keycloak OIDC and SAML adapters will change the session id after login. You can turn off this behavior check adapter config switches.

SAML SP Client Adapter Changes

Keycloak SAML SP Client Adapter now requires a specific endpoint, /saml to be registered with your IDP.

Migrating to 1.8.0

Admin account

In previous releases we shipped with a default admin user with a default password, this has now been removed. If you are doing a new installation of 1.8 you will have to create an admin user as a first step. This can be done easily by following the steps in Admin User.

OAuth2 Token Introspection

In order to add more compliance with OAuth2 specification, we added a new endpoint for token introspection. The new endpoint can reached at /realms/{realm}/protocols/openid-connect/token/introspect and it is solely based on RFC-7662.

The /realms/{realm}/protocols/openid-connect/validate endpoint is now deprecated and we strongly recommend you to move to the new introspection endpoint as soon as possible. The reason for this change is that RFC-7662 provides a more standard and secure introspection endpoint.

The new token introspection URL can now be obtained from OpenID Connect Provider’s configuration at /realms/{realm}/.well-known/openid-configuration. There you will find a claim with name token_introspection_endpoint within the response. Only confidential clients are allowed to invoke the new endpoint, where these clients will be usually acting as a resource server and looking for token metadata in order to perform local authorization checks.

Migrating to 1.7.0.CR1

Direct access grants disabled by default for clients

In order to add more compliance with OpenID Connect specification, we added flags into admin console to Client Settings page, where you can enable/disable various kinds of OpenID Connect/OAuth2 flows (Standard flow, Implicit flow, Direct Access Grants, Service Accounts). As part of this, we have Direct Access Grants (corresponds to OAuth2 Resource Owner Password Credentials Grant) disabled by default for new clients.

Clients migrated from previous version have Direct Access Grants enabled just if they had flag Direct Grants Only on. The Direct Grants Only flag was removed as if you enable Direct Access Grants and disable both Standard+Implicit flow, you will achieve same effect.

We also added built-in client admin-cli to each realm. This client has Direct Access Grants enabled. So if you’re using Admin REST API or Keycloak admin-client, you should update your configuration to use admin-cli instead of security-admin-console as the latter one doesn’t have direct access grants enabled anymore by default.
Option *Update Profile On First Login* moved from Identity provider to Review Profile authenticator

In this version, we added *First Broker Login*, which allows you to specify what exactly should be done when new user is logged through Identity provider (or Social provider), but there is no existing Keycloak user yet linked to the social account. As part of this work, we added option *First Login Flow* to identity providers where you can specify the flow and then you can configure this flow under *Authentication* tab in admin console.

We also removed the option *Update Profile On First Login* from the Identity provider settings and moved it to the configuration of *Review Profile* authenticator. So once you specify which flow should be used for your Identity provider (by default it’s *First Broker Login* flow), you go to *Authentication* tab, select the flow and then you configure the option under *Review Profile* authenticator.

Element *form-error-page* in web.xml not supported anymore

form-error-page in web.xml will no longer work for client adapter authentication errors. You must define an error-page for the various HTTP error codes. See documentation for more details if you want to catch and handle adapter error conditions.

IdentityProviderMapper changes

There is no change in the interface itself or method signatures. However there is some change in behavior. We added *First Broker Login flow* in this release and the method `IdentityProviderMapper.importNewUser` is now called after *First Broker Login flow* is finished. So if you want to have any attribute available in *Review Profile* page, you would need to use the method `preprocessFederatedIdentity` instead of `importNewUser`. You can set any attribute by invoke `BrokeredIdentityContext.setUserAttribute` and that will be available on *Review profile* page.

Migrating to 1.6.0.Final

Option that refresh tokens are not reusable anymore

Old versions of Keycloak allowed reusing refresh tokens multiple times. Keycloak still permits this, but also have an option *Revoke refresh token* to disallow it. Option is in in admin console under token settings. When a refresh token is used to obtain a new access token a new refresh token is also included. When option is enabled, then this new refresh token should be used next time the access token is refreshed. It won’t be possible to reuse old refresh token multiple times.

Some packages renamed

We did a bit of restructure and renamed some packages. It is mainly about renaming internal packages of util classes. The most important classes used in your application (for example `AccessToken` or `KeycloakSecurityContext`) as well as the SPI are still unchanged. However there is slight chance that you will be affected and will need to update imports of your classes. For example if you are using multiitenancy and `KeycloakConfigResolver`, you will be affected as for example class `HttpFacade` was moved to different package - it is `org.keycloak.adapters.spi.HttpFacade` now.
Persisting user sessions

We added support for offline tokens in this release, which means that we are persisting “offline” user sessions into database now. If you are not using offline tokens, nothing will be persisted for you, so you don’t need to care about worse performance for more DB writes. However in all cases, you will need to update standalone/configuration/keycloak-server.json and add userSessionPersister like this:

```
"userSessionPersister": {
    "provider": "jpa"
},
```

If you want sessions to be persisted in Mongo instead of classic RDBMS, use provider `mongo` instead.

Migrating to 1.5.0.Final

Realm and User cache providers

Infinispan is now the default and only realm and user cache providers. In non-clustered mode a local Infinispan cache is used. We’ve also removed our custom in-memory cache and the no cache providers. If you have realmCache or userCache set in keycloak-server.json to mem or none please remove these. As Infinispan is the only provider there’s no longer any need for the realmCache and userCache objects so these can be removed.

Uses Session providers

Infinispan is now the default and only user session provider. In non-clustered mode a local Infinispan cache is used. We’ve also removed the JPA and Mongo user session providers. If you have userSession set in keycloak-server.json to mem, jpa or mongo please remove it. As Infinispan is the only provider there’s no longer any need for the userSession object so it can be removed.

For anyone that wants to achieve increased durability of user sessions this can be achieved by configuring the user session cache with more than one owner or use a replicated cache. It’s also possible to configure Infinispan to persist caches, although that would have impacts on performance.

Contact details removed from registration and account management

In the default theme we have now removed the contact details from the registration page and account management. The admin console now lists all the users attributes, not just contact specific attributes. The admin console also has the ability to add/remove attributes to a user. If you want to add contact details, please refer to the address theme included in the examples.

Migrating to 1.3.0.Final

Direct Grant API always enabled

In the past Direct Grant API (or Resource Owner Password Credentials) was disabled by default and there was an option on a realm to enable it. The Direct Grant API is now always enabled and the option to enable/disable for a realm is removed.
Database changed

There are again few database changes. Remember to backup your database prior to upgrading.

UserFederationProvider changed

There are few minor changes in UserFederationProvider interface. You may need to sync your implementation when upgrade to newer version and upgrade few methods, which has changed signature. Changes are really minor, but were needed to improve performance of federation.

WildFly 9.0.0.Final

Following on from the distribution changes that was done in the last release the standalone download of Keycloak is now based on WildFly 9.0.0.Final. This also affects the overlay which can only be deployed to WildFly 9.0.0.Final or JBoss EAP 6.4.0.GA. WildFly 8.2.0.Final is no longer supported for the server.

WildFly, JBoss EAP and JBoss AS7 adapters

There are now 3 separate adapter downloads for WildFly, JBoss EAP and JBoss AS7:

- eap6
- wf9
- wf8
- as7

Make sure you grab the correct one.

You also need to update standalone.xml as the extension module and subsystem definition has changed. See Adapter Installation for details.

Migrating from 1.2.0.Beta1 to 1.2.0.RC1

Distribution changes

Keycloak is now available in 3 downloads: standalone, overlay and demo bundle. The standalone is intended for production and non-JEE developers. Overlay is aimed at adding Keycloak to an existing WildFly 8.2 or EAP 6.4 installation and is mainly for development. Finally we have a demo (or dev) bundle that is aimed at developers getting started with Keycloak. This bundle contains a WildFly server, with Keycloak server and adapter included. It also contains all documentation and examples.

Database changed

This release contains again a number of changes to the database. The biggest one is Application and OAuth client merge. Remember to backup your database prior to upgrading.
Application and OAuth client merge

Application and OAuth clients are now merged into Clients. The UI of admin console is updated and database as well. Your data from database should be automatically updated. The previously set Applications will be converted into Clients with Consent required switch off and OAuth Clients will be converted into Clients with this switch on.

Migrating from 1.1.0.Final to 1.2.0.Beta1

Database changed

This release contains a number of changes to the database. Remember to backup your database prior to upgrading.

iss in access and id tokens

The value of iss claim in access and id tokens have changed from realm name to realm url. This is required by OpenID Connect specification. If you’re using our adapters there’s no change required, other than if you’ve been using bearer-only without specifying auth-server-url you have to add it now. If you’re using another library (or RSATokenVerifier) you need to make the corresponding changes when verifying iss.

OpenID Connect endpoints

To comply with OpenID Connect specification the authentication and token endpoints have been changed to having a single authentication endpoint and a single token endpoint. As per-spec response_type and grant_type parameters are used to select the required flow. The old endpoints (/realms/{realm}/protocols/openid-connect/login, /realms/{realm}/protocols/openid-connect/grants/access, /realms/{realm}/protocols/openid-connect/refresh, /realms/{realm}/protocols/openid-connect/access/codes) are now deprecated and will be removed in a future version.

Theme changes

The layout of themes have changed. The directory hierarchy used to be type/name this is now changed to name/type. For example a login theme named sunrise used to be deployed to standalone/configuration/themes/login/sunrise, which is now moved to standalone/configuration/themes/sunrise/login. This change was done to make it easier to have group the different types for the same theme into one folder.

If you deployed themes as a JAR in the past you had to create a custom theme loader which required Java code. This has been simplified to only requiring a plain text file (META-INF/keycloak-themes.json) to describe the themes included in a JAR. See the themes section in the docs for more information.

Claims changes

Previously there was Claims tab in admin console for application and OAuth clients. This was used to configure which attributes should go into access token for particular application/client. This was removed and replaced with Protocol mappers, which are more flexible.
You don’t need to care about migration of database from previous version. We did migration scripts for both RDBMS and Mongo, which should ensure that claims configured for particular application/client will be converted into corresponding protocol mappers (Still it’s safer to backup DB before migrating to newer version though). Same applies for exported JSON representation from previous version.

**Social migration to identity brokering**

We refactored social providers SPI and replaced it with *identity brokering SPI*, which is more flexible. The Social tab in admin console is renamed to Identity Provider tab.

Again you don’t need to care about migration of database from previous version similarly like for Claims/protocol mappers. Both configuration of social providers and “social links” to your users will be converted to corresponding Identity providers.

Only required action from you would be to change allowed Redirect URI in the admin console of particular 3rd party social providers. You can first go to the Keycloak admin console and copy Redirect URI from the page where you configure the identity provider. Then you can simply paste this as allowed Redirect URI to the admin console of 3rd party provider (IE. Facebook admin console).

**Migrating from 1.1.0.Beta2 to 1.1.0.Final**

- WEB-INF/lib +standalone/configuration/providers+providers

**Migrating from 1.1.0.Beta1 to 1.1.0.Beta2**

- Adapters are now a separate download. They are not included in appliance and war distribution. We have too many now and the distro is getting bloated.
  KeycloakAuthenticatorValve
- JavaScript adapter now has idToken and idTokenParsed properties. If you use idToken to retrieve first name, email, etc. you need to change this to idTokenParsed.
- The as7-eap-subsystem and keycloak-wildfly-subsystem have been merged into one keycloak-subsystem. If you have an existing standalone.xml or domain.xml, you will need edit near the top of the file and change the extension module name to org.keycloak.keycloak-subsystem. For AS7 only, the extension module name is org.keycloak.keycloak-as7-subsystem.
- Server installation is no longer supported on AS7. You can still use AS7 as an application client.

**Migrating from 1.0.x.Final to 1.1.0.Beta1**

- RealmModel JPA and Mongo storage schema has changed
- UserSessionModel JPA and Mongo storage schema has changed as these interfaces have been refactored
- Upgrade your adapters, old adapters are not compatible with Keycloak 1.1. We interpreted JSON Web Token and OIDC ID Token specification incorrectly. **aud** claim must be the client id, we were storing the realm name in there and validating it.
**Migrating from 1.0 RC-1 to RC-2**

- A lot of info level logging has been changed to debug. Also, a realm no longer has the jboss-logging audit listener by default. If you want log output when users login, logout, change passwords, etc. enable the jboss-logging audit listener through the admin console.

**Migrating from 1.0 Beta 4 to RC-1**

- logout REST API has been refactored. The GET request on the logout URI does not take a session_state parameter anymore. You must be logged in in order to log out the session. You can also POST to the logout REST URI. This action requires a valid refresh token to perform the logout. The signature is the same as refresh token minus the grant type form parameter. See documentation for details.

**Migrating from 1.0 Beta 1 to Beta 4**

- LDAP/AD configuration is changed. It is no longer under the “Settings” page. It is now under Users→Federation. Add Provider will show you an “ldap” option.
- Authentication SPI has been removed and rewritten. The new SPI is UserFederationProvider and is more flexible.
- ssl-not-required +ssl-required +all +external +none
- DB Schema has changed again.
- Created applications now have a full scope by default. This means that you don’t have to configure the scope of an application if you don’t want to.
- Format of JSON file for importing realm data was changed. Now role mappings is available under the JSON record of particular user.

**Migrating from 1.0 Alpha 4 to Beta 1**

- DB Schema has changed. We have added export of the database to Beta 1, but not the ability to import the database from older versions. This will be supported in future releases.
- For all clients except bearer-only applications, you must specify at least one redirect URI. Keycloak will not allow you to log in unless you have specified a valid redirect URI for that application.
- Direct Grant API +ON
- standalone/configuration/keycloak-server.json
- JavaScript adapter
- Session Timeout

**Migrating from 1.0 Alpha 2 to Alpha 3**

- SkeletonKeyToken, SkeletonKeyScope, SkeletonKeyPrincipal, and SkeletonKeySession have been renamed to: AccessToken, AccessScope, KeycloakPrincipal, and KeycloakAuthenticatedSession respectively.
- ServleOAuthClient.getBearerToken() method signature has changed. It now returns an AccessTokenResponse so that you can obtain a refresh token too.
Adapters now check the access token expiration with every request. If the token is expired, they will attempt to invoke a refresh on the auth server using a saved refresh token.

Subject in AccessToken has been changed to the User ID.

Migrating from 1.0 Alpha 1 to Alpha 2

- DB Schema has changed. We don’t have any data migration utilities yet as of Alpha 2.
- JBoss and Wildfly adapters are now installed via a JBoss/Wildfly subsystem. Please review the adapter installation documentation. Edits to standalone.xml are now required.
- There is a new credential type “secret”. Unlike other credential types, it is stored in plain text in the database and can be viewed in the admin console.
- There is no longer required Application or OAuth Client credentials. These client types are now hard coded to use the “secret” credential type.
- Because of the “secret” credential change to Application and OAuth Client, you’ll have to update your keycloak.json configuration files and regenerate a secret within the Application or OAuth Client credentials tab in the administration console.

You can manage users in Anaconda Enterprise either by using your own Identity Provider or managing users from Anaconda Enterprise.

Using your own Identity Provider

- Connecting to your Identity Provider
- Configuring your Identity Provider

Managing users from Anaconda Enterprise

- Creating new users
  - Creating a new group
- Managing groups and roles

1.2.5 Admin Cheatsheet

Coming soon!

1.2.6 Troubleshooting

Coming soon!

1.3 Release notes

Anaconda Enterprise version 5.0.1 was released for the Anaconda Enterprise Innovator Program on March 8, 2017 and includes the following:
• Simplified, one-click deployment of data science projects and apps, 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.

1.4 Command reference - coming soon

Coming soon!

1.5 Glossary

1.5.1 Anaconda Enterprise Innovator Program
An early-adopter program that includes a preinstalled version of the Anaconda Enterprise platform for users to try using a public sandbox. All files on the system are public and visible to other users.

1.5.2 Application or app
A project that has been deployed.

1.5.3 Deployment
All the processes involved in getting a new software package up and running properly in its environment, including installing, configuring, running, testing and making necessary changes.

1.5.4 Projects
A folder that contains a configuration file together with data science scripts, notebooks and other files. Data scientists use projects to encapsulate data science projects and make them easily portable.

1.6 Help and support

To obtain help and support from within Anaconda Enterprise:

1. In the top right navigation bar, click the Help icon:
2. Select any of the following:
   - Documentation–A link to this documentation.
   - Getting Started with Anaconda Enterprise–a tutorial.
   - Support–A link to our support system.
   - Feedback–A link to our feedback system.

During the Innovator Program:
   - Get email support from enterprise_innovator@continuum.io.
   - Provide feedback by completing this 5-minute survey.

### 1.7 Early Adopter Program EULA

This Early Adopter Program Software End User License Agreement (“Agreement” or EULA”) is a binding legal agreement between you (“you” or “Licensee”) and Continuum Analytics, Inc. (“Continuum”) and governs Licensee’s access and use of the Early Adopter Program Software (defined below).

**BY CLICKING “I AGREE” OR OTHERWISE ACCESSING OR USING THE EARLY ADOPTER PROGRAM SOFTWARE, YOU AGREE THAT YOU HAVE READ, UNDERSTOOD, AND AGREE TO BE BOUND BY THIS AGREEMENT. IF YOU ARE USING THE EARLY ADOPTER PROGRAM SOFTWARE ON BEHALF OF A COMPANY, ENTITY, OR ORGANIZATION, THEN YOU REPRESENT AND WARRANT THAT YOU: (i) ARE AN AUTHORIZED REPRESENTATIVE OF THAT ENTITY WITH THE AUTHORITY TO BIND THAT ENTITY TO THIS AGREEMENT; AND (ii) AGREE TO BE BOUND BY THIS AGREEMENT ON BEHALF OF THAT ENTITY. IF YOU DO NOT AGREE, THEN YOU MAY NOT USE THE EARLY ADOPTER PROGRAM SOFTWARE.**

1. LICENSE. Subject to Licensee’s compliance with the terms and conditions of this Agreement, Continuum grants to Licensee a limited, nonexclusive, and revocable license to install and use one copy of the software Continuum provides to Licensee in connection with this Agreement (the “Early Adopter Program Software”) during the Term (defined below) in object-code form only, solely for the purpose of allowing Licensee to evaluate the Early Adopter Program Software. Any replacements, additions, or modifications to the Early Adopter Program Software provided to Licensee are included within the term “Early Adopter Program Software” and are subject to the terms of this Agreement. Licensee will provide, at its own expense, all equipment and third party software necessary to evaluate the Early Adopter Program Software.

2. TERM. This Agreement will begin on the date the Early Adopter Program Software is first made available by Continuum to Licensee for download and will continue until September 30, 2017 (“Term”), after which, this Agreement will automatically terminate. Either party may terminate this Agreement at any time with notice, including if the other
party materially breaches any provision of this Agreement. Upon the expiration or termination of this Agreement
the license granted in Section 1 will immediately terminate; and Licensee will uninstall and delete the Early Adopter
Program Software. Sections 2 and 3 - 8 will survive termination or expiration of this Agreement.

3. RESTRICTIONS. Except as otherwise explicitly provided in this Agreement or as may be expressly permitted by
applicable law, Licensee will not, and will not permit, encourage, or authorize third parties to: (a) modify, disassemble,
decompile, reverse engineer, rent, lease, loan, transfer, or copy any portion of the Early Adopter Program Software;
(b) circumvent or disable any security or other technological features of the Early Adopter Program Software; (c)
distribute, publicly display, publicly perform, or make the Early Adopter Program Software available to any third
parties; (d) use for competitive analysis, to guide or engineer competing or similar offering product or services; nor (e)
exceed any limitations described in any documentation relating to the Early Adopter Program Software or otherwise
communicated to Licensee by Continuum.

4. OWNERSHIP. Continuum owns all right, title, and interest, including all intellectual property rights, in and to
the Early Adopter Program Software, including any improvements, modifications, and enhancements to the Early
Adopter Program Software. Except for those rights expressly granted in this Agreement, no other rights are granted,
either express or implied, to Licensee.

5. FEEDBACK. Licensee has no obligation to provide Continuum any suggestions, comments or other feedback re-
garding the Early Adopter Program Software (“Feedback”). If Licensee nonetheless provides Feedback to Continuum
on any version or part of the Early Adopter Program Software, Licensee hereby assigns to Continuum all right, title,
and interest in and to the Feedback.

6. USAGE DATA COLLECTION. Continuum may collect certain non-personal usage data information such as error
tracebacks, execution history, and the like, solely for the purpose of bug analysis and feature enhancement. Continuum
will make every reasonable attempt to anonymize any Personally Identifiable Information (“PII”) collected in the
course of gathering such data.

7. WARRANTY DISCLAIMER. THE EARLY ADOPTER PROGRAM SOFTWARE IS A BETA PROGRAM PRO-
VIDED “AS IS” FOR LIMITED EVALUATION ONLY, AND CONTINUUM DOES NOT WARRANT THAT SAID
SOFTWARE WILL OPERATE WITHOUT ERROR OR INTERRUPTION. CONTINUUM SPECIFICALLY DIS-
CLAIMS ALL OTHER WARRANTIES, EXPRESS, IMPLIED (EITHER IN FACT OR BY OPERATION OF LAW),
OR STATUTORY, AS TO ANY MATTER WHATSOEVER. CONTINUUM EXPRESSLY DISCLAIMS ALL IM-
PLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUALITY, AC-
CURACY, TITLE, AND NON-INFRINGEMENT. CONTINUUM DOES NOT WARRANT AGAINST INTERFER-
ENCE WITH THE ENJOYMENT OF THE EARLY ADOPTER PROGRAM SOFTWARE. CONTINUUM DOES
NOT WARRANT THAT THE EARLY ADOPTER PROGRAM SOFTWARE IS ERROR-FREE OR THAT OP-
ERATION OF THE EARLY ADOPTER PROGRAM SOFTWARE WILL BE SECURE OR UNINTERRUPTED.
CONTINUUM DOES NOT WARRANT THAT ANY INFORMATION PROVIDED THROUGH THE EARLY
ADOPTER PROGRAM SOFTWARE IS ACCURATE OR COMPLETE OR THAT ANY INFORMATION PRO-
VIDED THROUGH THE EARLY ADOPTER PROGRAM SOFTWARE WILL ALWAYS BE AVAILABLE. CON-
INUUM EXERCISES NO CONTROL OVER AND EXPRESSLY DISCLAIMS ANY LIABILITY ARISING OUT
OF OR BASED UPON THE RESULTS OF LICENSEE’S USE OF THE EARLY ADOPTER PROGRAM SOFT-
WARE.

8. LIMITATION OF LIABILITY. CONTINUUM WILL NOT BE LIABLE TO LICENSEE OR ANY THIRD
PARTY FOR ANY INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES ARIS-
ING FROM OR OTHERWIE RELATED TO THIS AGREEMENT, INCLUDING BUT NOT LIMITED TO LOST
PROFITS, LOSS OF BUSINESS, LOSS OF GOODWILL, OR ANTICIPATED SAVINGS, EVEN IF CONTIN-
UUM HAS BEEN ADVISED OF THE POSSIBILITY OF THESE DAMAGES. UNDER NO CIRCUMSTANCES
WILL CONTINUUM’S TOTAL LIABILITY OF ANY OR ALL KINDS ARISING OUT OF OR RELATED TO
THIS AGREEMENT, REGARDLESS OF THE FORUM AND REGARDLESS OF WHETHER ANY ACTION OR
CLAIM IS BASED ON CONTRACT, TORT, OR OTHERWISE, EXCEED $100. THE LIMITATIONS SET FORTH
IN THIS AGREEMENT WILL APPLY EVEN IF ANY LIMITED REMEDY IS FOUND TO HAVE FAILED ITS
ESSENTIAL PURPOSE.

9. MISCELLANEOUS. This Agreement will be governed by the laws of the State of Texas without reference to
conflict of law principles. All disputes arising out of or related to this Agreement, will be subject to the exclusive jurisdiction of the state and federal courts located in Travis County, Texas, and the parties agree and submit to the personal and exclusive jurisdiction and venue of these courts. Licensee warrants that it will comply with all laws in connection with its performance of this Agreement, including applicable export laws and restrictions. Licensee will not assign this Agreement, directly or indirectly, by operation of law or otherwise, without the prior written consent of Continuum. Any assignment in violation of this Agreement is void. This Agreement binds and inures to the benefit of the parties and their successors and permitted assigns. This is the entire agreement between the parties relating to the subject matter hereof. No waiver or modification of this Agreement will be valid unless in writing signed by each party. If any part of this Agreement is found to be illegal, unenforceable, or invalid, the remaining portions of this Agreement will remain in full force and effect. The terms of any other document that conflicts with, or in any way purports to amend, any of the terms of this Agreement are hereby specifically objected to and will be of no force or effect.