Open for Innovation
KNIME

KNIME Business Hub Admin Guide

KNIME AG, Zurich, Switzerland
Version 1.12 (last updated on 2025-04-02)

Table of Contents

Introduction

KNIME Business Hub is a customer-managed Hub instance. Once you have a license for it and proceed with installation you will have access to Hub resources and will be able to customize specific features, as well as give access to these resources to your employees, organize them into Teams and give them the ability to manage specific resources.

This guide provides information on how to administrate a KNIME Business Hub instance.

To install a KNIME Business Hub please refer to the KNIME Business Hub Installation Guide - Embedded Cluster or the KNIME Business Hub Installation Guide - Existing Cluster.

A user guide is also available here, which contains instructions on how to perform team administration tasks. Team admins are designated by the global Hub admin, and have control over their team's allocated resources, can add users to their team, create execution contexts and have an overview of the team's deployments. In this way the Hub administration tasks are distributed and reallocated to those users that have a better overview of their own team necessities.

KNIME Business Hub editions

KNIME Business Hub is available in three different editions:

- Basic
- Standard
- Enterprise

All the features described in this guide are available for all the editions, unless explicitly pointed out.

However, please consider that some of the features might be limited by the resources available for the different editions.

In particular the Basic edition:

- Does not allow consumers
- It only allows 1 execution context, which is created during the installation. It is therefore not possible to create a new one unless the other one is deleted.
- It only allows 1 team. Therefore the creation of a new team is not possible.

For an overview of the available features and resources for different KNIME Hub offerings please refer to the pricing page on the KNIME website.

Users and Consumers

- **Consumers** on KNIME Business Hub have access to specific data apps and services available on KNIME Hub. In KNIME Business Hub Basic edition there are no consumers available. Only logged-in users that are members of the team have access to the workflows, spaces and deployments of the team. Standard and Enterprise edition instead allows consumers. They will have unlimited access to the data apps and services that are shared with them.
- **Users** on KNIME Business Hub are members of a team and have access to all the workflows, spaces and deployments of their team, and to the public spaces of the other teams. In KNIME Business Hub Basic and Standard edition licenses 5 users are included, while 20 are included for Enterprise edition licenses.
- Unlicensed users, instead, do not have read access to any of the resources of the KNIME Business Hub for Basic and Standard edition licenses, while they have read access in the Enterprise edition licenses.

Create and manage teams

A team is a group of users on KNIME Hub that work together on shared projects. Specific Hub resources can be owned by a team (e.g. spaces and the contained workflows, files, or components) so that the team members will have access to these resources.

Sign in to the KNIME Business Hub instance with the admin user name by visiting the KNIME Business Hub URL.

Then click your profile picture on the right upper corner of the page and select *Administration* to go to the KNIME Business Hub Administration page. Click *Teams* in the menu on the left. Here you will be able to see an overview of the existing teams and you will be able to manage them.

Create a team

To create a new team click the yellow plus button on the right.

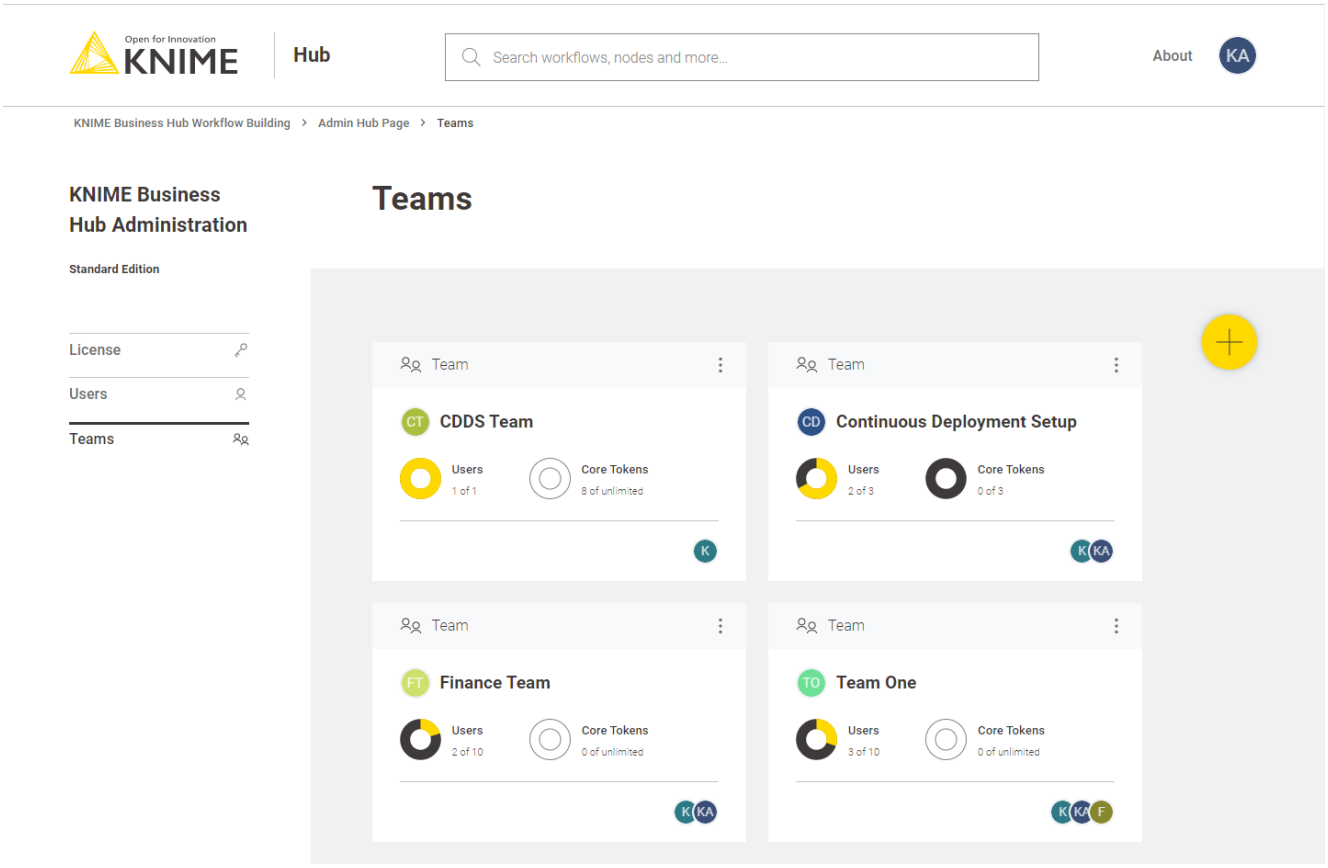


Figure 1. Create a new team in the KNIME Business Hub Administration page

After you create a new team you will be redirected to the new team's page. Here you can change the name of the team. To do so click the name of the team under the team logo on the left side of the page. The name of the team can also be changed at any point in time by

the team administrator.

From the team's page you can:

- Add members to the team
- Change their role to, for example, promote a user to team admininistrator role

Here you might for example want to assign the team to a team administrator. To do so click *Manage team* and enter the user name of the user you want to assign as a team administrator for the current team. Then click on the role and select *Member* and *Admin*. At the same time you might want to delete the global admin user name from the team members list. To do so click the bin icon corresponding to that user. Click *Save changes* to finalize the setting.

Allocate resources to a team

To allocate resources to a team navigate to the KNIME Business Hub Administrator page and select *Teams* from the menu on the left.

Here you can see an overview of the teams available, their allocated resourced, and of their current usage. Click the three dots on the right upper corner of the card corresponding to the team you want to allocate resources to.

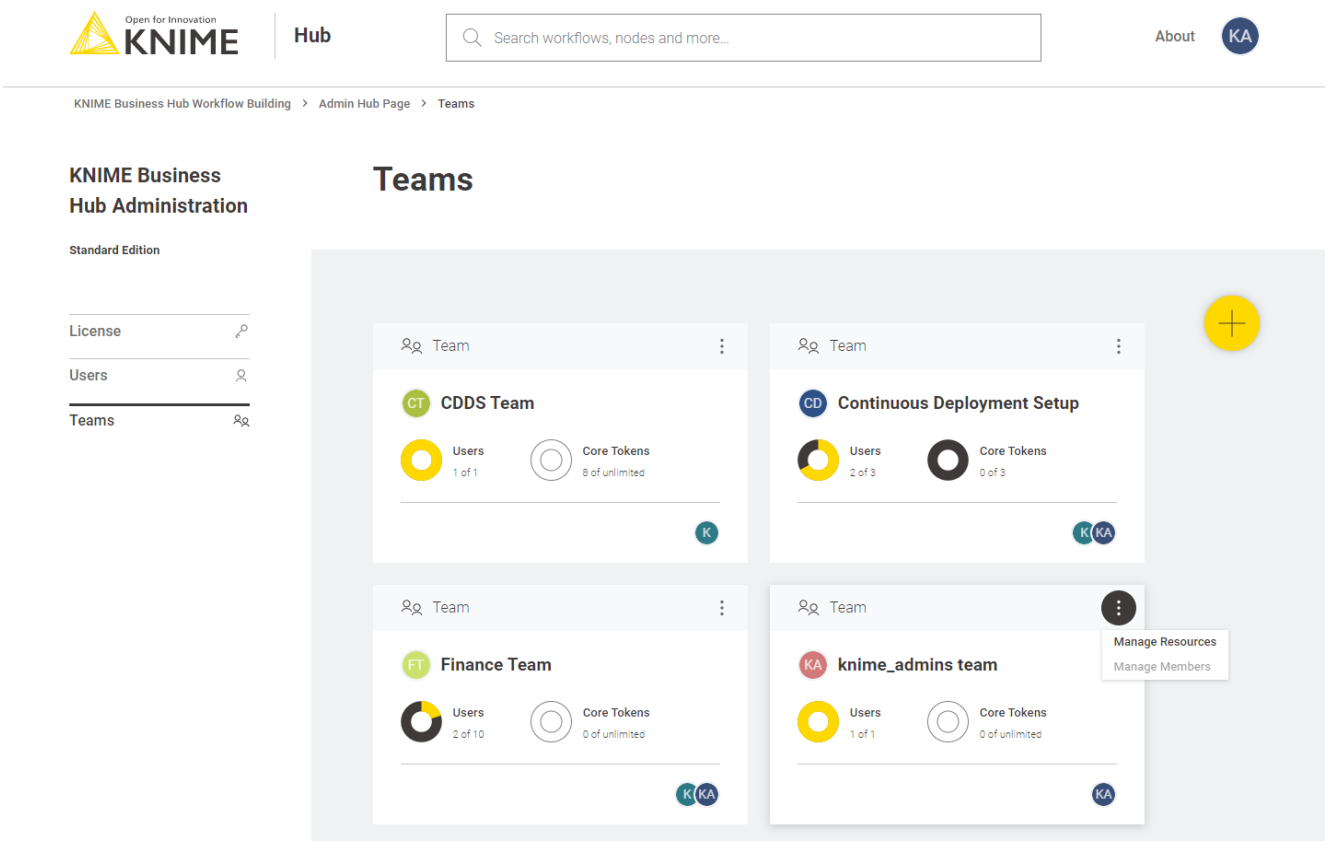


Figure 2. Manage resources of a team

Select *Manage resources* from the menu. A panel on the right will open where you can select the resources you want to allocate.

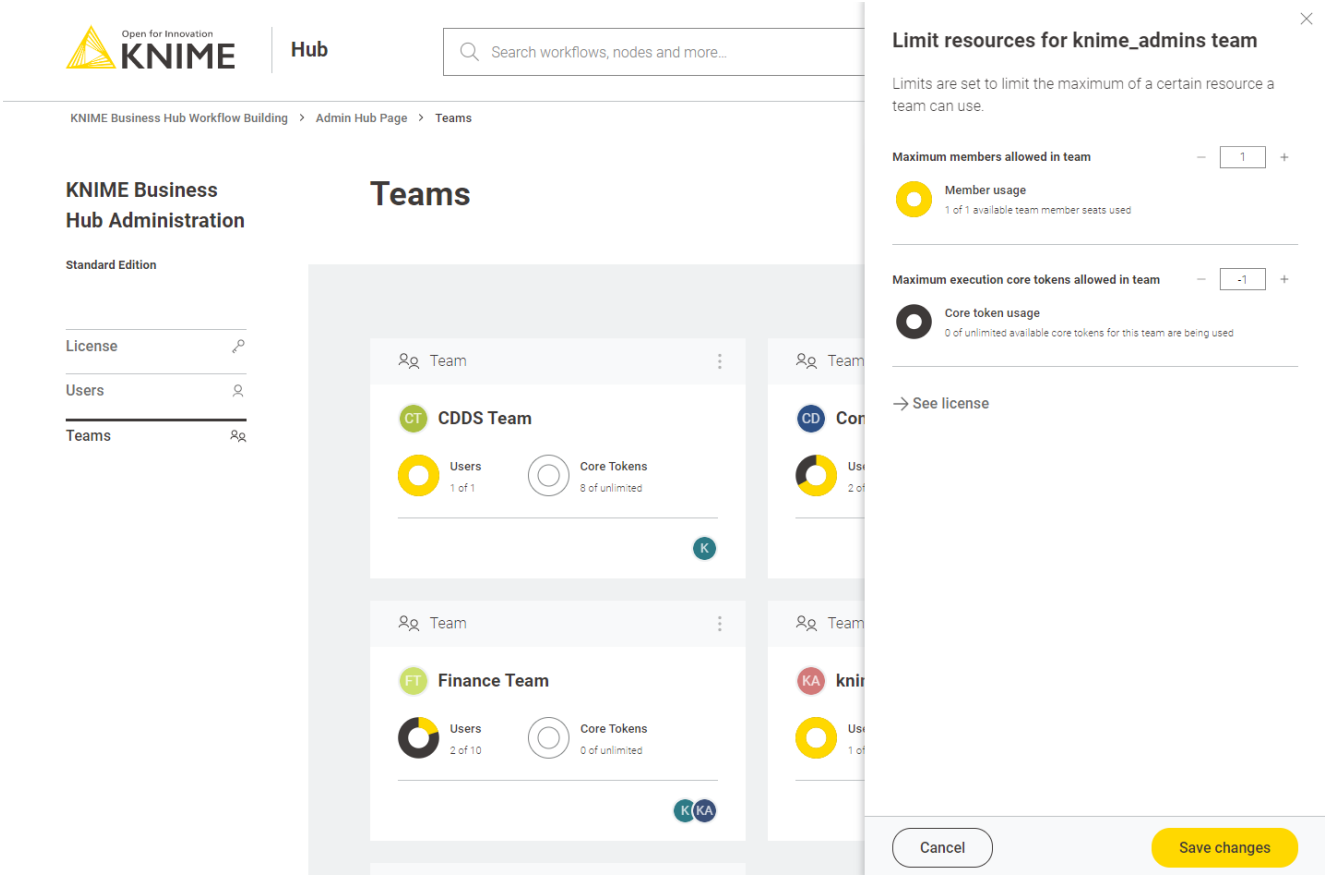


Figure 3. Allocate resources to a team

Here you can change:

- The maximum number of members allowed in that team
- The maximum number of execution vCore tokens allowed for that team

Click *Save changes* when you have set up the resources for the current team.

Manage team members

From the KNIME Business Hub Administration page you can also manage the team members.

Click the three dots on the right upper corner of the card corresponding to the team. From the menu that opens select *Manage members*. In the side panel that opens you can add members to a team, or change the team members role.

Delete a team

From the KNIME Business Hub Administration page you can also delete a team.

Click the three dots on the right upper corner of the card corresponding to the team. From the menu that opens select *Delete*. Be aware that this operation will delete also all the team resources, data and deployments.

Execution resources

As mentioned in the previous section you as an Hub admin can assign execution resources to each team.

Team admins will then be able to build execution contexts according to the execution resources that you assigned to their team. These execution contexts will then be dedicated specifically to that team.

As an Hub admin you can also create a shared execution context. Once you create one you can share it with multiple teams.

For an overview of all the available execution contexts click your profile icon on the top right corner of the KNIME Hub and select *Administration* from the drop-down.

You will be then redirected to the KNIME Business Hub administration page.

Here, select *Execution resources* from the menu on the left.

In this page you can see an overview of *All* the execution contexts available on the Hub.

From the toggle at the top you can filter to see only a specific type of execution contexts available in the Hub instance.

Select:

- *Shared*: Shared execution contexts are created by the Hub admin. They can be made available to multiple teams.
- *Dedicated*: Dedicated execution contexts are created by the team admins for their team. Dedicated execution contexts are exclusively used by a single team.

Each team can by default have a maximum of 10 execution contexts and 10000 jobs. As a KNIME Hub admin you can change these limits via a REST API call like the following:

```
``PUT https://api.<base-url>/execution/limits/{scopeId}/{limitKey}``
```

where `{scopeId}` is the team account ID and the `{limitKey}` is `account-execution-contexts` or `account-jobs` respectively.

Create a shared execution context

As an Hub admin you can create a shared execution context and make it available to multiple teams. To do so click the  button. A side panel opens where you can set up the new shared execution context.

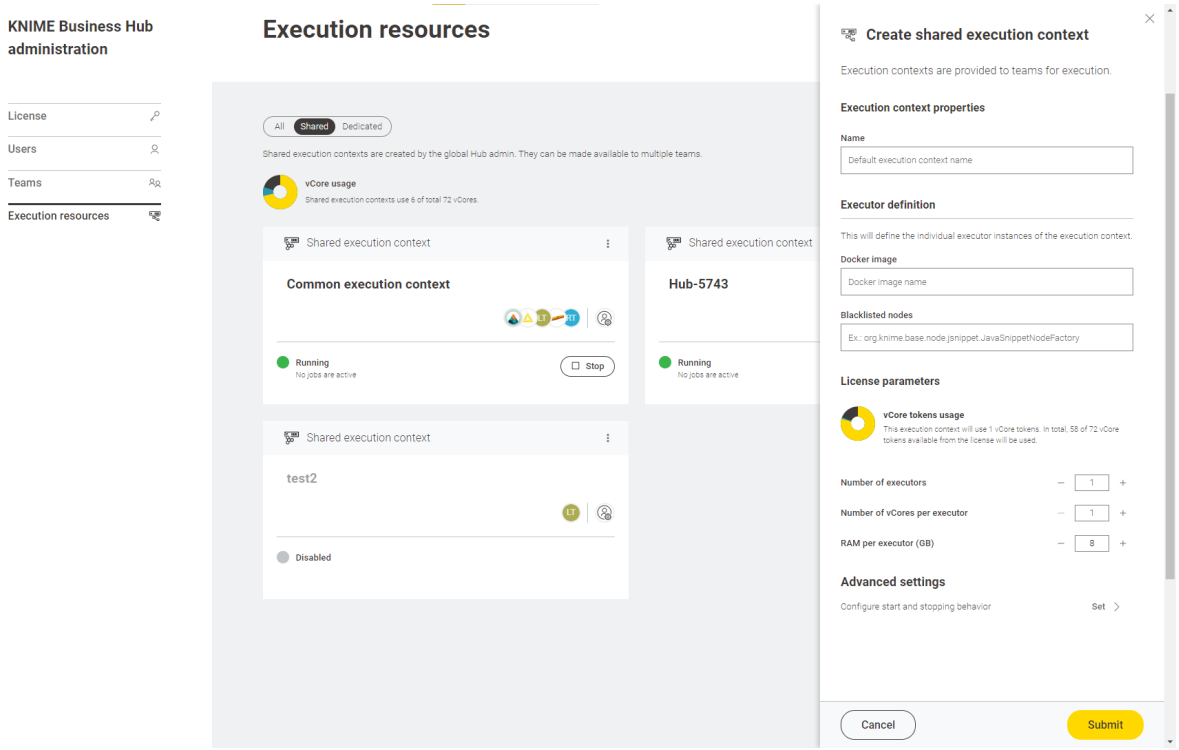


Figure 4. Create a shared execution context

Here you can give the execution context a name, set up which Docker image to use as the executor, give a list of blacklisted nodes, and assign the resources that the execution context will be able to use.

> Find more information about how to set up the execution context in the KNIME Business Hub User Guide.

Finally, you can configure wether you want the execution context to automatically start and stop. To do so click *Set* under *Configure start and stop behavior* and select *On* (the setting is *Off* by default) from the toggle on top. Then you can indicate the desired inactivity time (in minutes) for the execution context to stop.

The execution context will start automatically when a queued workflow needs to run and stop automatically when there are no more active or queued workflows.

Click *Submit* to create the execution context. A notification will appear where you can click *Manage access* to share the execution context with the teams.

At any time you can also manage the access to a shared execution context bx clicking the ⋮ icon in the corresponding tile and selecting *Manage access* from the menu.

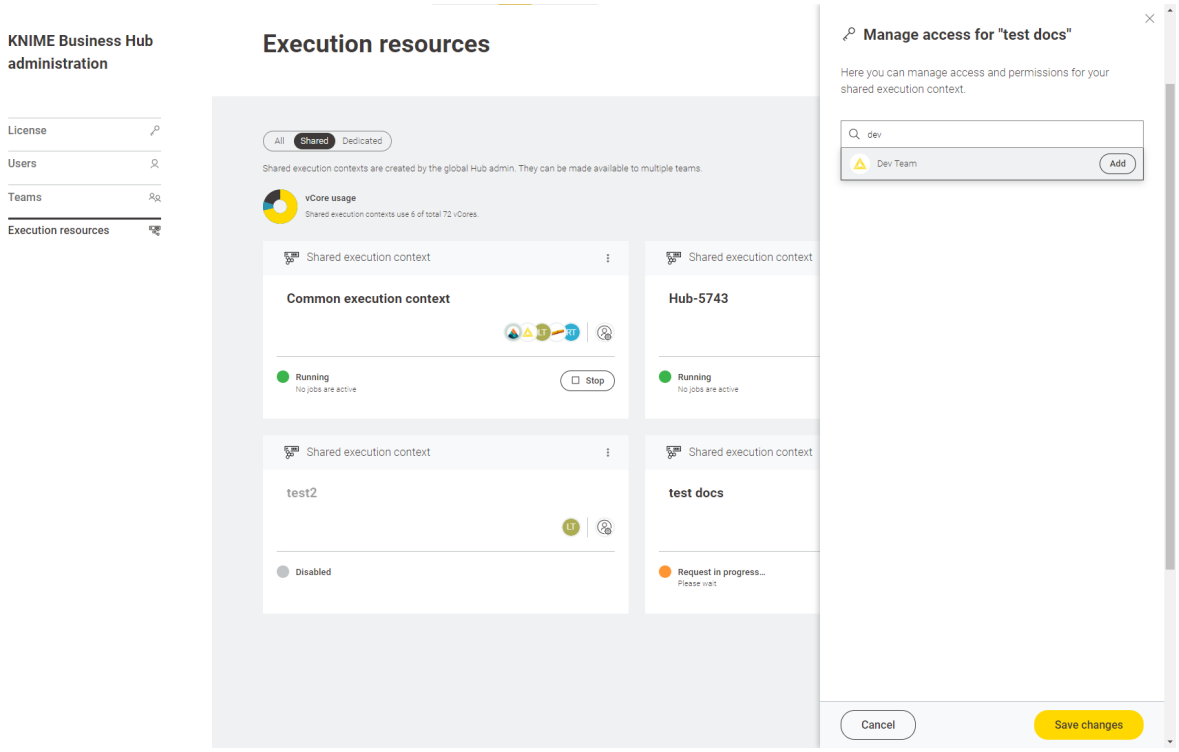


Figure 5. Manage access for a shared execution context

Manage shared execution contexts

Also from the *Execution resources* page you can have an overview about the current status of an execution context, which teams have access to it, how many jobs are running and also manage the execution context performing the following operations:

- *Start* and *Stop* an execution context by clicking the Start/Stop button in the tiles
- Click the ⋮ icon in the tile and from the menu that opens you can:
  - *Edit*: You can change the parameters and configurations in the right side panel that opens.
  - *Manage access*: Manage the access of teams to a shared execution context.
  - *Enable/Disable*: You will need first to delete the jobs associated to the execution context then proceed with disabling it.
  - *Delete*: As a Hub administrator you can delete a shared execution context. You will need to first, delete the jobs associated to the execution context then proceed with disabling it. Finally, you can delete the shared execution context.
  - *Show details*: Selecting this option will open a new page with a list of all the jobs that are running on that execution context, the usage of the execution context

(e.g. how many vCores are in use) and other information. You can also switch to the specific *Executor* tab to see more details about the executor.

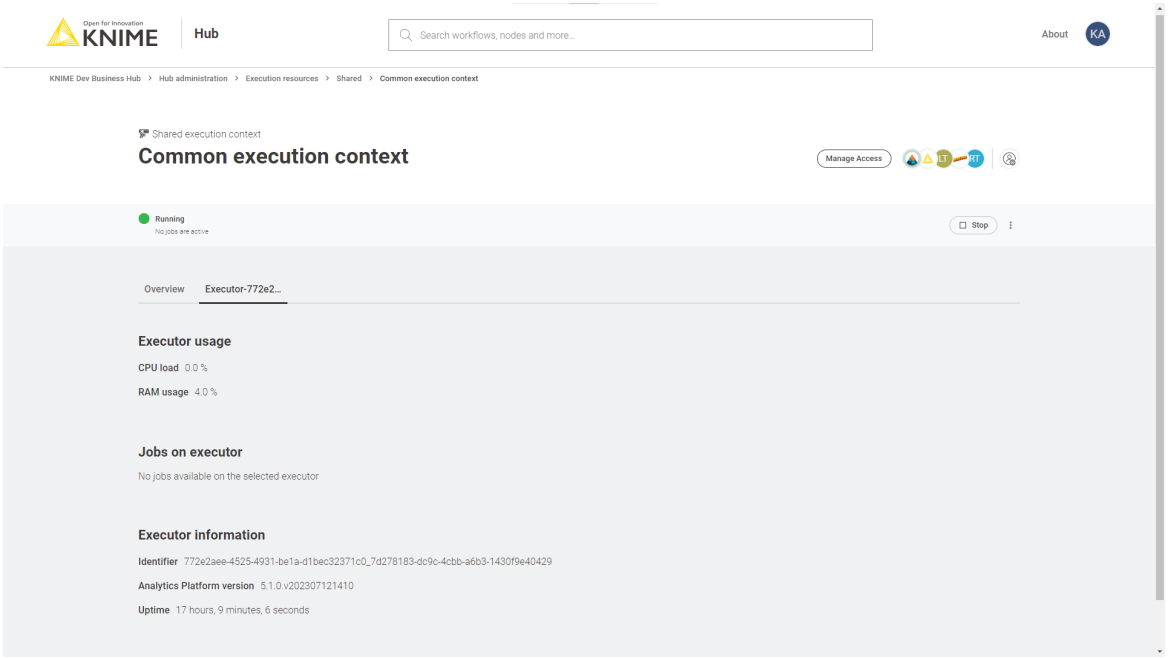


Figure 6. Additional executor information page

Advanced configuration of execution contexts

Execution contexts can be created and edited also via the Business Hub API.

Find more information on the available configurations in the Advanced configuration of execution contexts section in KNIME Business Hub User Guide.

Users management

Keycloak, an open source identity and access management solution, is embedded in KNIME Business Hub and is where users are managed in the backend.

However, if you want to see the users that have access to your KNIME Business Hub instance you can go to the KNIME Business Hub Administration page and select *Users* from the menu on the left. The list shows all users that have already logged into KNIME Business Hub.

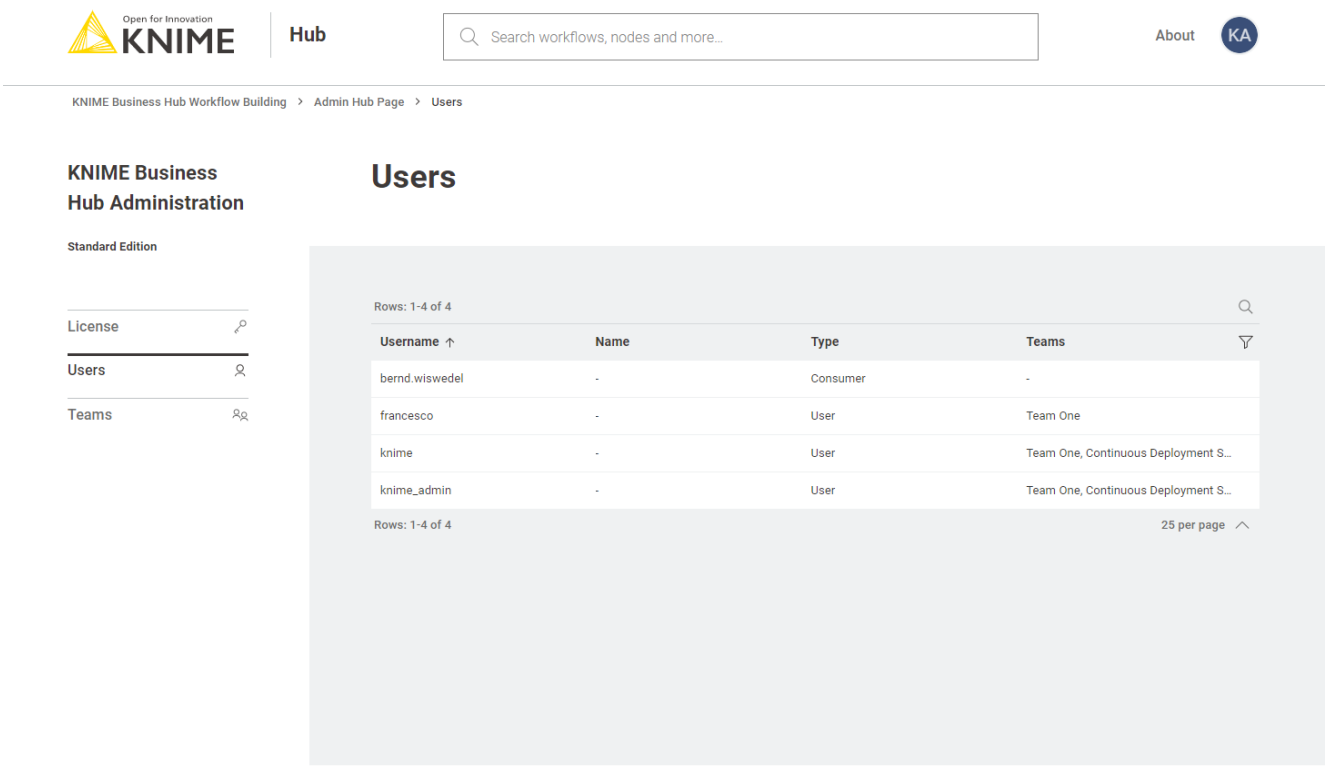


Figure 7. See users on KNIME Business Hub Administration page

Here you can filter the users based on their team, the type of users and their username and name. To do so click the funnel icon in the users list. You can also search the users by using the magnifier icon and typing the keyword in the field that appears.

Users that only exist in your identiy provider are not known to KNIME Business Hub. If you want to create users before they log in, e.g. in order to assign them to groups or share deployments with them, you can provision users with SCIM. See Provisioning users and groups with SCIM (Enterprise edition only) below for details.

Delete a user

To delete a user follow these steps in this order:

1. Delete the user from the KNIME Business Hub Administration page. Click the three dots and select *Delete*. You will need to confirm the user deletion in the window that opens by clicking *Delete user*. Be aware that this action will also delete all data from the deleted user and it will not be possible to restore the user.
2. Delete the user from Keycloak. Follow the steps in the next section to access Keycloak and manage users.

Access Keycloak for users management

1. First you will need to access the Keycloak admin console. To do so you will need the credentials that are stored in a kubernetes secret called `credential-knime-keycloak` in the `knime` namespace. To get the required credentials, you need to access the instance the Business Hub is running on and run the following command:

   ```
   kubectl -n knime get secret credential-knime-keycloak -o yaml
   ```

   This will return a file that contains the `ADMIN_PASSWORD` and the `ADMIN_USERNAME`. Please notice that they are both `base64` encrypted. In order to get the decrypted username and password, you can run the following commands:

   ```
   echo <ADMIN_PASSWORD> | base64 -d
   echo <ADMIN_USERNAME> | base64 -d
   ```

2. Then go to `http://auth.<base-url>/auth/` and log in.

Make a user Hub admin

The operation of promoting a registered user to the role of Hub admin is done in Keycloak.

To promote a user to Hub admin role, follow these steps:

1. First, access Keycloak admin console by going to `http://auth.<base-url>/auth/` and logging in.

   Follow the instructions in the section Access Keycloak for users management if you need to retrieve Keycloak credentials.

2. In the top left corner click the dropdown and select the "Knime" realm, if you are not there already.

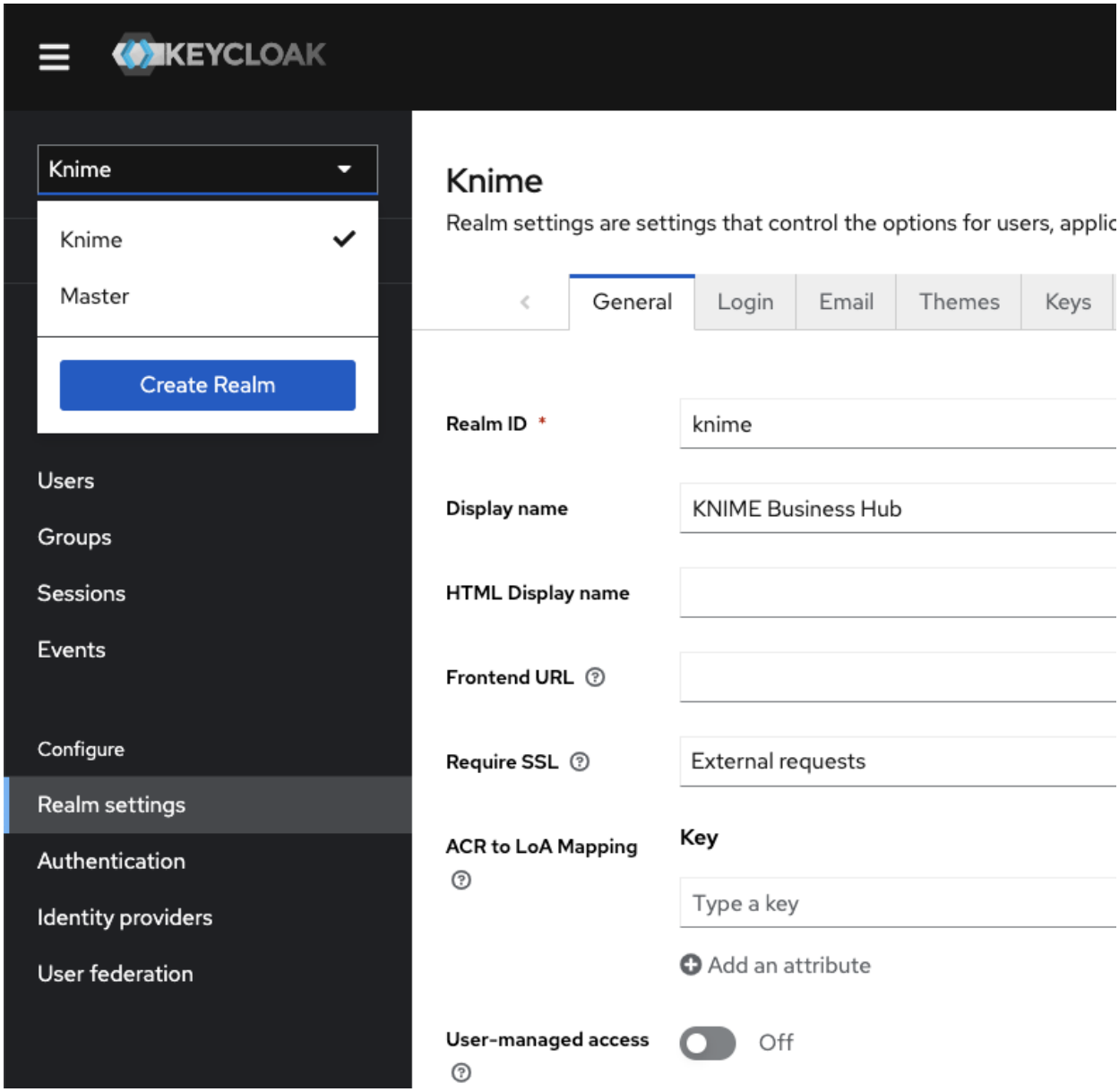


Figure 8. Select the "Knime" realm

3. Navigate to the *Users* menu and search for the user by name or email:

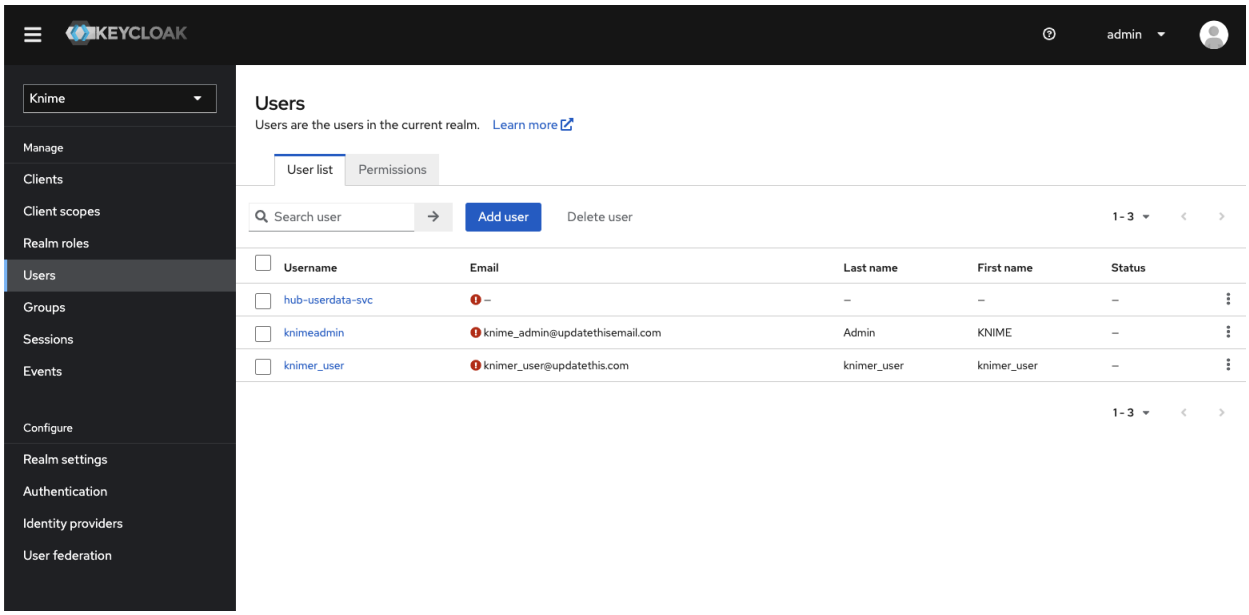


Figure 9. The Keycloak users menu

> In order for a user to appear in this list, it is necessary that they have logged into your KNIME Business Hub installation at least once.

4. Click the user and go to the *Groups* tab. Click *Join Group* and either expand the *hub* group by clicking it, or search for "admin". Select the admin group and click *Join*:

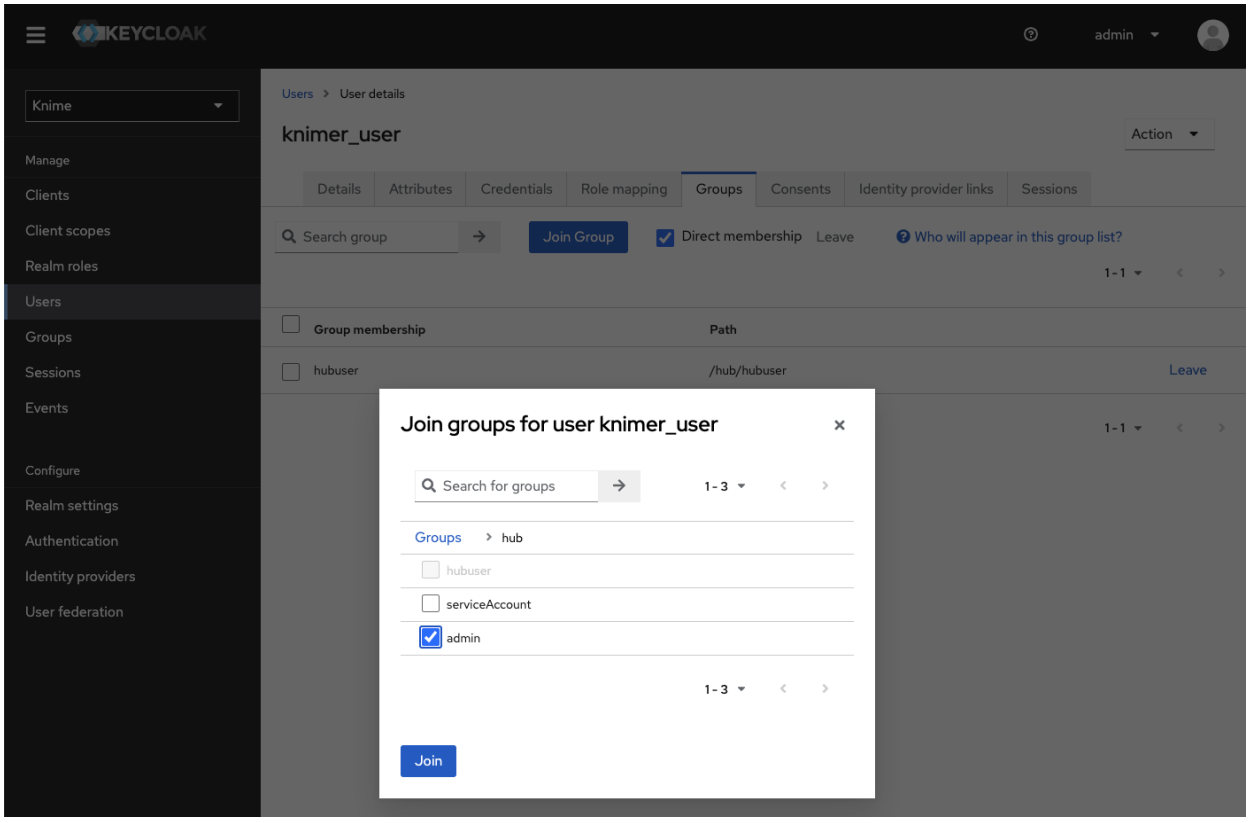


Figure 10. Making a user a Hub admin in Keycloak. If you are searching for the group then the group might show up under its full path "/hub/admin"

5. Done. The user now has the role of Hub admin, and can access the admin pages from within the Hub application to e.g., create teams or delete users.

Provisioning users and groups with SCIM (Enterprise edition only)

The System for Cross-domain Identity Management (SCIM) is an open industry standard for provisioning users and groups from an identity provider to an application. In contrast to authentication protocols such as OpenID/OIDC, a SCIM client in the identity provider actively pushes information about user and groups to the application.

KNIME Business Hub supports SCIM since version 1.10. This allows you to create users in KNIME Business Hub without them having to log in first. You can then assign them to teams or share deployments with them. In addition, groups can be provisioned with SCIM as well. They become *external* groups in KNIME Business Hub. In combination with the new nested groups feature they allow you to automatically assign users to teams based on group memberships in the identity provider.

You can enable SCIM on a KNIME Business Hub installation that already contains users. Existing KNIME Business Hub user accounts will be matched with accounts provisioned by SCIM based on the username and e-mail address.

If your usernames in the identity provider are e-mail addresses you likely have created a mapper in Keycloak that removes the domain part because previous KNIME Business Hub releases did not allow usernames to contain the `@` character. This has changed for version 1.10 so that SCIM clients can provision users with e-mail addresses as usernames without issues. However, this also means you have to disable/remove the corresponding mapper in Keycloak, otherwise the username created by SCIM does not match the username in the authentication token provided by Keycloak. After you have removed the mapper, existing accounts in KNIME Business Hub will automatically be updated when the user logs in the next time.

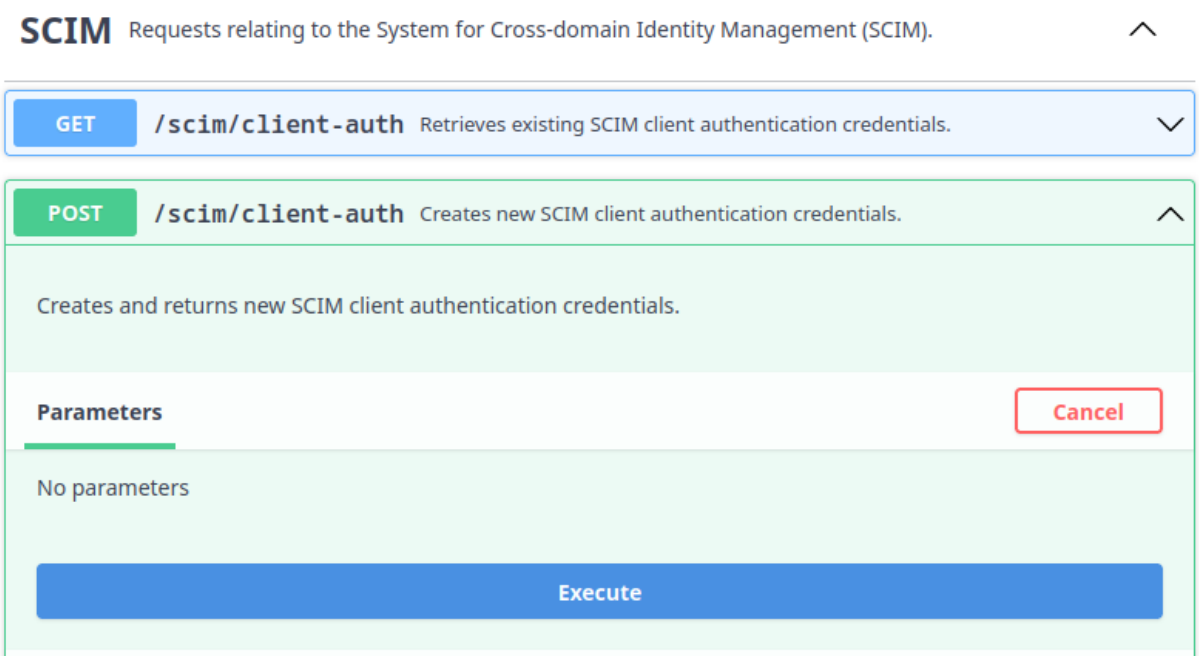
We have tested and hence support SCIM with Okta and Microsoft Entra ID as SCIM clients. Other SCIM clients may work as well but as the SCIM protocol is pretty extensive therefore there may be edge-cases which we haven't covered.

Create a SCIM application password

The first step for any kind of SCIM client is to create a dedicated application password with which the SCIM client authenticates against your KNIME Business Hub installation. This must be done by an administrator and can best be performed using the Swagger UI.

1. Log into your KNIME Business Hub installation as an administrator user.

2. Open the OpenAPI documentation of the account service functionality, which is available at `https://api.<base-url>/api-doc/?service=accounts-service`
3. Go to the *SCIM* section, expand the `POST` request box, click *Try it out*, and then click *Execute*.



4. The result will be displayed below the button. It contains a `pwId`, a `password`, and an `authToken`. For Okta as a SCIM client you will need the `pwId` and the `password`. For Microsoft Entra ID you will need the `authToken`. Please note down these values in a safe place because you will not be able to retrieve them again without deleting and re-creating the SCIM application password.

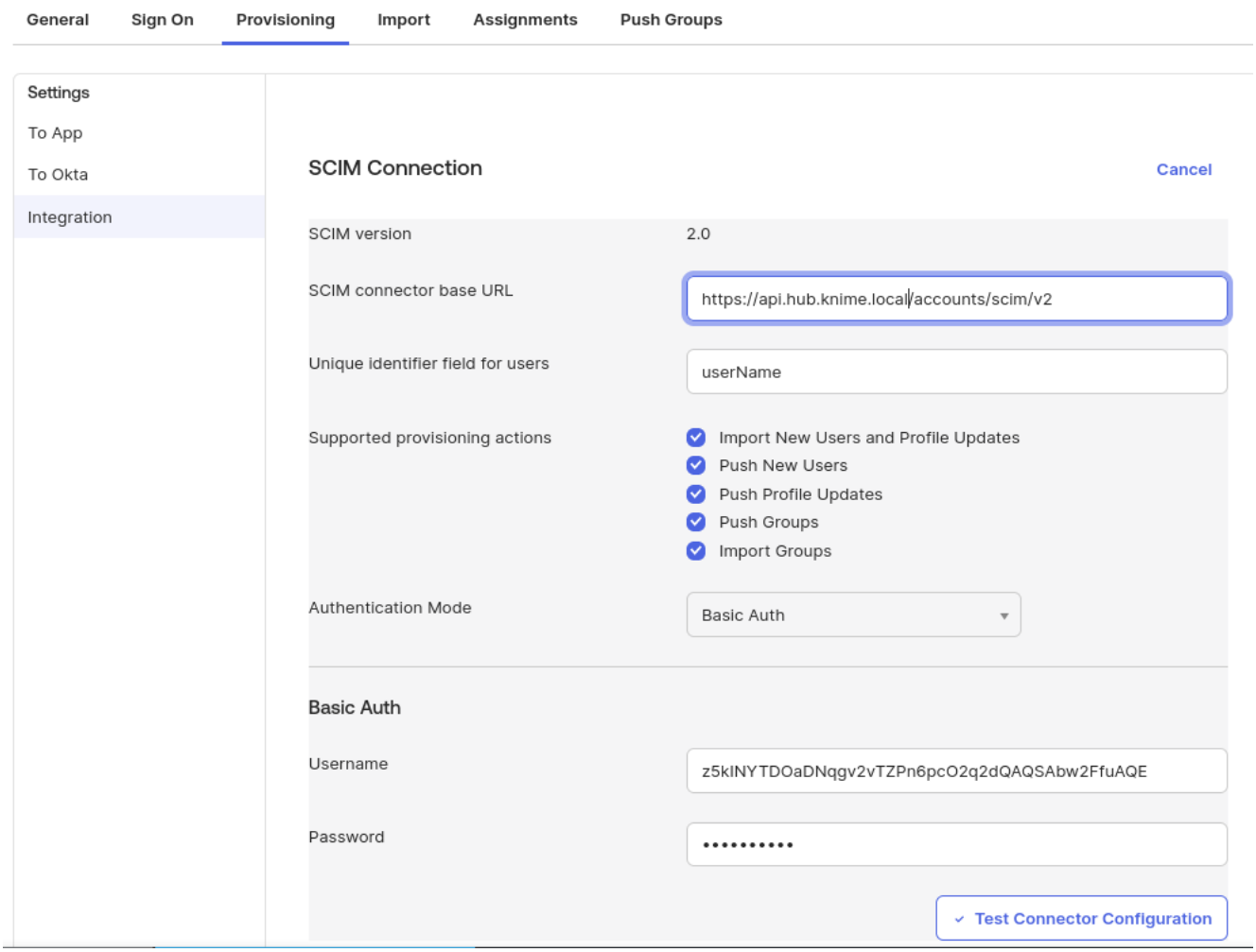
Configure Okta as a SCIM client

In order to set up Okta as a SCIM client for KNIME Business Hub, follow Okta's documentation about Adding SCIM provisioning to app integrations.

It is currently not possible to add SCIM to an OIDC app integration in Okta. But you can simply create a second integration for your KNIME Business Hub installation that is only responsible for provisioning via SCIM. For this create a *SWA app integration* as part of *Task 1* of the documentation.

When you configure the *SCIM Connection* of the app integration, provide the following values:

- Use `https://api.<your hub domain>/accounts/scim/v2` as the *SCIM connector base URL*.
- Use *userName* as *Unique identifier field for users*.
- Select all options for *Supported provisioning actions*.
- Select *Basic Auth* as *Authentication Mode*.
- From the application password created earlier use the `pwId` value as *Username* and the `password` value as *Password*.



After that you can assign users and/or groups to the application in the *Assignments* tab. Provisioning of the respective users will start automatically shortly afterwards. Likewise, group provisioning can be configured in the *Push Groups* tab.

Configure Microsoft Entra ID as a SCIM client

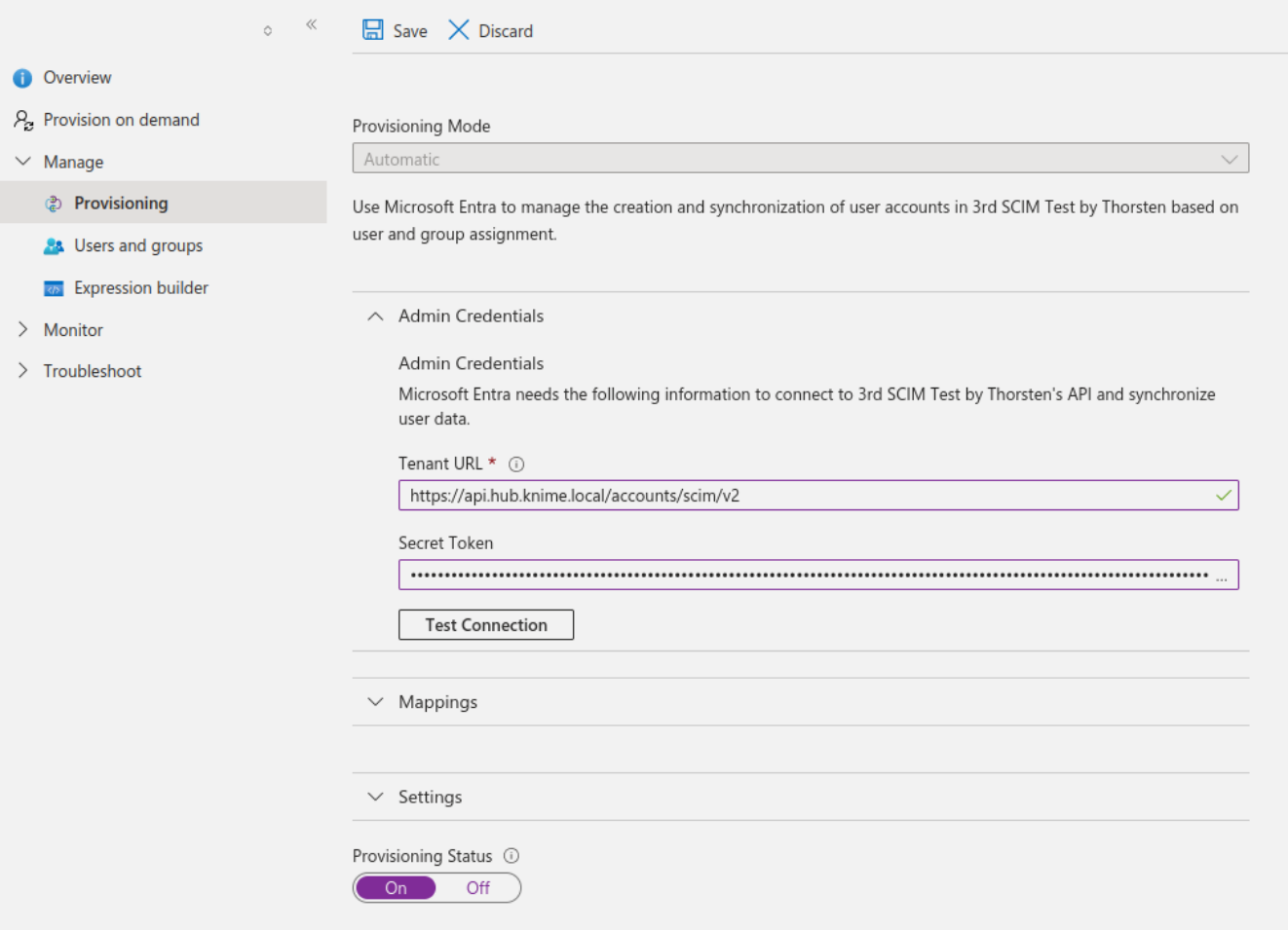
In order to set up Microsoft Entra ID as a SCIM client for KNIME Business Hub, follow

Microsoft's documentation about Integrating your SCIM endpoint with the Microsoft Entra provisioning service.

When you configure the *Provisioning* details of the Enterprise application, provide the following values:

- Use `https://api.<your hub domain>/accounts/scim/v2` as the *Tenant URL*
- Use the `authToken` value from the application password created earlier as the *Secret Token*

No changes are required in the *Mappings* section.



With Entra ID provisioning can take a long time, depending on the number of users and groups. For example, 250 users and 30 groups take about 40 minutes.

If your KNIME Business Hub installation is not reachable by Entra ID due to network restrictions, you can follow Microsoft's instructions about Configuring the connection via the provisioning agent

Assign user to teams based on external groups

If you have provisioned groups from your identity provider to KNIME Business Hub you can use these external groups to manage team memberships. Once changes to group memberships in the identity provider have been provisioned to KNIME Business Hub they will immediately be reflected in team memberships. This also allows you to assign users to teams before they have logged in.

In order to assign team membership based on an external group, you have to add the external group as a *nested group* to the team's *accountMember* group.

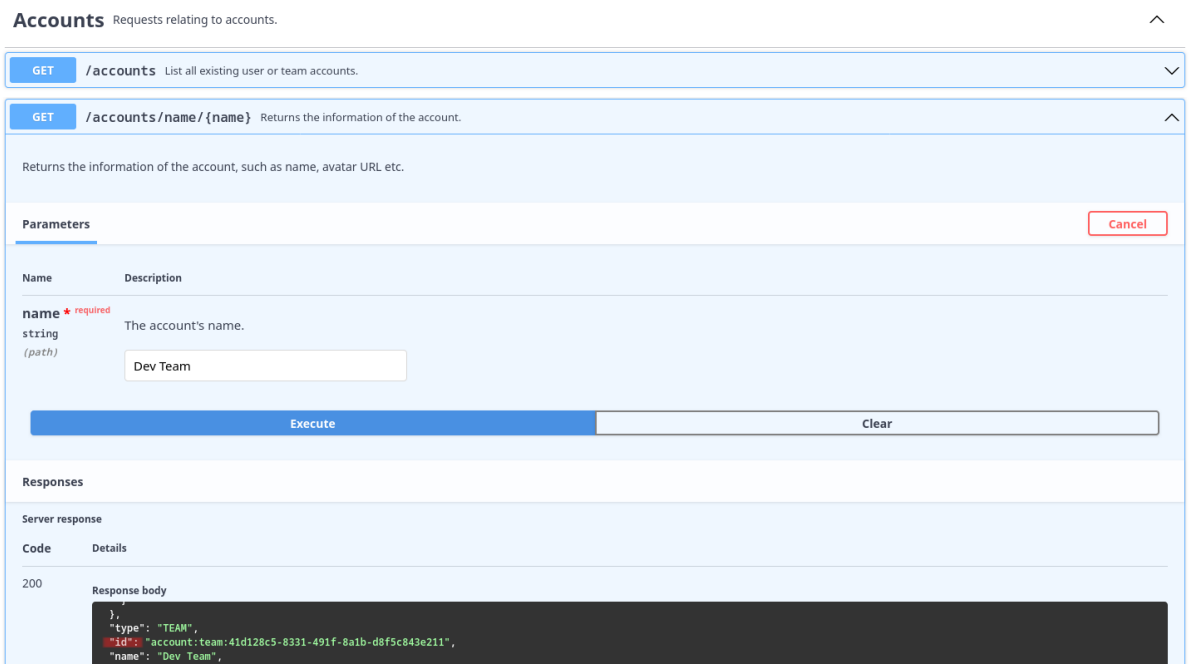
Currently, you can accomplish this by:

1. Using **API calls**, such as through the Swagger UI that you used to create the SCIM application password.
2. By using a dedicated **data application** deployed in your KNIME Business Hub instance.

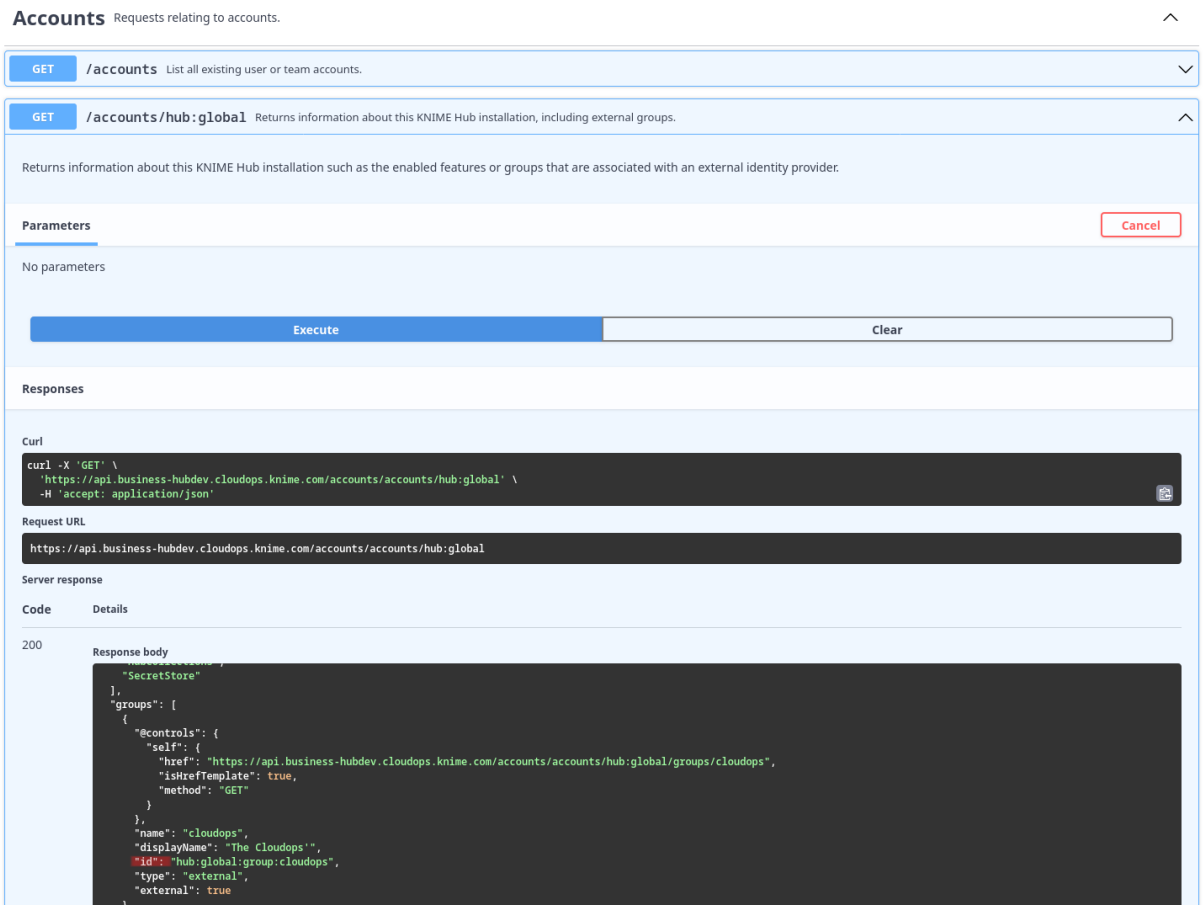
Assign users based on external groups via API calls

Follow these steps in order to add an external group as a nested group to a team group:

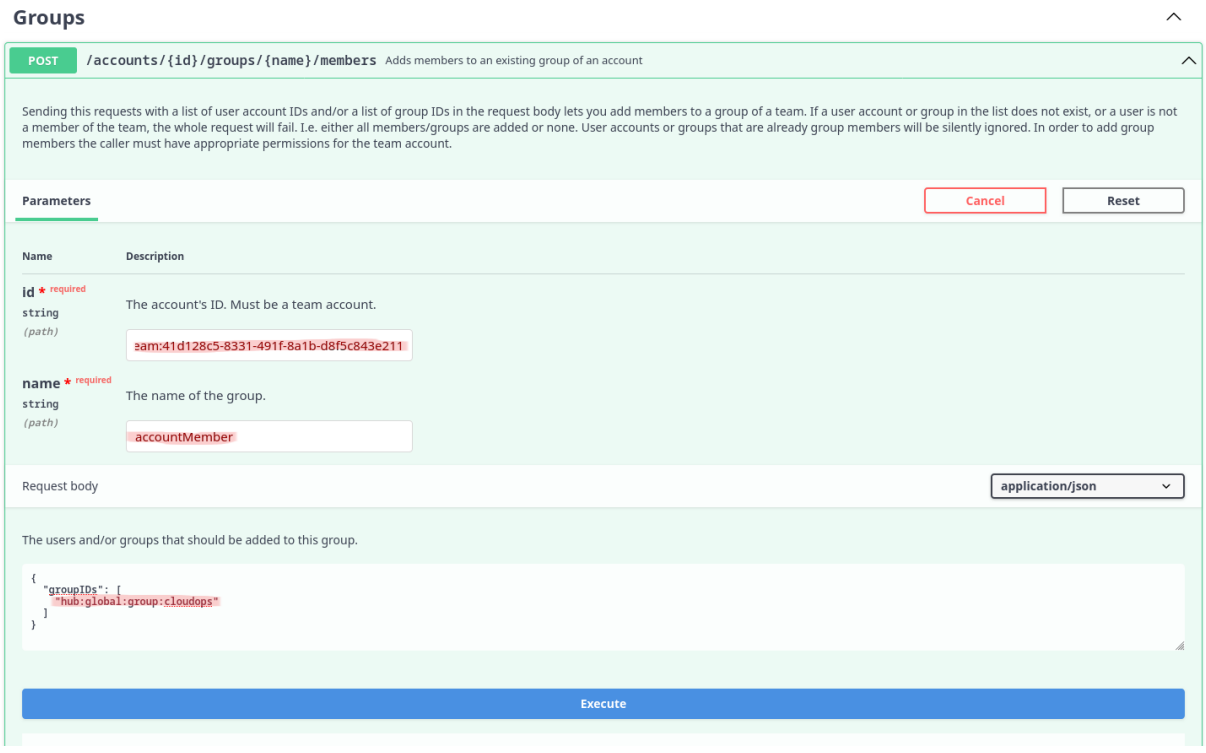
1. Find the team's ID. You can search teams by their name and then get the ID from the `id` field in the response.



2. Find the external group's ID. Since the external group's name in KNIME Business Hub may not be exactly the same as in the identity provider, the easiest way is to list all external groups, select the right one based on its `displayName`, and note down its `id`.



3. Add the external group as a member to the *accountMember* group of the desired team.



After that, the users in the external group will immediately show up as members of the respective team.

Please note that you must have enough users in your KNIME Business Hub license, otherwise this operation might fail due to unsufficient users. The team size limit will be updated automatically if the external group was provisioned by SCIM.

Assign users based on external groups via Data Application

Before running this application, ensure you have provisioned groups from your identity provider to KNIME Business Hub. Once you have done this, you can use these external groups to manage team memberships.

Data Application specifications

1. The aim of this application is to allow Global Admins to manage the assignment and deletion of users to teams based on external groups without having to do it manually.
2. Hence, only KNIME Business Hub Global Admins have permission to run this data application.
3. When using the data application in KNIME Business Hub, user authentication will align with the logged-in user's permissions, eliminating the need to provide an application password.

Data Application execution

1. Download the workflow from KNIME Community Hub.

   You can find the workflow in the KNIME Community Hub.

2. Upload the downloaded workflow to your KNIME Business Hub instance via KNIME Analytics Platform.
3. Deploy the workflow as a data application. You can learn more about how to do it here.
4. Afterward, execute the newly created data application deployment.

External groups - teams mapping

The one-page data application lets the Global Admin of KNIME Business Hub view the current mapping of the external groups and teams and make additions or deletions to modify it.

Current mapping

Below the main definitions is a table with three columns indicating the current mapping, the external groups, and the teams.

You can use the filters to narrow your research by team name or external group for convenience.

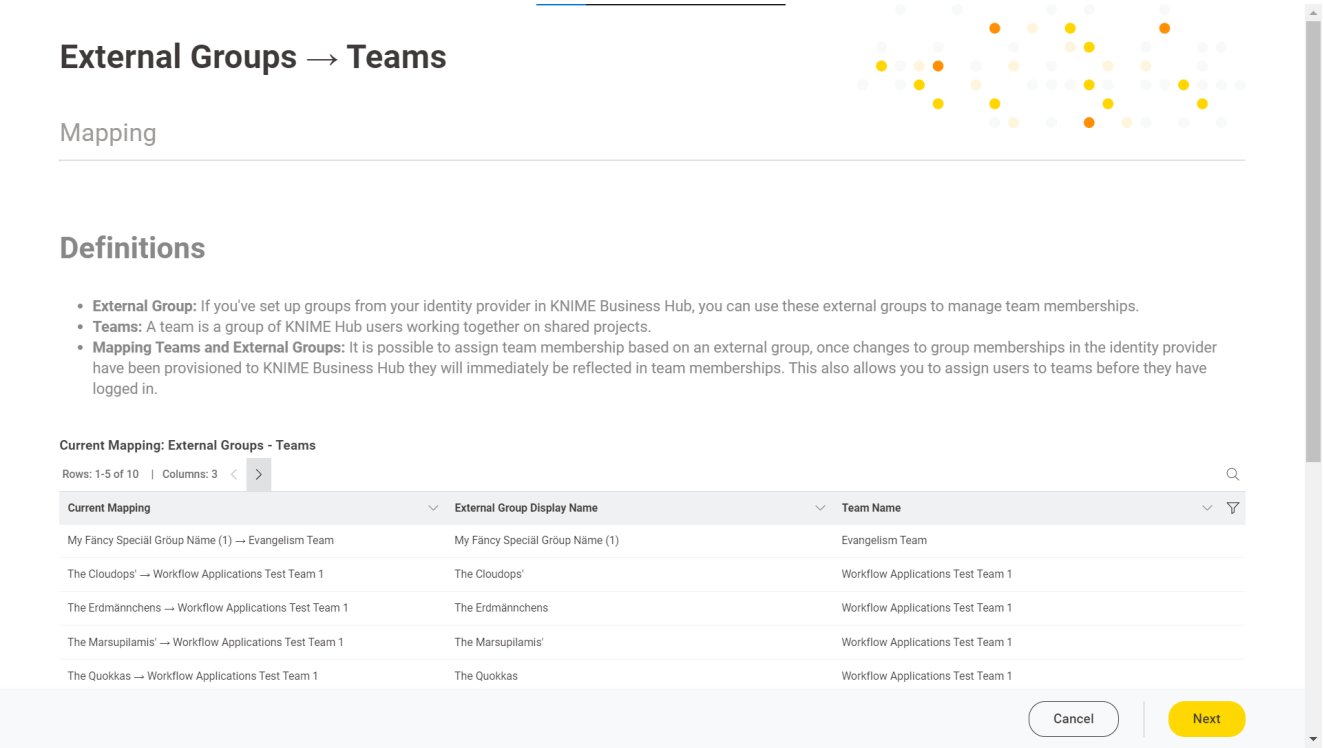


Figure 11. Table with the current external groups assigned to teams.

The table displays just five rows; use the right and left arrows to navigate.

Adding external groups to teams

Once you have a clear understanding of the current external groups assigned to teams, you can add new ones.

1. Choose the "Add external groups to teams" option to complete this task.
    a. The central search box allows you to add the "external groups - teams" pairs.
    b. To ease the research process, if too many options are available, you can **copy** either the *external groups* or the *team* names and paste them into the **search box**. This will display only items related to the pasted name.

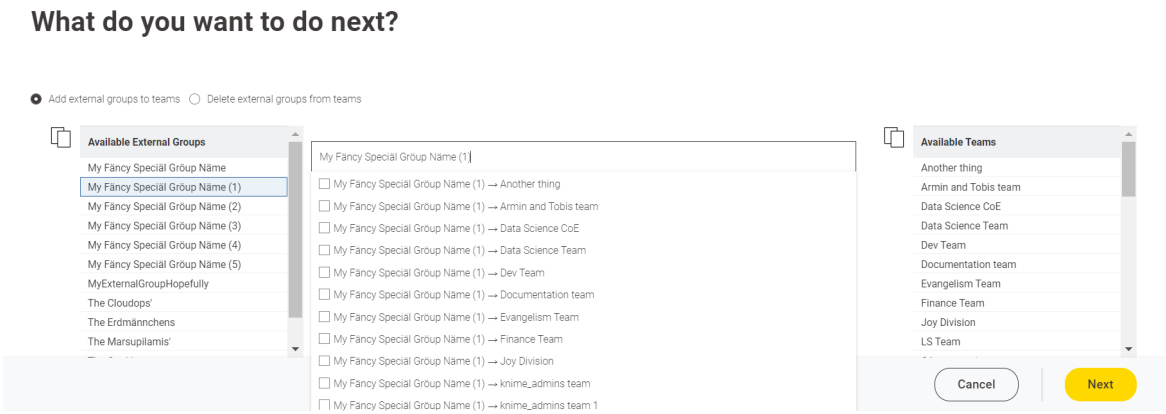


Figure 12. Copy the desired external group name and paste it into the search box.

2. After adding all the desired external groups to the target teams in the search box, click "Next." After a short wait, you will see the same page with the newly added external groups. You can use the table filters to find them.
3. Afterward, the users in the external group will **immediately** appear as members of the respective team.

Delete external groups from teams

To detach an external group from a specific team, you can follow these steps:

1. Select the "Delete external groups from teams" option.
2. Refer to the "Current Mapping: External Groups—Teams" table to determine which external groups are attached to which teams.
3. In the search box, enter the name of the external group or team that you want to delete.

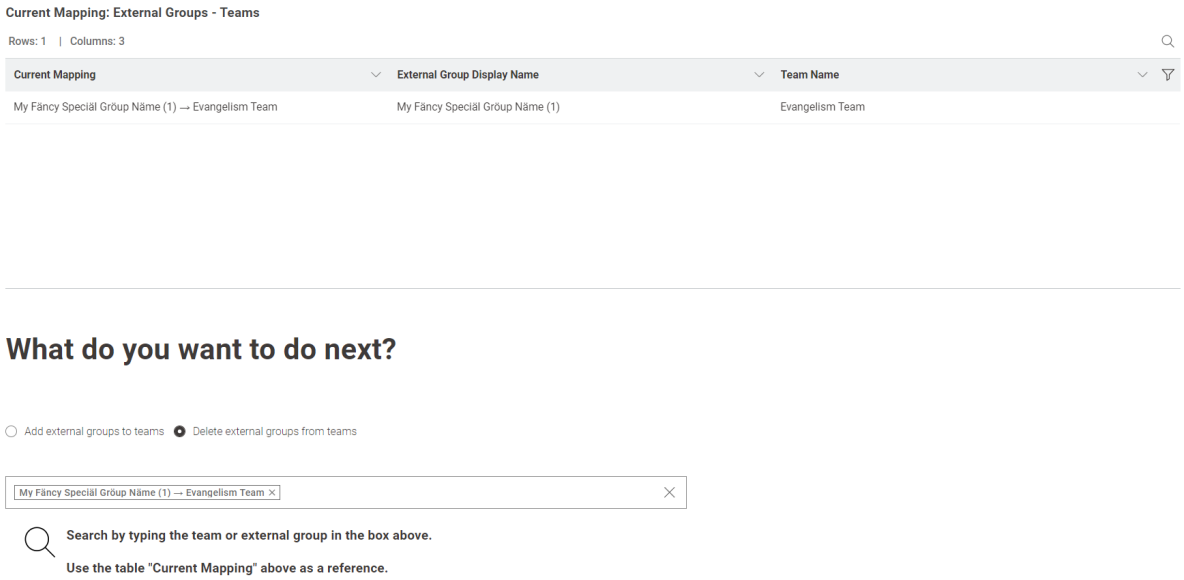


Figure 13. The external group that will be deleted from the team is added to the search box.

4. Once you have added all the external groups and their corresponding teams in the search box, click "Next."

After clicking "Next", all the users in the external group will **immediately** be removed as members of the respective team.

Warnings and errors

The data application may encounter various errors or issues during execution.

1. If the KNIME Business Hub API responds poorly, the data application has trouble communicating with it via REST requests. If this happens, a clear error message should be displayed to inform the user.
2. If external groups have yet to be provided for the current KNIME Business Hub, you must follow all the steps provided above before running the data application.
3. When running the data application for the first time, an information message indicates that no external groups are assigned to any team. To add them, use the search box and click "Next".

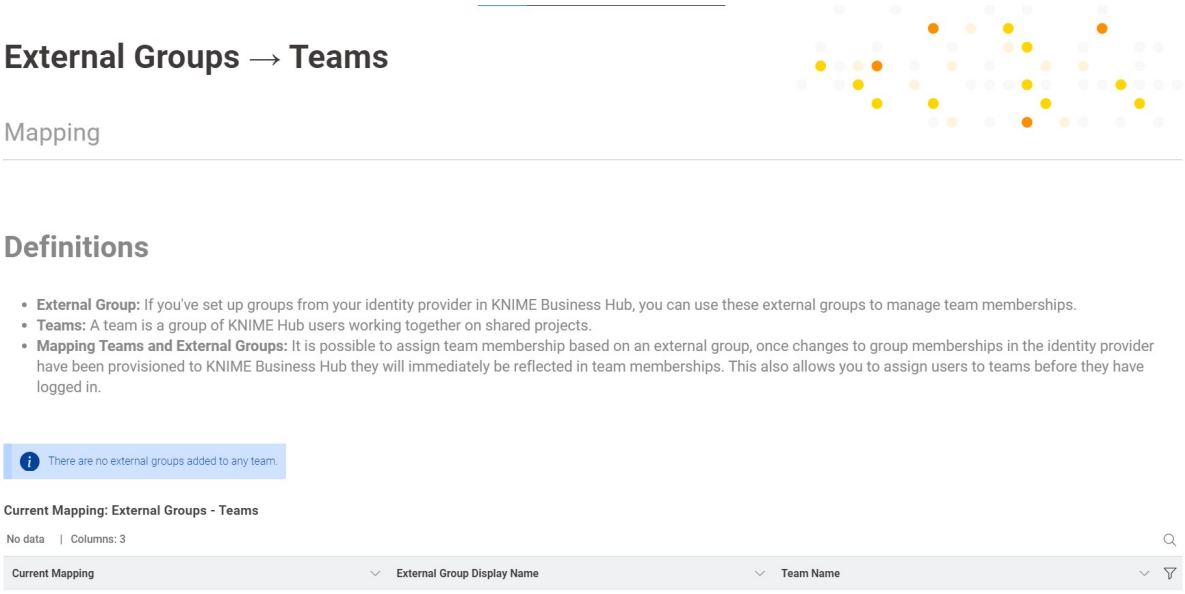


Figure 14. There are no external groups associated with teams.

4. Suppose a KNIME Business Hub team admin or member tries to run the data application. In that case, an error message will appear, telling the user that only Global Admins can execute the application.
5. If you proceed without selecting any item in the search box and click "Next", a warning message will be displayed, prompting you to return to the external groups teams mapping stage by clicking "Next" again.

Expose external groups inside KNIME Business Hub (Standard and Enterprise editions only)

In case you cannot use SCIM, there are two other possibilities for bringing external groups into your KNIME Business Hub installation. As a Global KNIME Hub administrator you can configure groups that are provided via an external identity provider to be exposed inside the KNIME Business Hub instance.

Two possible sources for your external groups are:

1. Groups are provided within the access token of your OIDC provider
2. Groups are imported from LDAP by federating the login

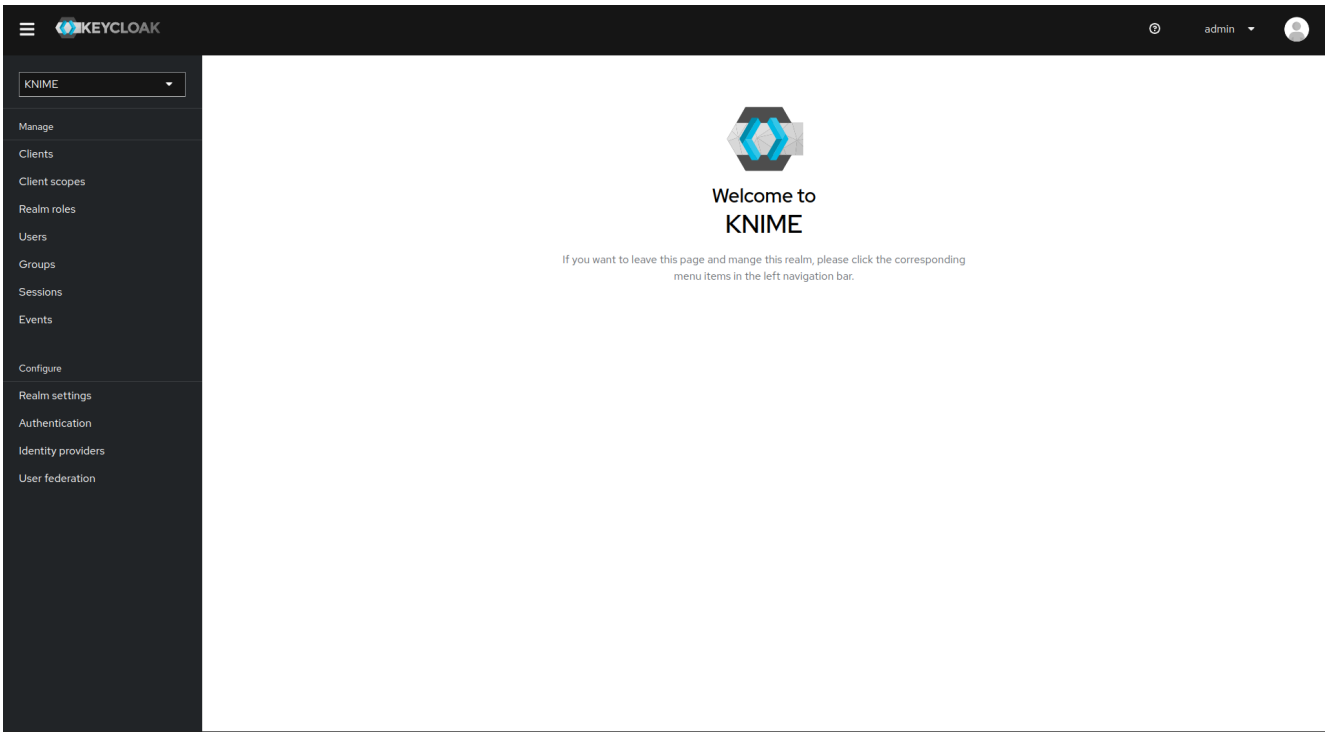
External OIDC provider

Assume you have an identity provider that provides groups through a **groups** claim in the access token.

```
{
    ...,
    "groups": [
        "finance",
        "marketing",
        "data"
    ]
}
```

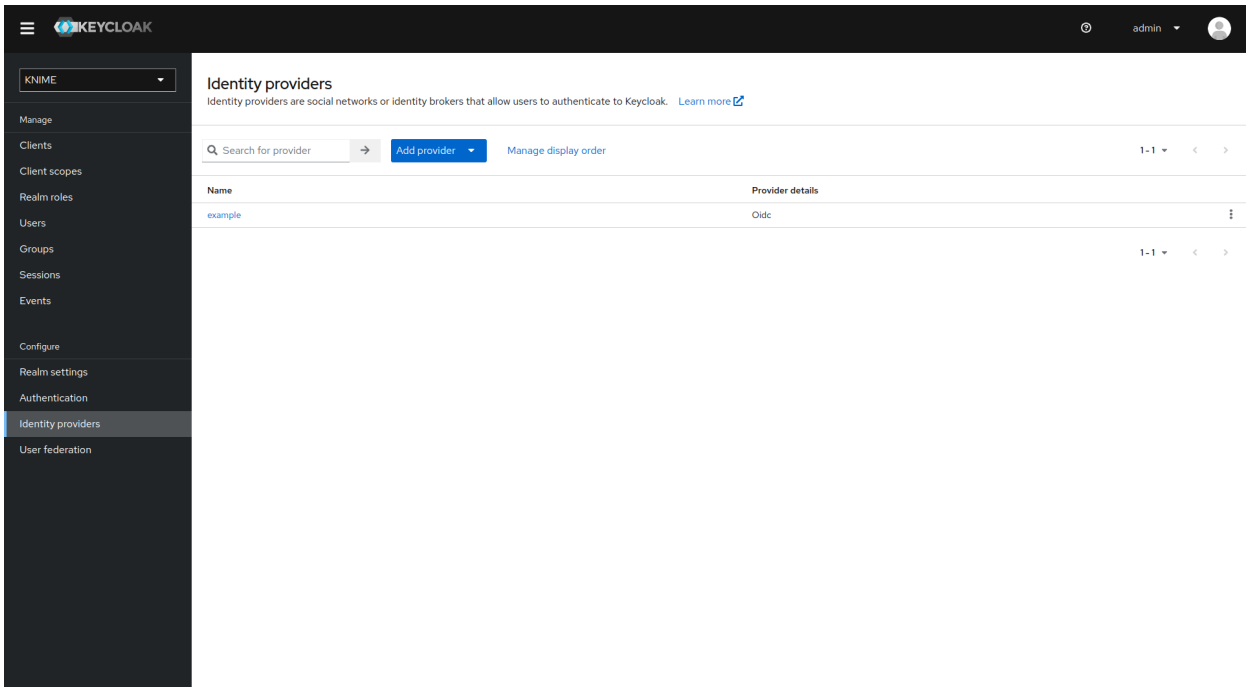
First you need to configure Keycloak in such a way that it can map these groups to a **user attribute**. The second step is to add a mapper that maps these user attributes into the Keycloak's tokens.

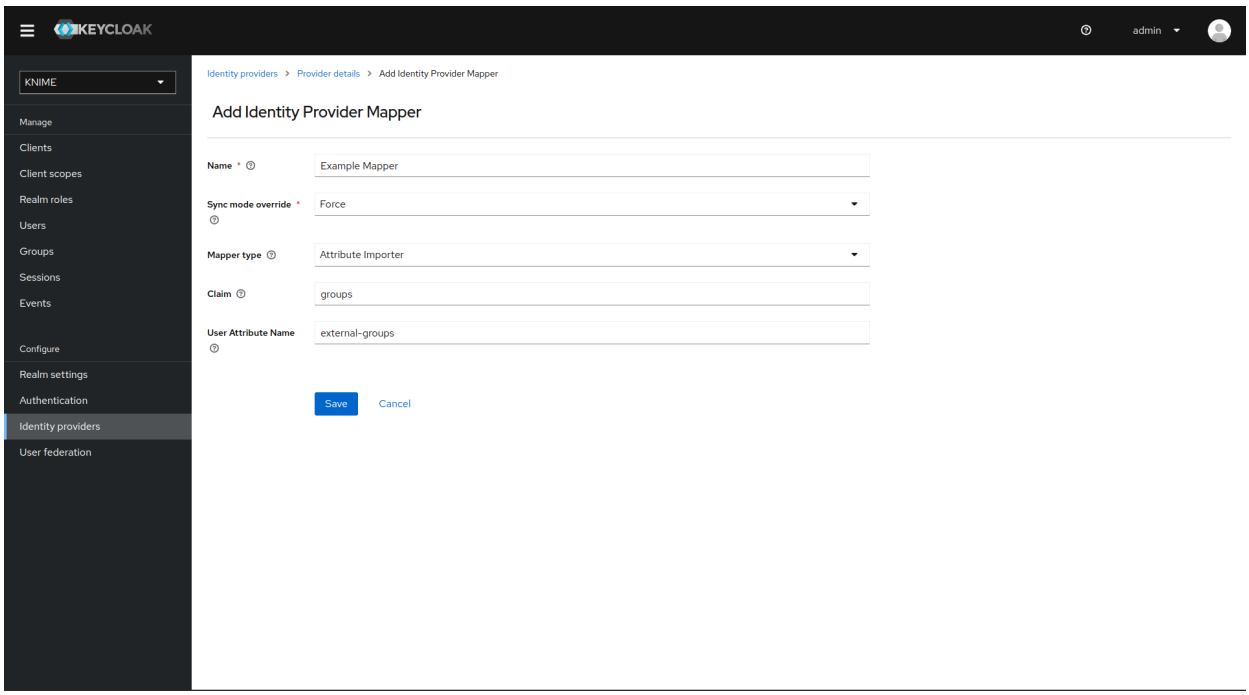
Your third-party identity provider should have been set up already. Keycloak has to be configured as follows:



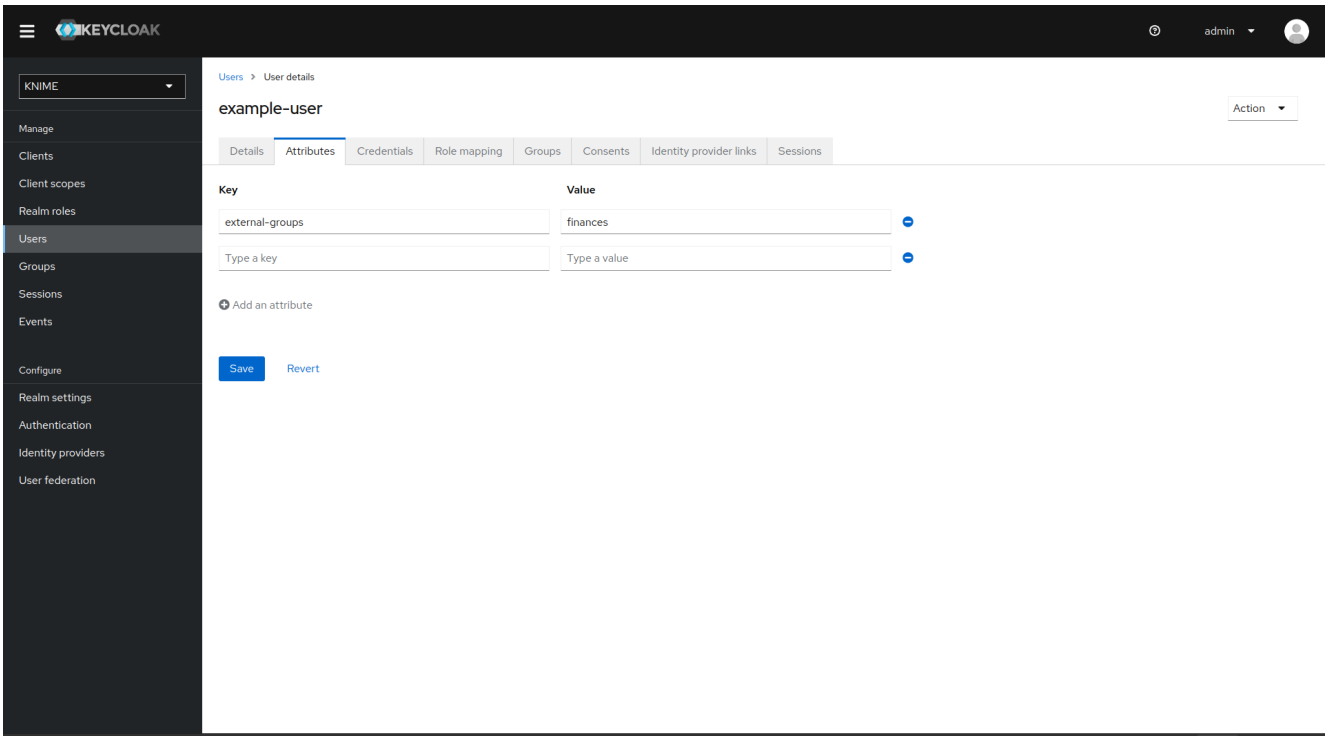
First step is to add an Attribute Importer Mapper.

1. In Keycloak select realm *Knime* in the top left dropdown menu
2. On the left tab select *Identity Providers*
3. Select your third-party provider
4. Switch to the tab *Mappers* and click on *Add mapper*
5. Provide a name for the mapper and set the *Sync mode override* to *Force* to ensure that the user's group memberships are updated upon every login
6. Set *Mapper type* to *Attribute importer*
7. Enter the *Claim* that contains the external groups in the original token (in our example *groups*)
8. In the *User Attribute Name* field enter *external-groups*
9. Click on *Save*





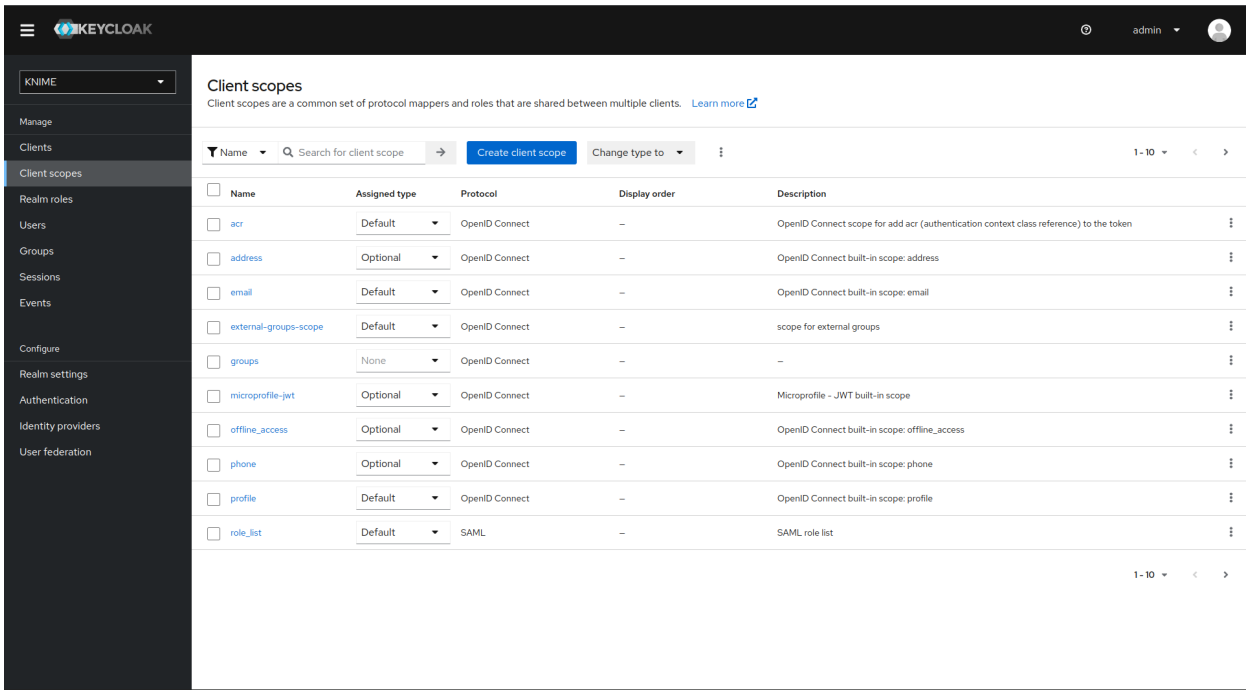
Now, every user in Keycloak who logged in after the mapper has been added will have an *external-groups* attribute associated like in the following picture:

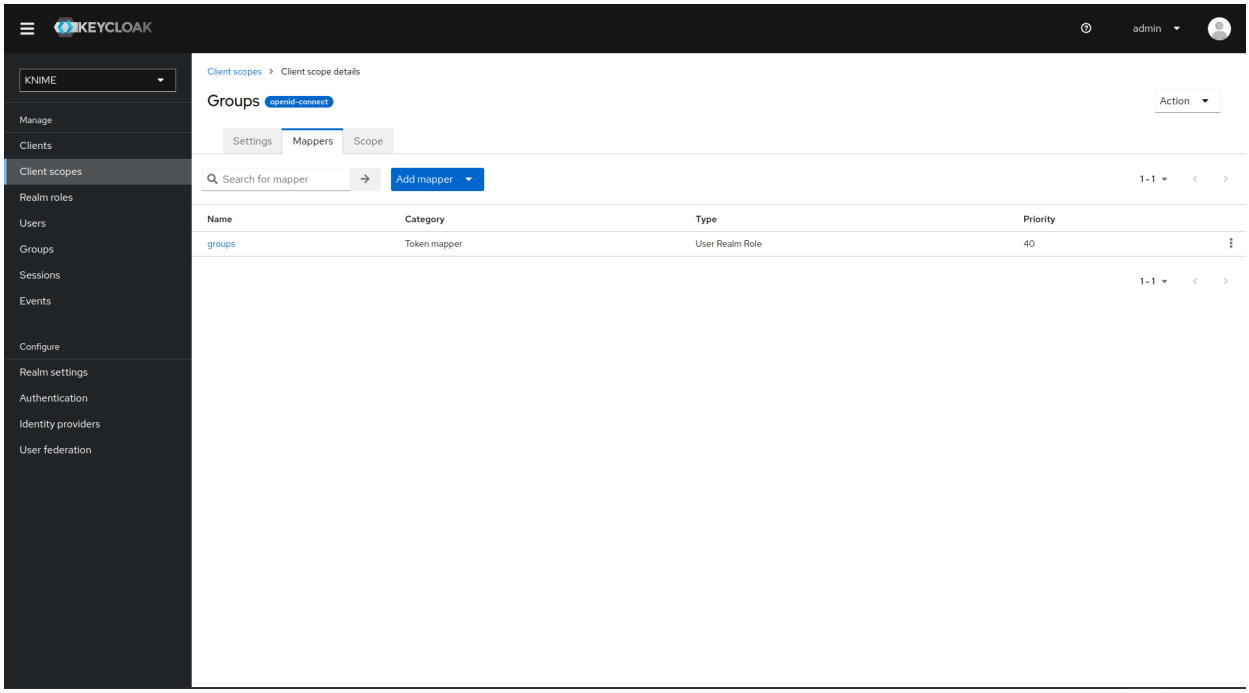


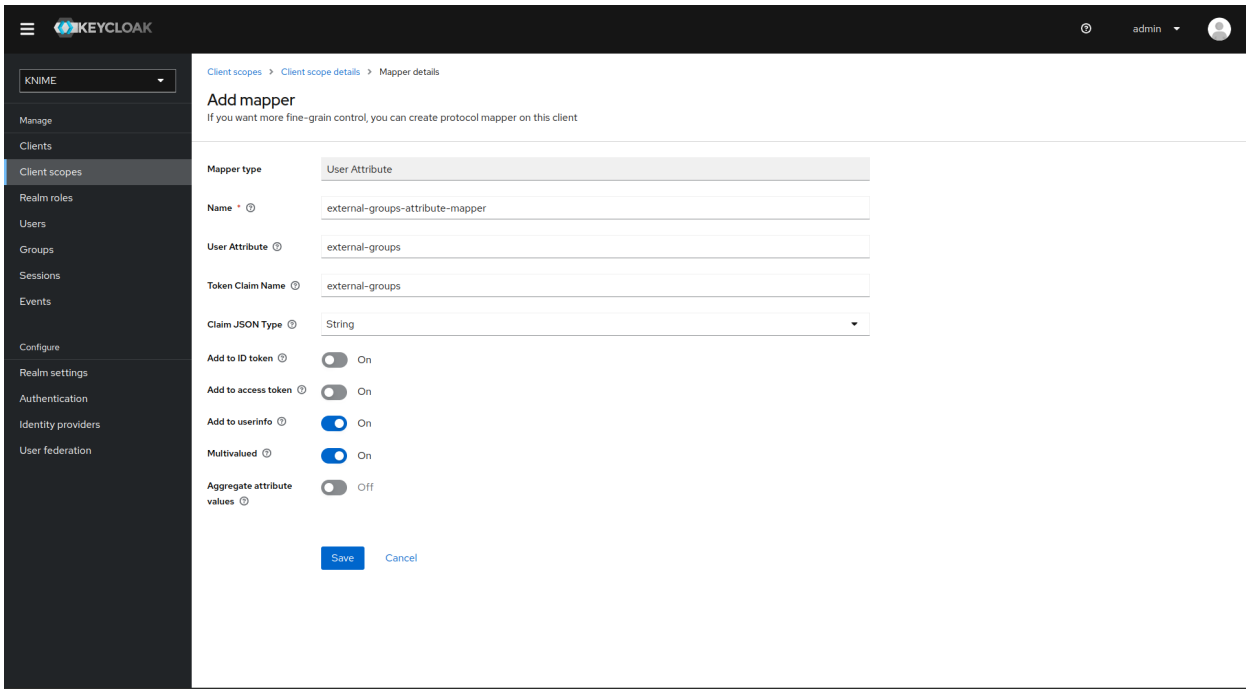
Now, the external groups are known to Keycloak. To expose them inside KNIME Business Hub they need to be mapped into the access tokens issued by Keycloak. For this a second mapper needs to be added, that maps the user attribute *external-groups* to a claim in the user's access token.

To do this you need to add a client scope, which includes a mapper for the user attribute.

1. On the left tab select *Client scopes*
2. Select *groups*
3. Switch to the tab *Mappers*
4. Click on *Add mapper > By configuration* and select *User Attribute* from the list
5. Provide a name, e.g. *external-groups-attribute-mapper*
6. Set both fields *User Attribute* and *Token Claim Name* to *external-groups*
7. Ensure that *Add to ID token*, *Add to access token* and *Aggregate attribute values* are turned off
8. Ensure that *Add to userinfo* and *Multivalued* are turned on
9. Click on *Save*







With both mappers in place, the external groups are part of userinfo response returned by Keycloak. By this, the external groups are exposed inside KNIME Business Hub. In order to enable external groups to be used for permissions and access management they need to be configured separately through the admin REST API as described in Enable external groups.

LDAP federation

Before proceeding with the following steps you need to have a user federation configured for an LDAP instance.

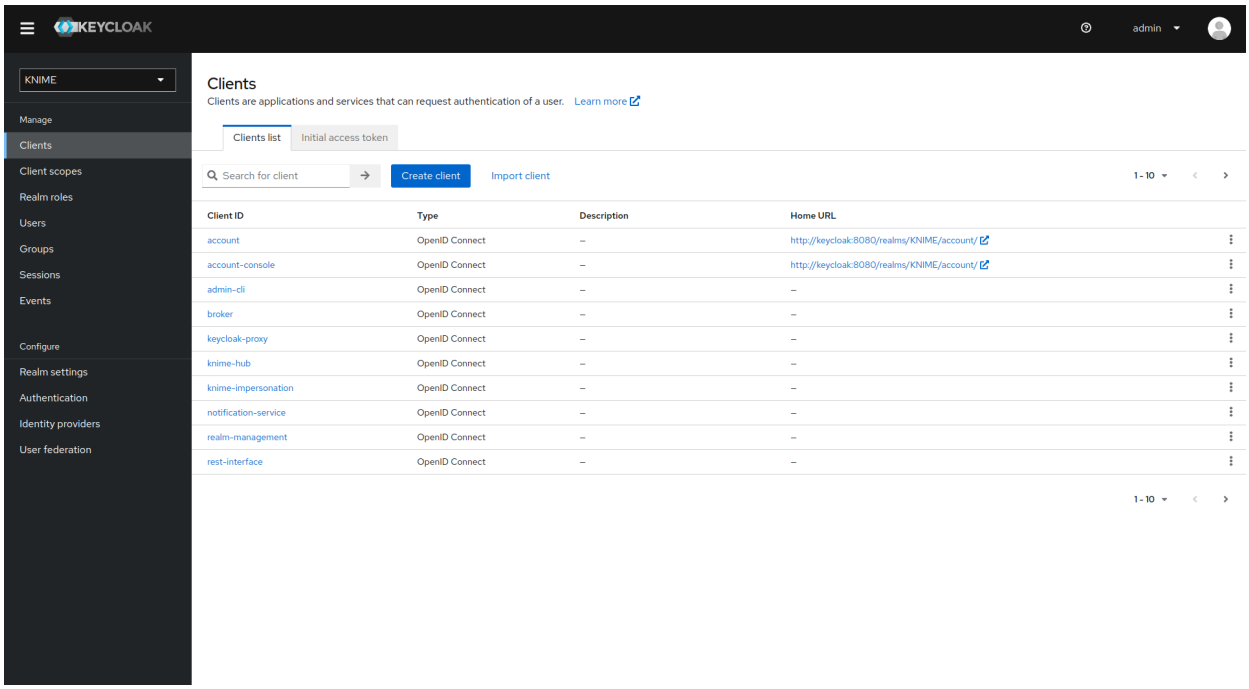
Once you have configured user federation for an LDAP instance that also supplies external group names, you need to configure mappers that map these groups into the access tokens used inside the Hub instance.

To ensure that groups from Keycloak groups and groups from LDAP are not mixed we recommend to treat external groups as realm roles.

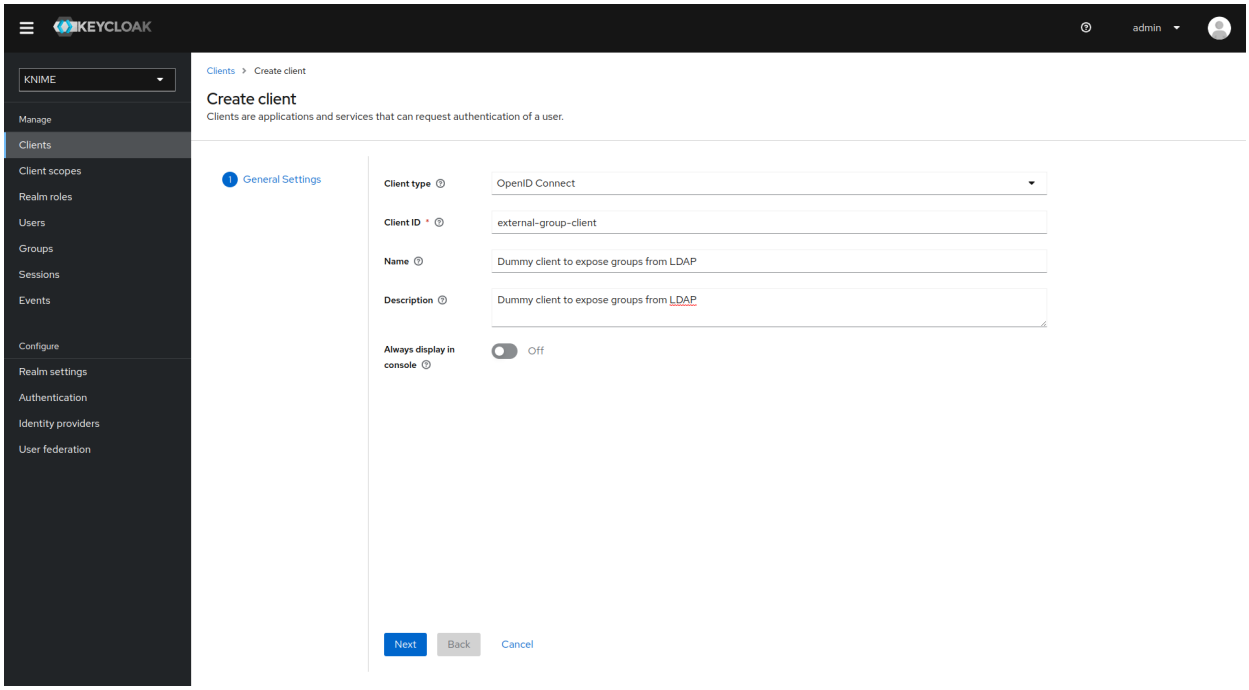
In order to do this we recommend to first create a dummy client for which roles can be created based on the LDAP groups. This will guarantee that any changes will be compatible with future changes to the KNIME Hub client in Keycloak.

To create a new client follow these steps:

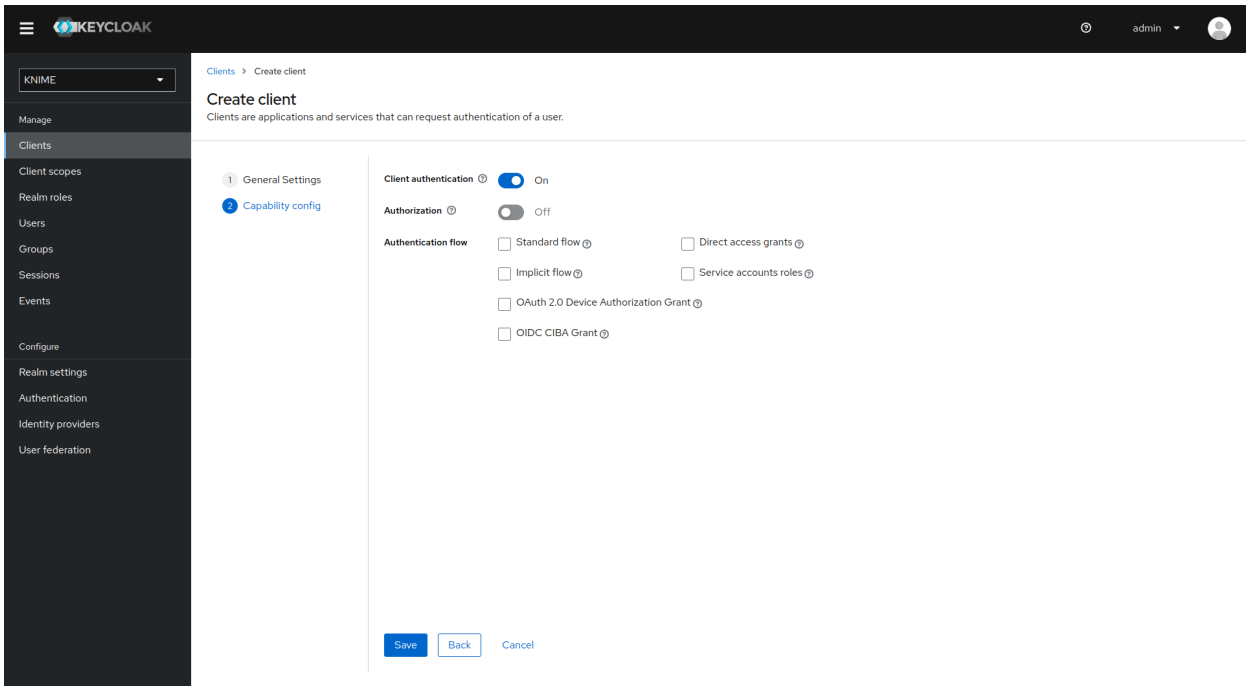
1. In Keycloak select realm *Knime* in the top left dropdown menu
2. On the left tab select *Clients* and click *Create client*



3. Set *Client type* to *OpenID Connect*
4. Enter a *Client ID* (in our example *external-group-client*), and a useful *Name* and *Description*
5. Click on *Next*

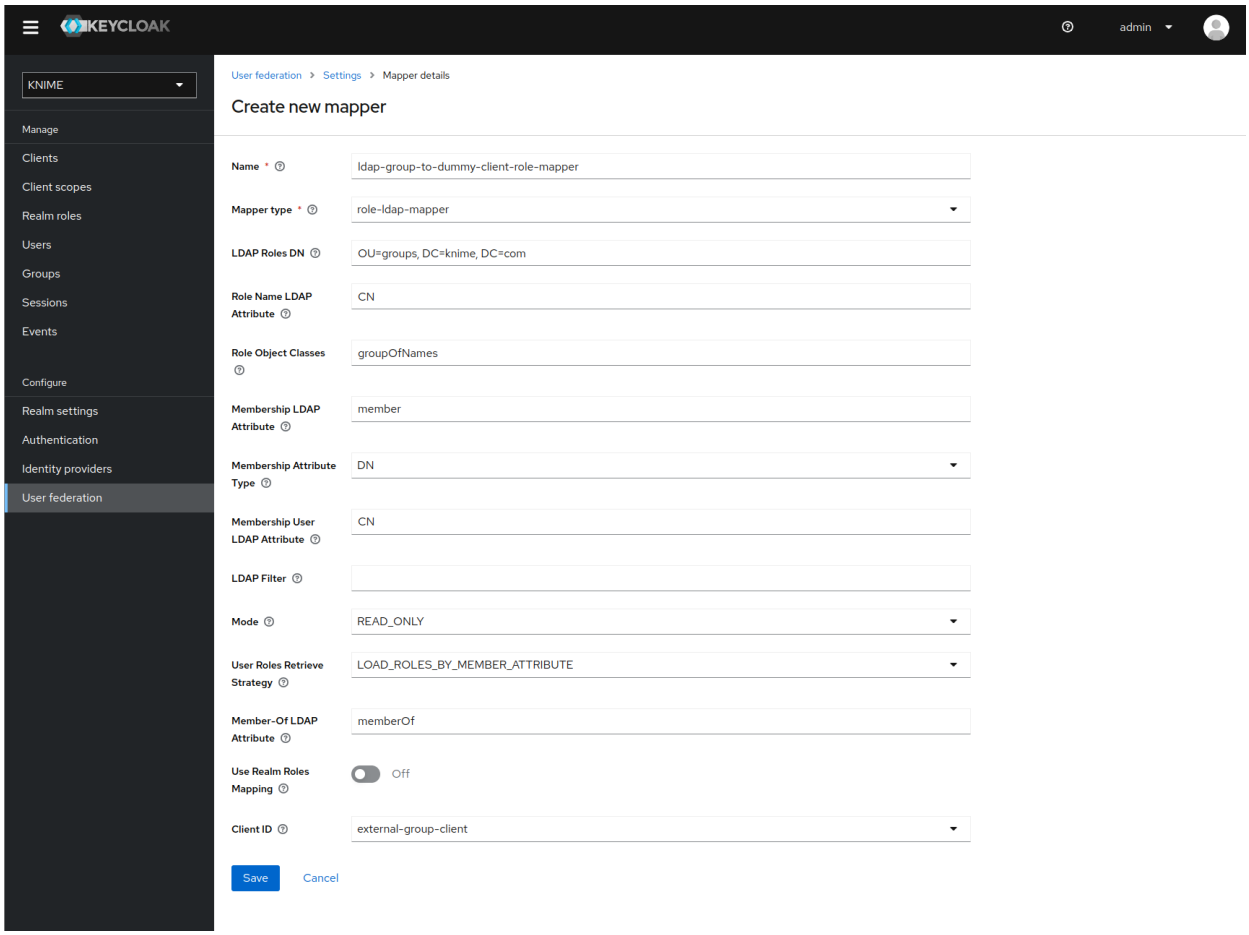


6. De-select all checkboxes of *Authentication flow* in the *Capability config* section, since this client will not require any capabilities
7. Enable *Client authentication*
8. Click on *Save*



Now that the dummy client is set up, you can proceed to create a mapper that maps the user groups from LDAP to roles inside the dummy client:

1. On the left tab select *User federation* and click on your LDAP configuration
2. Switch to the tab *Mappers*
3. Click on *Add mapper*
4. Provide a name, e.g. *ldap-group-to-dummy-client-role-mapper*
5. Set *Mapper type* to role-ldap-mapper
6. Setup the mapper according to your LDAP
7. Disable *User Realm Roles Mapping*
8. Set *Client ID* to the previously created dummy client (in our example *external-group-client*)
9. Click on *Save*

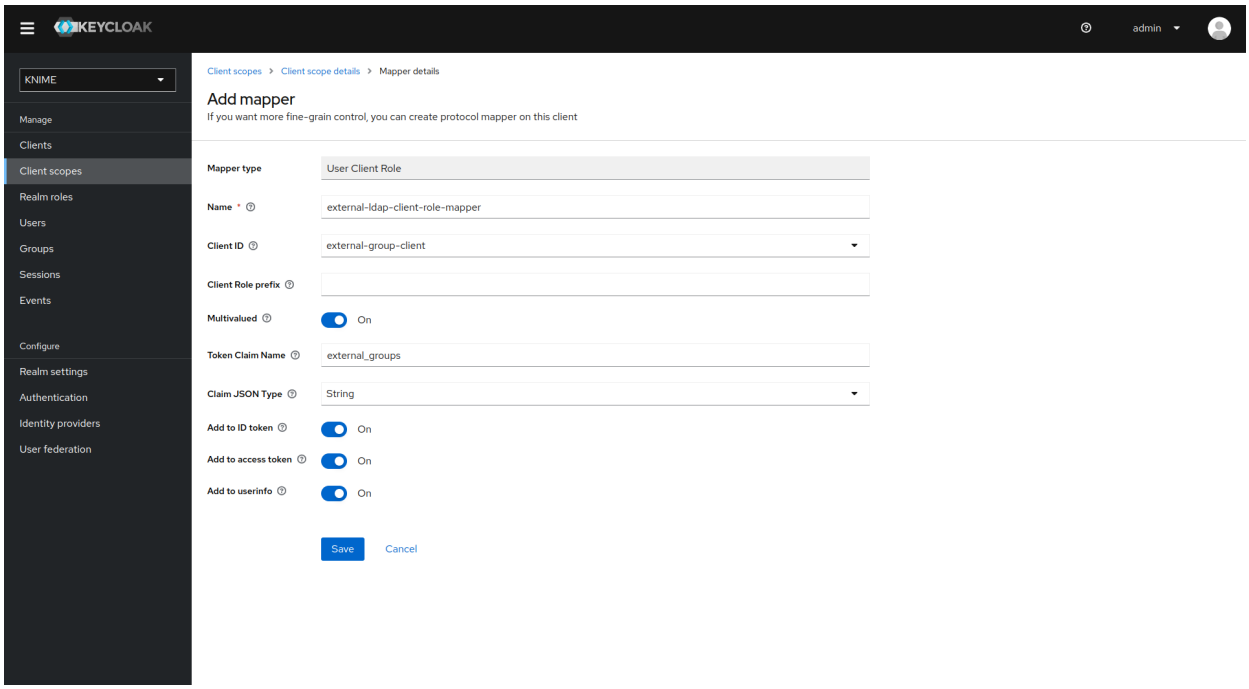


Now if a user logs in with the LDAP credentials the user's groups will be mapped to ad-hoc created client roles inside the *'external-group-client'*.

Next, you need to create a mapper that maps a user's realm roles from the dummy realm to the access tokens:

1. On the left tab select *Client scopes*
2. Select *groups*
3. Switch to the tab *Mappers*
4. Click on *Add mapper > By configuration* and select *User Client Role* from the list
5. Provide a name, e.g. *external-ldap-client-role-mapper*
6. Set *Client ID* to the previously created dummy client (in our example *external-group-client*)
7. Set *Token Claim Name* to *external-groups*
8. Set *Claim JSON Type* to *String*
9. Ensure that *Add to ID token*, *Add to access token*, *Add to userinfo*, and *Multivalued* are turned on

10. Click on *Save*



Enable external groups

Once you have configured the external groups in Keycloak you need to create the groups that you want to be available inside KNIME Business Hub.

To do so you have to make a `PUT` request to the corresponding endpoint:

```
PUT https://api.<base-url>/accounts/hub:global/groups/<external-group-name>
```

where <external-group-name> is the name of the group and it must match the group name in the external identity provider.

Docker executor images

In order to create execution contexts for their teams, team admins will need to reference the Docker image of the KNIME Executor that they want to use.

Public Docker executor images are made available by KNIME which correspond to the full builds of KNIME Executor versions 4.7.4 and higher.

The currently available executor images have the following docker image name:

- `registry.hub.knime.com/knime/knime-full:r-4.7.4-179`
- `registry.hub.knime.com/knime/knime-full:r-4.7.5-199`
- `registry.hub.knime.com/knime/knime-full:r-4.7.6-209`
- `registry.hub.knime.com/knime/knime-full:r-4.7.7-221`
- `registry.hub.knime.com/knime/knime-full:r-4.7.8-231`
- `registry.hub.knime.com/knime/knime-full:r-5.1.0-251`
- `registry.hub.knime.com/knime/knime-full:r-5.1.1-379`
- `registry.hub.knime.com/knime/knime-full:r-5.1.2-433`
- `registry.hub.knime.com/knime/knime-full:r-5.1.3-594`
- `registry.hub.knime.com/knime/knime-full:r-5.2.0-271`
- `registry.hub.knime.com/knime/knime-full:r-5.2.1-369`
- `registry.hub.knime.com/knime/knime-full:r-5.2.2-445`
- `registry.hub.knime.com/knime/knime-full:r-5.2.3-477`
- `registry.hub.knime.com/knime/knime-full:r-5.2.4-564`
- `registry.hub.knime.com/knime/knime-full:r-5.2.5-592`
- `registry.hub.knime.com/knime/knime-full:r-5.2.6-758`
- `registry.hub.knime.com/knime/knime-full:r-5.3.0-388`
- `registry.hub.knime.com/knime/knime-full:r-5.3.1-498`
- `knime/knime-full:r-5.3.2-563`
- `knime/knime-full:r-5.3.3-666`
- `knime/knime-full:r-5.4.0-62`
- `knime/knime-full:r-5.4.1-169`

- `knime/knime-full:r-5.4.2-185`

> With the release of 5.3.2 the docker image name **does not need** the `registry.hub.knime.com` prefix anymore. Find more information about KNIME Executor images provided by KNIME here.

> Please note that with the release of 5.2.2 KNIME executors will have HTML sanitization of old JavaScript View nodes and Widget nodes turned on by default. This ensures that no malicious HTML can be output. For more information check the KNIME Analytics Platform 5.2.2 changelog.

However you might want to customize the executor image, e.g. by adding specific extensions or default conda environments.

The following section explains how to do so.

Add extensions to an existing Docker image

In order to install additional extensions and features to the KNIME Executor image, you will need to first create a `Dockerfile`, build a Docker image from it, and finally make it available by pushing it to a registry. You can do this in two ways:

1. Via a data application, provided by us and described in the next section. Prerequisites:
    - The Execution Image Builder needs to be enabled: *KOTS Admin Console* > *Config* > *Execution Image Builder* > check *Enable Execution Image Builder*.
    - You need global admin privileges in order to build and push the image. Deploy the data app while configuring it with an application password of the global admin to make it available for others, e.g. team admins (the application password is not visible to others that way). If you use the data app not as a global admin, you can still create the Dockerfile, but you won't be able to build and push it to the registry.
2. Via a manual approach, as described below, if the Execution Image Builder service is not available. This might be the case for air gap installations, since the Execution Image Builder service references update sites, base images, and other external resources. Prerequisites:
    - `docker` should be installed.

If you need to install `docker` please make sure not to install it on the same virtual machine (VM) where the KNIME Business Hub instance is installed, as it might interfere with `containerd`, which is the container runtime used by Kubernetes.

Registry Prerequisites

Enabling the Image Builder allows to build an execution image in the cluster and to push it to a specified registry.

There are three different possibilities to specify a registry to which the Image Builder can have access:

1. If you are in an embedded cluster environment and want to use the default **Embedded Docker Registry**: activate the Embedded Docker Registry by going to *KOTS Admin Console > Config > Embedded Registry* and check the option *Enable Embedded Registry*. If you follow the manual approach, have the username and password ready that is defined in the KOTS Admin Console. See also KNIME Business Hub Installation Guide for more information.
2. If you are in an environment where the embedded registry is not available you will need to create a secret for the registry you want to use and point the Hub to the secret in the *Custom Execution Image Pull Secret* section of the KOTS Admin Console.
    a. First create a secret and add it to the cluster in the name spaces `hub` and `hub-execution` via the commands:

    ```
    kubectl -n hub-execution create secret docker-registry <secret-name> --docker
    -server=<registry-url> --docker-username=<username> --docker
    -password=<secret>
    kubectl -n hub create secret docker-registry <secret-name> --docker
    -server=<registry-url> --docker-username=<username> --docker
    -password=<secret>
    ```

    where `secret-name` is the name of the secret you want to create and `<registry-url>` is the URL of the registry you want to use.

    b. Point the Hub to the secret you added to the cluster. To do so go to the KOTS Admin Console, navigate to the *Execution Contexts* section and check the option *Enable Custom Execution Image Pull Secret*. Here under *Execution Image Pull Secret* add the `<secret-name>` from the command above. You will the be able to access the defined registry from the data app. If you are following the manual approach instead, you don't need to create the secret, but you will only need the URL, username, and password of the registry.

3. If you are in an air gapped environment instead you will need to specify a registry in the *Registry settings* tab of the KOTS Admin Console. If you follow the manual approach, have the username and password for this registry ready.

Building a Docker image requires enhanced container privileges. To be precise, the container needs to run as root with following capabilites:

- `CHOWN`
- `SETUID`
- `SETGID`
- `FOWNER`
- `DAC_OVERRIDE`

Depending on the size of the image, the build requires a lot of resources and can potentially slow down the KNIME Business Hub during the build time.

Using the Executor Image Builder data application

We provide an Executor Image Builder data application that will help you create the file, build an executor image, and push it to a registry to make it available when creating or editing an execution context. Download the data application, upload it to your KNIME Business Hub instance, and *Run* it.

Follow the steps 1 through 3 of the Data apps on KNIME Hub for detailed instructions.

The data app consists of five steps, displayed on five pages.

First Page: Settings

First, you'll find the basic settings.

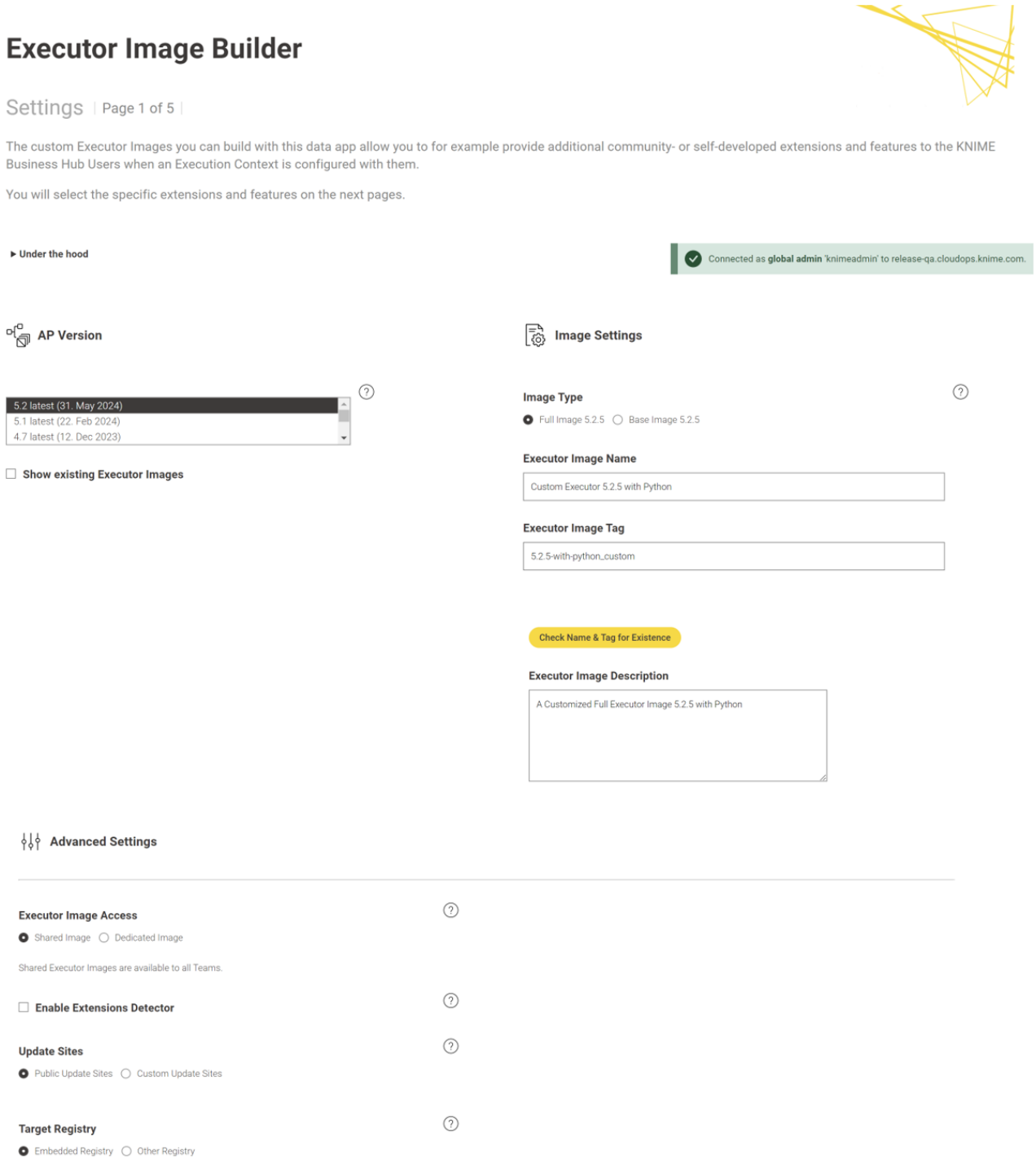


Figure 15. Executor Image Builder Settings on the first page as seen by a global admin.

If you hover over the setting or question mark in the data app, you'll find a detailed description of what each setting is for. Note that the settings related to image building and pushing are only available to global admins.

You proceed by clicking the *Next* Button.

Second Page: Extension Selection

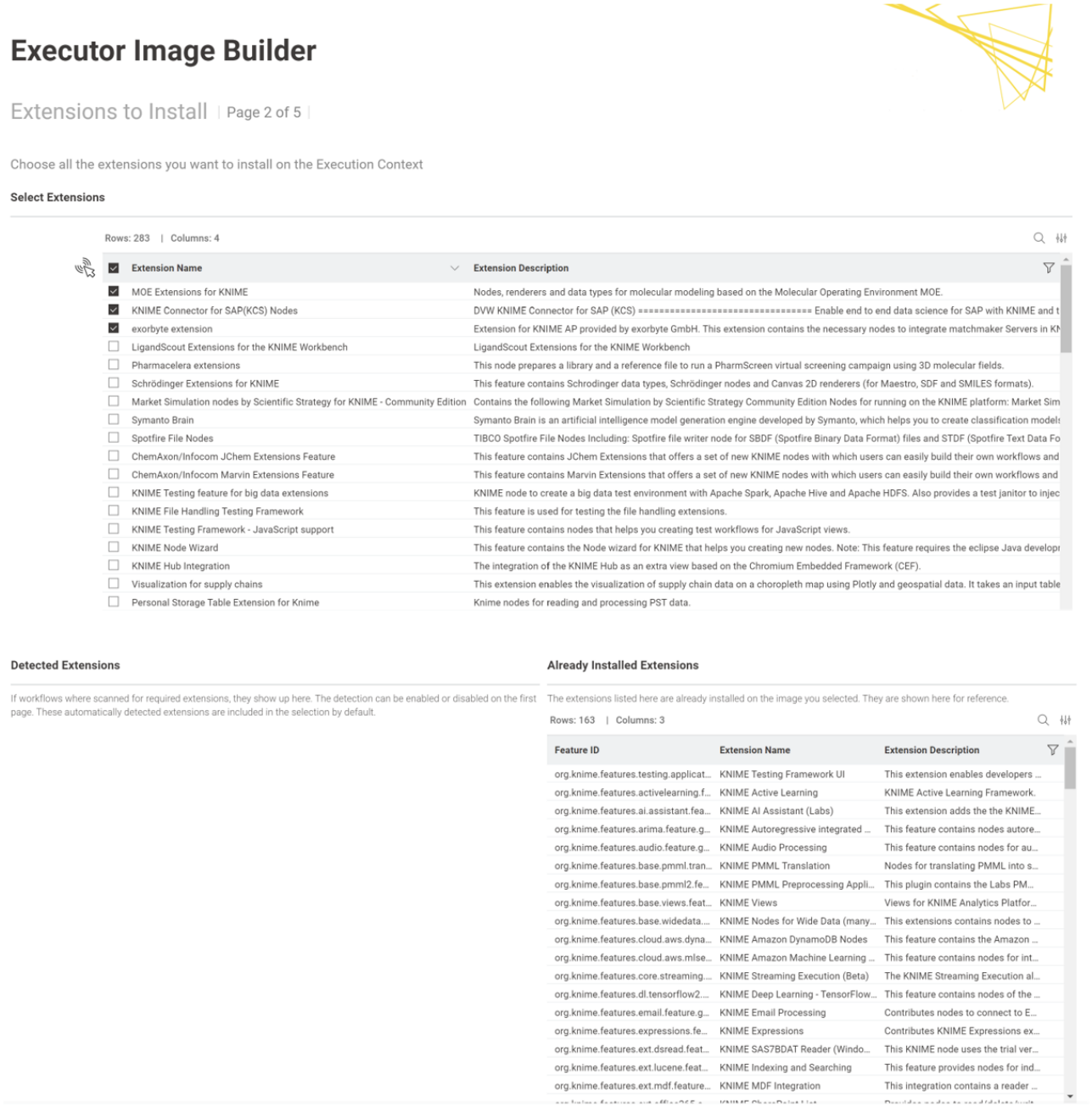


Figure 16. Executor Image Builder Extension selection on the second page.

This is the page where you select the extensions you want to install additionally. The bottom two tables review the extensions that were found by the automatic extension detection, as well as the extensions that are already installed in the base image.

Third Page: Integration Selection (R, Python)

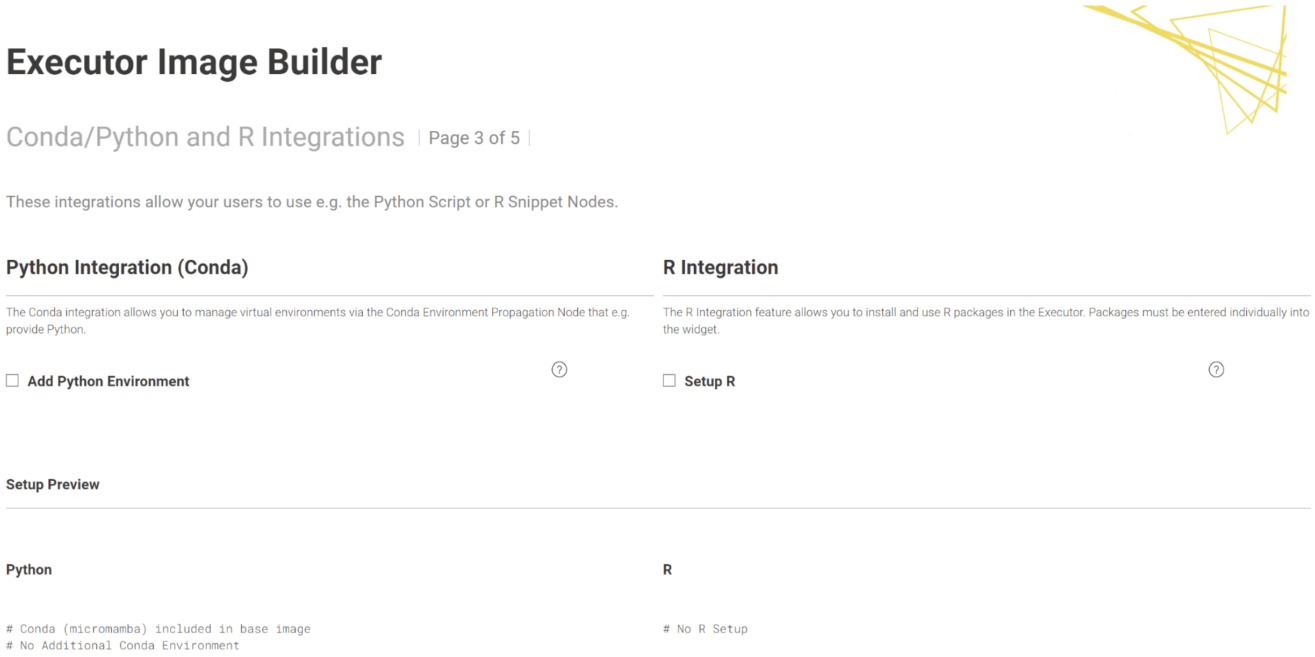


Figure 17. Executor Image Builder Python and R Integration configuration on the third page.

On the third page, you decide on whether to install conda/python, respectively additional default environments to the image, or the R integration.

Python

Learn more about how the KNIME Python Integration works with the KNIME Python Integration Guide

If your choice of image type (base or full) does not already include conda (specifically: micromamba as the management software of the virtual environments), you can choose to add it here. You need this, if the workflows you run on this executor contain e.g. a Conda Environment Propagation node.

If conda/micromamba is installed, you can further choose to provide a virtual environment that is already installed on the executor. This has the advantage that after a restart of the executor, the Conda Environment Propagation node executes faster, as the environment does not need to be recreated first. You add an environment by uploading a `.yml` file, which describes the environment. You can use the following example `.yml` file and modify it to your needs, e.g. by adding packages (and their versions) that you require:


```
name: py39_knime          # Name of the created environment
channels:                 # Repositories to search for packages
- conda-forge
dependencies:             # List of packages that should be installed
- python=3.9              # Python
- py4j                    # used for KNIME <-> Python communication
- nomkl                   # Prevents the use of Intels MKL
- pandas                  # Table data structures
- jedi=0.18.1             # Python script autocompletion
- python-dateutil         # Date and Time utilities
- numpy                   # N-dimensional arrays
- cairo                   # SVG support
- pillow                  # Image inputs/outputs
- matplotlib              # Plotting
- pyarrow=6.0             # Arrow serialization
- IPython                 # Notebook support
- nbformat                # Notebook support
- scipy                   # Notebook support
- python-flatbuffers<2.0 # because tensorflow expects a version before 2
- h5py<3.0                # must be < 3.0 because they changed whether str or byte is
returned
- protobuf>3.12           # Lower protobuf versions do not work with TensorFlow 2
- libiconv                # MDF Reader node
- asammdf=5.19.14         # MDF Reader node
- JPype1                  # Databases
```

To learn more about how to create conda environment files, visit the conda docs. If you want to pre-install multiple conda environments, please edit the Dockerfile manually on the next page of the data app.

In order for the executor to find the conda installation, the execution context needs to know where to find conda. This is done as a final step and is described below.

Unless modified, the path to the conda installation is `/home/knime/miniconda3/`, and the path to the environment (see below) is `<path to default conda environment dir>=<path to conda installation dir>/envs/<name of the env>`.

In the full images of KNIME Executor version 5.2 and above, where conda and Python are preinstalled, the path to conda is the same, i.e. `/home/knime/miniconda3/` and the paths to the default environments are `/home/knime/miniconda3/py2_knime` and `/home/knime/miniconda3/py3_knime`.

R

You can make R available as an integration if the workflows are expected to contain nodes from the KNIME R Scripting extension. Provide a comma-separated list of additional packages in the provided input field.

Fourth Page: Dockerfile Review

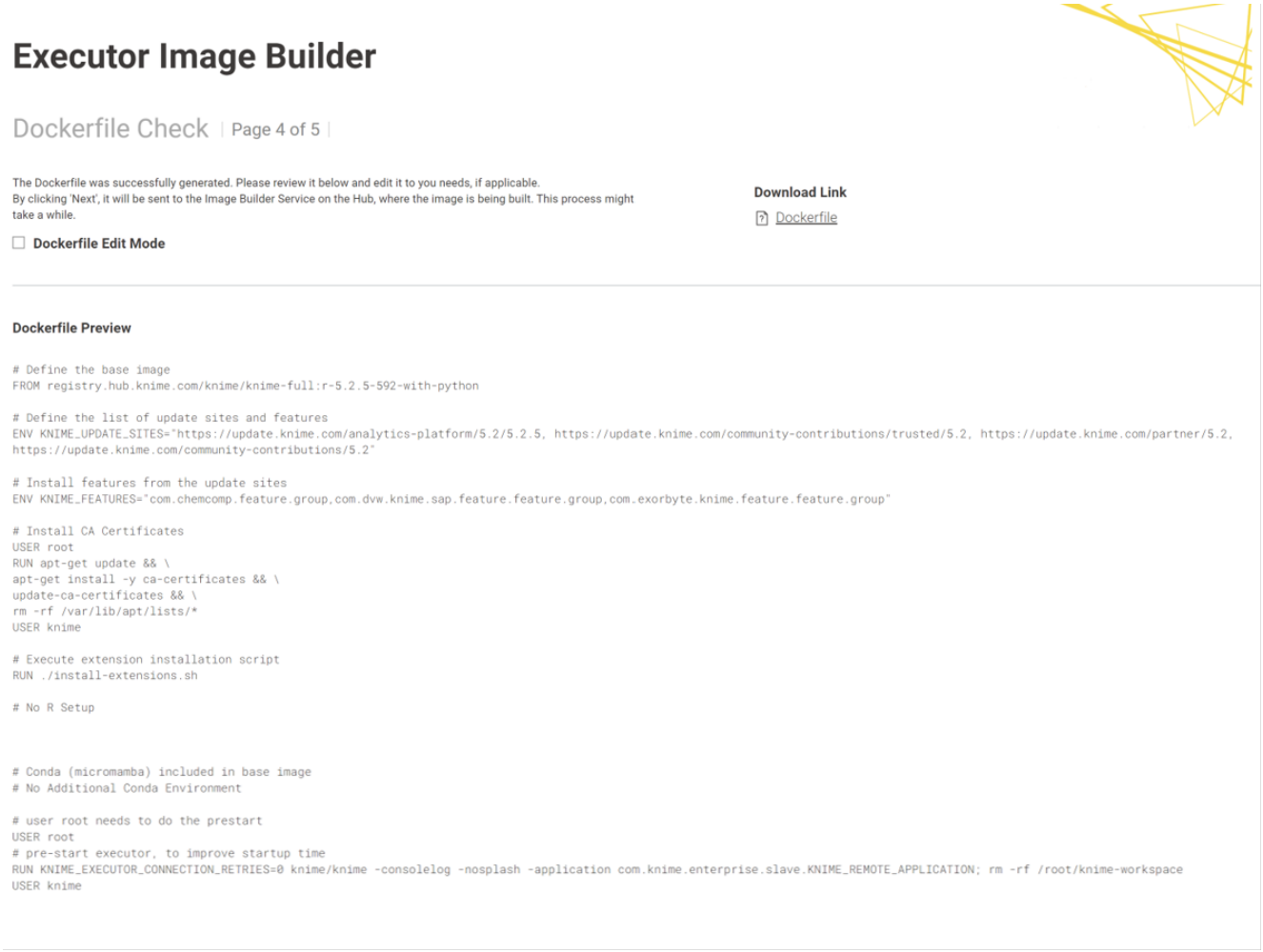


Figure 18. Executor Image Builder Dockerfile review on the fourth page. This is the final page for users that are not global admins.

Here you are able to review the Dockerfile and do manual edits, if needed. To do this, enable the Dockerfile Edit Mode, and change the Dockerfile based on your needs.

- If you are **not** connected as a **global admin**, you can proceed to build and push the Dockerfile manually.
- If you are the **global admin** and proceed to the next page, the Dockerfile is sent to the image builder endpoint, where the image is actually built, and then pushed to the executor image list.

Fifth Page: Dockerfile Building and Pushing

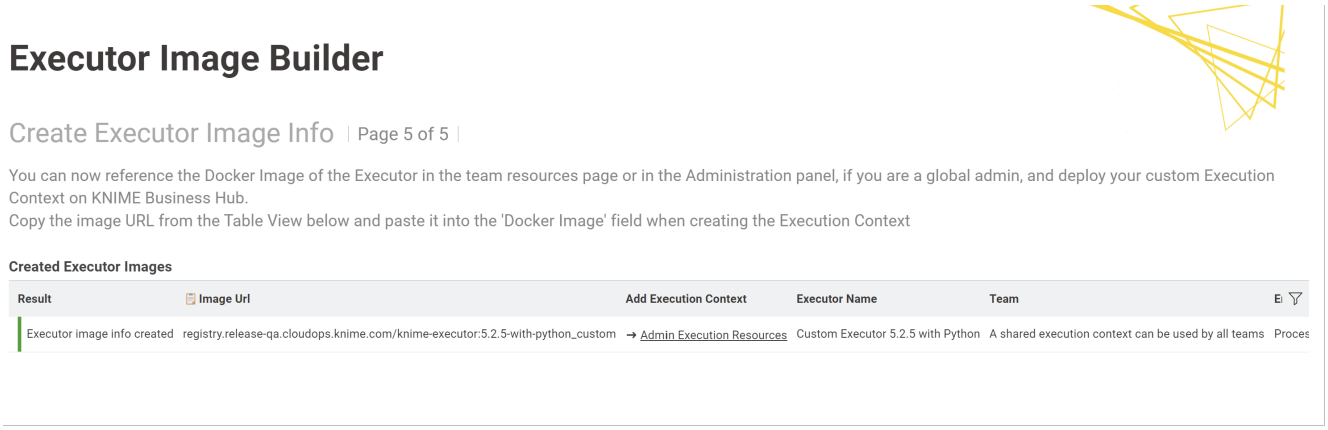


Figure 19. Executor Image Builder Dockerfile final page with the result of the building process. Copy the image URL and use it when creating or editing an execution context.

If you didn't encounter any errors during the building phase, the final image URL is shown. You can copy this, proceed to the Execution Resources page to create a new execution context, or edit an existing one, and paste the URL there.

Finally, go ahead and test the execution of a workflow containing a node from the freshly installed extensions or one that uses python.

Adding extensions manually

Instead of using the data app, you can also go through the steps manually. This might be necessary for air gap installations, as the service that builds the docker image would require access to e.g. update sites and other resources that are referenced in your docker file.

Create the Dockerfile

You can use the example below which demonstrates how to extend an existing executor image with a custom set of update sites and features.

```
# Define the base image
FROM registry.hub.knime.com/knime/knime-full:r-5.3.1-498

# Change to root user to be able to install system packages
USER root

# Update/upgrade package manager and install ca-certificates to enable ca certificates
that micromamba (for python) is asking for
RUN apt-get update && \
    apt-get upgrade -yq && \
    apt-get install -yq \
        ca-certificates && \
    # cleanup
    rm -rf /var/lib/apt/lists/*

# Change to knime user to handle extensions
USER knime

# Define the list of update sites and features
# Optional, the default is the KNIME Analytics Platform update site (first entry in the
list below)
ENV KNIME_UPDATE_SITES=https://update.knime.com/analytics-
platform/5.2,https://update.knime.com/community-contributions/trusted/5.2
# Install a feature from the Community Trusted update site
ENV KNIME_FEATURES="org.knime.features.geospatial.feature.group"

# Execute extension installation script
RUN ./install-extensions.sh
```

The `KNIME_UPDATE_SITES` environment variable determines the update sites that will be used for installing KNIME Features. It accepts a comma-delimited list of URLs. The `KNIME_FEATURES` environment variable determines the extensions which will be installed in the KNIME Executor. It accepts a comma-delimited list of feature group identifiers. A corresponding update site must be defined in the `KNIME_UPDATE_SITES` list for feature groups to be successfully installed. You can get the necessary identifiers by looking at *Help → About KNIME → Installation Details → Installed Software* in a KNIME instance that has the desired features installed. Take the identifiers from the "Id" column and make sure you do not omit the `.feature.group` at the end (see also screenshot on the next page). The base image contains a shell script `install-extensions.sh` which lets you easily install additional extensions in another `Dockerfile`.

The previous version of the Executor Image Builder allowed to create the Dockerfile, just like the Executor Image Builder Data App allows for non-global admins. You can use the resulting Dockerfile as a starting point.

Python Integration

If you want the executor to run workflows with the Conda Environment Propagation node, you need to install conda (respectively micromamba or any other conda environment manager). You can do so by adding the following lines to the Dockerfile:

```
# START Conda setup
# install additional packages
USER root
RUN apt-get update && \
apt-get install -y curl bzip2 && \
rm -rf /var/lib/apt/lists/*
USER knime
SHELL [ "/usr/bin/bash", "-o", "pipefail" ,"-c" ]
RUN set -x && \
mkdir -p ~/.local/bin && \
curl -Ls https://micro.mamba.pm/api/micromamba/linux-64/latest | tar -xvj -C
${HOME}/.local/bin/ --strip-components=1 bin/micromamba && \
$HOME/.local/bin/micromamba shell init -s bash -p ~/miniconda3
ENV MAMBA_ROOT_PREFIX=/home/knime/miniconda3
USER root
RUN ln -s /home/knime/.local/bin/micromamba /usr/local/bin/micromamba
USER knime
# legacy conda support
RUN micromamba install -y -n base conda -c conda-forge
# END Conda setup
```

If you additionally want to provide pre-installed environments, add them to the Dockerfile like below:

```
# START Python Environment Setup
RUN echo $'\
name: py39_knime            # Name of the created environment\n\
channels:                   # Repositories to search for packages\n\
- conda-forge\n\
dependencies:               # List of packages that should be installed\n\
- python=3.9                # Python\n\
- py4j                      # used for KNIME <-> Python communication\n\
- nomkl                     # Prevents the use of Intels MKL\n\
- pandas                    # Table data structures\n\
- jedi=0.18.1               # Python script autocompletion\n\
- python-dateutil           # Date and Time utilities\n\
- numpy                     # N-dimensional arrays\n\
- cairo                     # SVG support\n\
- pillow                    # Image inputs/outputs\n\
- matplotlib                # Plotting\n\
- pyarrow=6.0               # Arrow serialization\n\
- IPython                   # Notebook support\n\
- nbformat                  # Notebook support\n\
- scipy                     # Notebook support\n\
- python-flatbuffers<2.0    # because tensorflow expects a version before 2\n\
- h5py<3.0                  # must be < 3.0 because they changed whether str or byte is
returned\n\
- protobuf>3.12             # Lower protobuf versions do not work with TensorFlow 2\n\
- libiconv                  # MDF Reader node\n\
- asammdf=5.19.14           # MDF Reader node\n\
- JPype1                    # Databases\n\
' > /tmp/py39_knime.yml
RUN set -x && \
micromamba env create -f /tmp/py39_knime.yml
# END Python Environment Setup
```

Learn more about the KNIME Python Integration with the KNIME Python Integration Guide.

In order for the executor to find the conda installation, the execution context needs to know where to find conda. This is done as a final step and is described below.

Unless modified, the path to the conda installation is `/home/knime/miniconda3/`, and the path to the environment (see below) is `<path to default conda environment dir>=<path to conda installation dir>/envs/<name of the env>`.

In the full images of KNIME Executor version 5.2 and above, where conda and Python are preinstalled, the path to conda is the same, i.e. `/home/knime/miniconda3/` and the paths to the default environments are `/home/knime/miniconda3/py2_knime` and `/home/knime/miniconda3/py3_knime`.

Build a Docker image from the Dockerfile

Once the `Dockerfile` has been customized appropriately, you can build a Docker image from it by using the following command after replacing `<image_name>` and `<tag_name>` with actual values:

```
docker build -t <image_name>:<tag_name> .
```

This process can take a few minutes to be completed. In order to check if the new image has been built you can use the command `docker images`.

Push the Docker image to the Docker Embedded Registry

Finally you can push the image to the Docker Embedded Registry.

1. Authenticate against the registry with the credentials obtained from the *KOTS Admin Console > Config > Embedded Registry* via

   ```
   docker login --username <username> registry.<base-url>
   ```

   > If TLS is not configured, the registry URL must be added as an insecure registry.

2. Tag the previously created image with the format of the Embedded Docker Registry

   ```
   docker tag <old-name> registry.<base-url>/<new_name>
   ```

3. Push the image to the Embedded Docker Registry

   ```
   docker push registry.<base-url>/<image_name>:<tag_name>
   ```

4. Now the Docker image (e.g. `registry.hub.example.com/knime-full:5.2-with-additional-extension`) is available to create an execution context from the Hub UI.

Push the Docker image to the Container Embedded Registry for air gap installations

For air gap installations you will need to create a custom Docker image and push it to the Container Embedded Registry using `containerd` command line (`ctr`).

As Containerd is installed as container runtime in the cluster, you can make use of the `ctr` commands to pull and push the images into the embedded registry.

1. Build the image on a machine with access to the internet and installed `docker`:

```
docker build . -t registry.<base-url>/<image_name>:<tag_name>
```

2. Save the Docker image as a `tar` file on the machine where Docker is installed:

```
docker save -o docker.tar <tag_name>
```

3. Copy the image to the machine where the Hub instance is running
4. On the machine where the Hub instance is running, import the image into `containerd`:

```
ctr image import docker.tar
```

5. Tag the image:

```
ctr image tag <old-image_name>[<old-tag_name>] registry.<base-
url>/<image_name>:<tag_name>
```

6. Push the image to the Container Registry:

```
ctr images push --user knime -k registry.<base-url>/<image_name>:<tag_name>
```

   where `-k` parameter is to skip the TLS check, and `--user` parameter is to provide the username for the registry.

7. Now you can verify that the images are available on the Container Registry using the below endpoints:

```
http://registry.<base-url>/v2/_catalog
http://registry.<base-url>/v2/<repo>/tags/list
```

Push the Docker image to a non-embedded Registry

For non-embedded registries you will push the image from your local Docker instance to the remote registry, using the authentication required for that particular registry. Then you can use the image name to create the executor images, given that you have added the correct pull secrets to the KNIME Business Hub configuration.

Python and Conda in Docker images

When you create an Execution Context on KNIME Business Hub based on a full build you will have the Python environment bundled with the KNIME Python Integration available. If you need additional libraries or are using the Python 2 (legacy) extension, you need to create a custom Python environment to make them available on the Hub instance.

You can do this in several ways:

1. You can use the Conda Environment Propagation node in all your workflows using Python. To get started with the Conda Environment Propagation node, check out the KNIME Python Integration Guide. This has the advantage that no further setup is needed, and you are done with this guide. Any libraries installed using the Conda Environment Propagation node will be removed, however, when the executor restarts and are installed again next time the node executes, so libraries that are used often should be installed as part of the executor Docker image to save time. This is described in the following.
2. You can customizing the executor image. To do so, you need to create a Docker image with Python, either via the Executor Image Builder data application described above, or by creating (Dockerfile examples), building, and pushing the Dockerfile manually. Be sure to note down the paths where conda was installed, as you will need add them in the `.epf` file of the customization profile during the set up of the execution context. The default installation paths are:

   ```
   <path to conda installation dir>        = /home/knime/miniconda3/
   <path to default conda environment dir> = <path to conda installation
   dir>/envs/<name of the env>
   ```

3. Use an full image of version 5.2 and above, since starting with KNIME Executor version 5.2 the KNIME Python extension is already installed in the Docker images. There are two default environments installed, py2_knime and py3_knime:
   - `py2_knime`:

```
name: py2_knime             # Name of the created environment
channels:                   # Repositories to search for packages
  - defaults
  - conda-forge
dependencies:               # List of packages that should be installed
  - python=2.7              # Python
  - pandas=0.23             # Table data structures
  - jedi=0.13               # Python script autocompletion
  - parso=0.7.1             # Jedi dependency this is the last version
compatible with 2.7
  - python-dateutil=2.7     # Date and Time utilities
  - numpy=1.15              # N-dimensional arrays
  - cairo=1.14              # SVG support
  - pillow=5.3              # Image inputs/outputs
  - matplotlib=2.2          # Plotting
  - pyarrow=0.11            # Arrow serialization
  - IPython=5.8             # Notebook support
  - nbformat=4.4            # Notebook support
  - scipy=1.1               # Notebook support
  - jpype1=0.6.3            # Databases
  - protobuf=3.5            # Serialization for deprecated Python nodes
```

- py3_knime:

```
name: py3_knime             # Name of the created environment
channels:                   # Repositories to search for packages
  - defaults
  - conda-forge
dependencies:               # List of packages that should be installed
  - nbformat=4.4            # Notebook support
  - scipy=1.1               # Notebook support
  - pillow=5.3              # Image inputs/outputs
  - cairo=1.14              # SVG support
  - ipython=7.1             # Notebook support
  - numpy=1.16.1            # N-dimensional arrays
  - python=3.6              # Python
  - matplotlib=3.0          # Plotting
  - jpype1=0.6.3            # Databases
  - pyarrow=0.11            # Arrow serialization
  - jedi=0.13               # Python script autocompletion
  - python-dateutil=2.7     # Date and Time utilities
  - pandas=0.23             # Table data structures
  - libiconv=1.15           # MDF Reader node
  - asammdf=5.19.14         # MDF Reader node
```

If you choose to modify the executor image or use the full build of version 5.2 and above, you further need to set up the execution context for it to know where to find the conda/python installations. This is described in the section below.

Set up the execution context

Once you have created the Docker image with Conda/Python and the desired environments, create an execution context that uses the newly created Docker image.

Now you need to set up and customize the execution context. This process is described in the KNIME Python Integration Guide in detail, and the relevant parts are repeated here.

You specify the paths where the execution context will find the conda installation and environments in a customization profile applied it to the execution context.

1. Build the `.epf` file by following the steps in KNIME Python Integration Guide and exporting the `.epf` file. To export the `.epf` file from KNIME Analytics Platform go to *File > Export Preferences…*
2. Open the file and use only the parts related to Python/conda.

The `.epf` file could look like the following:

```
/instance/org.knime.conda/condaDirectoryPath=<path to conda installation dir>
/instance/org.knime.python3.scripting.nodes/pythonEnvironmentType=conda
/instance/org.knime.python3.scripting.nodes/python2CondaEnvironmentDirectoryPath=<path
to default conda environment dir>
/instance/org.knime.python3.scripting.nodes/python3CondaEnvironmentDirectoryPath=<path
to default conda environment dir>
```

Find more details on how to setup the `.epf` file in the Executor configuration section of the KNIME Python Integration Guide.

Now follow these steps to customize the execution context:

1. Build the `.zip` file containing the customization profile using the `.epf` file you just created.
2. Upload the customization profile `.zip` file to KNIME Business Hub.
3. Apply the customization profile to the execution context.

You are done, and can test the setup by running a workflow that contains a Conda Environment Propagation node.

Delete a custom Docker image

In this section you can find instructions on how to delete the Docker images that you pushed to the Embedded Docker Registry. This is especially important since the MinIO storage

allocated for the registry is, by default, limited to ~30 GB.

To check how much disk space is occupied you can run the command `kubectl exec -it -n minio minio-<id> -- /bin/sh -c "df -h"` with the correct `minio` pod id. The `/data` directory contains the space occupied by the registry.

The first step to delete the custom Docker images is to use the following script:

delete_registry_image.sh

```
#!/bin/bash

# exit when any command fails
set -e

registry='registry.<base-url>'

# concatenates all images listed in json file into single line string seperated with
blank
echo "Image Name:"
read images
echo "Image Tag (Space seperated for multiple tags or leave empty if all should be
deleted):"
read tags
echo "Registry User:"
read user
echo "Registry Password:"
read -s password

for image in $images; do

    if [[ -z $tags ]]
    then
        # get tag list of image, with fallback to empty array when value is null
        tags=$(curl --user $user:$password "https://${registry}/v2/${image}/tags/list" |
jq -r '.tags // [] | .[]' | tr '\n' ' ')
    fi

    echo "DELETING image: " $image
    echo "DELETING tags: " $tags

    # check for empty tag list, e.g. when already cleaned up
    if [[ -n $tags ]]
    then
        for tag in $tags; do
            curl --user $user:$password -X DELETE
"https://${registry}/v2/${image}/manifests/$(
                curl --user $user:$password -I \
                    -H "Accept: application/vnd.docker.distribution.manifest.v2+json" \
                    "https://${registry}/v2/${image}/manifests/${tag}" \
```

```
                | awk '$1 == "docker-content-digest:" { print $2 }' \
                | tr -d $'\r' \
            )"
            echo "DELETED:" $image "with tag" $tag
        done
        echo "Deleted images. To free disk space run the garbage collection command
inside the registry pod: 'bin/registry garbage-collect --delete-untagged /
etc/docker/registry/config.yml'. See documentation at:
https://docs.knime.com/latest/business_hub_admin_guide/index.html#garbage-collection"
    else
        echo "SKIP:" $image
    fi
done
```

To run the script you will need:

1. `jq`: `jq` is a lightweight and flexible command line JSON processor that is used to format the JSON output of `curl` calls. To install it on the machine where you want to run the shell script you can use the following command:

   ```
   sudo apt-get update
   sudo apt-get -y install jq
   ```

2. You will need to adapt the value `registry.<base-url>` at the line 6 of the script with the `<base-url>` of your KNIME Business Hub, e.g. for `hub.example.com` will be `registry.hub.example.com`.
3. You will need to know the Embedded Docker Registry username and password.

If your KNIME Business Hub instance does not have TLS enabled, the script will cause SSL certificate issues.

To solve this you can:

1. Change the script where a `curl` command is calling an `https://` endpoint to call an `http://` instead, or
2. Add `--insecure` to each line of the script with a `curl` command.

Identify Docker Image names and tags

To be able to run the script you will need to know the image names and tags present on the embedded registry that you want to delete. You can for example run the following `GET` requests against the Docker Registry API. Again you will need to first adapt the `<base-url>` entries to your specific Hub URL, and use your `<username>` and `<password>` for the Embedded

Docker Registry.

```
# listing images on the remote registry
$ curl -u <username>:<password> -X GET registry.<base-url>/v2/_catalog | jq
```

This should output a list of the Docker Images available:

```
{
    "repositories": ["executor-image-name1",
                     "executor-image-name2",
                     "executor-image-name3"]
}
```

Then you can run the following command for each Docker Image you are interested in, e.g. `executor-image-name1`, to retrieve the Image tag:

```
# listing tags on the remote registry
$ curl -u <username>:<password> -X GET registry.<base-url>/v2/<executor-image-
name>/tags/list | jq
```

This should output a list of the Docker Image's tags:

```
{
    "name": "executor-image-name1",
    "tags": ["tag1"]
}
```

Run the script

Now you can run the script to delete one or multiple image tags. The script will ask you to provide an Image name, one or more Image tag and the Embedded Docker Registry username and password.

```
$ ./delete-registry-image.sh

Image Name:
<image-name>
Image Tag (Space seperated for multiple tags or leave empty if all should be deleted):
<image-tag>
Registry User:
<usr>
Registry Password:
<pwd>
```

Run garbage collection on the `registry` pod

Since the above script only removes the image tags and manifests, leaving the actual image layers in storage, garbage collection is required to scan the registry storage and remove unreferenced or orphaned layers, reclaiming disk space and fully cleaning up after the deletion.

The garbage collection needs to be performed on one of the `registry` pods inside the `kurl` namespace.

You can either use the following commands via `kubectl` or use your preferred tool to manage the cluster, e.g. OpenLens.

1. Connect to the cluster where the Business Hub instance is installed
2. List the pods in the `kurl` namespace to find the `registry` pod on which you will run the garbage collection

   ```
   kubectl get pods -n kurl
   ```

   a. Identify the `registry` pod and proceed with the next steps

   b. You **only** need to perform garbage collection on one registry pod in the kurl namespace. There is no need to do it for the other registry pods. This is because all the registry pods in the Kubernetes cluster share the same underlying storage.
3. Next, open a shell into the selected registry pod, `<registry-name>`, and ensure you select the correct container

   a. Do **not** use the `registry-backup` container, to avoid risk of data loss

   b. Do not upload an image during garbage collection. If you were to upload an image while garbage collection is running, there is the risk that the image's layers are mistakenly deleted leading to a corrupted image.
4. Run the garbage collection command inside the shell:

   ```
   kubectl exec -it -n kurl <registry-name> -c registry -- /bin/sh -c "/bin/registry
   garbage-collect --delete-untagged /etc/docker/registry/config.yml"
   ```

After the garbage collection has run through, it takes a while to fully free the disk space. You can speed up the process by restarting the `registry` and `minio` pods. To check how much disk space is occupied now run the command `kubectl exec -it -n minio minio-<id> — /bin/sh -c "df -h"` with the correct `minio` pod id again. The `/data` directory contains the space occupied by the registry.

Even though all tags of an image are deleted, the image is still listed in the MinIO pod under the `minio` namespace when running a `GET` request to the `registry.<base-url>/v2/_catalog` endpoint. However, these images should not have any tags and therefore they do not occupy disk space anymore.

Customization profiles

Customization profiles are used to deliver KNIME Analytics Platform configurations from KNIME Hub to KNIME Analytics Platform clients and KNIME Hub executors.

This allows defining centrally managed:

- Update sites
- Preference profiles (such as Database drivers, Python/R settings)

A profile consists of a set of files that can:

- Be applied to the client during startup once the KNIME Analytics Platform client is configured. The files are copied into the user's workspace.
- Be applied to the KNIME Hub executors of the execution contexts.

Customization profiles can be:

- Global customization profiles that need to be uploaded and managed by the Global Admin. These can be applied across teams via shared execution context or to specific teams.
- Team's customization profiles, which are scoped to the team, can be uploaded either by a Global Admin or a team admin.

Once uploaded, the customization profile can then be downloaded or used in KNIME Analytics Platform clients and executors.

The access to the customization profile is not restricted meaning that anyone with the link can download it and use it.

Currently, customization profiles can be managed on KNIME Hub via REST or using the dedicated Customization Profile data application available here.

Structure of a customization profile

A customization profile minimally consists of a folder, named according to the *profile name*, containing at least one *preference file*. A preference file is a simple text file with the extension `.epf`.

Each line in a preference (`.epf`) file specifies key and value of a setting, separated by `=`.

```
<key1>=<value1>
<key2>=<value2>
# ...
```

If two lines specify identical keys, the value later in the file overrides a value specified earlier.

If the profile folder contains more than one `.epf` files, the files are read in lexicographic order.

A customization profile may contain additional arbitrary files. These are distributed as part of the profile and can be referenced in `.epf` files.

Variable replacement

It is possible to use variables inside the preference files (only those files ending in `.epf`) which are replaced on the client right before they are applied. This makes the Hub-managed customizations even more powerful. These variables have the following format: `${prefix:variable-name}`. The following prefixes are available:

- **`env`**: the variable is replaced with the value of an environment value. For example, `${env:TEMP}` will be replaced with `/tmp` under most Linux systems.
- **`sysprop`**: the variable is replaced with a Java system property.
  For example, `${sysprop:user.name}` will be replaced with the current user's name. For a list of standard Java system properties see the JavaDoc. Additional system properties can be defined via `-vmargs` in the `knime.ini`.
- **`profile`**: the variable will be replaced with a property of the profile in which the current preference file is contained in. Currently `location` and `name` are supported as variable names. For example, `${profile:location}` will be replaced by the file system location of the profile on the client. This can be used to reference other files that are part of the profile, such as database drivers:
  `org.knime.workbench.core/database_drivers=${profile:location}/db-driver.jar`

In case you want to have a literal in a preference value that looks like a variable, you have to use two dollar signs to prevent replacement. For example `$${env:HOME}` will be replaced with the plain text `${env:HOME}`. If you want to have two dollars in plain text, you have to write three dollars (`$$${env:HOME}`) in the preference file.

Once you use variables in your preference files, they are not standard Eclipse preference files anymore and cannot be imported as they are.

Create a customization profile

Follow the steps below to create a customization profile. You can export the preference file from a KNIME Analytics Platform installation, with the needed configuration.

Then create a folder with the preference file and any additional file that you might need to distribute with the customization profile. Finally, you compress the folder to a `.zip` format and

1. Set up the needed configuration in a local KNIME Analytics Platform installation.
2. Go to *File* → *Export Preferences* and select a location.
3. Open the created `.epf` file, and look for the lines related to your needed settings. Remove all other settings (as some contain e.g. local paths on your machine, which will inevitably cause issues when applying to another installation). You can also further modify the file with the customization options below.
4. Place the `.epf` file in a folder, together with any additional files that need to be distributed along the profile (e.g. database drivers).
5. Create a `.zip` from that folder.

Finally you can proceed with the next step, and upload the file to the KNIME Hub instance.

When creating a zip file on macOS using the built-in functionality, two files are automatically added that cause the next steps (i.e. applying the profile in Analytics Platform) to fail. There is a way to prevent creation of these files if creating the `.zip` via command line, see here. If in doubt, use a Windows or Linux machine to create the `.zip` file.

The customization profiles on the KNIME Hub instance are going to be accessible without user authentication. Therefore, they shouldn't contain any confidential data such as passwords.

For further details and an example on how to distribute JDBC driver files, go to the Hub-managed customization profiles section of the KNIME Database Extension Guide.

Customization options

Besides the preferences that are exportable by KNIME Analytics Platform, there are additional settings that can be added to the preference files to customize clients.

Since KNIME Analytics Platform 5.3, some complex settings can be specified in YAML

format.

1. Create a file `<filename>.yml` in the profile directory.
2. Reference the file in a preference (`.epf`) file by setting the following:

   ```
   /instance/org.knime.core/knime.core.ap-customization-
   configuration=${profile:location}/<filename>.yml
   ```

   where `<filename>.yml` is the name of the YAML file you created.

> The YAML file should declare the document version at the beginning of the file, i.e. `version: 'customization-v1.0'`. This is to ensure that the document can still be read in the future even if the format changes.

Restrict access to nodes

The property path `nodes.filter` allows to configure a sequence of *filters* which are evaluated on startup of the KNIME Analytics Platform.

The `scope` of a filter expresses to what extent access is restricted by this filter.

1. `view`: If a node does not match all filters with `view` scope, it can be loaded, configured and executed as part of a workflow, but it will not be presented as a choice in the node repository or similar places.
2. `use`: If a node does not match all filters with `use` scope, it will not be loaded and will not appear as a choice.

> `use` takes precedence over `view`. Meaning that a node that cannot be used can also never be viewed, regardless of whether it matches filters with the `view` scope.

Whether a node matches a filter is defined by its `predicate`, consisting of one or several regular expressions. The regular expression patterns are evaluated against the node factory class name. The value of `rule` (`allow` or `deny`) specifies whether the predicate is applied as-is or inverted.

Examples

- Completely ignore any Java Snippet or Row Filter nodes. Any other nodes are not affected.


```
version: 'customization-v1.0'
nodes:
  filter:
    - scope: use
      rule: deny
      predicate:
        type: pattern
        patterns:
          - .+JavaSnippet.+
          - .+RowFilter.+
        isRegex: true
```

- Completely ignore any Java Snippet node. Restrict node repository and related user interface elements to nodes from given modules.

```
version: 'customization-v1.0'
nodes:
  filter:
    - scope: use
      rule: deny
      predicate:
        type: pattern
        patterns:
          - .+JavaSnippet.+
        isRegex: true
    - scope: view
      rule: allow
      predicate:
        type: pattern
        patterns:
          - org\.<vendor>\.<package>\..+
          - org\.<vendor>\.<package>\..+
        isRegex: true
```

Custom UI menu entries

In the YAML customization file, use the property `ui.menuEntries` as illustrated by the following example.

```
version: 'customization-v1.0'
ui:
  menuEntries:
    - name: "My custom entry"
      link: "https://help.company.com/knime"
    - name: "Another entry"
      link: "https://support.company.com/contact"
```

These entries will appear in the *Help* menu revealed by the button at the top-right of the KNIME Analytics Platform user interface.

To configure custom menu entries in the **classic user interface**, use the following settings in a preference (`.epf`) file.

`/instance/com.knime.customizations/helpContact.buttonText=<label>`

If set together with `/instance/com.knime.customizations/helpContact.address` a button with the provided label will occur under `Help` in KNIME Analytics Platform. Clicking on the button will, depending on the `helpContact.address`, either open the default mail client or the default browser with the provided address.

`/instance/com.knime.customizations/helpContact.address=<uri>`

Sets the address of the support contact, e.g. `mailto:support@company` or `https://company/support`.
This option only takes effect in combination with `/instance/com.knime.customizations/helpContact.buttonText`.

`/instance/com.knime.customizations/documentation.buttonText=<label>`

Sets the label of the documentation button that can be found under `Help` in KNIME Analytics Platform. Clicking on the button will open the default browser and navigate to the documentation. If set to `-` the button will be hidden.

`/instance/com.knime.customizations/documentation.address=<uri>`

Sets the address of the documentation, e.g. `https://company/documentation` or `file:///sharedSpace/documentation`.
By default, the documentation address points to the KNIME Analytics Platform documentation.

Disallow storage of weakly encrypted passwords

You can use the `workflow.disablePasswordSaving` property to configure whether or not it should be possible to save weakly encrpypted passwords in workflows.

Example

- Prevent user to save passwords in workflows:

```
version: 'customization-v1.0'
workflow:
  disablePasswordSaving: true
```

Disable KNIME AI Assistant on KNIME Analytics Platform client version 5.4.0 or above

The `kai.disable` property allows you to disable the KNIME AI Assistant from the KNIME Analytics Platform.

> If you are using the KNIME Analytics Platform of version 5.3.x or below you need to set up the corresponding parameter in the `knime.ini` file.

Example

- Disable K-AI on KNIME Analytics Platform

```
kai:
    disable: true
```

Restrict which KNIME Hubs the KNIME AI Assistant is allowed to connect to

The `kai.hub.filter` property allows you to configure a sequence of filters that are evaluated on startup. These filters determine which KNIME Hubs can be selected as the backend for K-AI on the KNIME AI Assistant preferences page. This is similar to the `nodes.filter` property but with two key differences:

- **Filter target**: The filters match the host URL of a Hub (e.g., hub.knime.com for the KNIME Community Hub).

- **Scope**: There is no scope property as the filters only control which hubs K-AI is allowed to connect to.

Examples

- Prevent K-AI from connecting to KNIME Community Hub.

```
version: 'customization-v1.0'
kai:
  hub:
    filter:
      - rule: deny
        predicate:
          type: pattern
          patterns:
            - hub.knime.com
          isRegex: false
```

- Allow only hubs of a certain domain.

```
version: 'customization-v1.0'
kai:
  hub:
    filter:
      - rule: allow
        predicate:
          type: pattern
          patterns:
            - .+\.<custom domain>\.com
          isRegex: true
```

Mountpoints

`/instance/org.knime.workbench.explorer.view/defaultMountpoint/defaultMountpoints=<mount id1>,<mount id2>,…`

A comma separated list of default mountpoints that should be loaded,e.g. `LOCAL,EXAMPLES,My-KNIME-Hub`. Changes to this list only affects new workspaces, i.e. workspaces which already contain default mountpoints will still contain them even though they haven't been defined here. If this option is absent and `defaultMountpoint/enforceExclusion` isn't set to true then all default mountpoints will be added. The current default mountpoints are `LOCAL`, `EXAMPLES`, and `My-KNIME-Hub`.

`/instance/org.knime.workbench.explorer.view/defaultMountpoint/enforceExclusion=<true|false>`

If set to true then all default mountpoint not defined by `/instance/org.knime.workbench.explorer.view/defaultMountpoint/defaultMountpoints` will be removed on start up. Please note that if you want to use this option, the default mountpoints you want to include should **only** be listed in `/instance/org.knime.workbench.explorer.view/defaultMountpoint/defaultMountpoints`, and **not** in their full definition like when exporting the preferences.epf file from a KNIME Analytics Platform.

Update Sites

`/instance/com.knime.customizations/updateSite.uris=<uri>,<uri>,…`

Adds the provided addresses to the update sites.

`/instance/com.knime.customizations/updateSite.names=<name>,<name>,…`

The names that are shown under `Available Software Sites` for the provided update sites. Note that the number of names must match the number of provided URIs.

`/instance/com.knime.customizations/updateSite.default.disable=<true|false>`

Disables the default update sites added by KNIME after a fresh installation or update. If a user enables these update sites again they will remain enabled.

`/instance/com.knime.customizations/updateSite.default.forceDisable=<true|false>`

Disables the default update sites added by KNIME after a fresh installation or update. If a user enables these update sites again they will be disabled with the restart of their client.

Columnar Backend Memory configuration

The columnar backend makes use of off-heap memory to store data tables. The memory is managed by the columnar backend and is not part of the Java heap (which is configured via the Xmx setting). Therefore, the memory configured for the columnar backend must be taken into account to avoid out-of-memory errors.

Depending on the executor version, different preferences can be set.

KNIME Analytics Platform version 4.7.x

`/instance/org.knime.core.data.columnar/knime.core.data.columnar.use-defaults=<true|false>`

If true, the other preferences in this section are ignored.

`/instance/org.knime.core.data.columnar/knime.core.data.columnar.small-cache-size=<size>`

Size of cache for tables smaller than 1MB in MB. Example value: `32`.

`/instance/org.knime.core.data.columnar/knime.core.data.columnar.data-cache-size=2048`

Size of cache for other tables in MB. Example value: `2048`.

KNIME Analytics Platform version 5.x

`/instance/org.knime.core.data.columnar/knime.core.data.columnar.off-heap-limit=<size>`

Total off-heap memory limit in MB. Example value: `2048`.

Make sure to configure the memory settings according to the available resources on the executor. Make sure to leave enough memory for the Java heap and other processes running on the executor.

The default configuration for 5.1.0 to 5.1.2 is to use all memory that is not reserved for the Java heap. This configuration is likely to cause the executor to crash with an Out Of Memory Exception. For the default heap size of 60% of the available memory, we recommend setting the off-heap limit to 20% of the available memory.

On executor images before 5.1.2 and 4.7.7, an unfavorable interaction between the JVM and the glibc native memory allocator can cause higher than expected memory usage when using the Columnar Table Backend. Set the environment variable `MALLOC_ARENA_MAX=1` to prevent this issue. Also see, Columnar Backend High Memory Usage on Linux.

Proxy configuration

KNIME Analytics Platform has preferences to configure proxies platform-wide. More details on those, including the below-mentioned proxy providers, can be found here. To distribute proxy configurations to executors, use the following settings in a preference (`.epf`) file.

```
/instance/org.eclipse.core.net/proxiesEnabled=<true|false>
```

If true, proxies are enabled. Setting this to false corresponds to using the “Direct” proxy provider.

```
/instance/org.eclipse.core.net/systemProxiesEnabled=<true|false>
```

If true, the “Native” proxy provider is active, otherwise the “Manual” proxy provider is active. All following preferences are only relevant in the context of manually configured proxies.

```
/instance/org.eclipse.core.net/nonProxiedHosts=<pipe-separated list of excluded hosts>
```

A pipe-separated list of host names that should permanently be excluded from the manually configured proxies. It usually makes sense to set `localhost|127.0.0.1` as default here.

```
/instance/org.eclipse.core.net/proxyData/<HTTP|HTTPS|SOCKS>/host=<host>
/instance/org.eclipse.core.net/proxyData/<HTTP|HTTPS|SOCKS>/port=<port>
/instance/org.eclipse.core.net/proxyData/<HTTP|HTTPS|SOCKS>/hasAuth=<true|false>
```

Preferences for proxy addresses per protocol (HTTP, HTTPS, or SOCKS), and whether the proxy requires authentication. Proxy credentials cannot be configured in preference (`.epf`) files but have to be supplied separately, as described below.

Proxy authentication

Configured proxy credentials are persisted in the Eclipse secure storage, which by default is located within the user's home directory. It is encrypted using a master password.

If a proxy requires authentication (i.e. setting `hasAuth=true`), the secure storage is queried for that proxy host, in conjunction with the master password. This should work out-of-the-box on Windows, macOS, and Linux for the KNIME Analytics Platform. However, KNIME Hub executors need additional configuration.

1. The user's home directory will not be resolved correctly when a KNIME Hub executor runs as system service, since it is not associated to a system user. Hence, the storage location must be configured explicitly in the `knime.ini` file.
2. Usually, either a UI prompt for the master password appears on KNIME startup, or the master password is sourced relative to the user's home directory. Both are not feasible using a headless executor running as system service. Hence, the master password must also be configured in the `knime.ini` file.
3. Do not forget to restart the executor service after applying the following configuration changes.

Custom secure storage location
By default, the secure storage is stored in `~/.eclipse/org.eclipse.equinox.security/secure_storage` where `~` denotes the user's home directory. In order to configure a custom `<storage location>`, you must edit the `knime.ini` file and enter the following **before** the line containing `-vmargs`.

```
-eclipse.keyring
<storage location>
```

Custom secure storage password
To define a custom master password, you first need to disable active master password providers by deselecting all entries in the preferences at *General > Security > Secure Storage*. If this preferences page is not available for you in KNIME Modern UI, try switching to the preferences in the KNIME Classic UI.

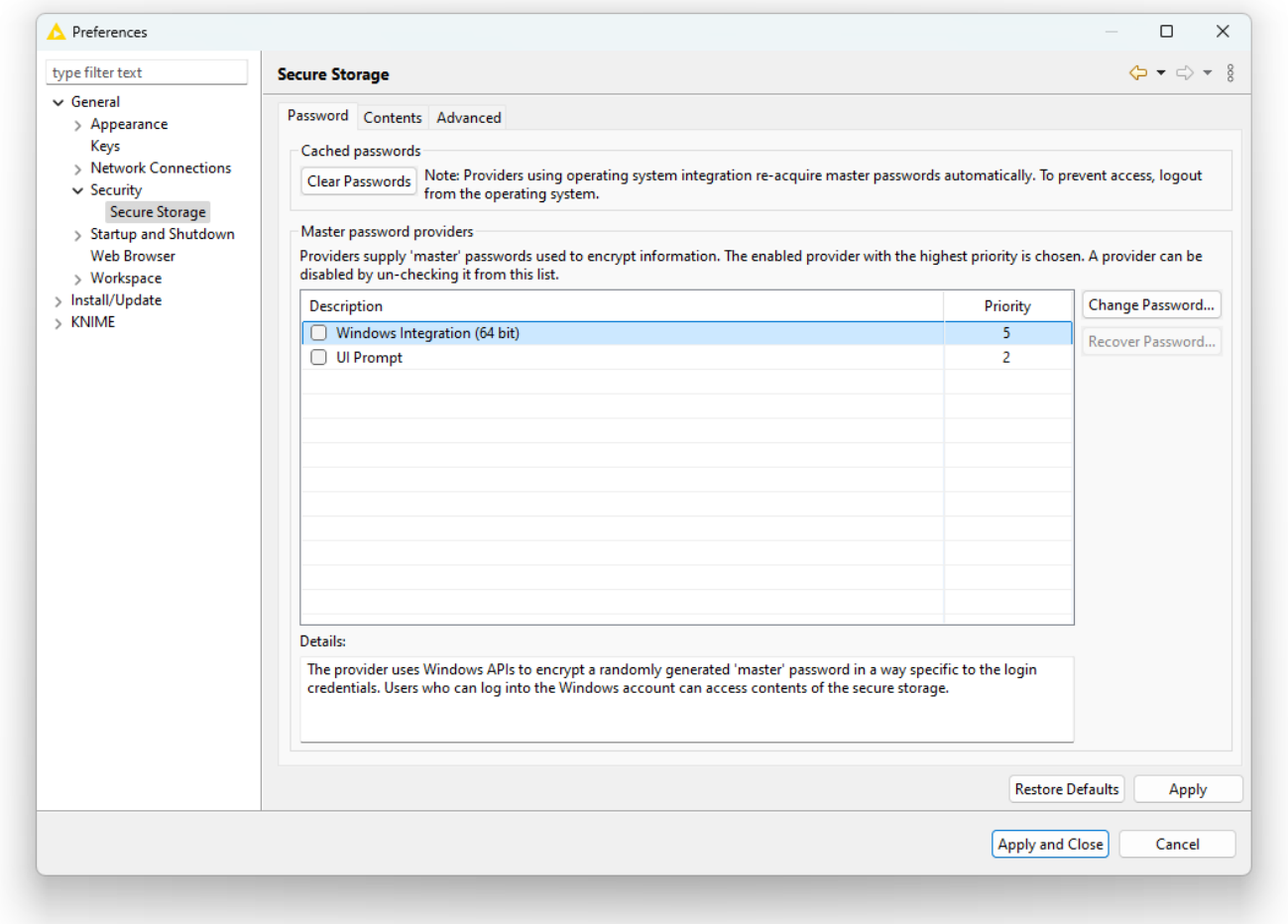


Figure 20. The Secure Storage preferences page.

Alternatively — without using the UI — create the file `<install-dir>/configuration/.settings/org.eclipse.equinox.security.prefs` with the following content. Here, `<install-dir>` denotes the installation directory of the KNIME Hub executor. This file is generated for you if you disable the master password providers on the preferences page.

```
eclipse.preferences.version=1
org.eclipse.equinox.security.preferences.disabledProviders=org.eclipse.equinox.security.
windowspasswordprovider64bit,org.eclipse.equinox.security.osxkeystoreintegration,org.ecl
ipse.equinox.security.linuxkeystoreintegrationjna,org.eclipse.equinox.security.ui.defaul
tpasswordprovider
```

In order to configure a custom `<password location>`, you must edit the `knime.ini` file again, and enter the following **before** the line containing `-vmargs`. At this custom location, create a text file containing nothing but your new master password.

```
-eclipse.password
<password location>
```


Further details on proxies

More useful proxy configuration options are listed here.

Environment variables as fallback
Starting from KNIME version 5.3, environment variables are supported as fallback when choosing the “Manual” or “Native” proxy provider. This is the format required for the proxy environment variables.

```
<http|https|socks|all>_proxy=<user>:<password>@<host>:<port>
```

In addition, the `no_proxy` variable defines a comma-separated list of hosts that are excluded from proxies defined in environment variables. Proxy authentication is directly specified via the user information prefix `<user>:<password>@` in the variable value.

Prior to KNIME version 5.3, environment variables were supported as part of the “Native” proxy provider on Linux systems, with the only difference being that `all_proxy` was not supported there.

Specifically on Linux, we recommend using environment variables in conjunction with the “Native” proxy provider, given that you accept a less secure storage of proxy credentials. This method is simple and independent of users and configuration directories.

Bootstrapping customization profiles behind a proxy
If the profile location of the preferences themselves are hidden behind a network proxy, preferences obviously cannot be used to define the proxy configuration. For this case, the proxy has to be configured in the `knime.ini` file using Java networking properties. This proxy is independent of proxy configurations mentioned above, and will not be used by the KNIME Hub executor for anything except fetching preference profiles.

- To use the equivalent of the “Manual” proxy provider, add the `-Djava.net.useSystemProxies=false` property and the required properties for your proxy configuration (see Java’s documentation).
- To use the equivalent of the “Native” proxy provider, add the `-Djava.net.useSystemProxies=true` property. Unfortunately, it is a bit unclear what Java defines as system properties. We could not verify that environment variables work on Linux with this method. The Java documentation only provides this information about system proxies.

> On Windows systems, macOS systems, and GNOME systems it is possible to tell the java.net stack, setting this property to true, to use the system proxy settings (all these systems let you set proxies globally through their user interface).
>
> — Java SE 17 & JDK 17 docs, Networking Properties

Upload a customization profile

After creating a customization profile, you need to upload it to KNIME Business Hub. You can do it via REST or via data application.

Upload via Data Application

1. If you have not done so already, on KNIME Business Hub, create an application password for the user uploading the profile.
   a. For global customization profiles, use a global admin user.
   b. For team-scoped customization profiles, the user can be either a global admin or a team admin user.
2. Download the workflow:

   Click here and download the workflow.
3. Upload the downloaded workflow to your KNIME Business Hub instance.
4. Deploy the workflow as a data application. You can learn more about how to do it here. In the configuration dialog, you can set the application password and the Hub URL for the user uploading the customization profile.

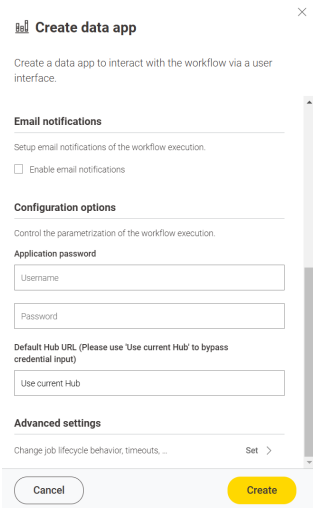


Figure 21. Deployment configuration of the data app.

If you set the "Default Hub URL" configuration default value to "Use current Hub," user authentication on KNIME Business Hub will reflect the logged-in user's permissions.

5. Since we configured the connection directly in the deployment configuration, the data application will, by default, open on the *Customization Profile Operations* page with the *Upload* option selected.

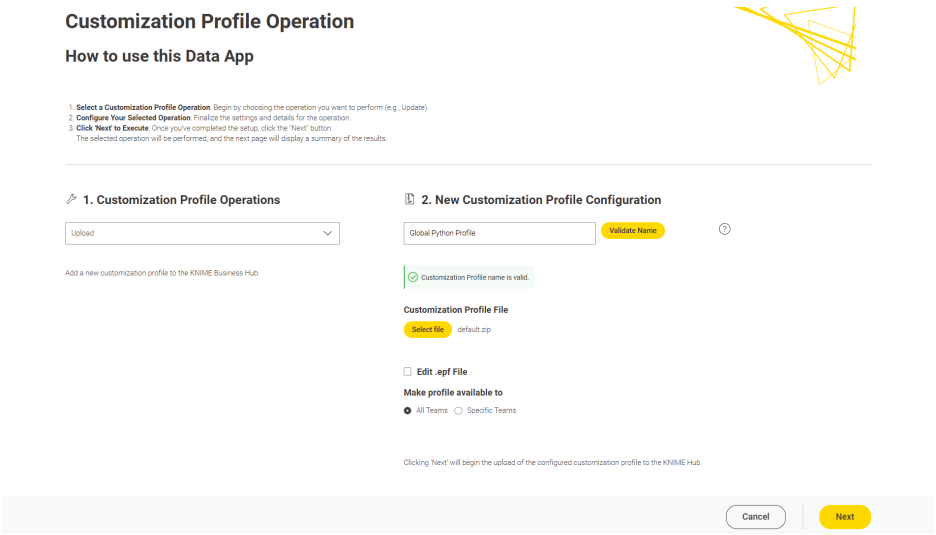


Figure 22. The upload option is selected in the "Customization Profile Operations" drop-down menu.

6. In the section *New Customization Profile Configuration*, you can configure your customization profile using the following steps:
    a. **Name Your Customization Profile:** Provide a unique name for the customization profile.
        i. Use the *Validate Name* button to ensure no duplicate customization profile names exist, as uploading two with the same name is not allowed.

b. **Upload Your File:** Upload your customization profile file. Refer to the documentation for guidance on crafting a valid file.

c. **Edit .epf File (Optional):**

   i. Select the one you wish to edit if the uploaded ZIP file contains multiple `.epf` files.

   ii. You can edit content only for files with `.epf` and `.yml` extensions, as other file types are not editable within this data app.

d. **Set Customization Type:** Specify the type of customization profile:

   i. *Global Admin:* You can choose the customization profile scope if logged in to the Data App as a global admin.

      A. Global scope: After being uploaded to KNIME Business Hub, the global customization profile can be applied to a shared or a team execution context.

      B. Team scope: When a customization profile with team scope is uploaded, it can be applied to the team's executors. It is possible to upload the same customization profile to multiple teams. The global admin can upload the customization profile to any team.

   ii. *Team Admin:* If logged in as a team admin, you can only upload the customization profile as a team-scoped type. You can upload the same customization profile for multiple teams. The availability of teams is determined by the permissions of the user who is logged in.

7. On the *Results* page, a table displays the results of the customization profile upload operation.

   If everything goes well, the column *Results* will show the value *Success* in green, indicating the operation was successful. You can now apply it to a KNIME Analytics Platform installation or to a KNIME Business Hub executor.

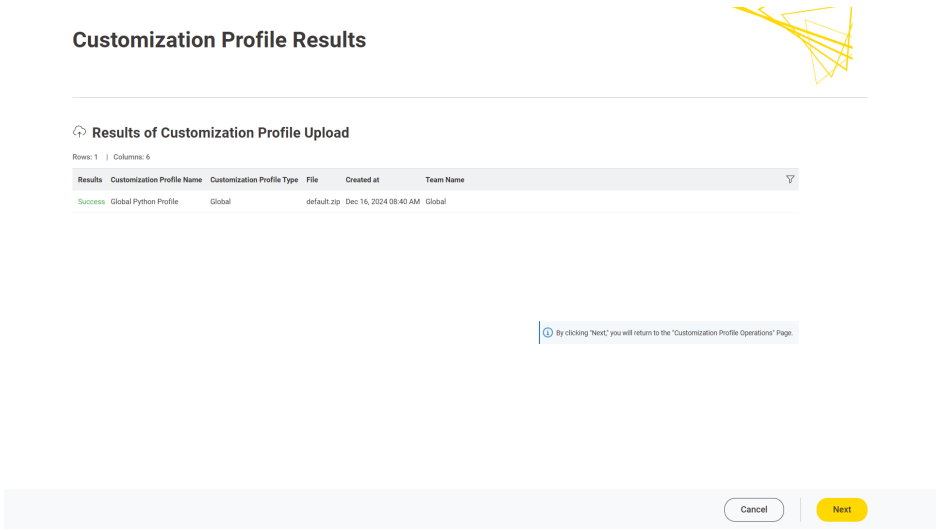


Figure 23. The customization profile has been uploaded successfully to KNIME Hub.

8. **Error Handling**: The data application is equipped to identify and display errors encountered during operations, whether they occur while retrieving data from the KNIME Business Hub API or executing actions via the API. Errors are captured both server-side and application-side and are displayed in the "**Error Overview**" table. This table provides a detailed summary, including:
    a. The API call status code.
    b. The error message.
    c. The specific part of the data application where the error occurred.

    Users can consult this table to diagnose issues and take appropriate corrective actions before retrying the operation.
9. Click *Next* to finish the operation.You can return to the *Customization Profile Operations* page to perform additional actions or close the data application directly.

Upload via REST request

1. If you have not done so already, on KNIME Business Hub, create an application password for the user uploading the profile.
    a. For global customization profiles, use a global admin user.
    b. For team's customization profiles, the user can be either a global admin user or a team admin user.
2. Send a `POST` request to `https://api.<base-url>/execution/customization-profiles` with the following set up:
    a. Authorization: select Basic Auth, using username and password from the created application password

b. Body: select form-data as request body type and add the following entries:

   i. Add a new key, set content to “File”. Name of the key needs to be “content”. The value of the key is the `.zip` file containing the profile, and the content type is `application/zip`.

   ii. Add a new key, set content to “Text”. Name of the key needs to be “metadata”.

      A. The value of the key is:

         For global customization profiles enter:

```
{
  "name": "<profile_name>",
  "scope": "hub:global"
}
```

         For team’s customization profiles you first need to obtain the `<team_ID>` of the team you want to upload the customization profile to. To do so you can use the following `GET` request:

```
GET api.<base-url>/accounts/identity
```

         You will find the `<team_ID>` in the body response under `teams`:

```
...
    "teams": [
        {
            "id": "account:team:<team_ID>",
            "name": "<team_name>",
            ...
        }
    ]
...
```

         Then the value of the key is:

```
{
  "name": "<profile_name>",
  "scope": "account:team:<team_ID>"
}
```

      B. Set the content type for the second key to `application/json`

c. When using Postman there is no need to manually adjust the headers

3. Send the request

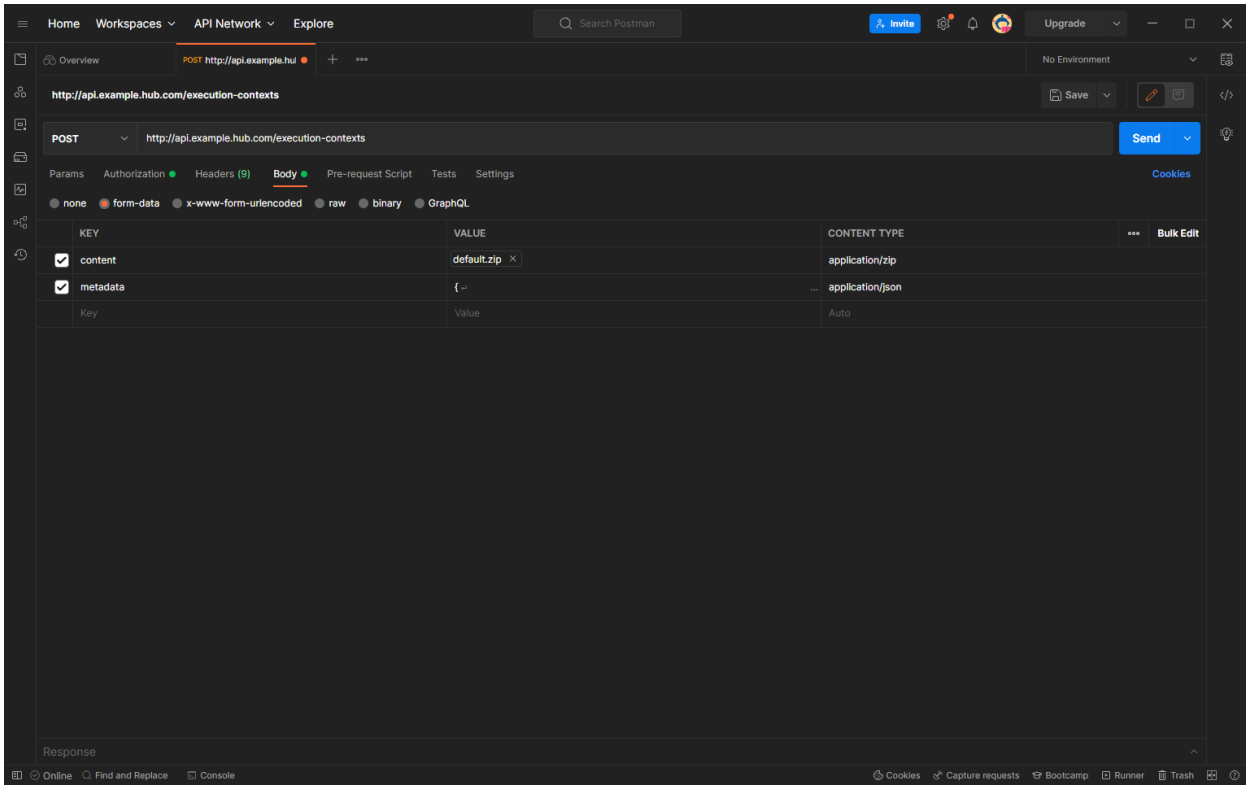


Figure 24. Set up Postman for a REST call to upload a customization profile to a KNIME Hub instance

4. If successful, you will receive a 201 status response. Make note of the created `<profile_ID>`, as this will be used to refer to the profile when requesting it.

You can refer to the API documentation at the following URL for more details about the different calls you can make to the Customization Profiles endpoint.

```
http://api.<base-url>/api-doc/execution/#/Customization%20Profiles
```

Apply a customization profile

Apply a customization profile to KNIME Hub executor

Apply via Data App

To apply a customization profile to all executors running in a KNIME Business Hub execution context, we can use the data application. However, the type of user logged in affects the

operations that can be performed. Refer to the table below for a comprehensive overview.

Table 1. Apply customization profile: user rights

User type	Eligible customization profile	Customization profile type	Eligible Execution Context
Global Admin	All uploaded within the KNIME Business Hub instance	Global	Shared and any team specific execution context
		Team-scoped	Any team execution contexts
Team admin	Only those uploaded in teams in which the user has team admin rights	Team-scoped only	Only the team execution contexts where the customization profile was uploaded. Shared execution contexts are not eligible

To provide a better understanding of the table, here are some examples to demonstrate its functionality:

- A global admin can choose one team-scoped customization profile and apply it to any team-scoped execution context within the KNIME Business Hub instance. For instance, the global admin can apply *Team A's customization profile* to the team-scoped execution context of *Team B*.
- A Global Admin can also select a **Global** customization profile and apply it to any shared and team-scoped execution context within the KNIME Business Hub instance.
- A team admin can **only** choose team-scoped customization profiles uploaded within the teams with admin rights. For example, they can only apply a customization profile uploaded in *Team A* within the Team A execution contexts.

1. Learn how to download the data app from Community Hub, upload and deploy it in KNIME Business Hub, and authenticate with your application password in the Upload a customization profile section.

2. Select *Apply* in the *Customization Profile Operation* menu to apply a Customization Profile.
3. The table *Potential Customization Profile-Context Combinations* displays the **possible combinations** between the customization profiles accessible to the logged-in user and the available execution contexts.#

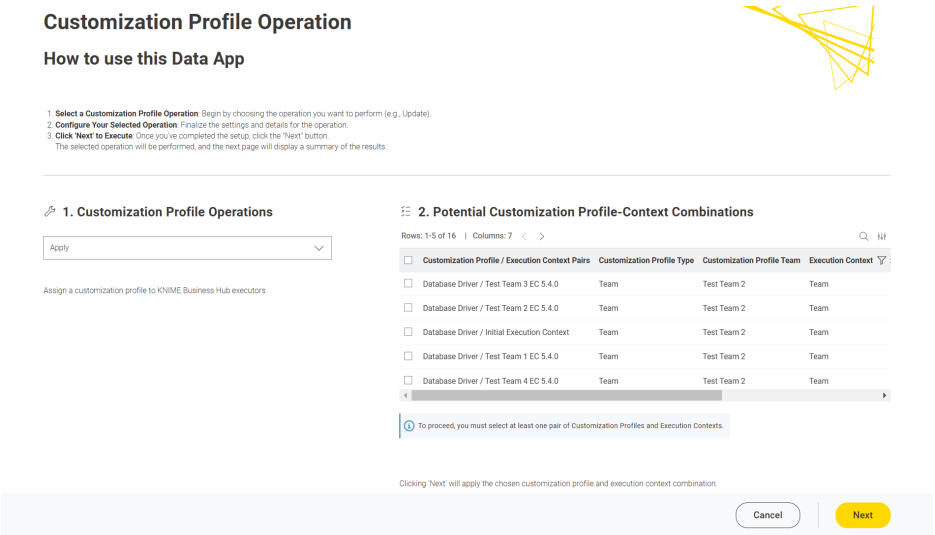


Figure 25. Multiple combinations of potential customization profile execution contexts are available for a global admin.

4. As mentioned above, a **Global admin** can apply a *team-scoped* customization profile to multiple team-scoped execution contexts or a *global customization profile* to any execution context in KNIME Hub. This may lead to **many options**, so it's advisable to use the table filters to narrow down the choices.
5. Select the desired **pair** of customization profile and execution context from the table, then click *Next*.

Figure 26. We want to apply the team-scoped customization profile from Test Team 2 to Test Team 1's team-scoped execution context.

On the *Results* page, a table displays the results of the customization profile application operation.

If everything goes well, the column *Results* will show the value *Success* in green, indicating that your customization profile has been successfully applied to KNIME Business Hub executors for the selected execution contexts.

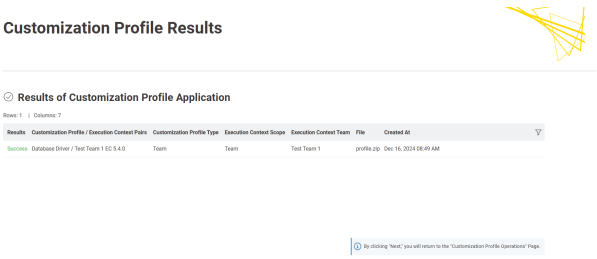


Figure 27. The user applies a JDBC database driver within a team-scoped execution context.

6. **Error Handling**: The data application is equipped to identify and display errors encountered during operations, whether they occur while retrieving data from the KNIME Business Hub API or executing actions via the API. Errors are captured both server-side and application-side and are displayed in the **"Error Overview"** table. This table provides a detailed summary, including:
    a. The API call status code.
    b. The error message.
    c. The specific part of the data application where the error occurred.

    Users can consult this table to diagnose issues and take appropriate corrective

actions before retrying the operation.

7. Click *Next* to finish the operation.You can return to the *Customization Profile Operations* page to perform additional actions or close the data application directly.

Apply via REST request

In order to apply a customization profile to all executors running in a KNIME Business Hub execution context, you will need to send a request to the execution context endpoint.

First you need to get the execution context ID. To do so you can use the following `GET` request to get a list of all the execution contexts that are owned by a specific team:

```
GET api.<base-url>/execution-contexts/account:team:<team_ID>
```

> If you are a global admin you can also `GET` a list of all the execution contexts available on the Hub instance with the call `GET api.<base-url>/execution-contexts/`.

Now you can apply the new customization profile to the selected execution context.

You will need to obtain the `<profile_ID>` using the following `GET` request:

1. For global customization profiles:

   ```
   GET api.<base-url>/execution/customization-profiles/hub:global
   ```

2. For team's customization profiles:

   ```
   GET api.<base-url>/execution/customization-profiles/account:team:<team_ID>
   ```

   > Refer to the above section to find out how to get the `<team_ID>`.

Then you need to update the execution context by using the following `PUT` request:

```
PUT api.<base-url>/execution-contexts/<execution-context_ID>
```

You need to select `Body > raw > JSON` and add the following to the request body:

```
{
"customizationProfiles": [
    "<profile_ID>"
  ]
}
```

This will cause a restart of all executor pods of this execution context, after which the profile will be applied.

> At present, this will also terminate all jobs currently running on the executor.

Update a customization profile

It is possible to update customization profiles uploaded to KNIME Business Hub. This can be done even if the customization profile *has already been applied* to the KNIME Business executor via execution context. You can update two features:

1. The customization profile **name**.
2. The customization profile zip file can be overwritten. Refer to the create a customization profile section to learn how to make one.

Update via Data Application

Users can use the customization profile data app to **update** customization profiles previously uploaded to the KNIME Business Hub.

Depending on the user's role as a Global Admin or a Team Admin, they can access and update specific customization profiles:

- **Global Admin:** can **update** all uploaded customization profiles within the current KNIME Business Hub instance, either team or global-scoped customization profiles.
- **Team Admin:** can only update team-scoped customization profiles from their own teams.

---

1. Learn how to download the data app from Community Hub, upload and deploy it in KNIME Business Hub, and authenticate with your application password in the Upload a customization profile section.
2. Select *Update* in the *Customization Profile Operations* drop-down menu to update a customization profile.

3. A table titled *Overview of Uploaded Customization Profiles* will display a list of customization profiles available to the logged-in user. The table includes details such as:
    a. The type of profile (global or team-scoped).
    b. The upload location.
    c. The creation date.
    d. The file used for the upload.

    > Only one customization profile can be updated at a time.

4. After reviewing the table, select the profile you want to update from the "**Select Customization Profile**" drop-down menu. The format of the selection will appear as a `customization profile name (upload location)`.

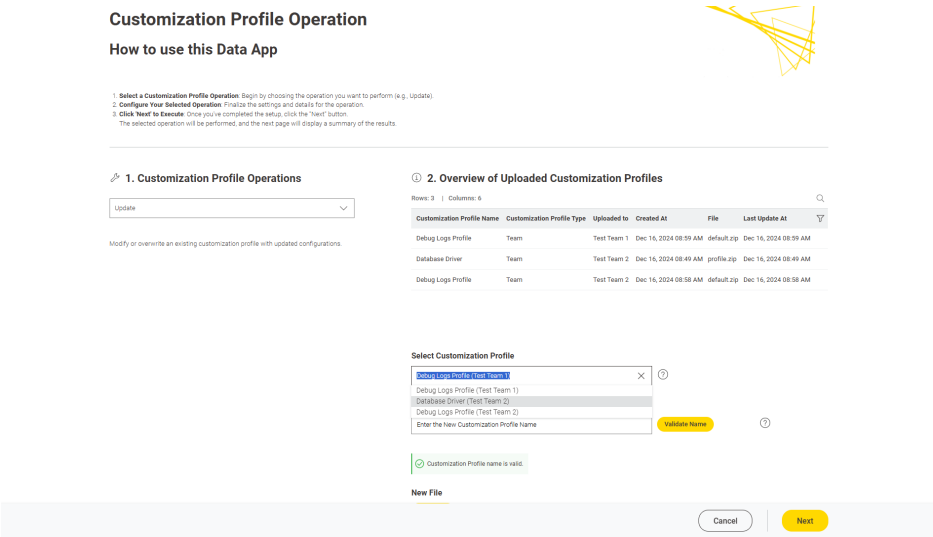


Figure 28. A team admin could select only team-scoped customization profiles.

5. Once a profile is selected, you can:
    a. Assign a **new name** to the customization profile.
    b. Upload a **new ZIP file** for the customization profile.
    c. If you choose "**Edit .epf file**", you can directly edit the content of files with `.epf` or `.yml` extensions within the data app. Other file types cannot be edited.

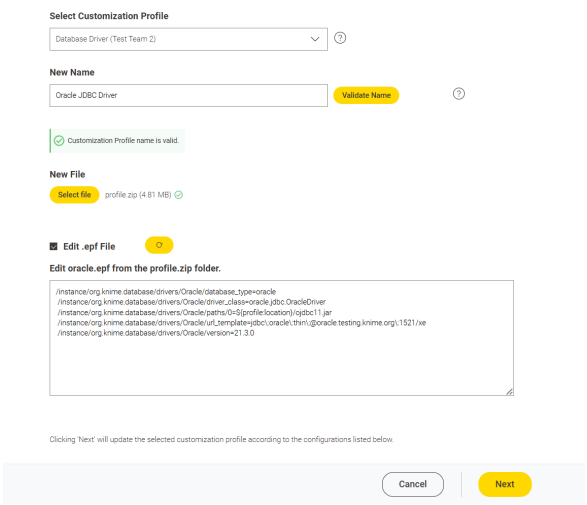


Figure 29. The .epf from the zip folder file can be edited within the data app.

6. On the *Results* page, a table will display the outcome of the update operation: If successful, the "**Results**" column will show Success in green, indicating the update was applied.

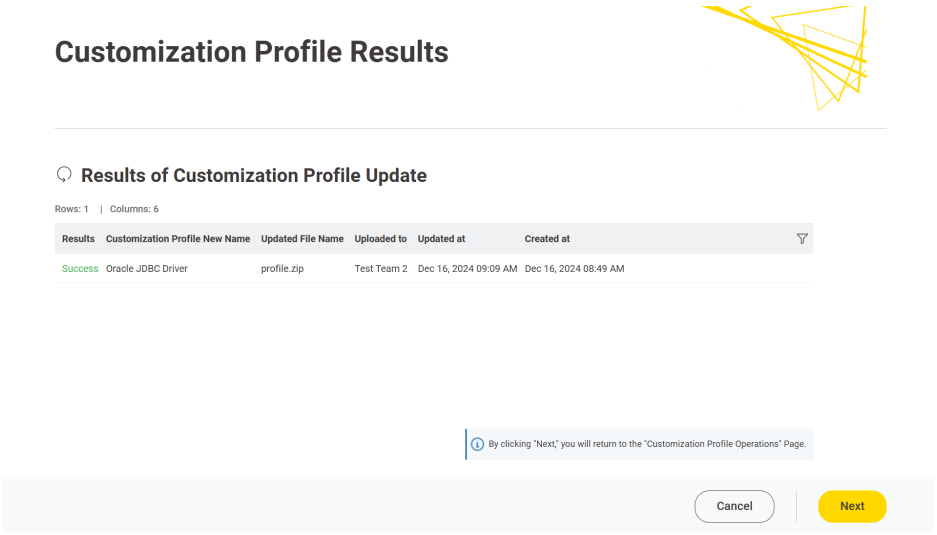


Figure 30. The outcome of updating one customization profile.

The updated customization profiles will be **automatically applied to execution contexts or KNIME Analytics Platform clients using them.**

7. **Error Handling**: The data application is equipped to identify and display errors encountered during operations, whether they occur while retrieving data from the KNIME Business Hub API or executing actions via the API. Errors are captured both server-side and application-side and are displayed in the "**Error Overview**" table. This table provides a detailed summary, including:
    a. The API call status code.
    b. The error message.

c. The specific part of the data application where the error occurred.

   Users can consult this table to diagnose issues and take appropriate corrective actions before retrying the operation.

8. Click *Next* to finish the operation.You can return to the *Customization Profile Operations* page to perform additional actions or close the data application directly.

Update via REST request

Updating a customization profile is like replacing an existing profile with a new file and name.

If you want to update a customization profile via REST request, the process is similar to the uploading process. The only difference is that instead of a POST Request, you need to perform a **PUT Request** and specify the ID of the customization profile.

1. If you have not done so already, on KNIME Business Hub, create an application password for the user uploading the profile.
   a. For global customization profiles, use a global admin user.
   b. For team's customization profiles, the user can be either a global admin user or a team admin user.
2. To begin, you will need to get the list of all uploaded customization profiles on KNIME Business Hub:

   You can obtain the `<profile_ID>` using the following `GET` request:

   For global customization profiles:

   ```
   GET api.<base-url>/execution/customization-profiles/hub:global
   ```

   For team's customization profiles:

   ```
   GET api.<base-url>/execution/customization-profiles/account:team:<team_ID>
   ```

   Refer to the above section to learn how to get the `<team_ID>`.

3. **Updating** a customization profile is similar to uploading it via a REST Request. However, unlike uploading, we only need to provide the name and file of the customization profile for updating. So, we don't need to provide the scope as it remains the same.
   a. Send a `PUT` request to `https://api.<base-url>/execution/customization-`

`profiles/<customization-profile-ID>` with the following set up:

b. Authorization: select Basic Auth, using username and password from the created application password

c. Body: select form-data as request body type and add the following entries:

i. Add a new key, set content to “File”. Name of the key needs to be “content”. The value of the key is the `.zip` file containing the profile, and the content type is `application/zip`.

ii. Add a new key, set content to “Text”. Name of the key needs to be “metadata”.

A. The value of the key is the same for global and team-scoped customization profiles:

```
{
  "name": "<profile_name>"
}
```

B. Set the content type for the second key to `application/json`

d. When using Postman there is no need to manually adjust the headers

4. Send the request

Detach a customization profile

It’s important to know that detaching a customization profile is not the same as deleting it. When you detach a customization profile, it isn’t removed from KNIME Business Hub. Instead, it just means that the customization profile won’t be applied to the KNIME Analytics Platform or the KNIME Business executors.

However, the customization profile is still available in KNIME Business Hub, so it can be reused again whenever needed.

Detach a customization profile from a KNIME Business Hub executor

Detaching a customization profile applies **only** to those customization profiles applied to a KNIME Business Hub executor via execution context. Separating a customization profile from its execution context is the prerequisite step to deleting a customization profile from a KNIME Business Hub instance.

Detaching a customization profile from an execution context can also be done if the user no

longer wants to apply the customization to the executors.

Detach via Data Application

The data application allows users to **detach** customization profiles applied to execution contexts in KNIME Business Hub.

For instructions on how to apply a customization profile to an execution context, refer to this section.

The customization profiles available for detachment depend on the user type:

- **Global Admin:** can **detach** all applied customization profiles within the current KNIME Business Hub instance, either team or global-scoped customization profiles.
- **Team Admin:** can only detach team-scoped customization profiles from their own teams.

1. Learn how to download the data app from Community Hub, upload and deploy it in KNIME Business Hub, and authenticate with your application password in the Upload a customization profile section.
2. Select *Detach* from the *Customization Profile Operations* drop-down menu to begin the detachment process.

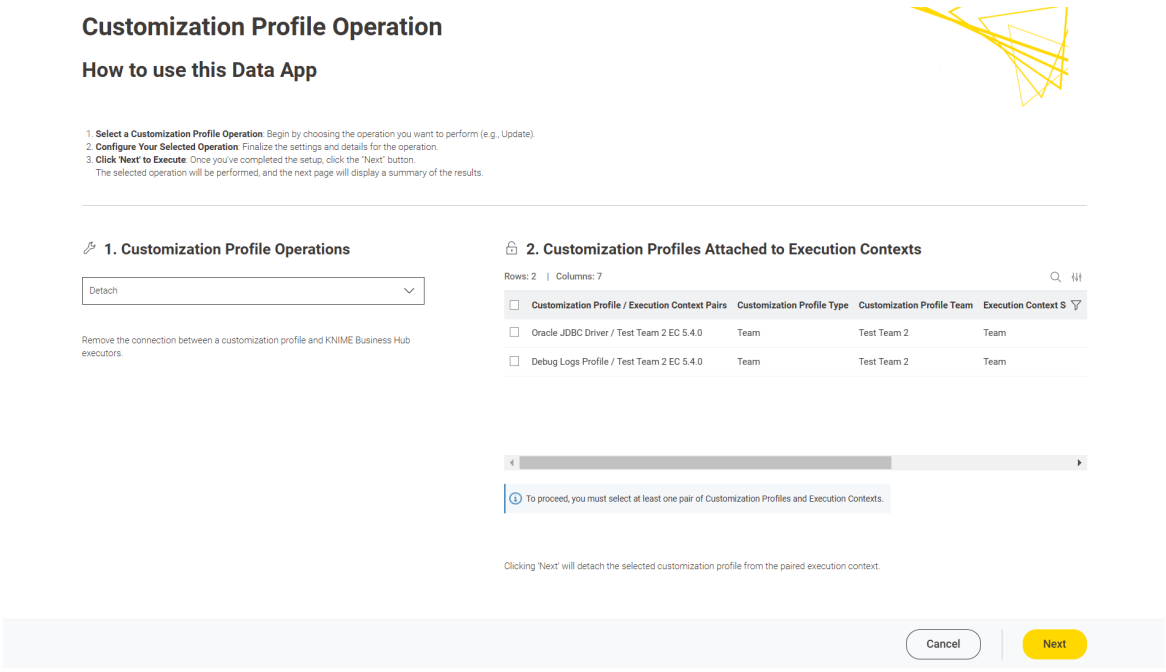


Figure 31. Team admin can detach customization from its teams.

3. Use the table titled *Customization Profiles Attached to Execution Contexts* to select the desired customization profile and execution context **pairs** you wish to detach.

a. *Global admins* may encounter a large table with all possible combinations available for detachment.

b. Use the provided **filters** to narrow the options, such as by filtering by customization profile upload location or execution context type.

> You can select multiple customization profiles for detachment.

4. Ensure you select at least one customization profile and execution context pair from the table. Once you have made a selection, click "**Next**" to proceed.

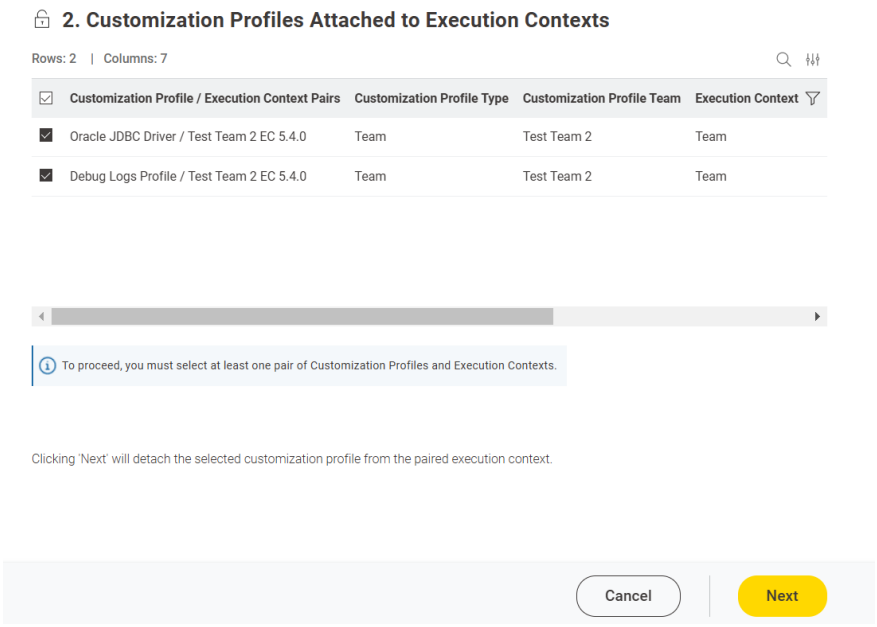


Figure 32. Two team-scoped customization profiles will be detached.

5. On the *Results* page, a table displays the outcome of the detach operation: If successful, the "**Results**" column will show Success in green, indicating the detachment was applied.

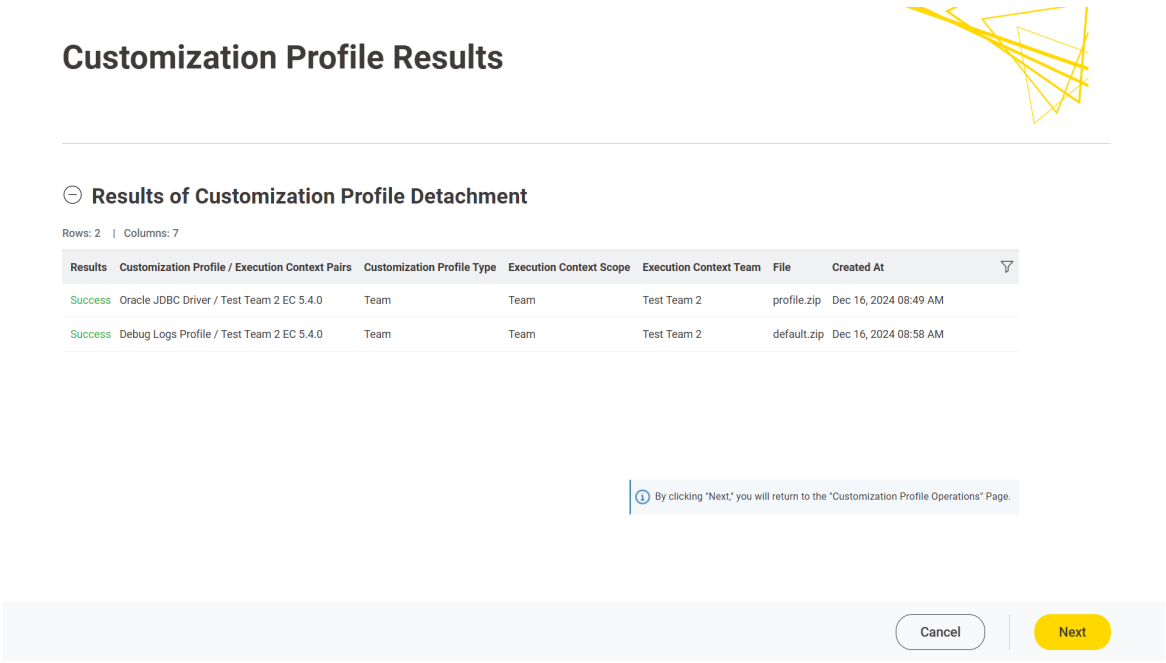


Figure 33. The outcome of detaching two customization profiles.

> Detaching a customization profile from an execution context doesn't remove it from the Business Hub or other execution contexts using the same profile.

6. **Error Handling**: The data application is equipped to identify and display errors encountered during operations, whether they occur while retrieving data from the KNIME Business Hub API or executing actions via the API. Errors are captured both server-side and application-side and are displayed in the **"Error Overview"** table. This table provides a detailed summary, including:
    a. The API call status code.
    b. The error message.
    c. The specific part of the data application where the error occurred.

    Users can consult this table to diagnose issues and take appropriate corrective actions before retrying the operation.
7. Click *Next* to finish the operation.You can return to the *Customization Profile Operations* page to perform additional actions or close the data application directly.

Detach via REST request

You can detach a customization profile from a KNIME Business Hub execution context, which is the inverse of applying it. The steps for detaching a customization profile are similar

to applying one.

To detach a customization profile from all executors running in a KNIME Business Hub execution context, you must send a request to the execution context endpoint, not including the customization profile ID that you want to detach.

1. If you have not done so already, on KNIME Business Hub, create an application password for the user uploading the profile.
   a. For global customization profiles, use a global admin user.
   b. For team's customization profiles, the user can be either a global admin user or a team admin user.
2. First, you need to get the execution context ID. To do so, you can use the following `GET` request to get a list of all the execution contexts that are owned by a specific team:

   ```
   GET api.<base-url>/execution-contexts/account:team:<team_ID>
   ```

   If you are a Global Admin you can also `GET` a list of all the execution contexts available on the Hub instance with the call `GET api.<base-url>/execution-contexts/`.

3. Retrieve the existing customization profiles in the execution context from the above Get Request response. Look for a key in the JSON body similar to:

   ```
   "customizationProfiles" : [ "<customization-profile-ID_1>",<customization-profile-ID_2>" ]
   ```

4. Now, you can detach the target customization profile from the selected execution context. To do so, you need to update the execution context by using the following `PUT` request:

   ```
   PUT api.<base-url>/execution-contexts/<execution-context_ID>
   ```

5. To detach a customization profile, e.g. `<customization-profile-ID_1>`, from the target execution context, follow these steps. Select `Body > raw > JSON` and ensure you do not include the customization profile you wish to remove. Use the syntax of the request body shown below:

```
{
"customizationProfiles": [
    "<customization-profile-ID_2>"
  ]
}
```

6. If the target execution context has only **one** customization profile attached, you can detach it by doing an empty request.

```
{
"customizationProfiles": []
}
```

Delete a customization profile

Deleting a customization profile from KNIME Business is a straightforward operation, and you can do it either through a REST request or the data application. Please follow the step-by-step guide below to understand how it works.

Please note that the type of user determines which customization profiles they can delete:

- **Global Admin:** can **delete** all customization profiles within the current KNIME Business Hub instance, either team or global-scoped customization profiles.
- **Team Admin:** can only **delete** team-scoped customization profiles from their own teams.

Deleting a customization profile from KNIME Business Hub requires first **detaching** it from any execution context where it was applied.

Please refer to the Detach a customization profile from a KNIME Business Hub executor section to understand how to detach a customization profile.

Delete via Data Application

1. Learn how to download the data app from Community Hub, upload and deploy it in KNIME Business Hub, and authenticate with your application password in the Upload a customization profile section.
2. Select *Delete* from the *Customization Profile Operations* drop-down menu to delete a customization profile from your KNIME Business Hub instance.

3. Review the table titled *Customization Profiles Eligible for Deletion*, which lists all customization profiles that can be deleted.
    a. Only customization profiles that are **fully detached from all execution contexts** are displayed in this table. Customization profiles still attached to execution contexts are excluded to prevent errors during the deletion process.

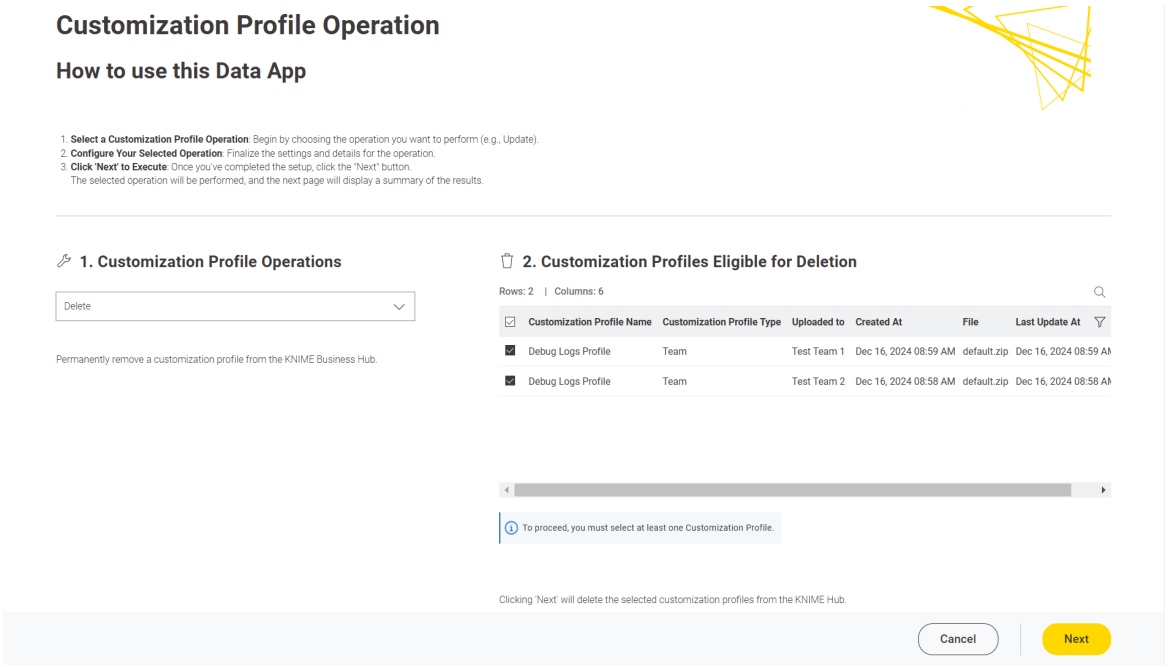


Figure 34. Team admins can only delete customization profiles from teams they are admin of.

4. Select the customization profiles you want to delete from the table and click "**Next.**"
    a. You can choose to delete multiple customization profiles at the same time.

> Clicking "**Next**" will immediately delete the selected customization profiles from KNIME Hub. Ensure your selection is accurate before proceeding, as this action cannot be undone.

5. On the *Results* page, a table displays the outcome of the delete operation: If successful, the "**Results**" column will show Success in green, indicating the deletion was applied.

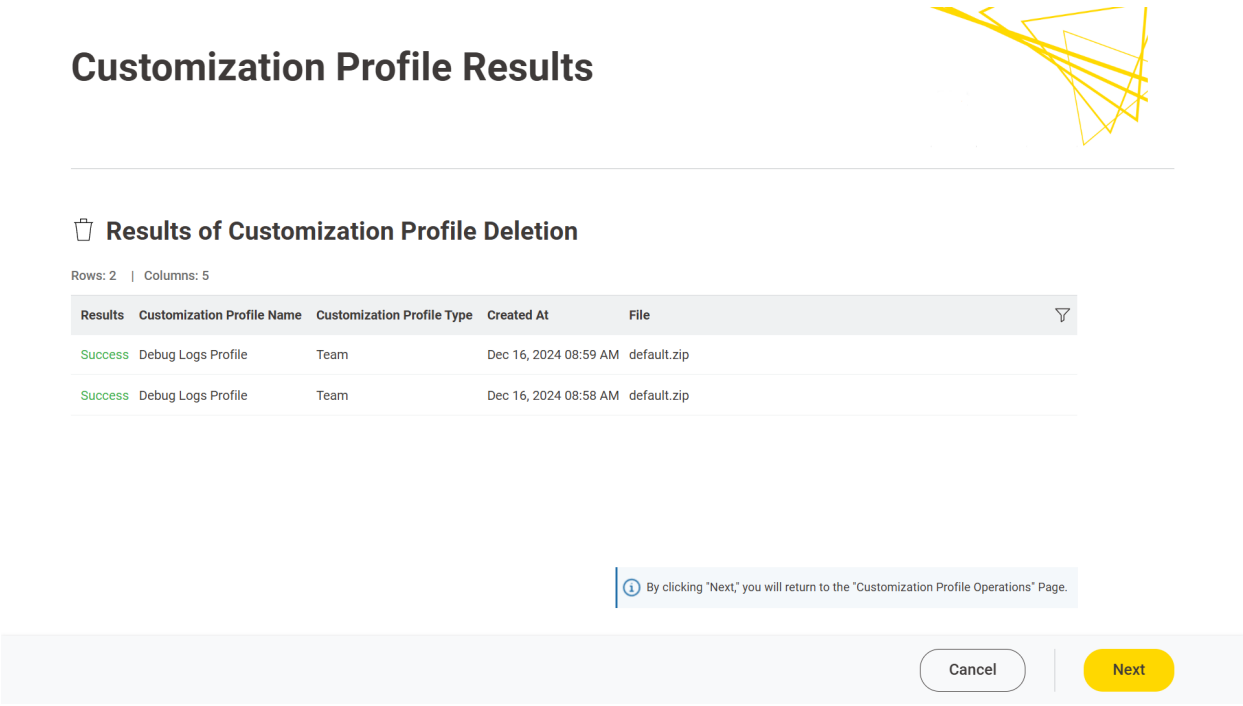


Figure 35. The customization profile deletion from KNIME Business Hub was successful.

6. **Error Handling**: The data application is equipped to identify and display errors encountered during operations, whether they occur while retrieving data from the KNIME Business Hub API or executing actions via the API. Errors are captured both server-side and application-side and are displayed in the "**Error Overview**" table. This table provides a detailed summary, including:
    a. The API call status code.
    b. The error message.
    c. The specific part of the data application where the error occurred.

    Users can consult this table to diagnose issues and take appropriate corrective actions before retrying the operation.
7. Click *Next* to finish the operation.You can return to the *Customization Profile Operations* page to perform additional actions or close the data application directly.

Delete via REST request

Deleting a customization profile from KNIME Business Hub is possible via a REST request.

Below are the steps to accomplish this:

1. If you have not done so already, on KNIME Business Hub, create an application password for the user uploading the profile.

a. For global customization profiles, use a global admin user.

b. For team's customization profiles, the user can be either a global admin user or a team admin user.

2. To start, you need to retrieve the list of the uploaded customization profiles to KNIME Business Hub:

   You can obtain the `<customization_profile_ID>` using the following `GET` request:

   For global customization profiles:

   ```
   GET api.<base-url>/execution/customization-profiles/hub:global
   ```

   For team's customization profiles:

   ```
   GET api.<base-url>/execution/customization-profiles/account:team:<team_ID>
   ```

   Refer to the above section to find out how to get the `<team_ID>`.

3. Once you have the `<customization-profile-ID>` that you want to delete, perform a DELETE Request.

   a. Send a `DELETE` request to `https://api.<base-url>/execution/customization-profiles/<customization-profile-ID>` with the following set up:

      i. Authorization: select Basic Auth, using username and password from the created application password

      After a successful deletion, you will receive a 20* status code.

Advanced configuration

This section covers some of the configuration settings that are available for your KNIME Business Hub instance.

The following configurations are available in the KOTS Admin Console and can be changed after the installation and first minimal configuration steps are concluded successfully.

You can access the KOTS Admin Console via the URL and password you are provided in the output upon installation.

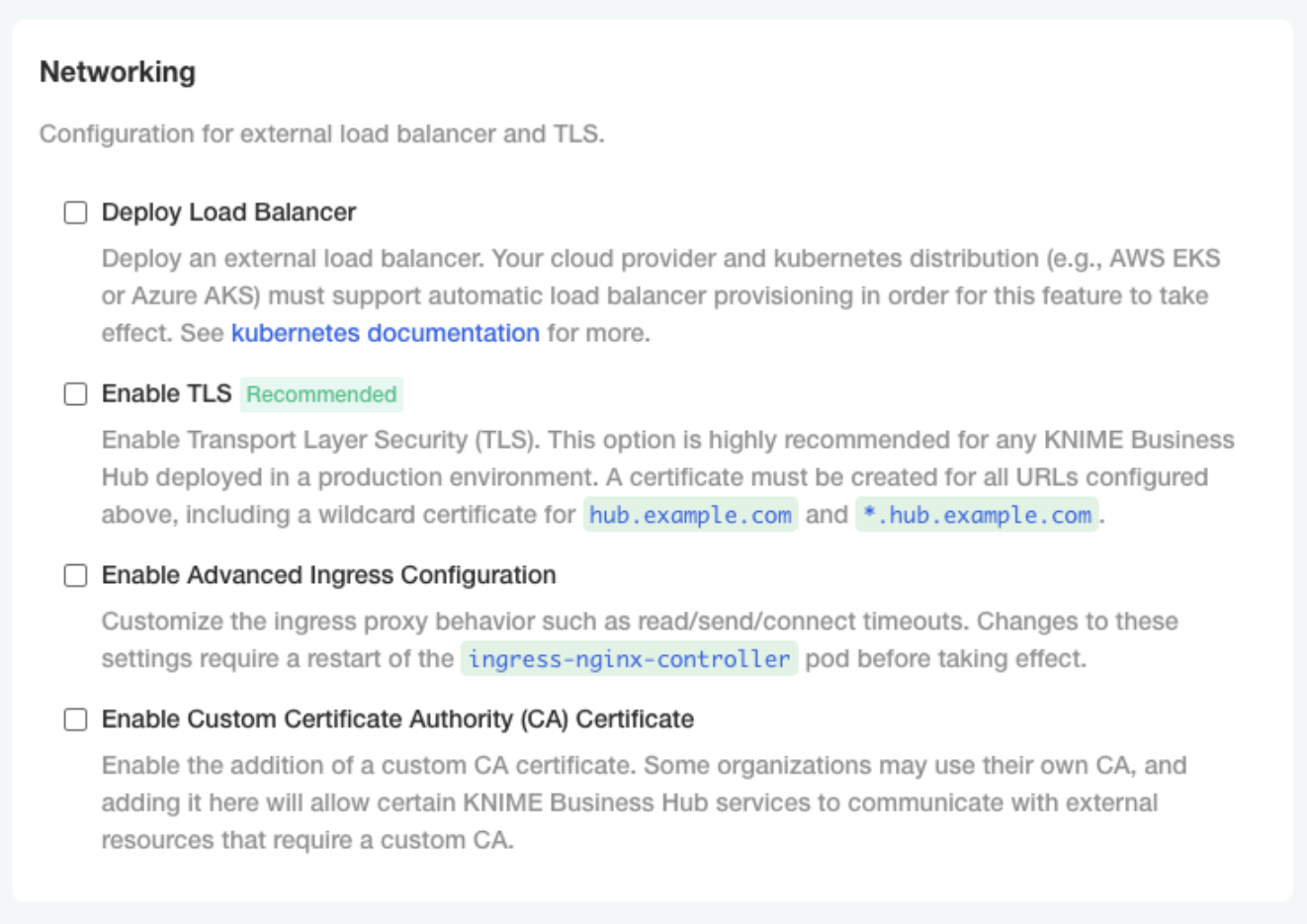
Configure networking

In the "Networking" section of the KOTS Admin Console you can:

- Deploy an external load balancer for traffic ingress: this feature takes effect only if your cloud provider and kubernetes distribution support automatic load balancer provisioning.
- Enable Transport Layer Security (TLS): the encryption protocol that provides communications security is highly recommended especially for KNIME Business Hub instances deployed in a production environment.

> Please, be aware that if TLS is not enabled some HTTPS-only browser's features will not be available. For example, it will not be possible for a user to copy generated application passwords.

- Enable advanced ingress configuration: you can customize the ingress proxy behavior, for example configuring the read/send/connect timeouts.



Configure TLS

If you enable the Transport Layer Security (TLS) you need to have a certificate that is valid for all the URLs defined during the installation. We recommend to create a wildcard certificate for `<base-url>` and `*.<base-url>`, e.g. `hub.example.com` and `*.hub.example.com`.

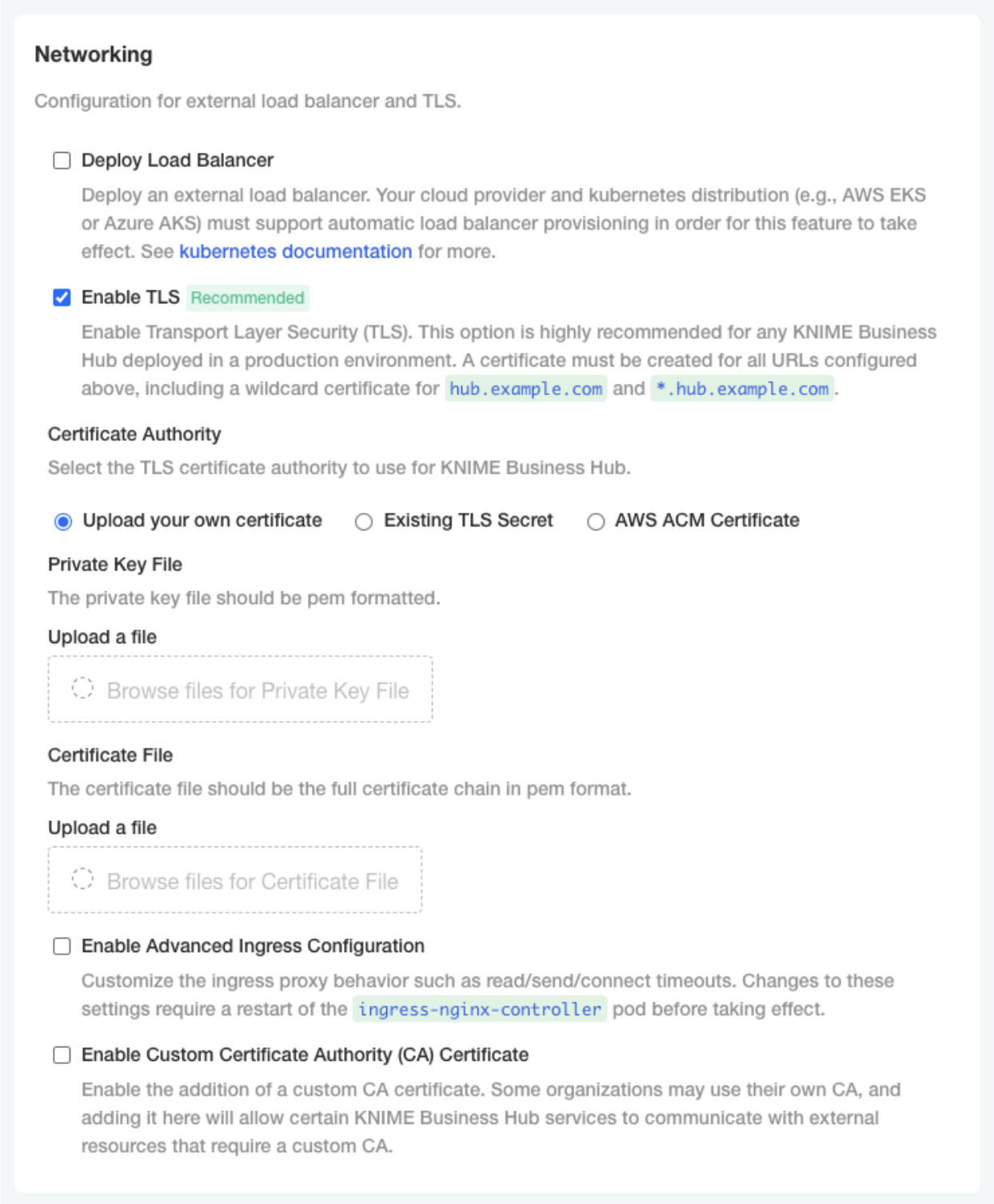
Check *Enable TLS* in the "Networking" section of the KOTS Admin Console.

- **Upload your own certificate**: Select *Upload your own certificate* to be able to upload the certificate files.

  You will need an unencrypted private key file and a certificate file that contains the full certificate chain. In the certificate chain the server certificate needs to be the first in the PEM file, followed by the intermediate certificate(s). You usually can get a certificate from your company's IT department or Certificate Authority (CA).

  Another possibility, if you have a public domain name, is to use `letsencrypt` to obtain a certificate.

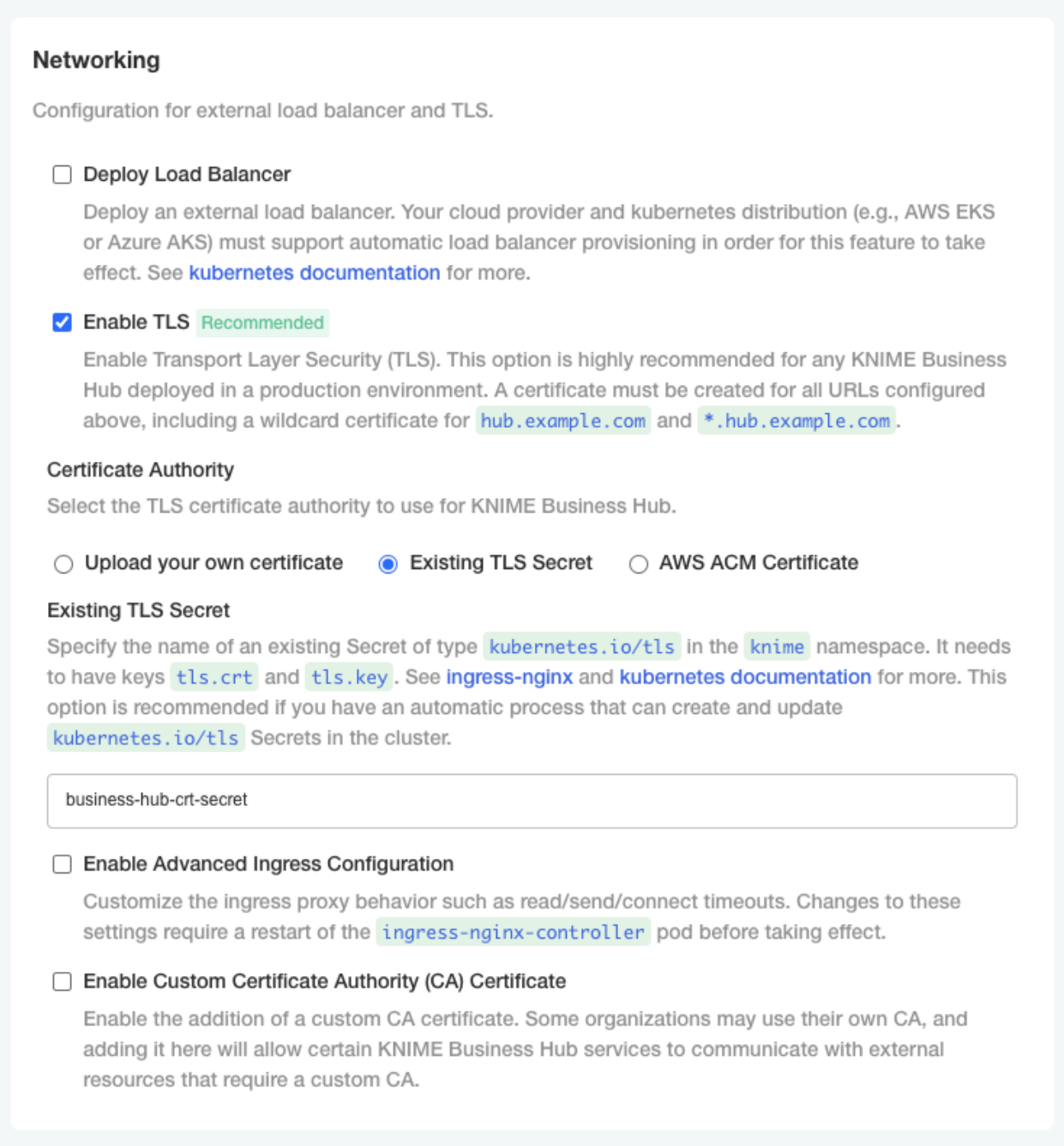
  Both certificates need to be PEM formatted as requested by the `ingress-nginx-controller` (see the relevant documentation here).



- **Existing TLS Secret**: Select *Existing TLS Secret* to specify the name of of an existing Secret of type `kubernetes.io/tls` in the `knime` namespace. It needs to have keys `tls.crt` and `tls.key`, which contain the PEM formatted private key and full chain certificate.

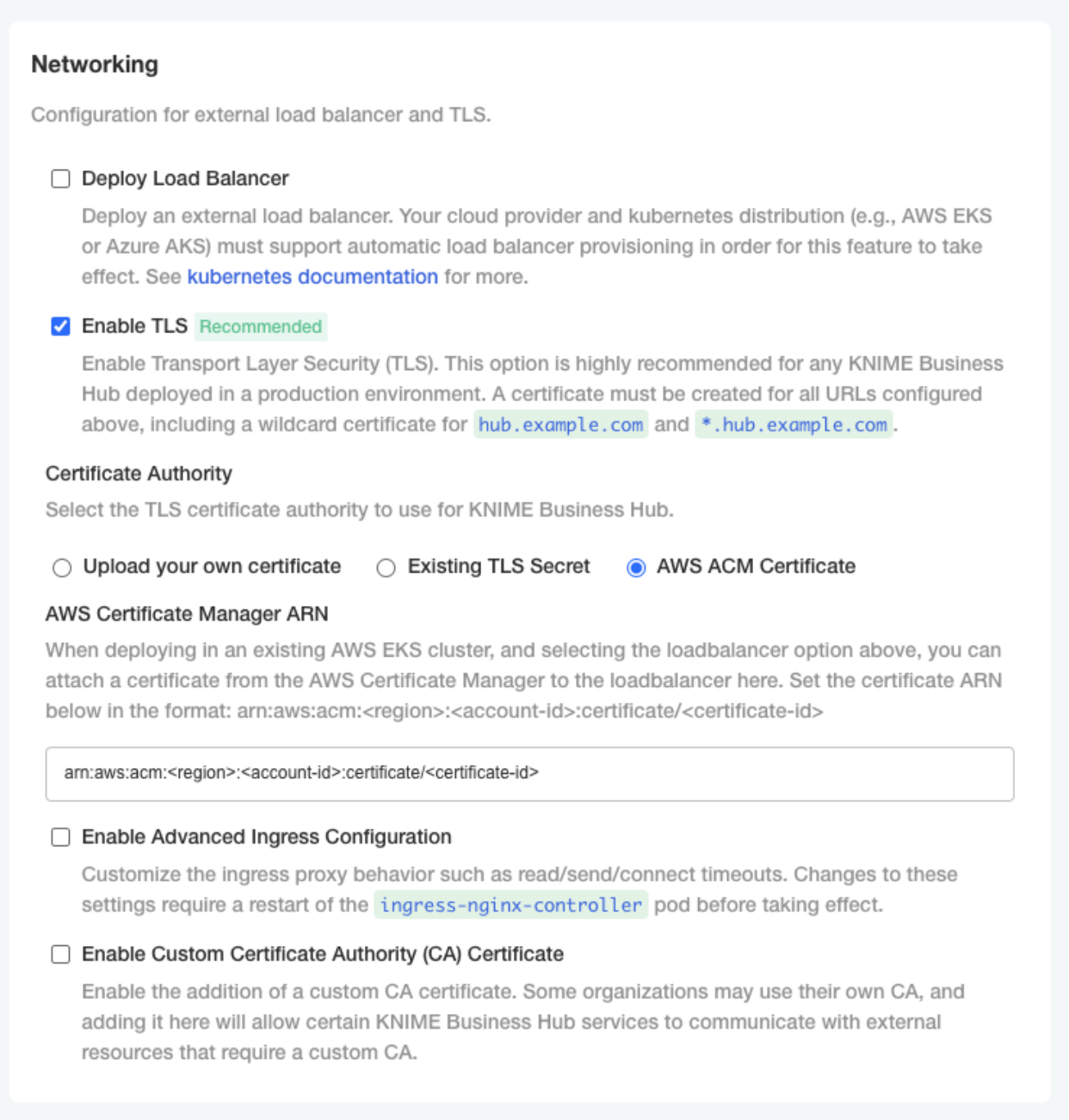
  This option is recommended if you have an automatic process that can create and renew `kubernetes.io/tls` Secrets in the cluster, like the cert-manager project.

  See ingress-nginx and kubernetes documentation on TLS secrets for more details.



- Select *AWS ACM Certificate* if, instead, you have deployed an AWS Elastic Load Balancer (ELB). In this case you can use AWS Certificate Manager (ACM) and set the certificate as an annotation directly on the Loadbalancer. You can find more information in AWS documentation for ACM here.

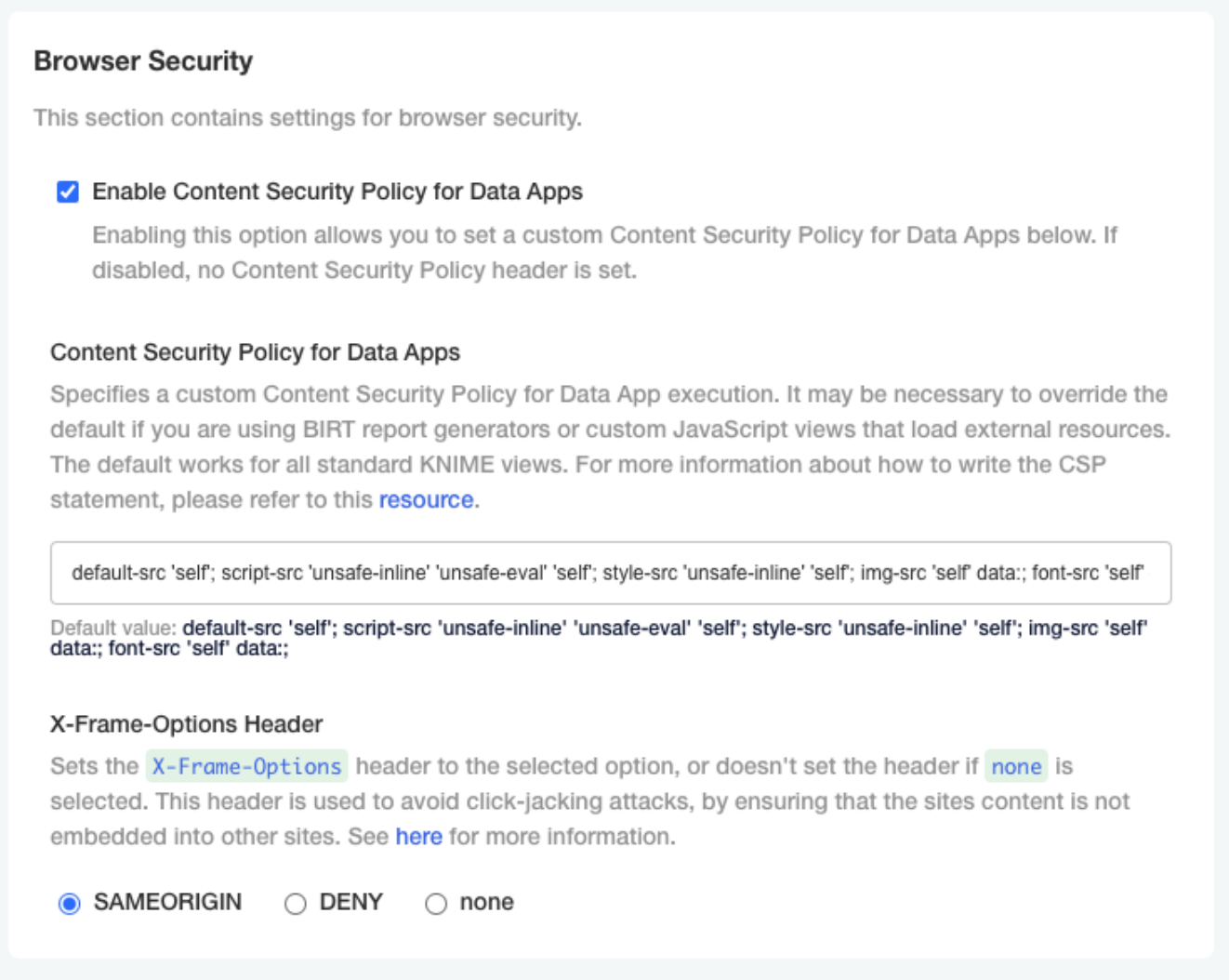
  Once you obtained the certificate Amazon Resource Name (ARN) in the form `arn:aws:acm:<region>:<account-id>:certificate/<certificate-id>`, insert the ARN in the corresponding field as shown in the image below.



Configure Browser Security

In the "Browser Security" section of the KOTS Admin Console you can:

- Specify a custom Content Security Policy for Data App execution. It may be necessary to override the default if you are using custom JavaScript views that load external resources. The default works for all standard KNIME views. For more information about how to write the CSP statement, please refer to this resource.
- Configure the X-Frame-Options header being set by webapps. This header is used to avoid click-jacking attacks, by ensuring that the sites content is not embedded into other sites. See here for more information.



Job viewer feature configuration

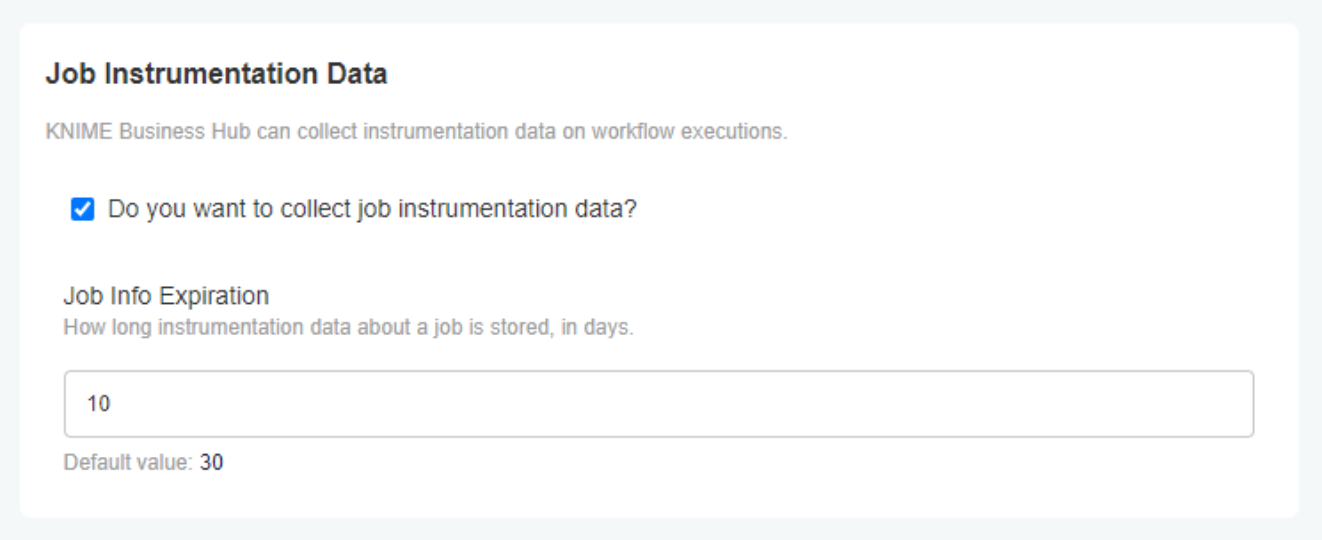
With the job viewer users are able to inspect a job at its current state in a browser based read only mode, without the need to open their local KNIME Analytics Platform.

In order for this feature to be available you will need: * **KNIME Analytics Platform 5.2.3** or newer to be used as executor * Load balancers / proxies in front of Business Hub need to be **websocket compatible**, and the `ws.<baseurl>` DNS entry needs to be set up as its shown on the KOTS admin console.

Activate and configure Job Instrumentation Data

To make use of the service that stores job's data you need to check the relative option in the "Job Instrumentation Data" section of the KOTS Admin Console. By default the collected data are deleted after 30 days. Here, you can also change this value to the amount of desired days.

You can use the Workflows Jobs Monitoring data application for obtaining insights about the status of workflow executions, usage of execution resources across different teams, or identify faulty jobs.



Installation of AI services (Enterprise edition only)

The AI service is a Business Hub Enterprise feature that enables end-users to connect the AI features of their Analytics Platform (such as the KNIME AI Assistant and Code Generation) to KNIME Business Hub.

The AI service is configured via the KOTS Admin Console.

The configuration consists of three parts:

- LLM provider
- Disclaimer & welcome messages
- AI history access groups

LLM provider

Currently, it is possible to configure the AI service to use either OpenAI or Azure OpenAI as backend.

To configure OpenAI as LLM provider do the following steps:

1. Create an OpenAI account
2. Generate an API key on the API keys page. It is recommended to create a fresh API key that is used exclusively by the AI service.

To configure Azure OpenAI as LLM provider do the following steps:

1. Create an Azure account and get access to Azure OpenAI
2. Create an Azure OpenAI resource to be used by the service and enter it in the KOTS Admin Console. It is recommended to create a separate resource that is used exclusively by the AI service to avoid any interference with other applications. See the Azure OpenAI documentation to learn how to create a resource.
3. Deploy a GPT 3.5 Turbo, a GPT 4 and an embeddings model with API version 2023-07-01-preview or newer. The embeddings model must be a text-embedding-ada-002 for the AI service to function properly. See the Azure OpenAI documentation for more details.

Models usage

The GPT 3.5 Turbo and the embeddings model are used for the Q&A mode of the KNIME AI Assistant.

The GPT 4 model is used for code generation and the build mode of the KNIME AI Assistant.

The reason for this distinction is that the GPT 4 model is more capable than the GPT 3.5 Turbo model but also much slower to respond and more expensive. If response time and cost are no concerns, GPT 4 can also be used for the Q&A mode.

Disclaimer & welcome messages

The KOTS Admin Console also allows to customize the initial messages displayed by the KNIME AI Assistant. The disclaimer is not shown by the KNIME AI Assistant if it is left empty.

AI history access groups

The AI service also exposes an endpoint that allows to retrieve the requests that were made to the AI service via the KNIME AI Assistant. The KOTS Admin Console allows to configure which Keycloak groups can access the endpoint.

It is recommended to only give access to admins.

The address depends on the hub but it is typically located at `https://api.<base-url>/ai-history/kai`.

GET requests to this endpoint need to have the authorization header set with a valid bearer token from the Hub.

You have the possibility to filter data by date directly within the endpoint, for instance:

```
https://api.<base-url>/ai-history/code/python?start_time=2023-12-
31T00:00:00&end_time=2024-04-10T00:00:00
```

The format of the parameters `start_time` and `end_time` needs to be `yyyy-mm-ddThh:mm:ss` as in the example above.

A data application is available to help you monitor and govern K-AI usage in your KNIME Business Hub instance. Find more information and a step-by-step guide here.

Node affinity

Node affinity makes it possible to ensure that cluster resources intended for a specific task, e.g. execution resources, run on a specific set of nodes. There are two roles that each pod is grouped into: `core` and `execution`. Pods in the `core` group consist of KNIME Business Hub control plane resources, and pods in the `execution` group relate to execution contexts.

In order to use the node affinity feature in your KNIME Hub cluster, you can apply one or both of the following labels to nodes within your cluster:

- `hub.knime.com/role=core`
- `hub.knime.com/role=execution`

To label a node, you can execute the following command (where `<node-name>` is the name of the node you want to label):

```
kubectl label node <node-name> hub.knime.com/role=core
```

For more information about labeling nodes, see the Kubernetes documentation.

Pods will have to be restarted in order to be rescheduled onto labeled nodes. You can use the following example commands to restart the pods in a live cluster:

- `kubectl rollout restart deployment -n istio-system`
- `kubectl rollout restart deployment -n hub`
- `kubectl rollout restart deployment -n knime`
- `kubectl delete pods --all --namespace hub-execution`

This command will restart all execution context pods.

There are a few things to note about the behavior of this feature:

- Node affinity uses a "best effort" approach to pod scheduling.
  - If one or both of the `hub.knime.com/role` labels are applied, cluster resources will attempt to be scheduled onto the nodes based on their role.
  - If no nodes have a `hub.knime.com/role` label, pods will be scheduled onto any available node.
  - If labeled nodes reach capacity, pods will be scheduled onto any available node.

    - If a labeled node is shut down, pods will be rescheduled onto other nodes in the cluster with a preference towards using nodes that have a matching label.
    - Node affinity for KNIME Business Hub uses the `preferredDuringSchedulingIgnoredDuringExecution` approach (see the Kubernetes documentation for more details).

- It is possible to use only one of the labels above, e.g. labeling nodes for the `execution` role but not specifying any node labels for the `core` role.

Scalability options for selected Hub services

KNIME Business Hub comes with preconfigured resources for the various services. These resource allocations will be fine for average sized deployments. Depending on factors like number of users, execution load, and type of usage, it will be necessary to give more resources to certain services.

Enabling scalability options for selected Hub services adds more flexibility in setting scalability and resource usage options for your Hub instance services. Your Business Hub pods will be able to scale according to load in an automatic way.

You can specify resource usage for the following Hub services:

- Accounts
- Catalog
- Execution Rest Interface
- Search
- Websocket Proxy
- Search-sync-service
- Artemis
- Elasticsearch

In order to enable the scalability settings go to the KOTS Admin Console and under the *Config* tab, go to the *Global* section and check the option *View Scalability Settings*.

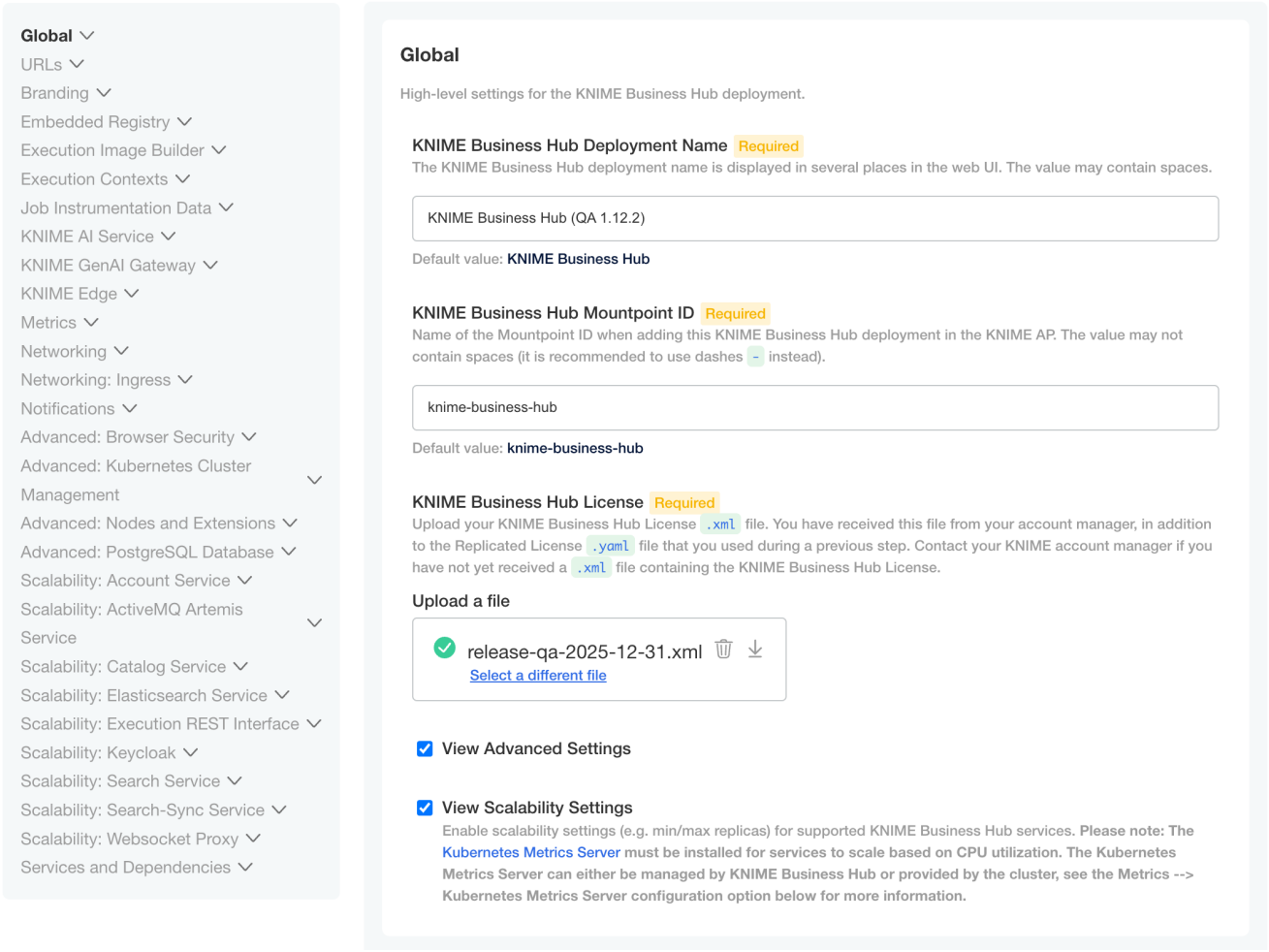


Figure 36. Enable scalability settings for you Hub instance from the KOTS Admin Console

Specify resource usage for Hub services

Once you have enabled the feature by checking the *View Scalability Settings* option, you can configure the following settings for the desired service:

- Minimum replicas
- Maximum replicas
- Target CPU Utilization
- CPU Resources
- Memory Resources

You can do so from the KOTS Admin Console by going to the *Scalability* section of each service, e.g. *Scalability: Accounts Service* for the accounts service and so on. In Figure 37 you can see an example of how to set up the scalability for the accounts service.

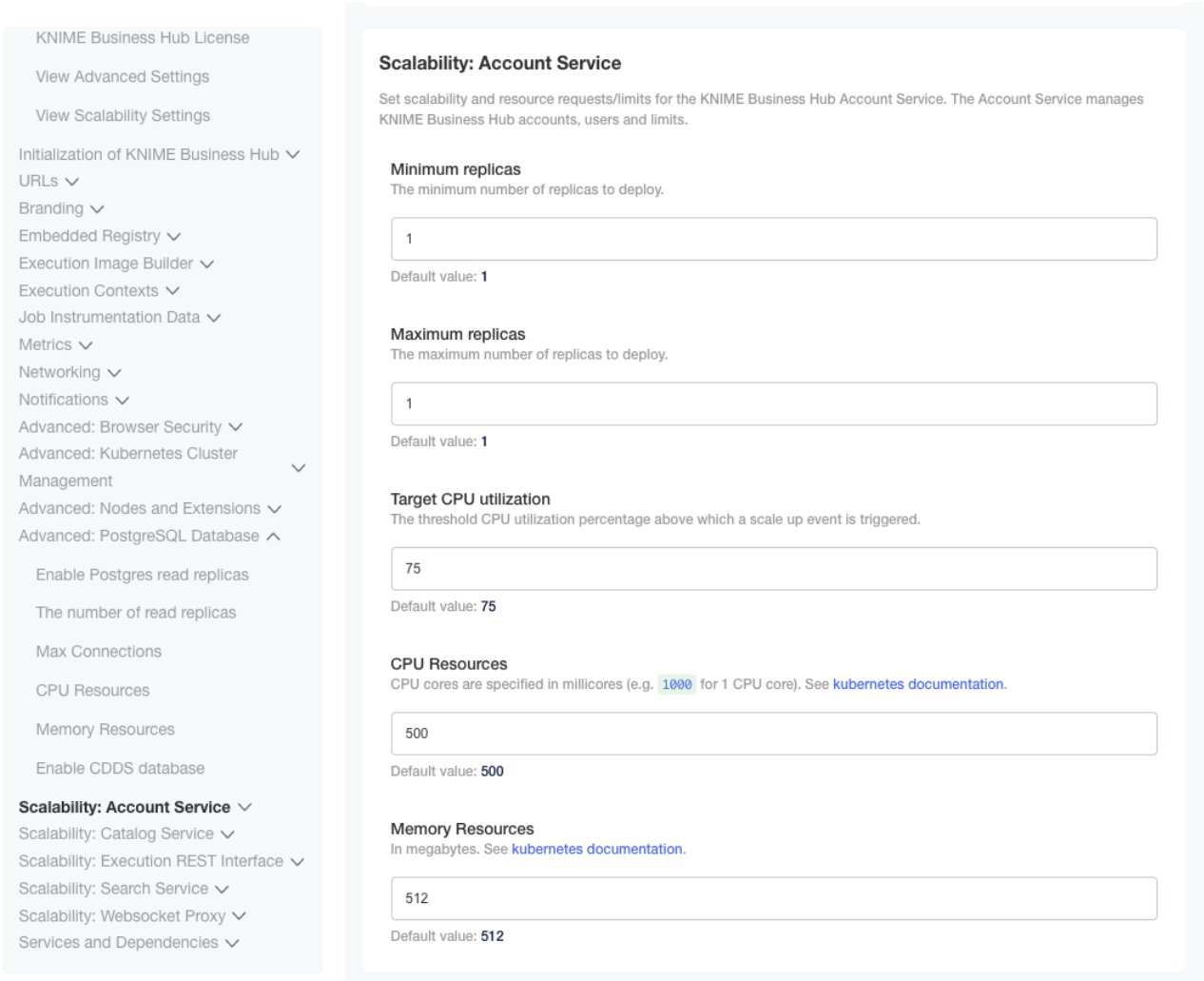


Figure 37. Configure scalability settings for Hub services - Accounts service example

KNIME GenAI Gateway (Enterprise edition only)

The GenAI Gateway allows Hub admins to centrally configure GenAI models inside of the Business Hub which can then be accessed in KNIME workflows with dedicated nodes. The GenAI Gateway supports chat and embeddings models which can be queried using the KNIME Hub Chat Model Connector and KNIME Hub Embeddings Model Connector, respectively. Both nodes use the KNIME Hub Authenticator to connect to the Hub that is running the GenAI Gateway.

This is a feature of KNIME Business Hub - Enterprise edition.

Installation

The GenAI Gateway is part of the KNIME Business Hub Enterpise offering. It can be enabled via the KOTS admin page by checking the *Enable GenAI Gateway* checkbox in the *KNIME GenAI Gateway* section.

If you have a KNIME Business Hub Enterprise license and the **section is not shown**, notify your KNIME contact person to update your license.

Model management

The Business Hub admin can manage the available models on the Hub administration page by selecting the GenAI Gateway menu entry. The page lists the already registered models and allows to add new or remove existing models.

Models are added via the  button which will open a sidepanel where the model configuration can be added.
The following configurations have to be provided for all model vendors:

- **Name**: The name under which the model will be available in the respective connector node. This name is also displayed on GenAI Gateway admin page.
- **Description**: An optional description of the model that is displayed on the GenAI Gateway admin page.
- **Type**: The type of the model, i.e. either *Chat* or *Embedding*. This information is used to filter the model list for the respective connector nodes and is also displayed on the GenAI Gateway admin page.

See below for more details and examples for the individual vendors.

OpenAI

OpenAI is an AI research and deployment company most widely known for their ChatGPT chat bot. They also offer an API that provides access to their LLM and embeddings models.

It requires the following parameter to be configured:

- **Model**: The name of the OpenAI model.
- **API Key**: A valid OpenAI API key.

Here is an example for configuring the GPT-3.5 Turbo model:

```
Model: gpt-3.5-turbo
API Key: <your OpenAI API key>
```

Azure OpenAI

Azure OpenAI provides access to OpenAI's GenAI models with the security and enterprise promise of Azure. It allows to manage details of deployed Azure OpenAI Resources and endpoints. This includes which region of the world the workloads are computed in and which models are deployed. For more details on the different configuration options please refer to the official documentation.

It requires the following parameters to be configured:

- **Model**: The name of the deployed model prefixed with `azure/`
- **API Key**: One of the API keys of the deployment.
- **API Base**: The endpoint URL of the deployment.
- **API Version**: The API version of the deployment.

Here is an example configuration for a deployment with an endpoint at `https://example.openai.azure.com` that deploys the model `my-gpt4` with API version `2024-05-01-preview`.

```
Type: Chat
Model: azure/my-gpt4
API Key: <One of the deployment API keys>
API Base: https://example.openai.azure.com
API Version: 2024-05-01-preview
```

Amazon Bedrock

Amazon Bedrock is a service managed by AWS that provides access to a variety of GenAI models from different AI companies.

It requires the following parameters to be configured:

- **Model**: The name of the model prefixed with `bedrock/`
- **Additional Model Params**
  - **aws_access_key_id**: The ID of the AWS access key to use for authentication.
  - **aws_secret_access_key**: The AWS secret key to use.
  - **aws_region_name**: The region in which to operate the resources.

Here is an example for configuring Amazon's Titan Text Express model:

```
Model: bedrock/amazon.titan-text-express-v1
Additional Model Params:
{
    "aws_access_key_id": "...",
    "aws_secret_access_key": "...",
    "aws_region_name": "eu-central-1"
}
```

Google AI Studio

Google AI Studio allows to access Google's Gemini LLMs.

It requires the following parameters to be configured:

- **Model**: The name of the model to configure prefixed with `gemini/`.
- **API Key**: A valid Goolge AI Studio API key.

Here is an example for configuring Google Gemini 1.5 Pro:

```
Model: gemini/gemini-1.5-pro-latest
API Key: <Your Goggle AI Studio API key>
```

Anthropic

Anthropic is an AI company most well-known for their Claude models.

It requires the following parameters to be configured:

- **Model**: The name of the model without any prefix.
- **API Key**: A valid Antropic API key.

Here is an example of configuring their latest model Claude 3.5 Sonnet:

```
Model: claude-3-5-sonnet-20240620
API Key: <Your Anthropic API key>
```

OpenAI API Compatible Providers

OpenAI's API is also used by a variety of other providers for example … It is also used by a number of inference solution that allow hosting your own model such as …

In order to configure such a model, the following parameters are required:

- **Model**: The name of the model prefixed with ´openai/`. This distinction from models hosted by OpenAI is crucial.
- **API Key**: A valid API key for the provider if it requires one.
- **API Base**: The base URL of the OpenAI-compatible API.

Hugging Face Hub Dedicated Inference Endpoints

Hugging Face Hub is the platform for accessing open source GenAI models. Via its Dedicated Inference Endpoints it also provides an easy way to quickly deploy a variety of LLMs and embeddings models to the cloud of your choice.

It requires the following parameters to be configured:

- **Model**: The repository of the model prefixed with `huggingface`.
- **API Key**: A valid Hugging Face API key.

- **API Base**: The URL of the deployed endpoint.

Here is an example of configuring a dedicated endpoint that deploys Mistral /B Intruct 0.3:

```
Model: huggingface/mistralai/Mistral-7B-Instruct-v0.3
API Key: <A valid Hugging Face API key>
API Base: https://example.eu-west-1.aws.endpoints.huggingface.cloud
```

Create a collection (Enterprise edition only)

It is possible to create collections on your KNIME Business Hub instance.

KNIME Collections on KNIME Hub allow upskilling users by providing selected workflows, nodes, and links about a specific, common topic.

One example of a collection can be found on KNIME Community Hub here.

This is a feature of KNIME Business Hub - Enterprise edition.

In order to create a new collection page you need to be a global admin of your KNIME Business Hub instance.

The creation of a collection is possible via REST API, and a description of the different configurations can be found in your KNIME Business Hub API doc at the following URL:

```
api.<base-url>/api-doc/?service=catalog-service#/Collections
```

e.g. `api.hub.example.com/api-doc/?service=catalog-service#/Collections`.

In order to create a collection the items (i.e. workflows and nodes) that are collected need to be stored and accessible on the same KNIME Business Hub instance where collection is created.

To create the collection you will need then to build a json file with the schema that is available in the API doc in the Collections section, under the `POST /collections` request description.

The following is an example that would allow you to build a collection, similar to the one available on KNIME Community Hub here.

In the first section you can for example set up a title, a description, a so-called hero, which is the banner image at the top right of the example collection page, and tags:

```
{
  "title": "Spreadsheet Automation",
  "description": "On this page you will find everything to get started with spreadsheet
automation in KNIME",
  "ownerAccountId": "account:user:<global-admin-user-id>",
  "hero": {
    "title": "New to KNIME?",
    "description": "Get started with <strong>KNIME Analytics Platform</strong> to import
all the examples and nodes you need for spreadsheet automation right now!",
    "actionTitle": "Download",
    "actionLink": "https://www.knime.com/downloads"
  },
  "tags": [
    "Excel",
    "XLS"
  ],
```

Next you can add different sections and subsections, each with a title and a description, choose a layout, and select the `itemType` such as *Space*, *Component*, *Workflow*, *Node*, *Extension*, or *Collection*. For each of these items you will need to provide the `id` under which they are registered in your Business Hub installation.

The `id` for workflows, spaces, components, and collections can be build by taking the last part of their URL, after the `~`, and adding a * at the beginning. For example, the following workflow on the KNIME Community Hub has URL https://hub.knime.com/-/spaces/-/latest/~1DCip3Jbxp7BWz0f/ so its `id` would be `*1DCip3Jbxp7BWz0f`. The `id` for node and extensions instead needs to be retrieved with a REST call, for example to the `search` endpoint of your KNIME Business Hub instance.

```
  "sections": [
    {
      "title": "Workflow examples",
      "description": "Some subtitle text here. Can have <strong>bold format</strong>",
      "iconType": "Workflow",
      "subsections": [
        {
          "title": "How to do basic spreadsheet tasks in KNIME",
          "description": "Some examples on how to do common things",
          "layout": "SingleColumn",
          "numberOfTeaseredItems": 2,
          "items": [
            {
              "title": "Click Here!",
              "itemType": "Link",
              "absoluteUrl": "https://knime.com"
            },
            {
              "id": "*SJW5zSkh1R3T-DB5",
              "itemType": "Space"
            },
            {
              "id": "*vpE_LTbAOn96ZOg9",
              "itemType": "Component"
            },
            {
              "id": "*MvnABULBO35AQcAR",
              "itemType": "Workflow"
            },
            {
              "showDnD": true,
              "id": "*yiAvNQVn0sVwCwYo",
              "itemType": "Node"
            },
            {
              "id": "*bjR3r1yWOznPIEXS",
              "itemType": "Extension"
            },
            {
              "id": "*QY7INTkMW6iDj7uC",
              "itemType": "Collection"
            }
          ]
        }
      ]
    }
  ]
}
```


Administrator workflows

The workflows described in this section of the documentation aim to support KNIME Business Hub administrators and heavy KNIME Business Hub users to configure, clean up, monitor, and administrate their Business Hub instance.

Workflows overview

You can access the workflows on the KNIME Community Hub in a public space owned by KNIME. Additionally, you can find them on a dedicated collection page. To use them, download the workflows from the Community Hub and upload them to an existing team space in your KNIME Business Hub installation.

Business Hub has three types of user roles (global admin, team admin, and team member). All users with access to the space where you uploaded the workflows can run them. The user's role defines their allowed actions when running the different workflows.

The workflows can be run ad-hoc or as a data app deployments that can be shared with other users not having access to the space where the workflows reside.

Below the list of administrator workflows, click them for further details:

- Admin Dashboard
- Manage All Existing Jobs
- Delete Old Versions
- Monitor User Usage
- K-AI in KNIME Business Hub
- Workflow Jobs Monitoring
- Proxy Diagnostics

Admin Dashboard

Overview

This workflow provides detailed insights into usage patterns of KNIME Business Hub, enabling effective and actionable analysis.

You can download the workflow from the KNIME Community Hub.

Workflow specs

The workflow collects data accessible to the user and then provides an overview of the current Hub usage in the form of a dashboard.

Depending on your user role on the KNIME Business Hub, if you execute this workflow, you will have the following permissions:

- **Global admin:** can view Admin Dashboard of all the users and teams on the Hub.
- **Team admin:** can view Admin Dashboard of the teams they administer.

To run the data app in KNIME Business Hub, it is possible to use an ad-hoc execution or run the data app deployment. Please follow this link for more information about data app deployments in KNIME Business Hub.

Deployment configuration

This workflow can be deployed as a data app deployment. By default, it uses the current Hub authentication by setting the default value to "Use current Hub." This option bypasses the need to input the Hub URL and the application password. The user must have the correct rights to access the data.

Alternatively, you can provide the following information to deploy the workflow as a data app:

1. **Hub URL**: The URL of your KNIME Business Hub instance, e.g. "https://my-business-hub.com".
2. **Application Password ID**: User-associated application password ID.
3. **Application Password**: User-associated application password.

To create an **application password**, follow these steps.

Data app

Below are listed the steps for the data app:

1. **Data Collection on Hub Usage.** Once the app is running, a progress bar will indicate the data collection process, which might take some time. This component can be edited by the user as needed.
2. **Admin Dashboard.** After data collection is complete, this dashboard provides an overview of the current Hub usage.

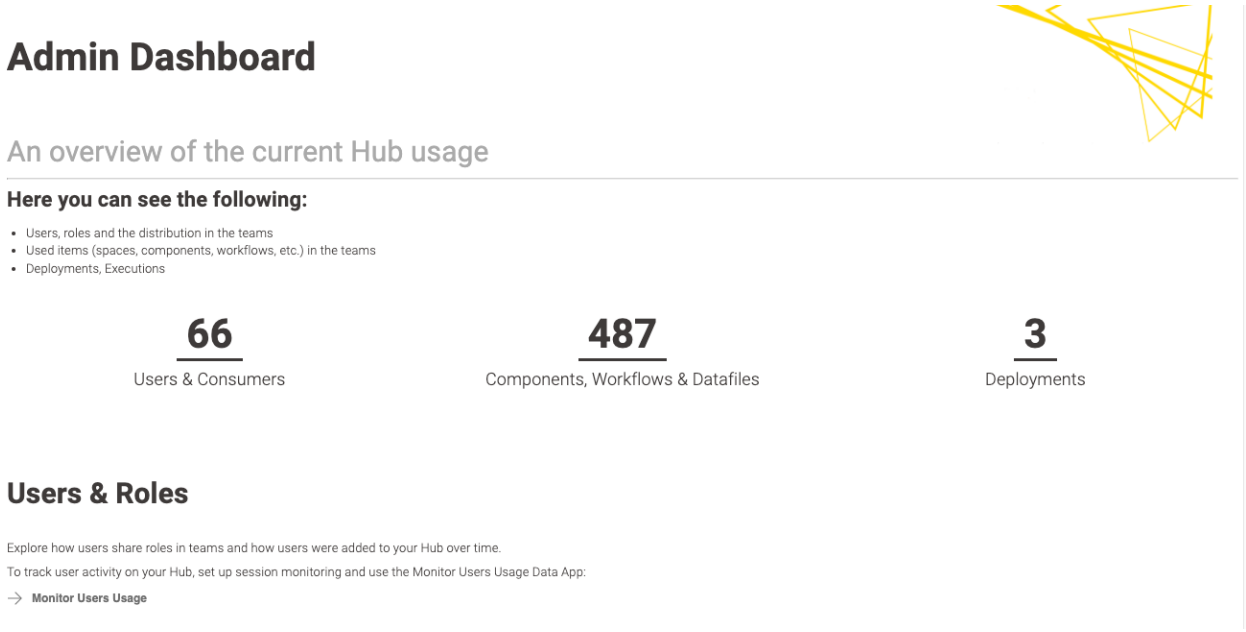


Figure 38. Admin Dashboard: An overview of the current Hub usage

The Admin Dashboard provides the following visualizations:

- Users, roles, and their distribution across teams. It enables the exploration of how roles are shared within teams and how users have been added to the Hub over time.

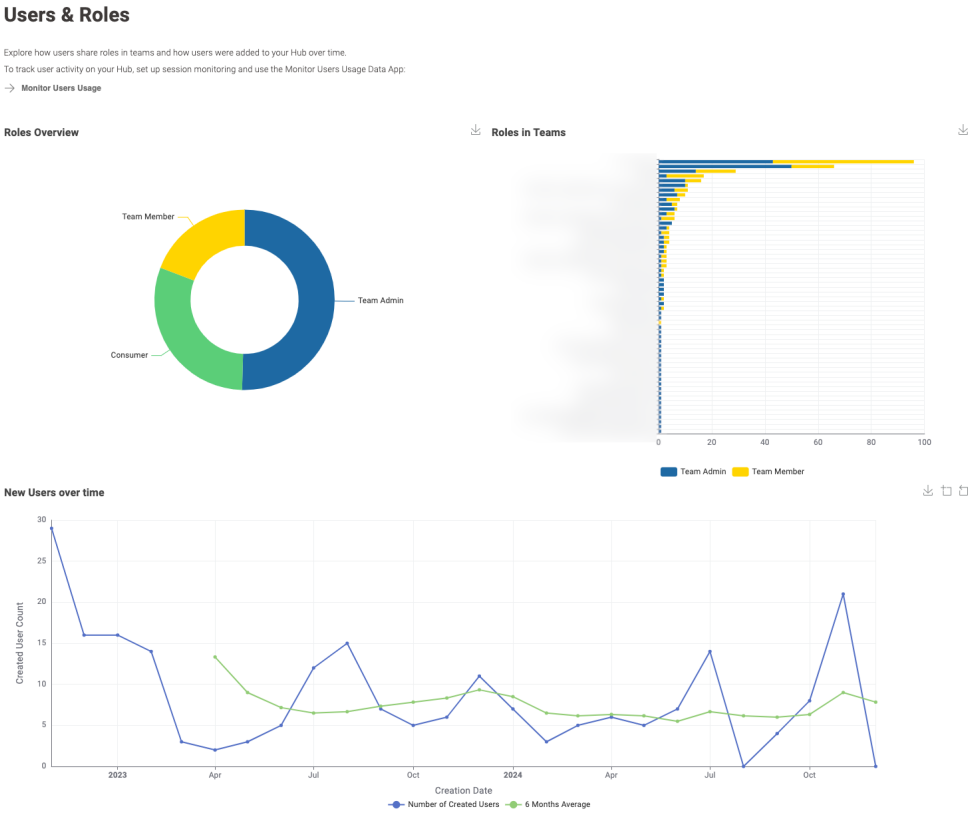


Figure 39. Admin Dashboard: Users & Roles Overview

- Used items (spaces, components, workflows, etc.) in the teams. It includes information on the distribution of item types across your Hub's teams, their growth trends over time, and item size distribution. Access items directly from the provided table to address issues such as empty spaces.
- Deployments, Executions. This part lets you explore deployments and shared execution context between different teams.

Manage All Existing Jobs

Overview

Job definition: Whenever a workflow is executed on KNIME Business Hub, it creates a job. Jobs can come from ad-hoc executions or deployments. The Hub's user interface allows users to explore these jobs from different touchpoints, including the Execution Resources and Deployment pages within each team, or the Administration panel for global admins.

The "Manage All Existing Jobs" workflow, deployed as a data app in KNIME Business Hub, displays **all existing** jobs, allowing users to select and delete them. Existing jobs are those in either the executor's memory or the object store (i.e. swapped).

Global or team admins can use the data app to **automate job management** across multiple execution contexts.

You can download the workflow from the KNIME Community Hub. After downloading, upload the workflow to your KNIME Business Hub instance and run it as a data app.

Workflow specs

Without applying any time range, the data app lists all existing jobs accessible to the user for the following execution types: ad-hoc executions, triggers, data apps, schedules or shared deployments.

The data app lists all types of job statuses, including the ones that have failed. For a detailed list click here.

The jobs a user could retrieve depends on the user role:

- **Global admin:** can recover all jobs in any team and space.

- **Team admin:** can recover all jobs within the team where it is an admin.
- **Team member:** can recover jobs from any team and space where it is a member (no matter the user's rights on space items). It also includes shared deployments from teams where the user is not a member.

The user's ability to delete jobs is determined by their role:

- **Global admin:** can delete all jobs in any team and space from any execution type.
- **Team admin:** can delete all jobs of the teams of which it is an admin from any execution type.
- **Team member:** can delete only self-generated jobs from any execution type, team and space of which it is a member (no matter the user's right on space items). It also includes deployments shared with the user from teams where the user is not a member.

Deployment configuration

This workflow must be deployed as a data app.

You can provide the following information to deploy the workflow:

1. **Hub URL**: The URL of your KNIME Business Hub instance, e.g. "https://my-business-hub.com".

It is possible to use the current Hub authentication by using the default value to "Use current Hub." This option bypasses the need to input the Hub URL and the application password. The user must have the correct rights to access the data.

2. **Username**: User-associated application password ID.
3. **Password**: User-associated application password.

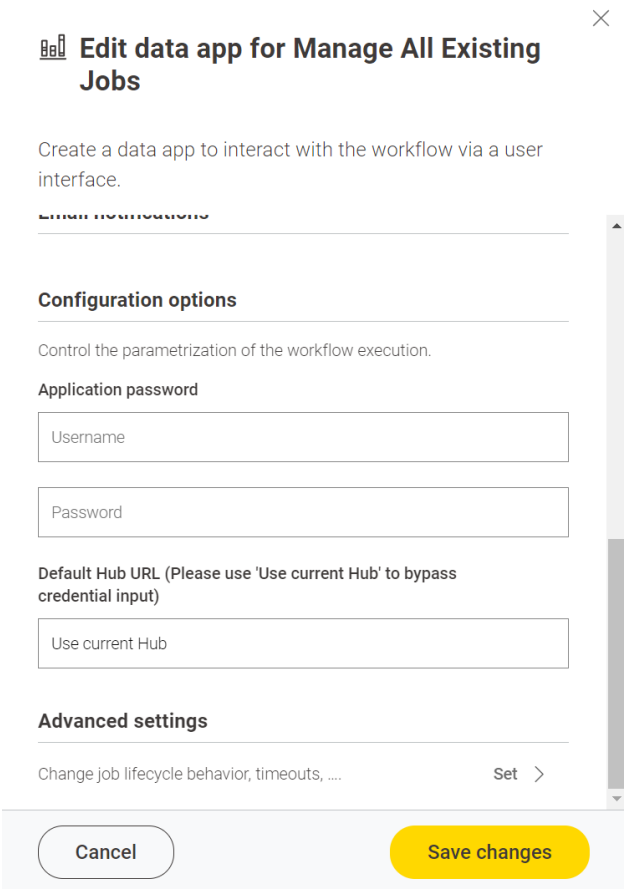


Figure 40. "Use current Hub" in the deployment configuration dialogue

If you want to know how to create an **application password**, follow these steps.

Data app

After deploying the workflow as a Data App, you can run it. To do so, follow the instructions.

Below is a brief description of each page in the data app:

1. **Available Jobs:** This feature provides a list of jobs with detailed information, including workflow name, team, job state, execution context, deployment type, duration, and execution timestamp. Users can open the job in the job viewer. If the job is a data app in the "Interaction Required" state, users can reopen or terminate the data app as needed.

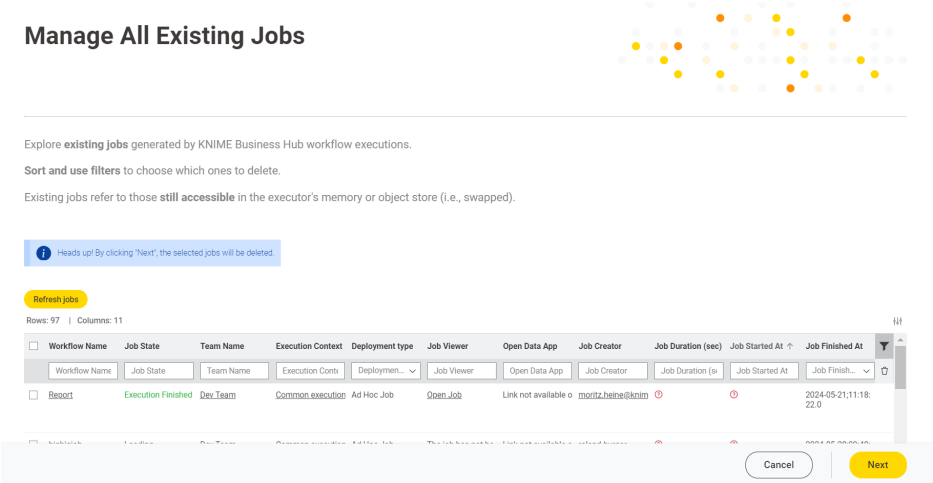


Figure 41. Available jobs table in the data app

a. As shown in the image above, you can use the table **filters** to narrow your research and **sort** the columns by clicking the column header.

b. By selecting the jobs from the table and clicking the "Next" button, the application will attempt to delete them in the next step.

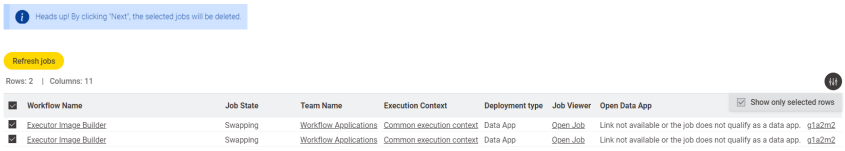


Figure 42. The selected jobs will deleted after clicking the "Next" button.

c. There is a **"Refresh jobs"** button to fetch the latest status of the KNIME Business Hub instance, as more jobs could be created while the data app is running.

d. The table displays **100 jobs**. Use the pagination arrows at the top to navigate the table.

e. Two exceptions related to the job's deployment information:

   i. When you run a workflow in KNIME Business Hub using the Run button, it generates a job, but the deployment name is unavailable. Therefore, the deployment type is "Ad Hoc Job."

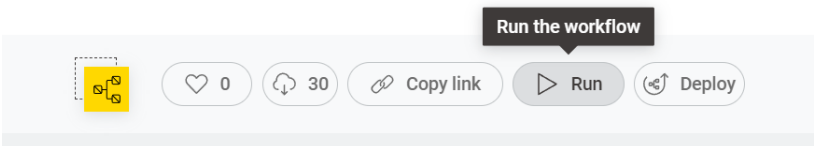


Figure 43. Ad-hoc execution via the Run button.

   ii. If the deployment that generated a job is no longer available in the KNIME Business Hub at the time the current data app is executed (due to deletion

or the end of a scheduled deployment), it will not be possible to retrieve the deployment information. Instead, a message will be displayed in the "*Deployment type*" column: "*Not Available*."

2. **Deletion Results:** The table displays job deletion results, including job name, deletion result, team name, deployment type, and workflow name.

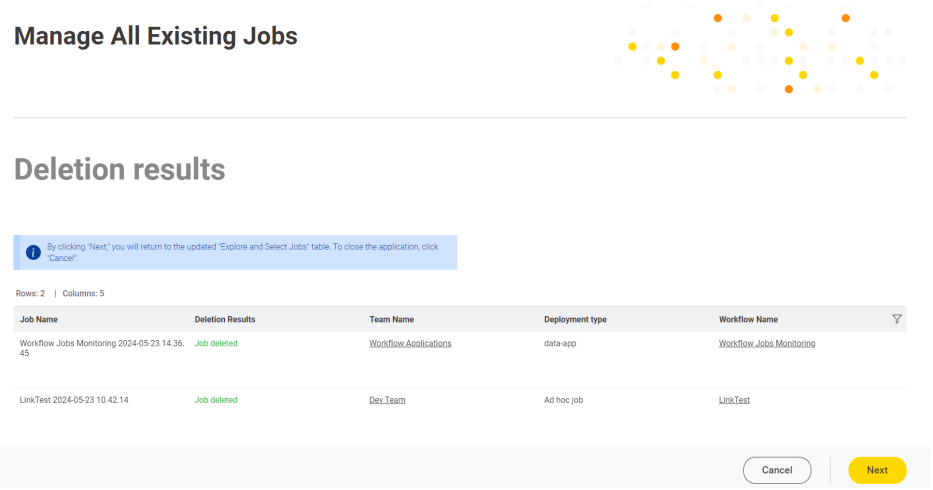


Figure 44. Deletion results table in the data app

   a. If the data app cannot delete a job, an error message will be displayed in the deletion result column.

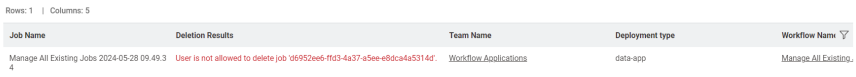


Figure 45. The logged users has insufficient rights to delete the job.

   b. By clicking the "Next" button, the user will return to the updated "Explore and Select Jobs" table. To close the application, click "Cancel".

3. **Warnings and errors**:

   a. The data app interacts with the KNIME Business HUB API using `GET` requests to fetch the jobs and `DELETE` requests to delete them. It also ensures that a meaningful error is displayed in case of server-side issues or problems with the API call.

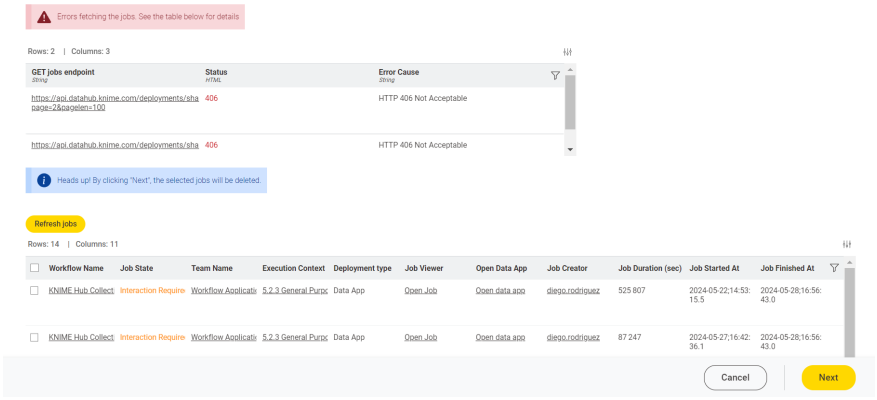


Figure 46. GET Request error displayed in the data app

b. If the user doesn't select any jobs in the "Explore and Select Jobs" table and then clicks "Next", the data app will display a warning on the "Deletion Results" page. Clicking "Next" again redirects you to the "Explore and Select Jobs" table.

Figure 47. Warning message displayed in the data app

c. Also if no jobs are availablein the KNIME Business Hub instance for the logged user, a message will be displayed.

Delete Old Versions

Overview

The workflow aims to delete old item versions that aren't involved in any deployment. The sought effect of this operation is to avoid a massive proliferation of item versions within the Business Hub installation, impacting disk space.

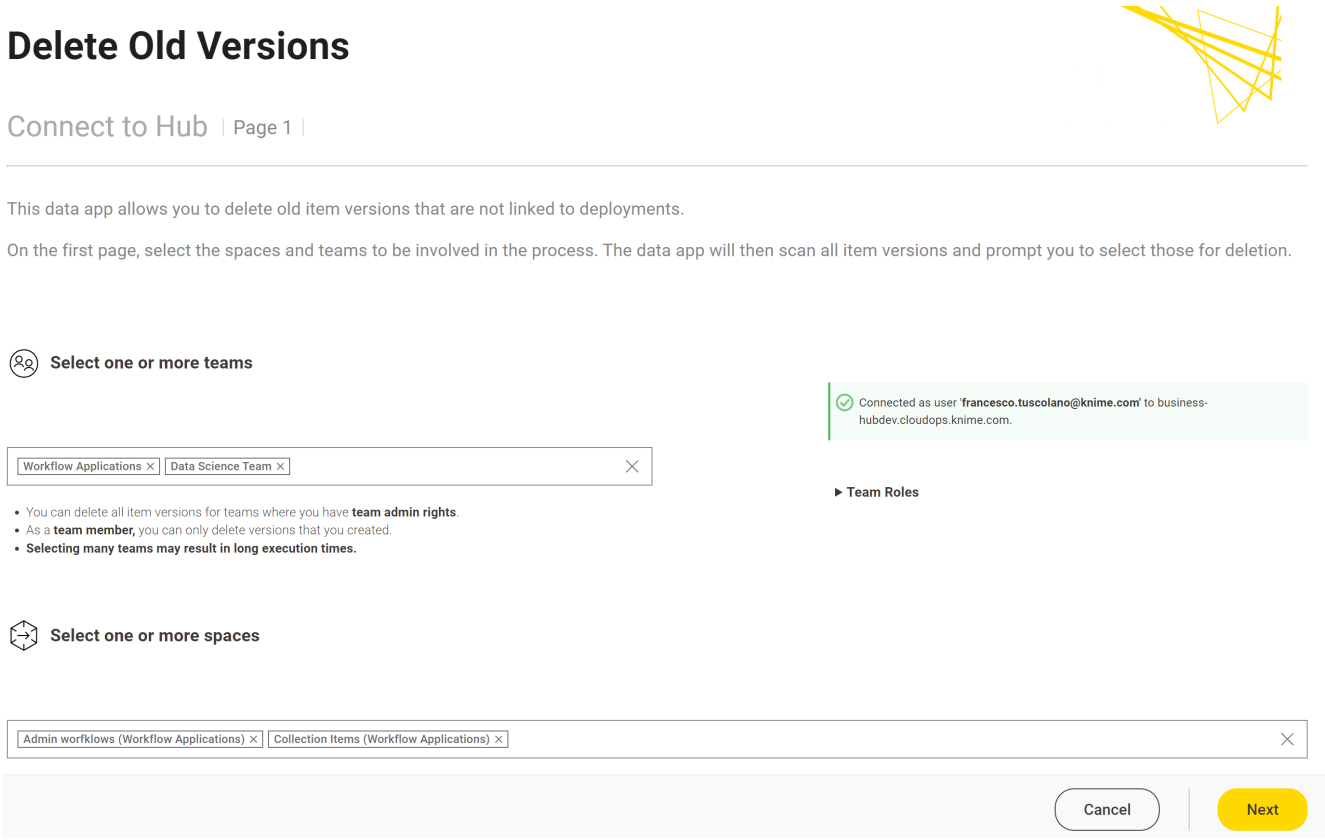
Workflow specs

The workflow deletes all item versions older than the specified number of days, e.g. older than seven days.

The deletion will only be applied to the selected teams and spaces using the workflow as a data app.

The latest item version will not be deleted, even if it is affected by the rule. Additionally, also item versions that are involved in deployments are exempt from the rule, to avoid breaking the deployment.

Initially, all items are retrieved from the selected Hub teams and spaces, which you will choose in the first step. Afterward, all available versions are displayed in a table, allowing you to select the ones you wish to delete.



Depending on your user role on the KNIME Business Hub, if you execute this workflow, you will have the following permissions:

Global admin: Can delete every item version not used in any deployment from any team on the KNIME Business Hub instance.

Team admin: Can only delete the item versions not involved in any deployment in the team where it's an admin.

Team member: Can only delete the item versions not involved in any deployment in the teams where it's a member. The Team member must have "Edit" permissions for the targeted spaces to perform the version deletion.

Deployment configuration

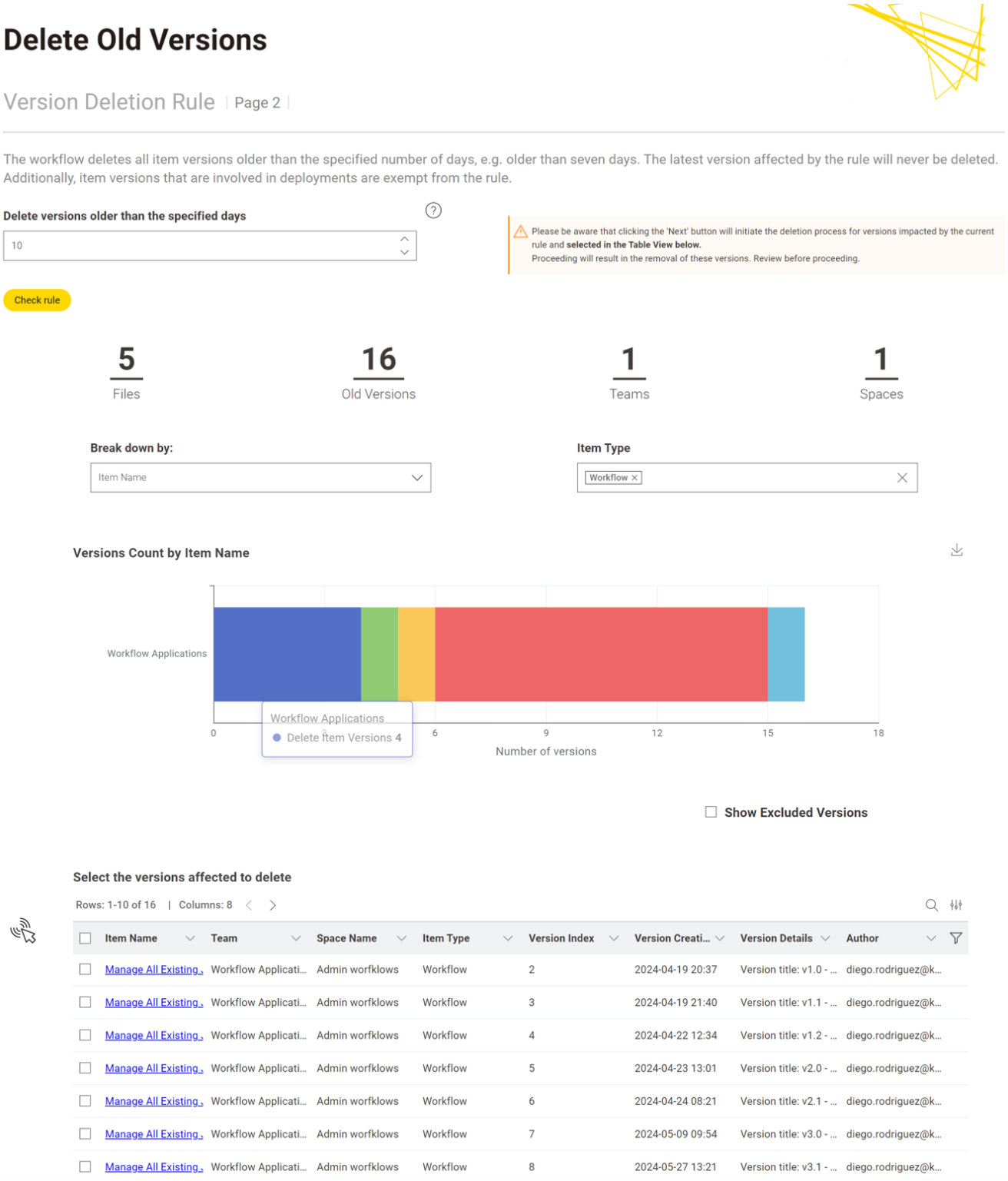
This workflow can be deployed as a data app

Data app

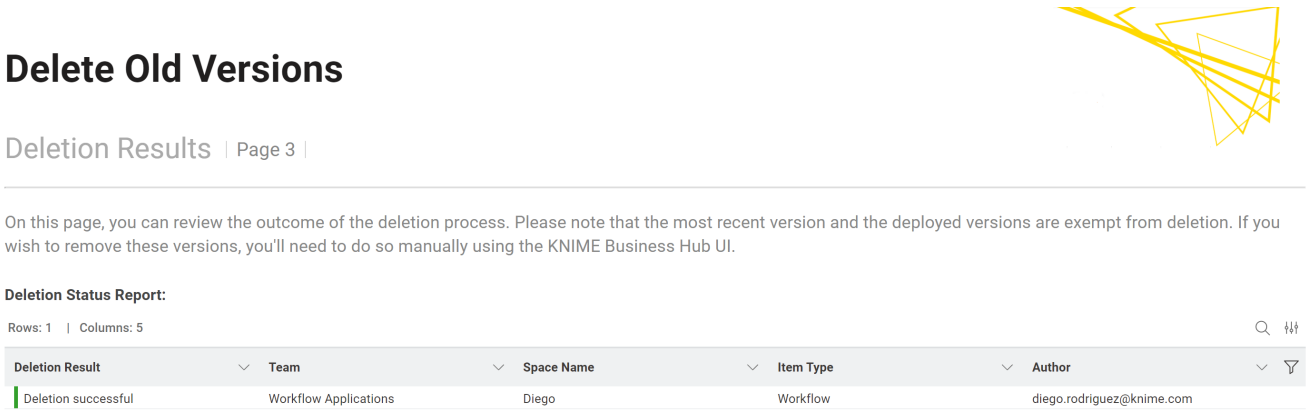
After deploying the workflow as a data app, you can run it. To do so, follow these instructions.

Below you can find the steps for the data app:

1. **Business Hub connection**: The workflow automatically connects to the Hub instance where it is running.
2. **Team and space selection**: Use the widgets in the first page to select the teams and associated spaces from which the workflow should delete the old versions.
3. **Define version deletion rule**: Here, you can set a version deletion rule. All versions affected by the rules are avaialbe for the selection inside the table view. You can also visualize the versions distribution among different variables like teams, spaces and single items. Optionally, you can check which versions are affected by the rules but excluded from the selection.



1. **Deletion result**: A table showing the deletion result with the version information will appear by default.



1. In the Team Deployments page or the workflow page, Deployments section, you can check the number of executions and their status.

Monitor User Usage

Overview

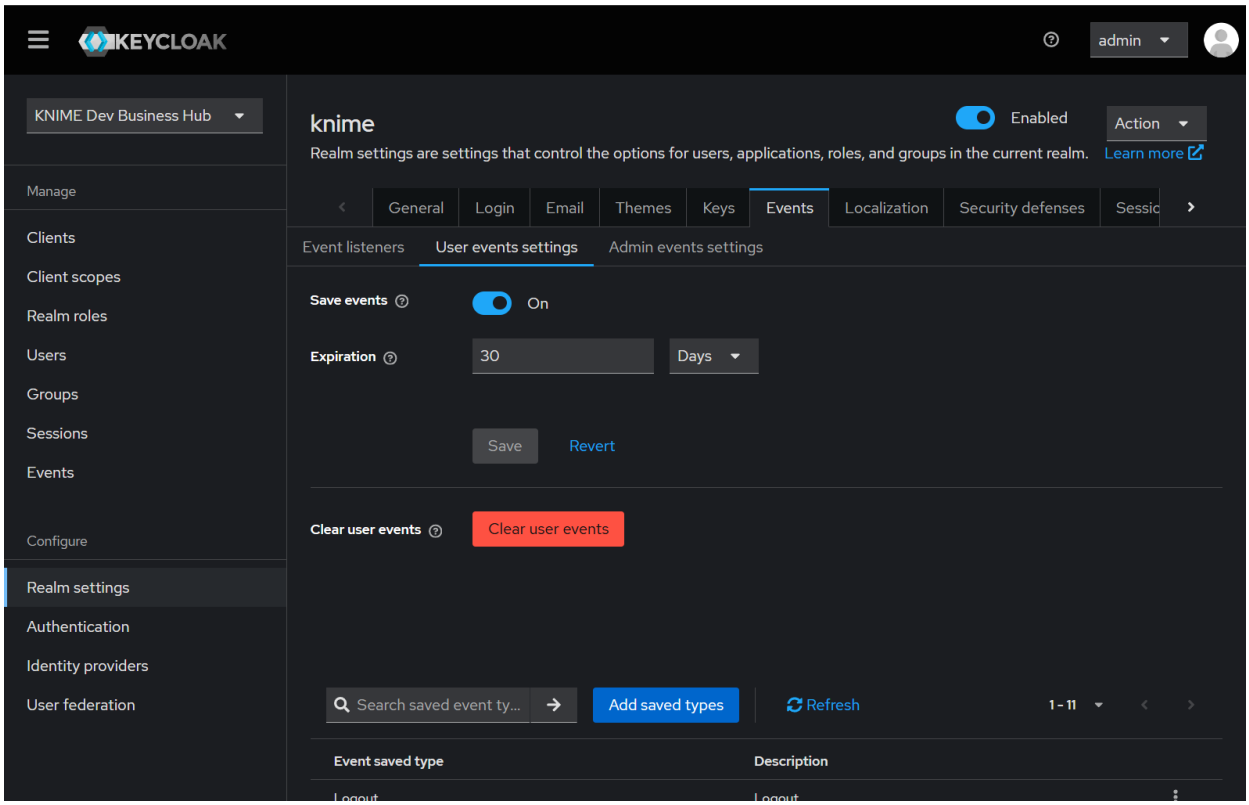
This data app lets you investigate KNIME Business Hub user sessions to gain insights on the KNIME Business Hub usage behavior. It requires the setup of a Keycloak Client that stores this data for a given amount of time, which is described below.

The data app informs you about users that where active during the logging period, and those who where not. It displays the sessions over time, highlighting periods of high usage. Furthermore, it breaks down the sessions individually, showing if people use your KNIME Business Hub via browser, Analytics Platform, or both.

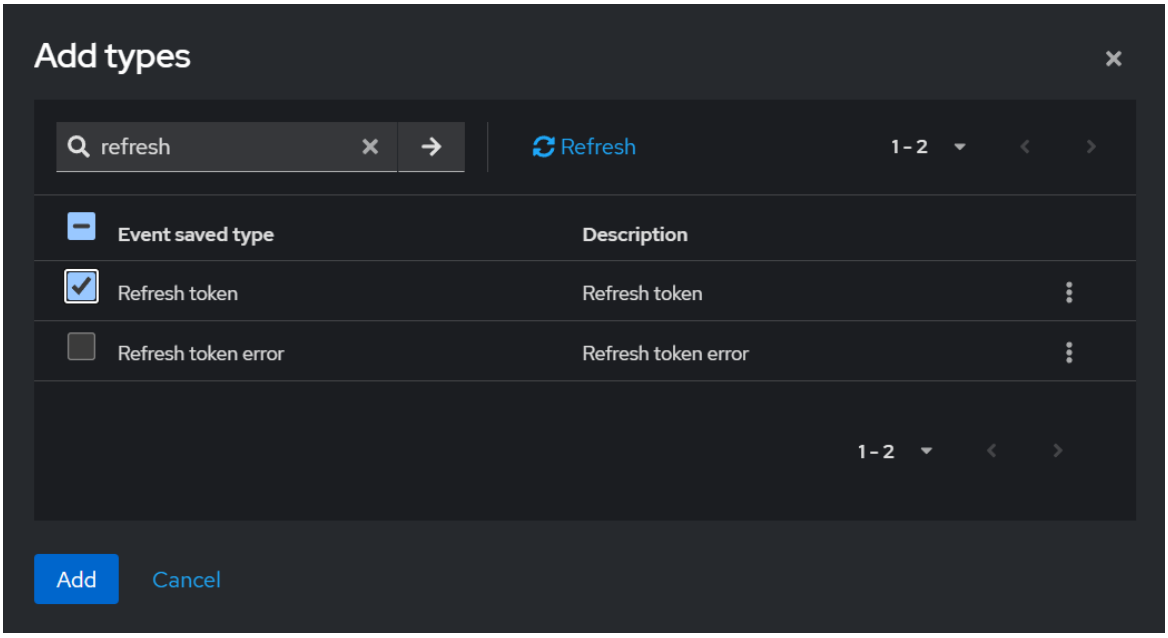
Keycloak Client Setup

Here you will configure keycloak such that it stores user events, and setup a service client that has permissions to retrieve these events. This service client will be used by the data app to retrieve the user events.

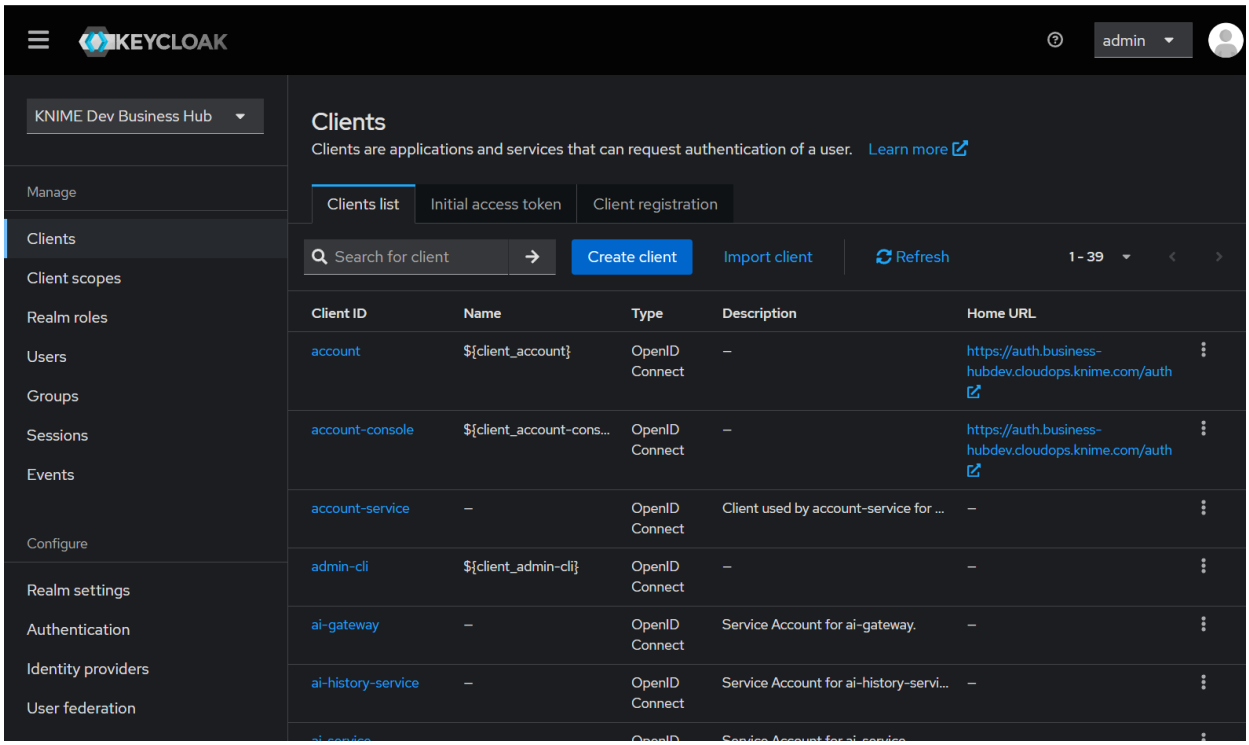
1. Login into Keycloak Admin Console: *"https://auth.my-business-hub.com" > Administration Console*. Learn how to retrieve the credentials here.
2. Select your Realm (usually `knime`), and from *Realm Settings > Events > User events settings*, activate "Save events" to save KNIME Business Hub Users events. You can configure how many days to keep the user's events.



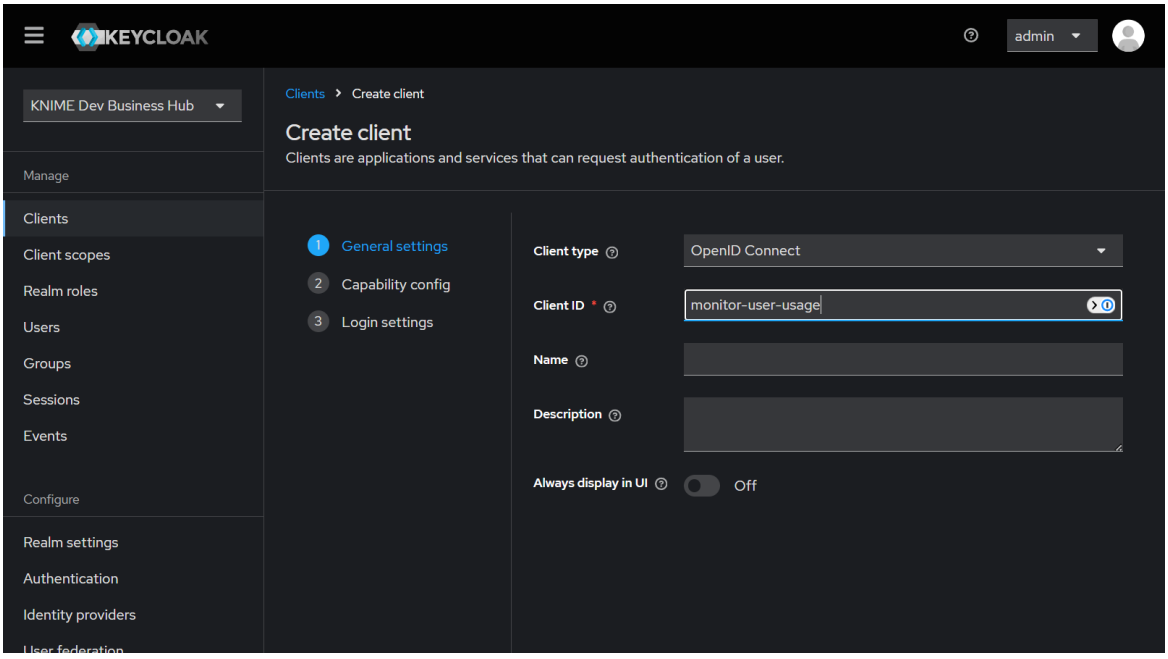
a. Select "Add saved types" and add the refresh token event to the list of events that are saved.



3. Create a new Client from *Clients* > *Create Client*.

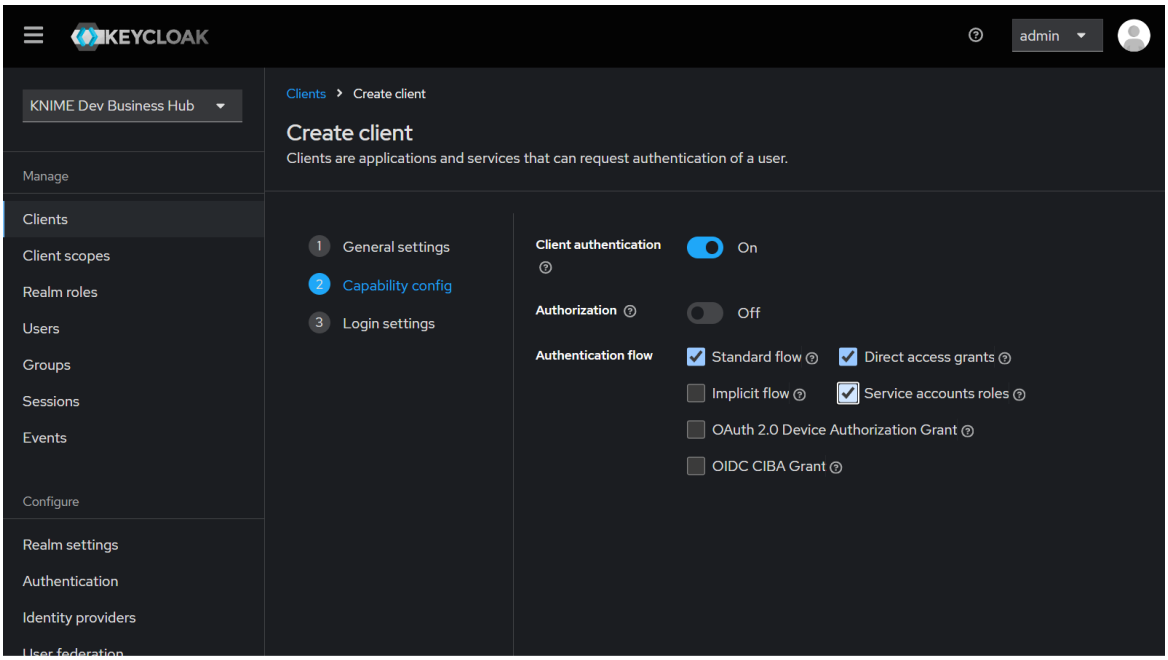


a. Give it a Client ID (`monitor-user-usage` in the screenshot). Optionally, add a name and a description to make clear what this client is for.

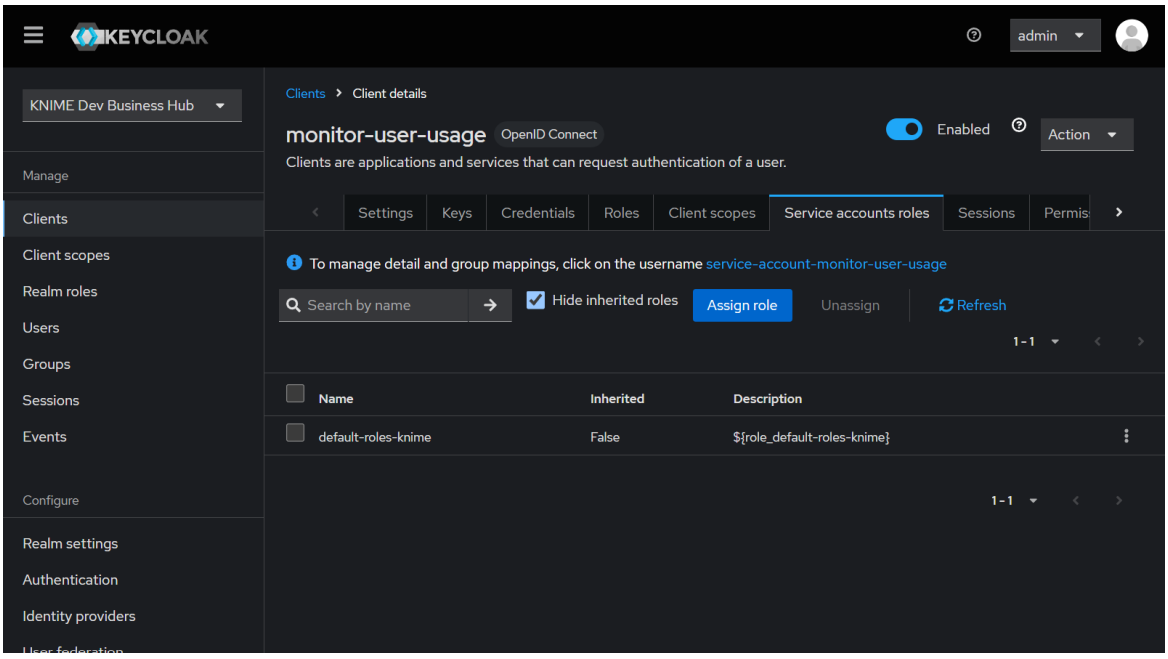


b. In the "Capability config" section, activate "Client authentication". Further, select "Service account roles" to allow you to authenticate this client to Keycloak and retrieve the access token dedicated to this client. The "Login settings" don't need to be configured.

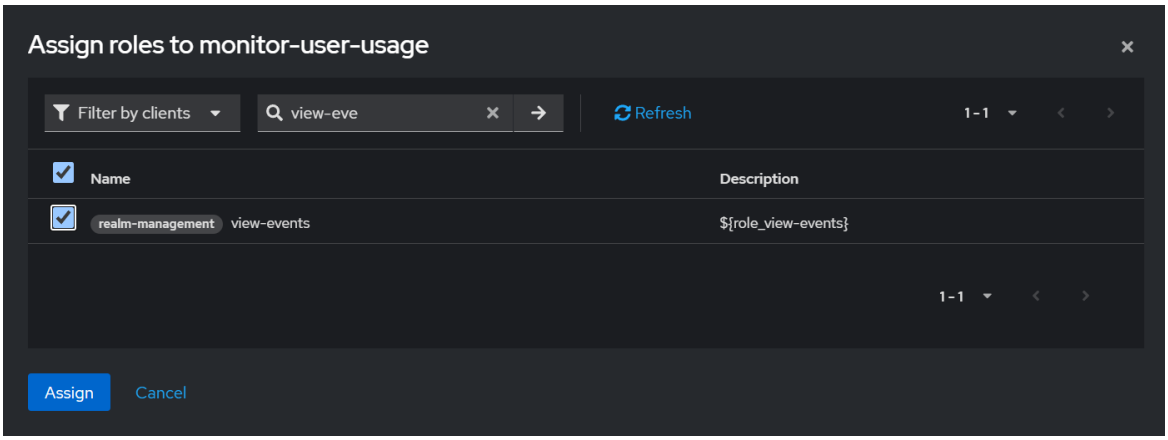
c. Save your new client.



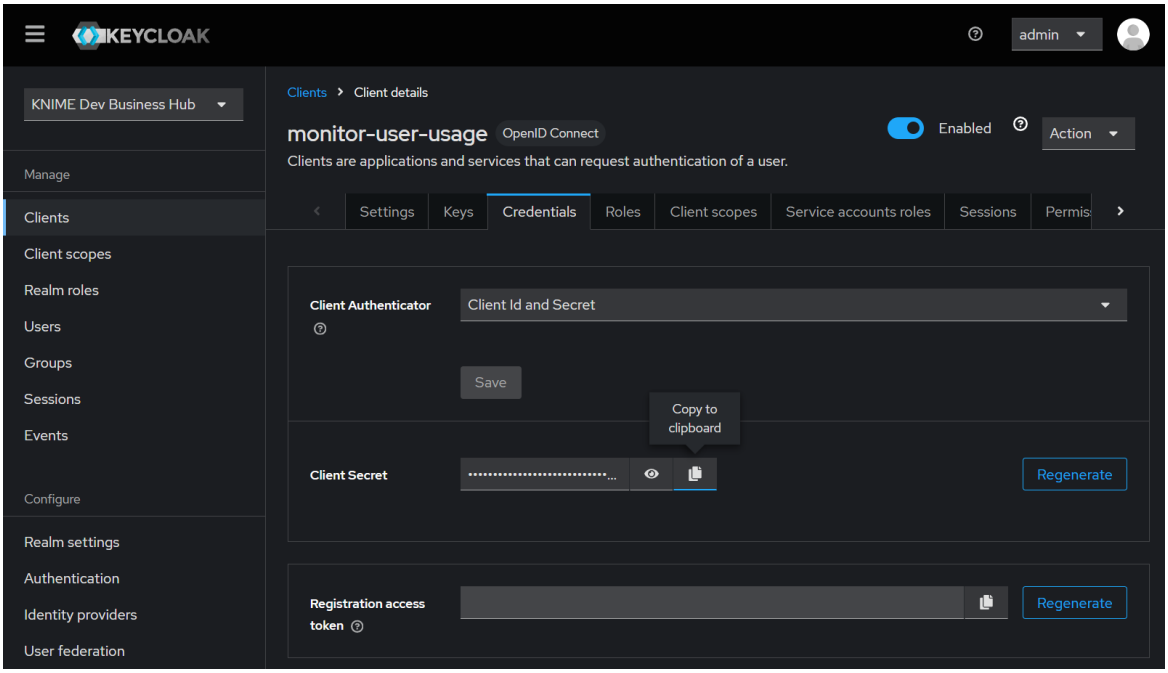
4. In *Clients > Client ID (Column)*, find and click on your new Client (`monitor-user-usage`).
   a. Go to the "Service account roles" tab and click the "Assign role" button.



   b. Via "Filter by clients", search for "view-events" and for "manage-users" and assign both roles to the service account associated with your client.



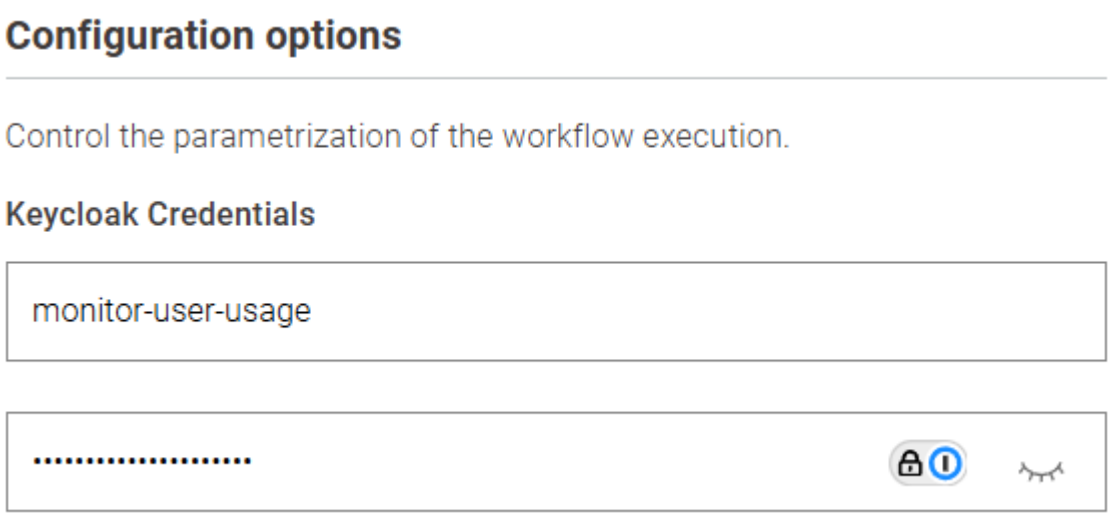
5. Finally retrieve the Client ID and Client secret:
    a. Go to *Clients > Client ID (Column)* and choose your client (`monitor-user-usage`)
    b. Click on the "Credentials" tab
    c. Leave as a Client Authenticator the "Client ID and Secret" option.
    d. Copy the Client ID from the top of the tab (`monitor-user-usage` in the screenshot below).
    e. Copy the Client's Secret.



Use the retrieved Client ID and Secret in the data app deployment of the Monitor User Usage workflow.

Deployment configuration

The workflow is designed to be deployed as a data app. Please provide the following keycloak service client credentials configuration to deploy the workflow.



This way, the credentials are safely stored with the deployment, and you can manage access for other Hub users.

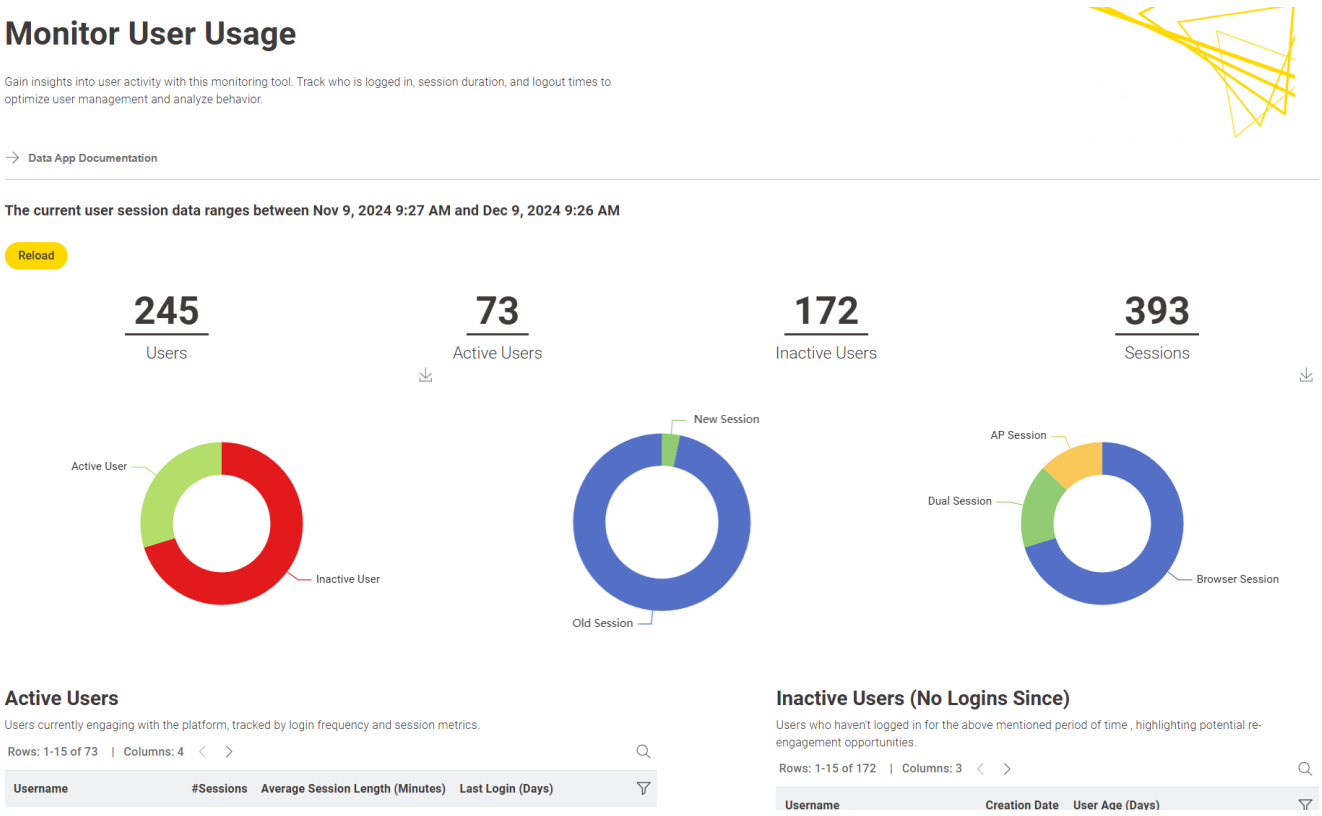
Alternatively, you can run the data app ad-hoc, entering the credentials each time you execute.

Note that the data app assumes you are using the default keycloak realm name `knime`. In case it differs for you, you can configure the `Monitor User Usage` component directly in the workflow to update the keycloak realm name.

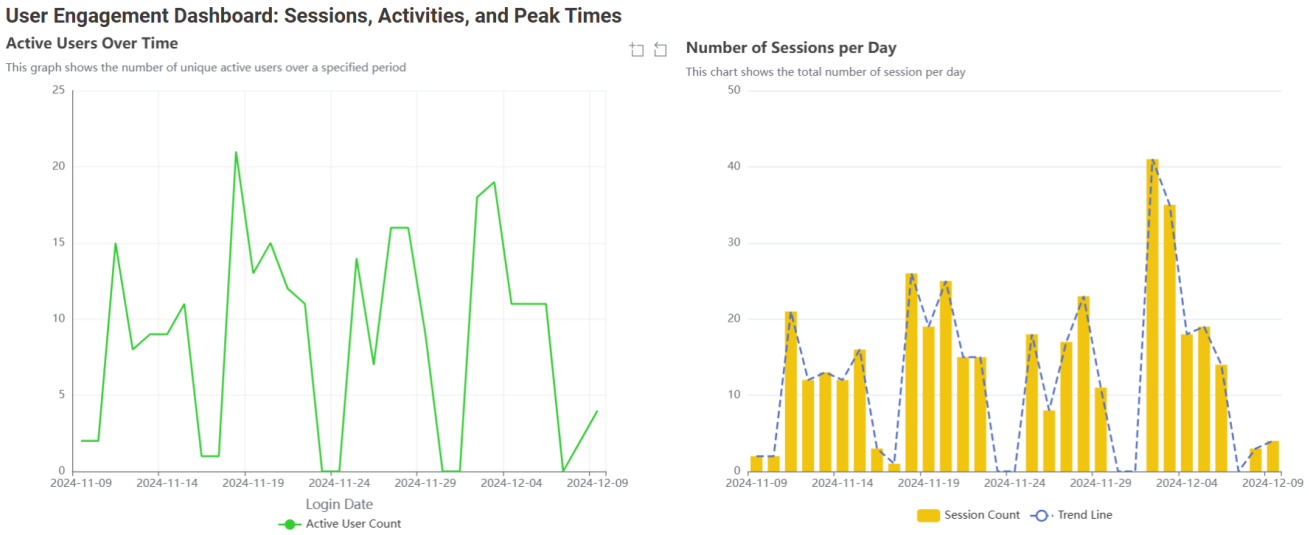
Data app

The data app consists of four sections. The first section gives you an overview over the logging period, lets you reload the data, and shows basic statistics such as the number of users and sessions or the counts of active sessions. "AP Sessions" are sessions where the user was connected via the Analytics Platform (AP) and "Browser Sessions" where the user was connected via the browser only, e.g. to execute data apps. "Dual Sessions" are sessions where the user was logged in via AP and Broswer simultaneously.

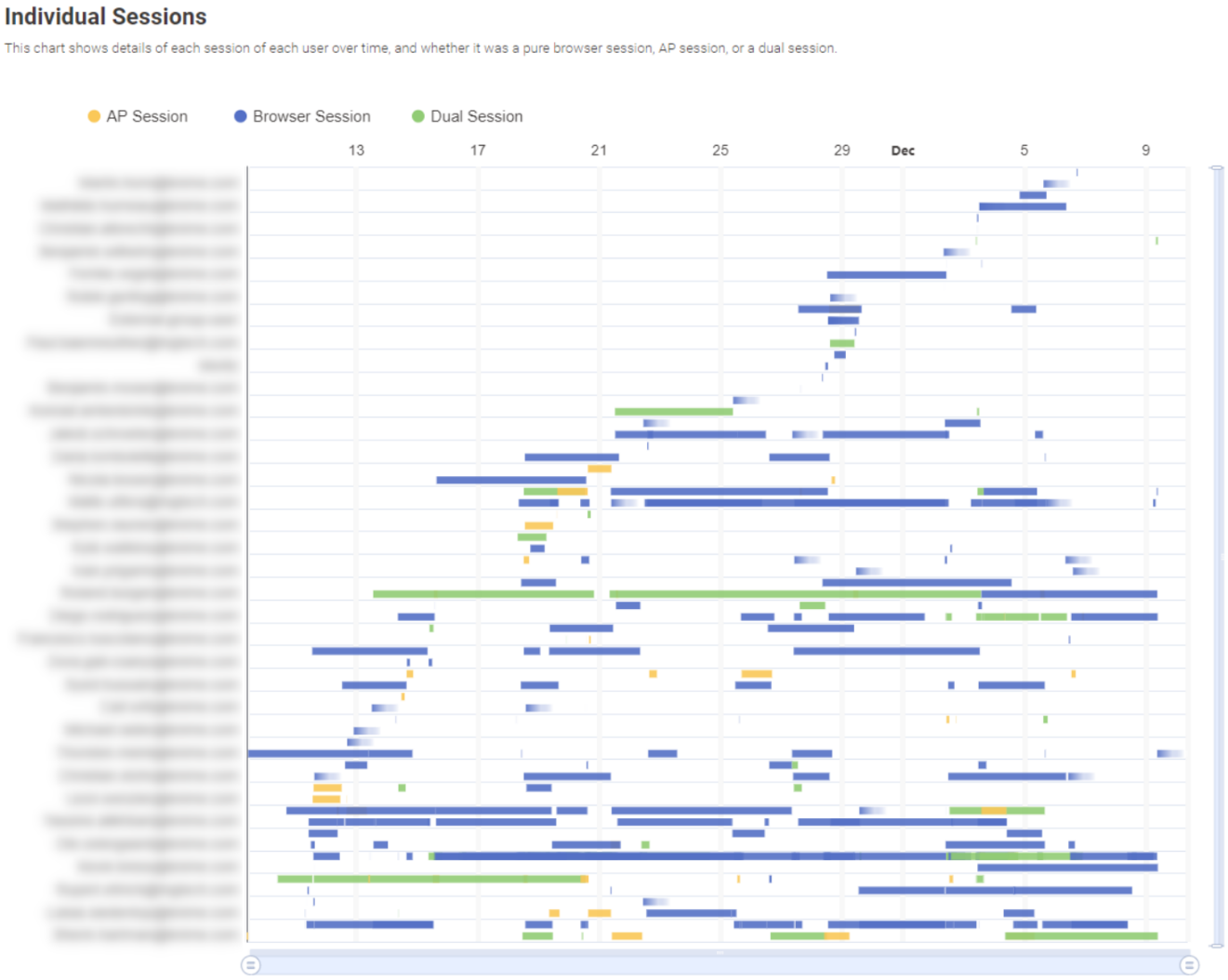
The next sessions shows lists of active and inactive users. The usernames are linked to their pages on the Hub, in case you want to learn more about an individual.



The third section shows time-development of the sessions. The example below nicely shows that on weekends almost no usage happens, and indicates that the latest week was slightly more busy than the weeks before.



In the final section, a gantt chart shows individual sessions of individual users, distinguishing again Browser, AP and Dual sessions.



Note that not every session has a logout event, e.g. when the AP or browser is closed during a session. If no logout event is present, the sessions end is approximated by the latest refresh token event (which happens every 5 minutes by default). If there us still no logout date, the sessions are shown blurred in the Gantt chart.

Monitoring K-AI in KNIME Business Hub

A data app is available to help KNIME Business Hub admins monitor and govern K-AI usage.

Download the data app and upload it to your KNIME Business Hub instance and *Run* it or deploy it as a data app.

> Follow the steps 1 to 3 (or 1 to 5 if you want to create a data app deployment) of the Data apps on KNIME Hub guide to know how to do this.

Data access

To know more about the AI history access groups, please consult the KNIME Business Hub Admin Guide.

Data structure

Every user on the Hub can interact with K-AI. When users open their KNIME Analytics Platform and initiate a prompt, a conversation with K-AI begins. The conversation concludes when the user closes the *Assistant* window. Each question the user asks is a prompt, and the corresponding answer is a response. This results in three distinct levels of K-AI usage data, since each user has multiple conversations consisting of several prompts and responses.

Table 2. An example of usage data for K-AI

User ID	Account created (year)	Conversation ID	User Prompt	K-AI Response	Timestamp	Tokens consumed
37373	2022	1	Hi!	Hello! How can I assist you today?	2024-01-01T00:00:00	330
37373	2022	1	What node can I use to close a loop with the Group Loop Start node	You can use the "Loop End" node to close a loop with the "Group Loop Start" node.	2024-01-01T00:05:00	1220

User ID	Account created (year)	Conversation ID	User Prompt	K-AI Response	Timestamp	Tokens consumed
37373	2022	2	How can I read an Excel file?	To read an Excel file in KNIME, you can use the "Excel Reader" node from the "KNIME Excel [...]	2024-01-05T20:00:00	880

Based on the data in the table, on January 1st, 2024, a user (identified as 37373) interacted with K-AI for five minutes and made two different prompts with varying token consumption values. On January 5th, the same user issued a prompt inquiring about how to read an Excel File. This prompt was stored with a different Conversation ID, indicating that the K-AI assistant was closed between the two conversations.

Deploying the data app

To run the data app in KNIME Business Hub, it is possible to use an Ad hoc execution or run the data app deployment. Please follow this link for more information about data app deployments in KNIME Business Hub.

In the data app deployment configuration, the current Hub authentication can be used if the logged user has permission for the correct rights to access the data. Otherwise, a user with permissions must provide their application password, and it is necessary to input the Hub URL.

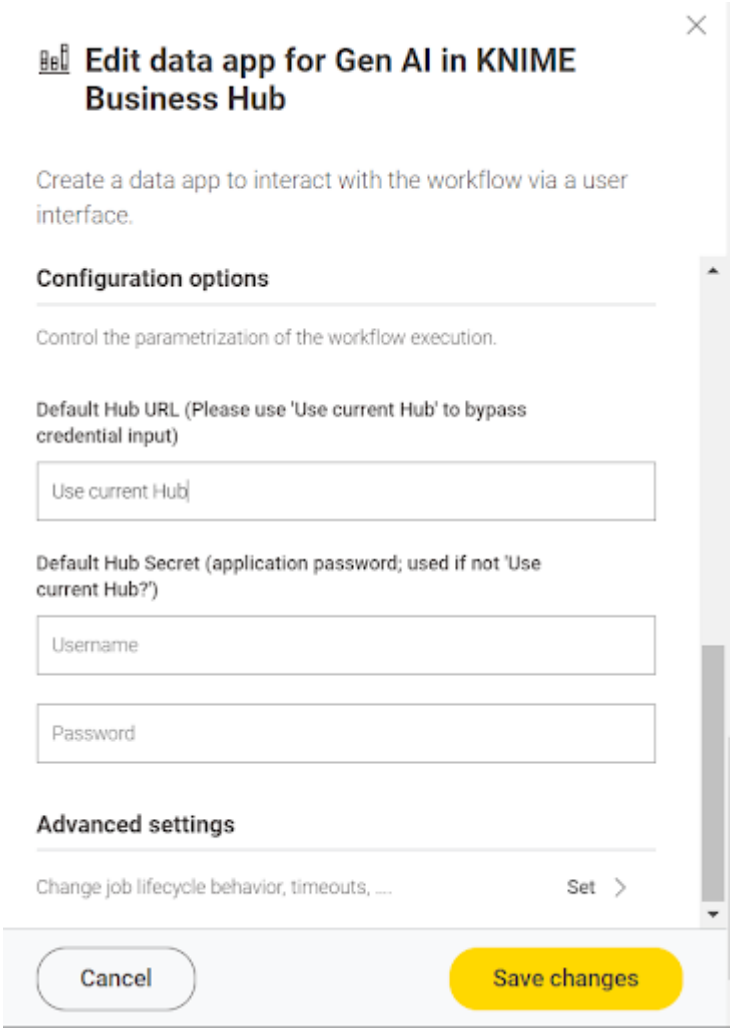


Figure 48. Deployment using Use current Hub to bypass credential input

Data app overview

Perspective selection

The data app is divided into four different perspectives to facilitate the answer to various questions:

- Users: Who is using K-AI?
  This perspective can answer user-related questions like "Are the new KNIME Business Hub users using K-AI?"
- Conversations: How are users using K-AI?
  This dashboard answers questions such as how the user prompts are distributed by K-AI mode (Q&A, Workflow builder, etc.).
- Cost: How much does it cost to use K-AI?
  The estimated cost associated with K-AI can be analyzed based on the tokens

consumed.

- All Dimensions: This perspective combines all attributes and metrics from three previous perspectives to create a more articulated analysis.

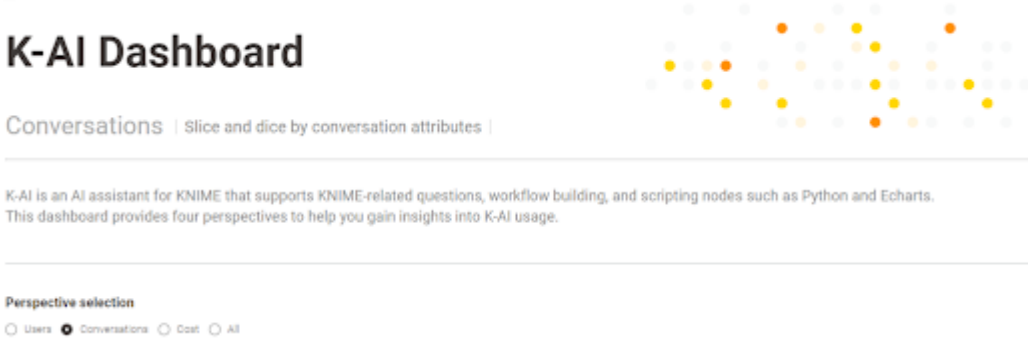


Figure 49. Perspective selection in the K-AI Dashboard

Slice and dice

There are different slicing filters and dice attributes for every perspective. The term *Slice* refers to filtering the data by different attributes. *Dice* refers to splitting chart metrics by the same attributes. The *Slice and Dice Legend* provides a comprehensive explanation of all of them.

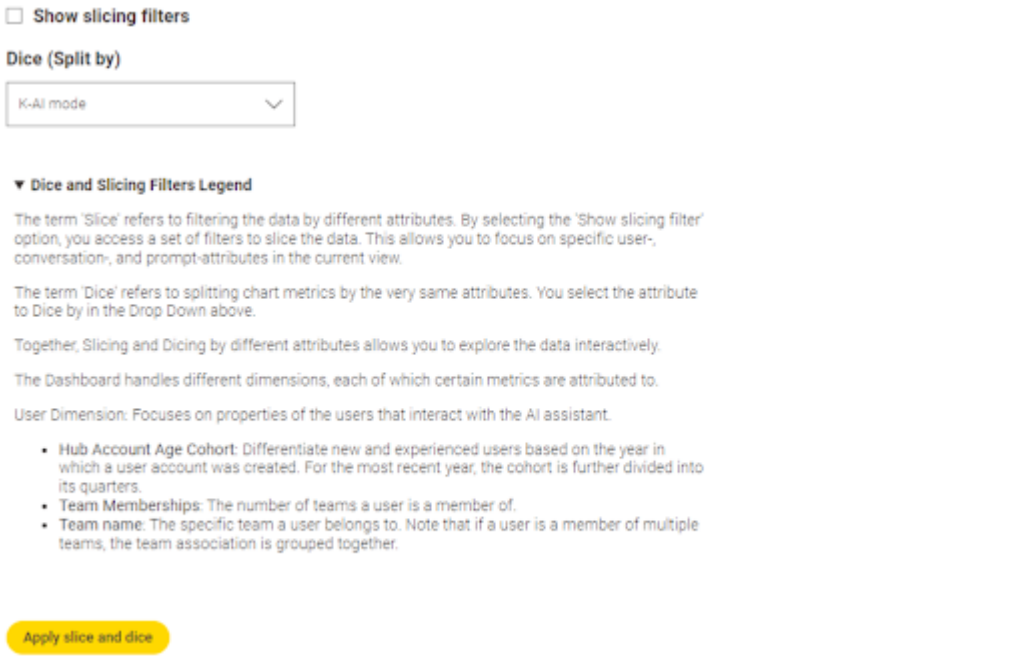


Figure 50. Slice and Dice Filter Legend in the K-AI Dashboard

Slicing filters

The slicing filters isolate data based on dimension attributes. For instance, from the *Conversation* perspective, you can slice the data only to analyze K-AI interactions related to

Python.

The user slicing filters will filter users and all their conversations, while the conversation slicing filters will filter specific conversations with all their prompts. Cost slicing filters only apply to the prompts.

Based on the Table 2, the user with ID "37373" is filtered out along with all its conversations if you slice by "Hub account age cohort," which equals 2024 in the *User* perspective. Conversation ID 1 is filtered out if you slice by "Conversation duration buckets" equals "10-20 min," and finally, if you slice by "Prompt and response token buckets" equals "1000-2000 tokens," only row 2 from the table above will be kept.

It is important to note that the slicing filters work sequentially from left to right. This means that if you select to view only K-AI interactions within Python nodes, the filtered output of the first slicing filter will be passed to the downstream filters. Please see Figure 51 for a visual representation.

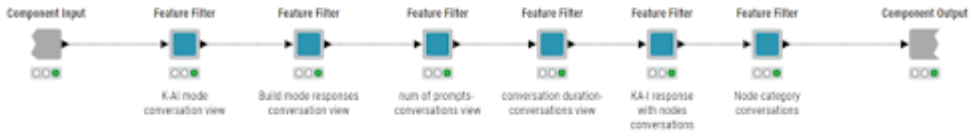


Figure 51. The slicing filters in the underlaying workflow

Dice

Analyzing K-AI usage metrics by different attributes can help you understand how data trends are formed over time. For instance, in the *Conversations* perspective, you can observe the number of user prompts over time, split by "K-AI Mode" or "Conversation duration (minutes)" attributes.

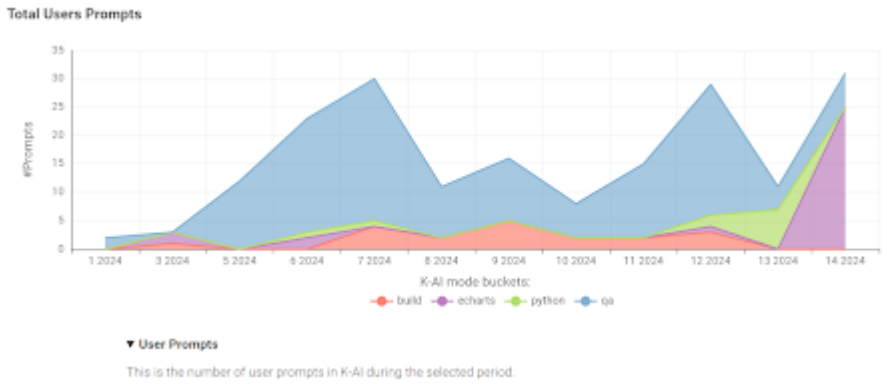


Figure 52. Total user prompts dice by K-AI mode

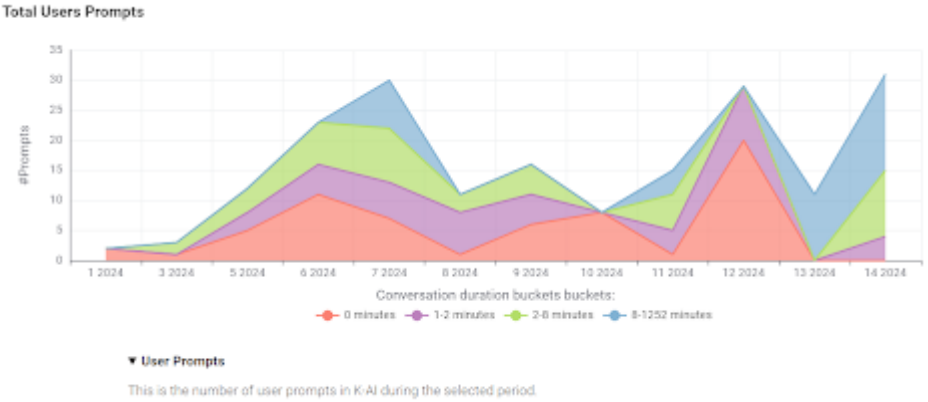


Figure 53. Total user prompts dice by Conversation duration

It is possible to select the attribute to *Dice* in the drop-down menu:

Figure 54. Dice drop-down menu

Data app usage

Please note that if you modify the time range, the data app will retrieve data from the KNIME Business Hub API endpoints again. Depending on the amount of data and the time range, it may take some time to re-execute.

Instead, the granularity feature only re-executes the charts within a different time granularity (from year to day).

When switching between perspectives,e.g., from Users to Conversations, the dice and slicing

filters are reset, and the charts are re-executed.

Customization

This workflow, is a blueprint you can customize to meet your needs. For instance, you can remove or reorganize some slicing filters, choose to have only the *Cost* perspective, eliminate other perspectives, or be inspired to create your attributes.

AI History Service

The aforementioned data app obtains its data via the AI History Service which provides access to historical user queries. You can query this service yourself. It offers the following HTTP GET endpoints:

- `https://api.<base-url>/ai-history/kai`
- `https://api.<base-url>/ai-history/code`
- `https://api.<base-url>/ai-history/code/[language]`

Endpoint details

Endpoint	Description
`/ai-history/kai`	For queries to the K-AI AI assistant.
`/ai-history/code`	For queries to AI code generation assistants in the corresponding KNIME nodes (Python Script, Generic ECharts View, Expression). This endpoint returns the history for all languages.
`/ai-history/code/[language]`	For queries to AI code generation assistants in the corresponding KNIME nodes (Python Script, Generic ECharts View, Expression). This endpoint returns the history for a specific language, where [language] can be 'python', 'echarts', or 'knime_expression'.

Optional query parameters

Parameter	Type	Description
`feedback`	Boolean	Include user feedback when set to `true`.
`start_time`	ISO 8601 datetime, e.g., `2024-07-15T15:53:00+05:00`	Start of the time range for history retrieval.
`end_time`	ISO 8601 datetime, e.g., `2024-07-15T15:53:00+05:00`	End of the time range for history retrieval.

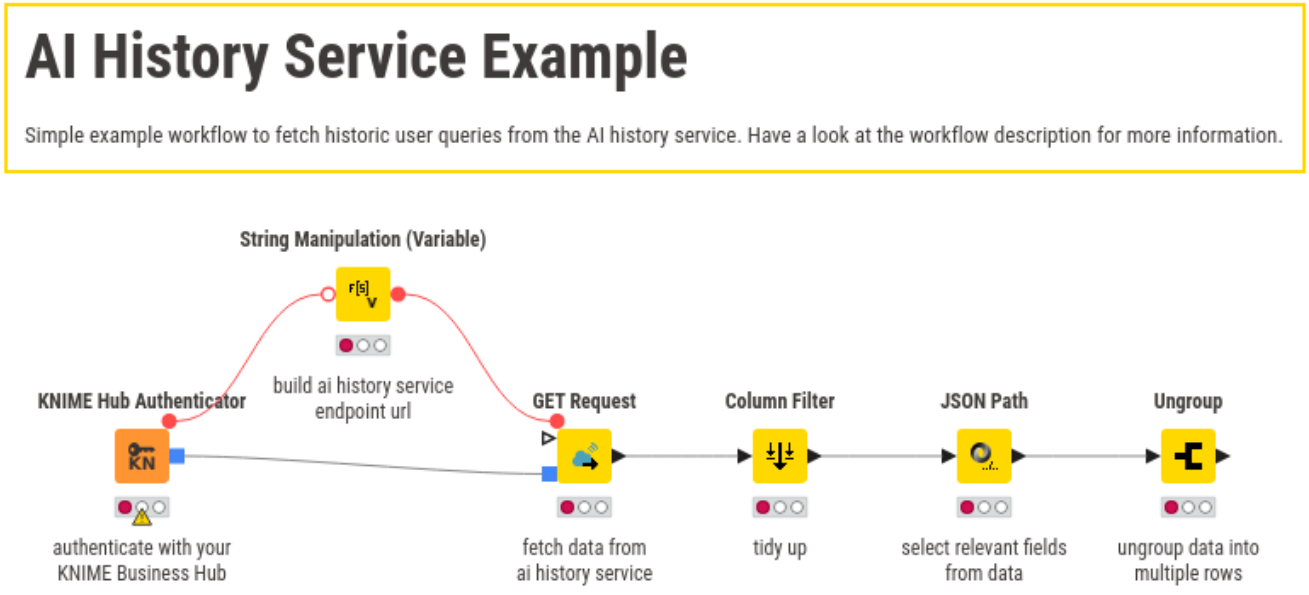
Example

```
https://api.<base-url>/ai-history/kai?feedback=true&start_time=2024-07-15T15:53:00+05:00
```

All endpoints require authentication and your user account must belong to an appropriate Keycloak group. For more information see here.

Fetching data via KNIME

On the KNIME Community Hub you can download a prototypical workflow that shows how to fetch this data:



To fetch the data with this workflow you will perform the following steps:

1. Authenticate to the Hub via the **KNIME Hub Authenticator** node
2. Fetch the data via the **Get Request** node
3. Select fields from the data via the **JSON Path** node
4. Ungroup the data into multiple rows via the **Ungroup** node

Workflows Jobs Monitoring

Introduction

Whenever an execution is requested on KNIME Business Hub, a corresponding job is created. Executions can originate from various sources such as ad-hoc executions, deployments, or through the Call Workflow node. The Hub's user interface provides access to explore these jobs from different touchpoints. For instance, you can navigate to the *Execution Resources* and *Deployment* pages within each team, or access the *Administration* panel if you are a global admin of the Hub.

However, this data app offers a deeper dive into job executions, providing insights such as:

- Status of workflow executions
- Usage of execution resources across different teams
- Trends in executions over time
- Identification of lengthy executions and faulty jobs

With this data app, you can gain a comprehensive understanding of your workflow executions and optimize processes accordingly.

This application accesses the jobs data collected when the Job Instrumentation Data service is activated for the KNIME Business Hub instance. This is managed from the KOTS Admin Console, as explained in the Activate and configure Job Instrumentation Data section.

Data app usage

The workflow is stored here and is compatible with KNIME Business Hub versions 1.10 and above. It can be executed either locally or directly on the Business Hub with executors versions 5.2.3 and above.

If you choose to run the workflow locally, you will need to manually input the Hub URL and the

<<Application Passwords in the String Configuration node.

However, if you opt to run it on the Hub, you can connect directly by selecting "Use current Hub" as the Hub URL, which is also set as the default value. Upon triggering the data app, it will initiate by downloading up to 1000 jobs metadata for each team the logged-in user is associated with.

We **strongly advise against** deploying this workflow with a global admin application password stored in the configuration nodes. This would grant all users running the data app access to all jobs or open the job and perform API calls with a global admin role, which leads to a potential security risk. Instead, the **recommended practice** is to deploy the data app with the default value set to "Use current Hub". By doing so, each user running the data app will only see the jobs they have permission to access.

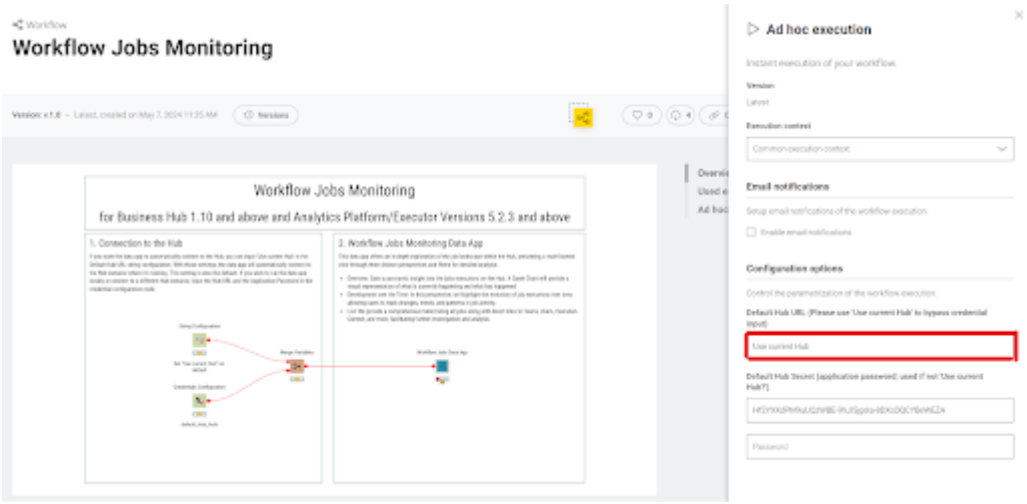


Figure 55. Configuring the Workflows Jobs Monitoring data app

Any user on the Hub can execute the data app. However, the displayed data of the fetched jobs varies significantly based on the user's role within their teams.

Job visibility permissions are structured as follows:

- Global admins have access to view all jobs across all teams on the Hub.
- Team admins are granted access to view all jobs within their respective teams.
- Team members can view all jobs they have triggered.

Once the data app is executed you will be able to see that there are three different sections:

- *Query Parameters*,
- *Filters*, and

- *Perspective*.

Query Parameters

This section enables you to adjust the number of jobs fetched for each team you are a member of. Additionally, it provides connection details such as user roles, the Hub instance you are connected to, the teams you are associated with, and a summary of your permissions.

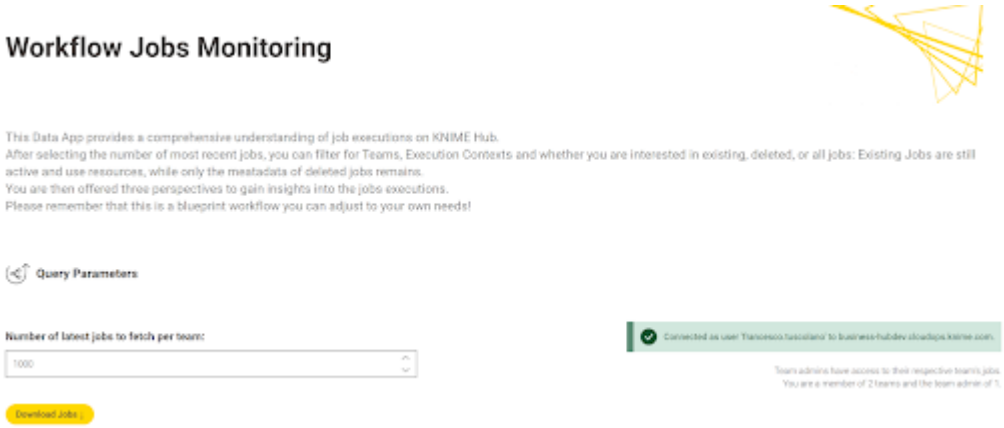


Figure 56. Query Parameters section of the Workflows Jobs Monitoring data app

Filters

This section empowers the data app with filters to slice the downloaded data, considering that users may not be interested in all teams and execution contexts they have access to. Note that the team filter is deactivated if the user can only access jobs from a single team. Additionally, users can choose to visualize all available jobs or only those that are deleted or existing. It is important to remember that deleted jobs are those that have been removed from the executor, with only metadata surviving. Existing jobs, on the other hand, can still be accessed and are retained within the Executor.

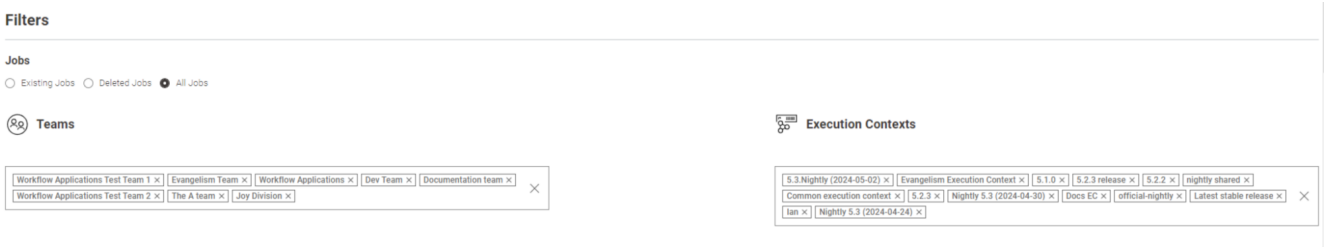


Figure 57. Filters section of the Workflows Jobs Monitoring data app

Perspective

This is the central component of the data app, offering insights into job executions. You have the flexibility to choose among three distinct perspectives, each providing a unique view of the data. While these sections use the same dataset, they are designed to shape and present the information in varied ways, ensuring a comprehensive understanding of the job executions.

Three distinct perspectives are available:

- Overview: Gain a panoramic insight into the jobs executions on the Hub. A Gantt Chart will provide a visual representation of what is currently happening and what has happened.
- Development over the Time: In this perspective, we highlight the evolution of job executions over time, allowing users to track changes, trends, and patterns in job activity.
- List: We provide a comprehensive table listing all jobs along with direct links to Teams, Users, Execution Context, and more, facilitating further investigation and analysis.

Overview

This section provides a concise overview and basic statistics concerning the executions. It includes counts of job states, the number and types of executions by team, the total number of fetched jobs or workflows, and other relevant metrics.

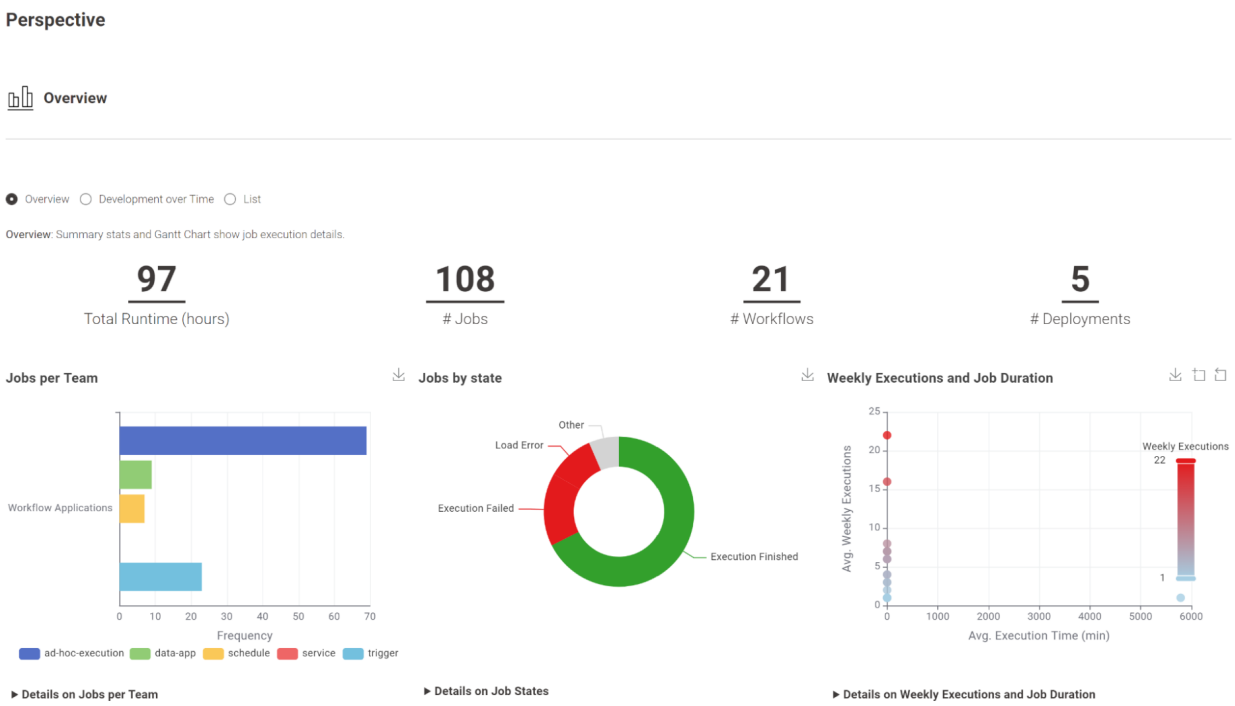


Figure 58. Overview section of the Workflows Jobs Monitoring data app

As you scroll down the Overview page, you will encounter a Gantt Chart visually depicting the executions on the current Hub instance. Here, the horizontal axis represents time intervals, while workflows are listed on the Y-axis. The length of each bar is proportional to the duration of the execution across the time axis. Hovering over the bars reveals additional information about the job. By default, the color of the bars corresponds to the job state, but you have the option to customize this setting and color the bars according to teams.

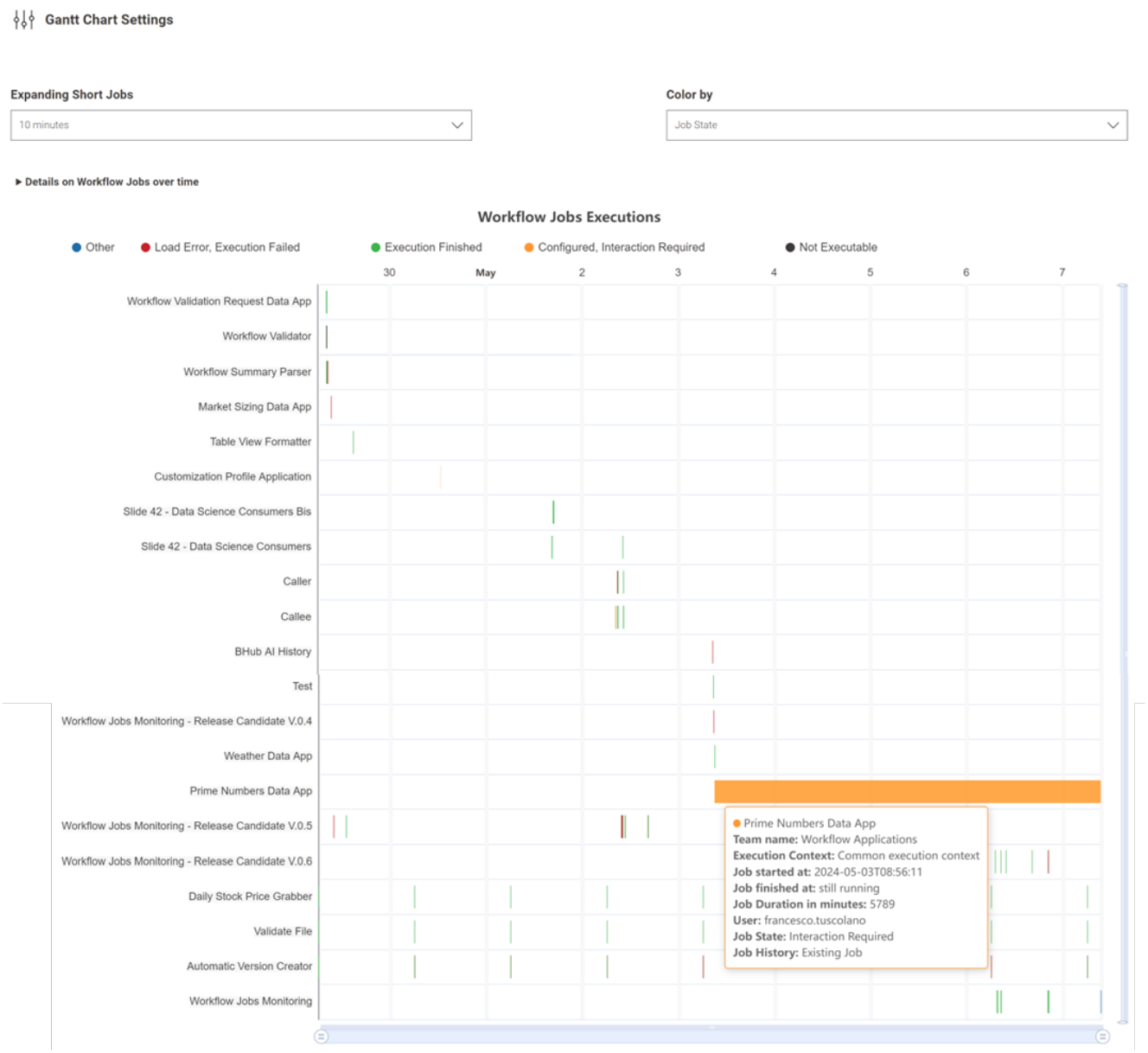


Figure 59. Gantt Chart Settings section of the Workflows Jobs Monitoring data app

Development over the time

This perspective dives deeper into job execution trends over time through a combination of tables and stacked area charts. It provides insights into execution counts, execution times, and highlights the workflows and deployments that are most frequently executed. Stacked area charts are an effective tool for identifying trends in execution. Moreover, we integrated a single selection widget that allows for the breakdown of the charts by three variables of

interest, such as Execution Contexts, Deployment types, and teams, facilitating the understanding of resource allocation dynamics. Additionally, the table views feature links that redirect users to the workflows' locations on the Hub, enhancing accessibility and navigation.

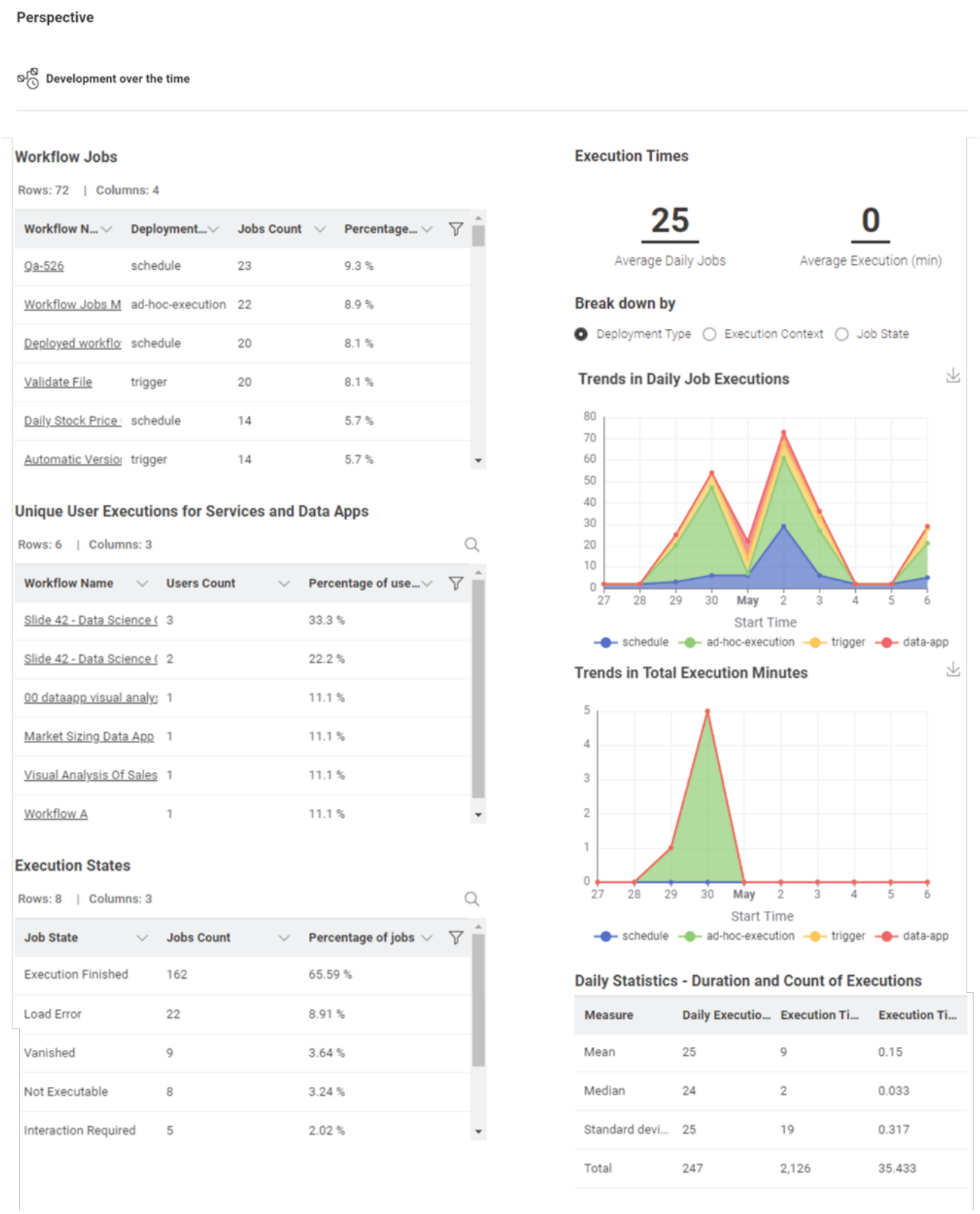


Figure 60. Development over the time section of the Workflows Jobs Monitoring data app

List

The final perspective presents the fetched job data in a well-organized table, with all available resources linked and jobs sorted from the most recent to the earliest. This comprehensive list of jobs allows users to inspect every detail for each job, including the workflow name, team, job state, deployment type, duration, and timestamp of the execution. Moreover, if the job is still existing and accessible in the executor, users can open it in the job viewer. Additionally, if the job is a data app in the "Interaction Required" state, users can reopen the data app and, if necessary, terminate the execution to free up memory.

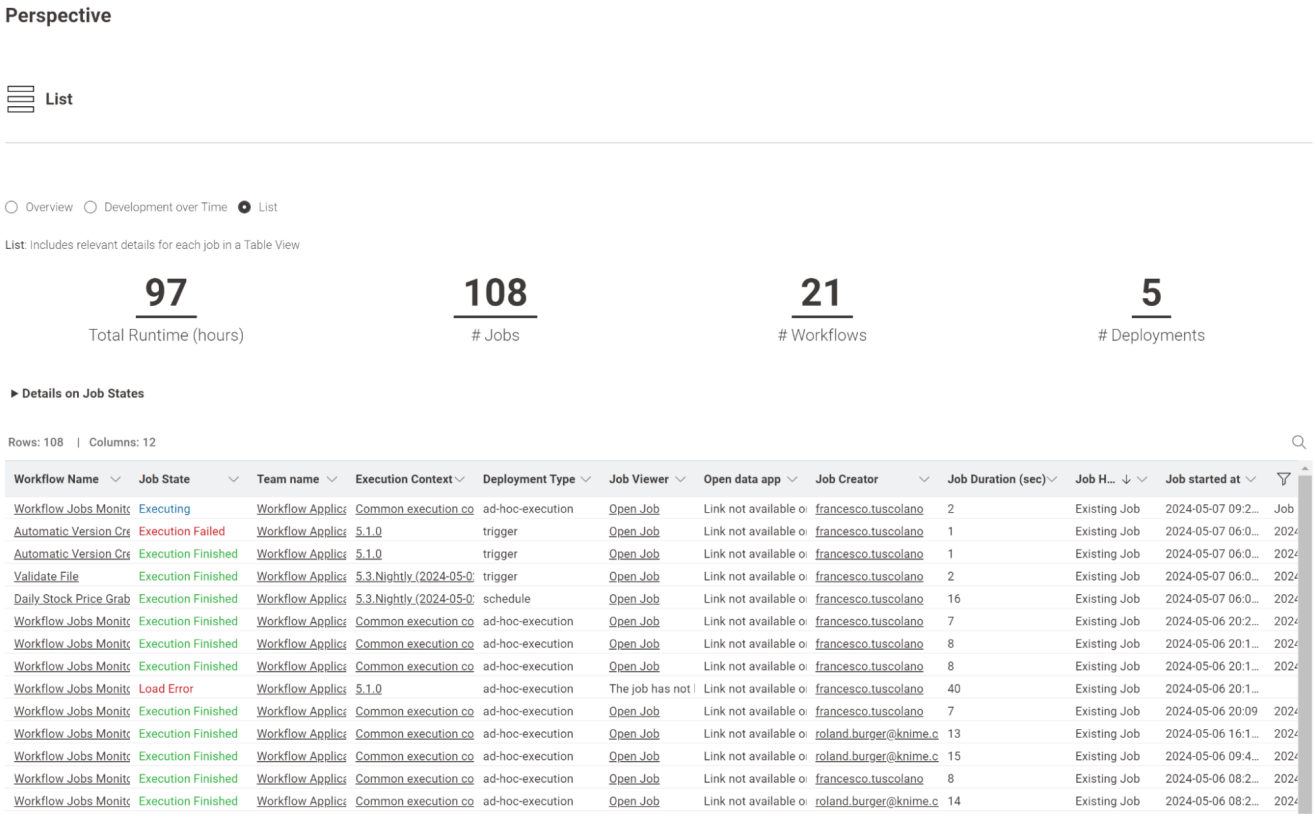


Figure 61. List section of the Workflows Jobs Monitoring data app

Business Hub metrics via Grafana Dashboards

As an administrator you have access to metrics from Kubernetes, KNIME Business Hub services, and tools like Keycloak and MinIO to understand KNIME Business Hub operations better and troubleshoot performance issues more effectively, via Grafana Dashboards.

In this section you will learn how to enable Prometheus to collect and store metrics and use Grafana Dashboards to consume the collected metrics.

- **Grafana** is used to visualize metrics collected from your Kubernetes cluster.
  For **embedded cluster installations**, Grafana comes preconfigured with dashboards tailored to Kubernetes components like nodes, pods, and services. These dashboards provide out-of-the-box visibility into cluster health, performance, and resource usage, simplifying the monitoring and debugging of KNIME Business Hub.
  For KNIME Business Hub, Grafana depends on Prometheus and the Prometheus Operator to be available in the cluster.
- **Prometheus** is a monitoring system that scrapes metrics from Kubernetes components and services. It uses Kubernetes service discovery to automatically find and monitor targets within the cluster, providing a robust and dynamic way to track performance metrics.
  The Prometheus Operator simplifies and automates the deployment, configuration, and management of Prometheus instances in a Kubernetes cluster. It provides Kubernetes-native custom resources such as Prometheus, ServiceMonitor, and PodMonitor for configuring Prometheus instances.
  For KNIME Business Hub, the Prometheus Operator operator manages one or more Prometheus instances which scrape and store metrics from various services in the cluster.

> For **existing cluster installations** additional steps are necessary to use monitoring via Prometheus Metrics and Grafana Dashboards, such as installing Prometheus and Grafana and deploying Prometheus Operator. Contact your designated (technical) account manager to know more about monitoring on an existing cluster.

Enabling Prometheus Metrics and Grafana Dashboards

In KOTS Admin Console, under the *Metrics* section, ensure that the *Enable Prometheus Metrics* and *Enable Grafana Dashboards* options are selected. Deploy the configuration changes if needed.

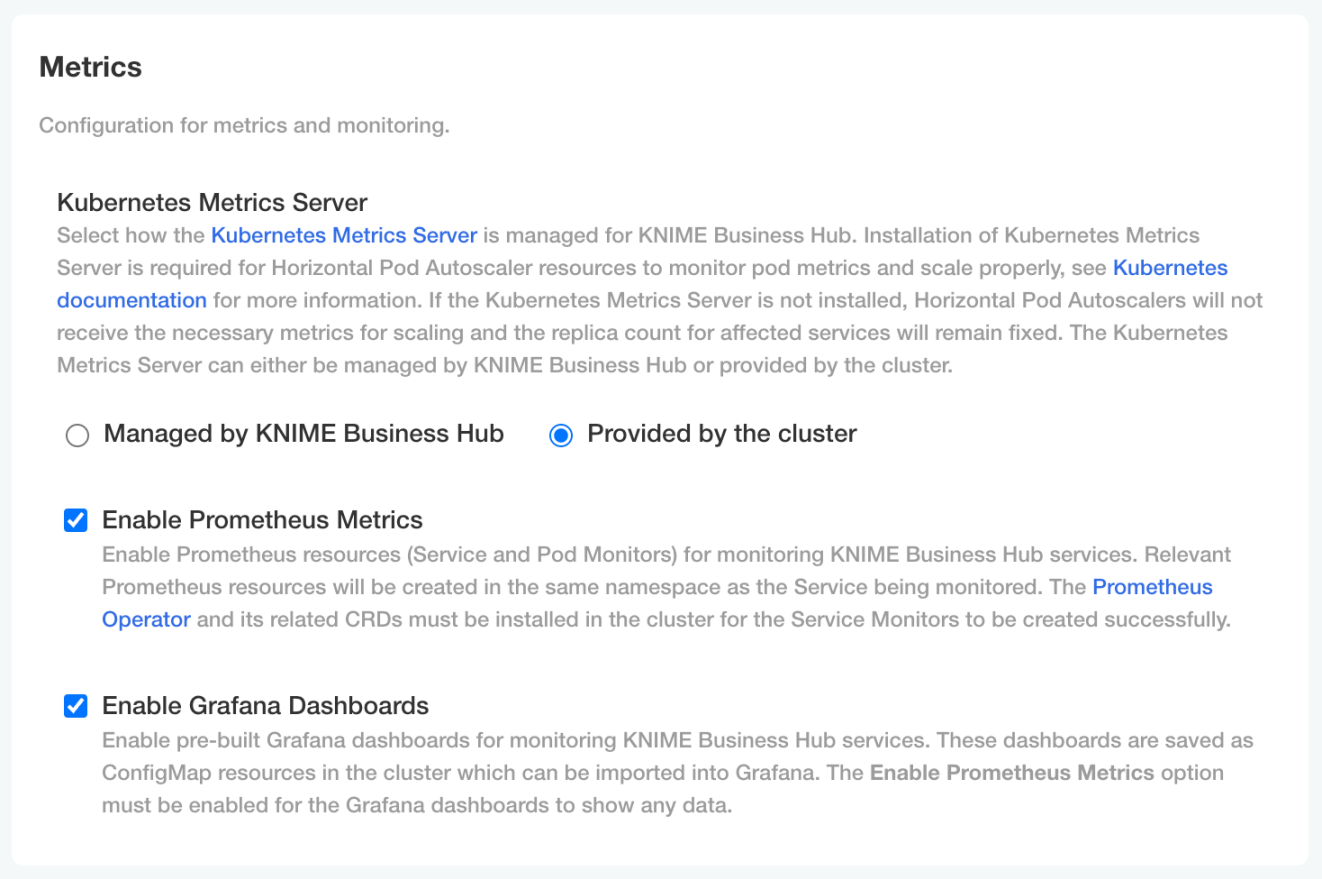


Figure 62. Enable metrics and dashboards in KOTS Admin Console.

Grafana

Accessing Grafana

To access Grafana follow these steps:

1. **Retrieve Grafana login credentials**: The Grafana login credentials are saved in a Kubernetes Secret (called `grafana-admin` for embedded cluster installations) within the `monitoring` namespace. The commands below can be used to retrieve and decrypt the admin username and password.


```
## Find the name of the Grafana admin credentials secret.
kubectl get secrets -A | grep grafana-admin

## Retrieve Grafana admin username.
## The secret name may be different depending on cluster type and install method.
kubectl get secret -n monitoring grafana-admin -o jsonpath="{.data.admin-user}" |
base64 -d

## Retrieve Grafana admin password.
## The secret name may be different depending on cluster type and install method.
## The output will likely show a % character appended to the end, which should be
omitted.
kubectl get secret -n monitoring grafana-admin -o jsonpath="{.data.admin-
password}" | base64 -d
```

2. **Access the Grafana Console**: In order to access Grafana, the `grafana` service needs to be port-forwarded to `localhost`. Below is an example command which port-forwards the `grafana` service to `localhost:9000`.

   ```
   ## Find the name of the Grafana service.
   kubectl get services -A | grep grafana

   ## Port-forward the Grafana service to localhost:9000.
   ## The service name may be different depending on cluster type and install method.
   kubectl -n monitoring port-forward service/grafana 9000:80
   ```

   After port-forwarding the `grafana` service, the web console should be accessible locally via `http://localhost:9000`. Enter the credentials that were retrieved from the `grafana-admin` secret and login.

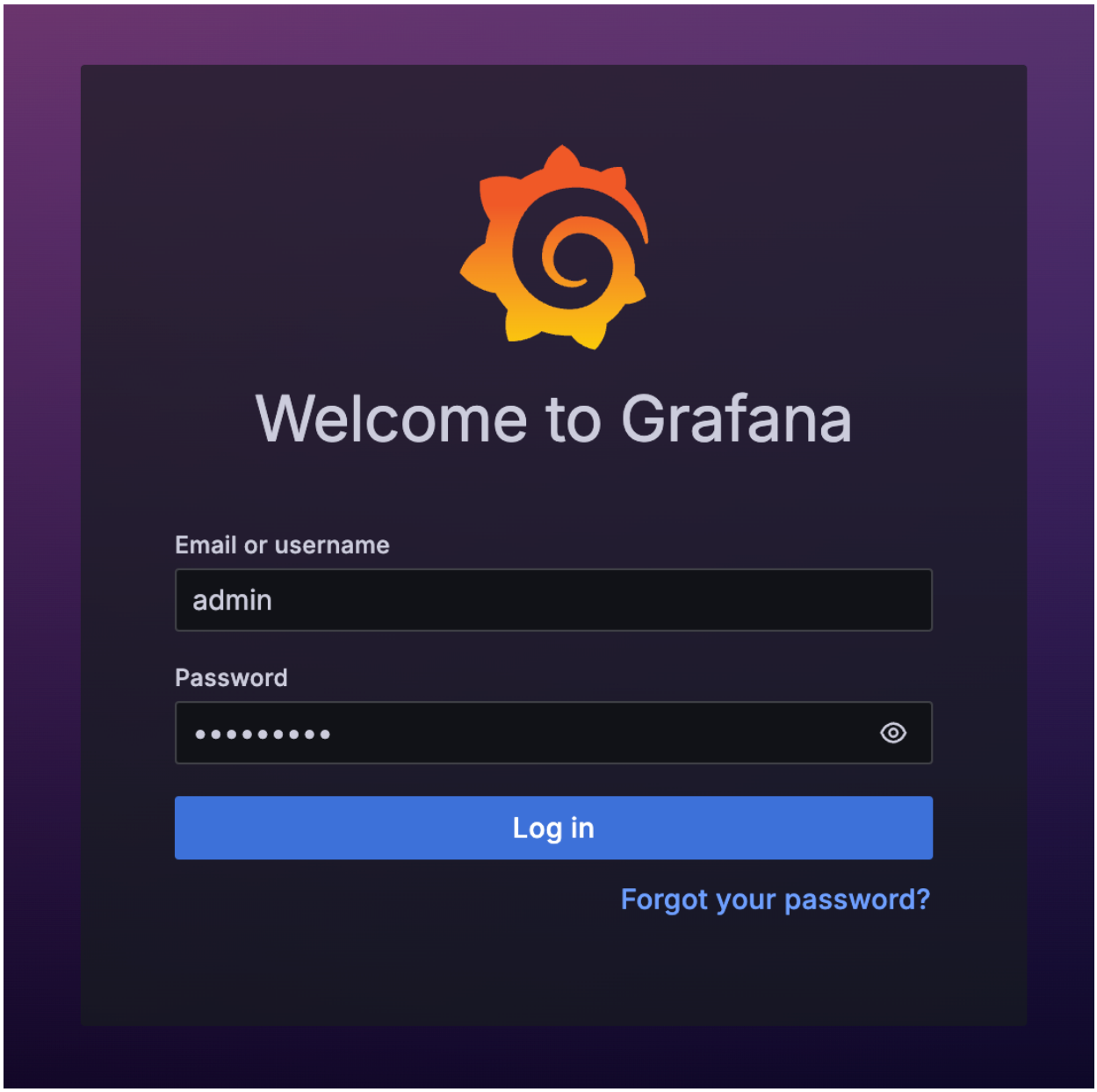


Figure 63. Log in to Grafana

Grafana web console usage

After following the above instructions, you should now have access to the Grafana web console. Navigate to *Dashboards* on the left-hand sidebar, and multiple dashboards should display. This collection includes a number of dashboards from KNIME Business Hub as well as some generic Kubernetes dashboards sourced from the `kube-prometheus-stack` Helm chart, which is installed by default as a plugin for KURL clusters.

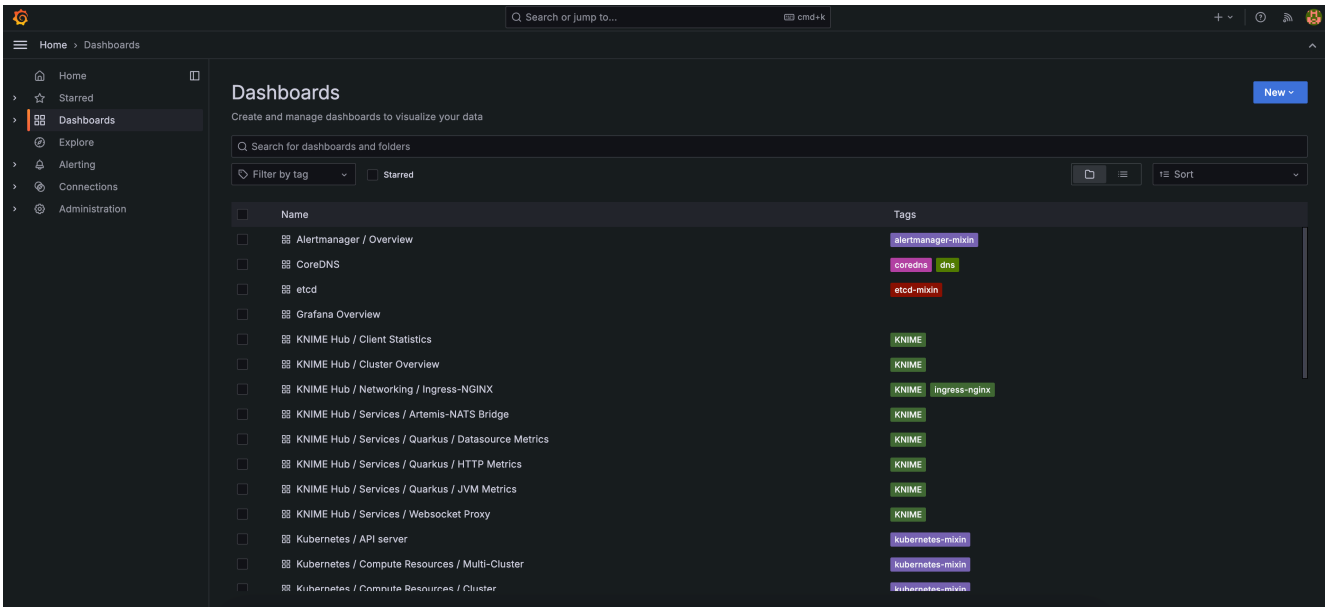


Figure 64. Grafana dashboards available in KNIME Business Hub

You can optionally use the tag filter control to filter the list of dashboards down to KNIME dashboards only.

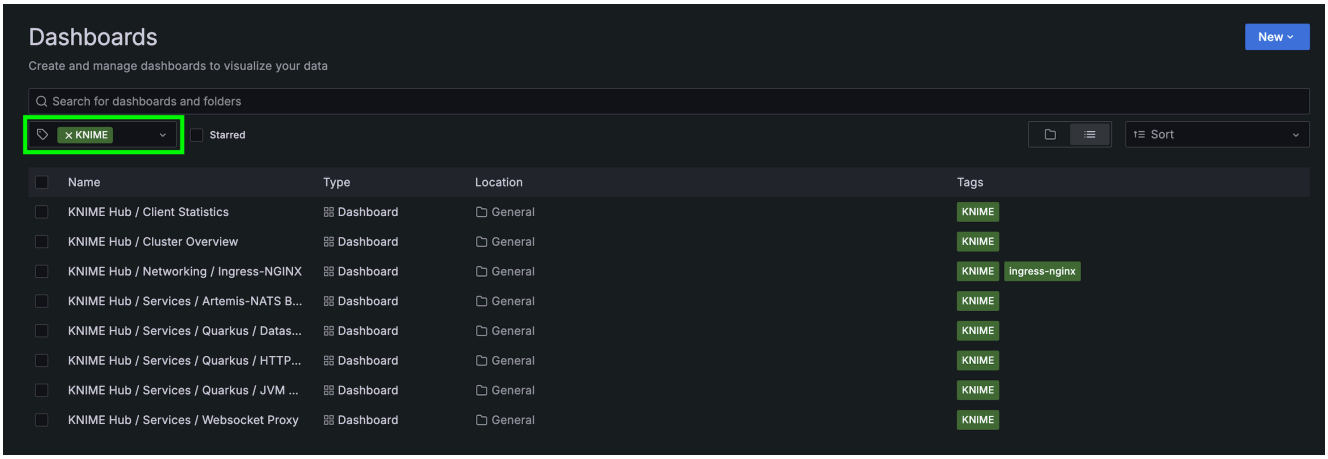


Figure 65. Grafana dashboards filtered by `KNIME` tag

You can then click on any given dashboard to view and interact with it.

See Grafana Docs: Dashboards for more information on using Grafana dashboards.

Each dashboard with the `KNIME` tag has a *Datasource* parameter which defaults to “Prometheus”. The *Datasource* parameter dynamically populates with any compatible Grafana Datasources (must be of type Prometheus). This allows pointing dashboards to a custom datasource with a different name than the default. For any custom datasources, they must be properly configured in order for data to display in the dashboards.

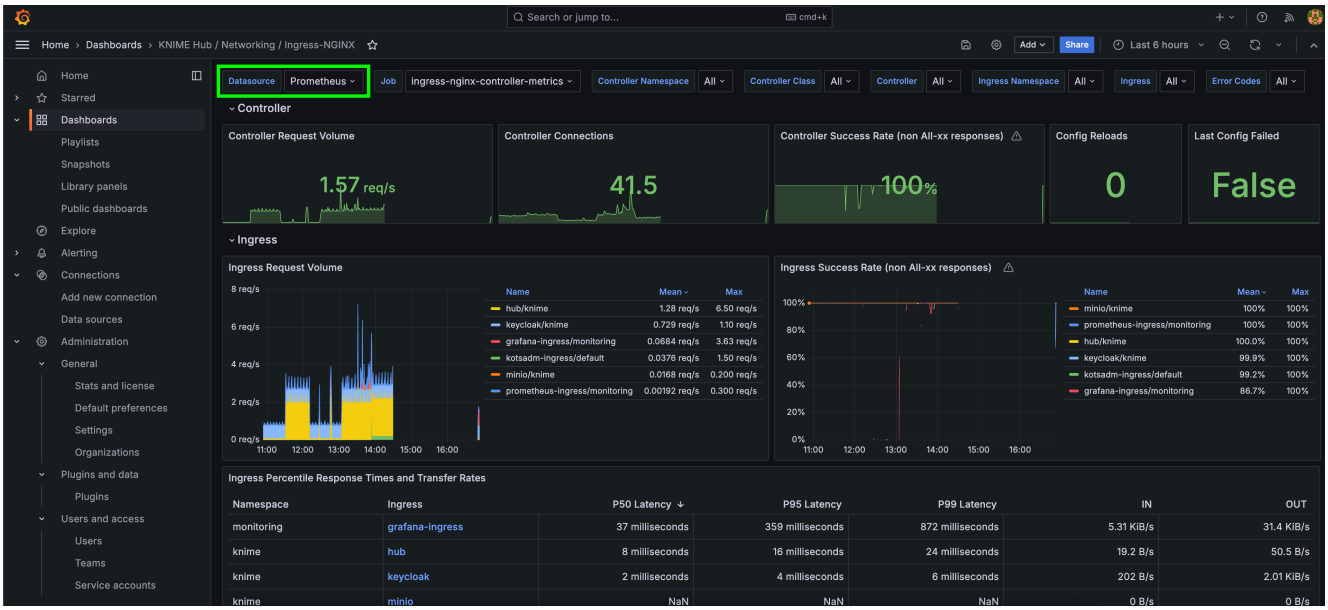


Figure 66. Example of a Dashboard and Datasource parameter

Troubleshooting

- *Dashboard shows no data*: If your dashboard shows no data, it is likely because the Enable Prometheus Metrics option has not been enabled (and deployed) via the KOTS Admin Console.

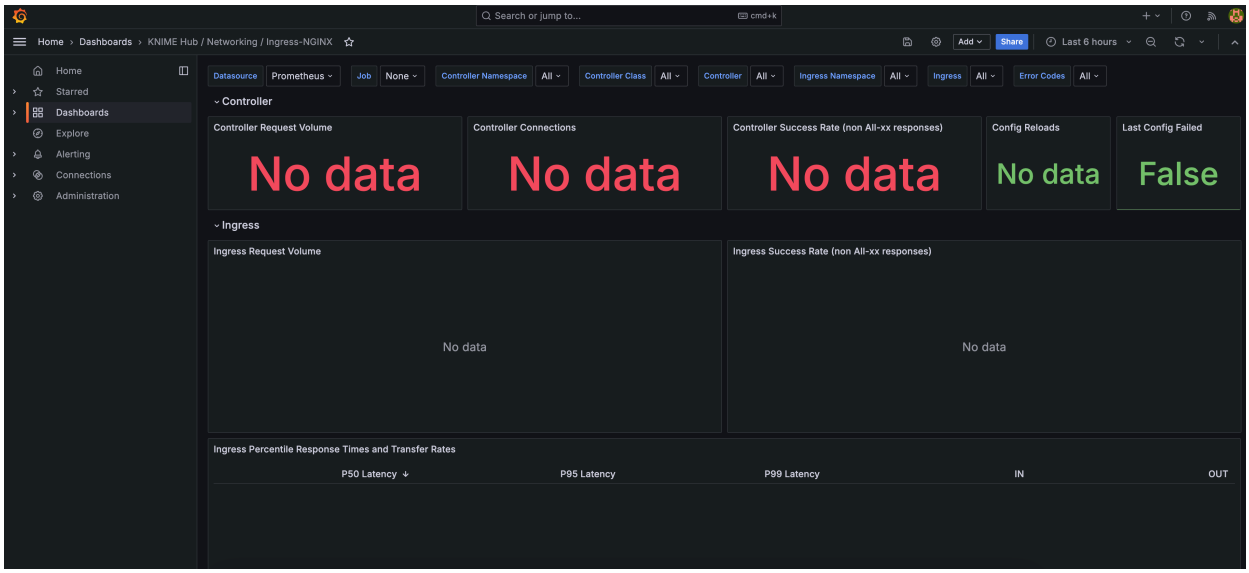


Figure 67. Dashboard does not show any data

- *Deployment errors are shown when Prometheus Metrics are enabled*: If you attempt to enable the Enable Prometheus Metrics and see deployment errors like the ones below, that indicates that the Custom Resource Definitions (CRDs) for the Prometheus Operator are not installed. For example, you might have an Existing Cluster installation of Business Hub.


```
------- ingress.nginx -------
Error: UPGRADE FAILED: resource mapping not found for name: "ingress-nginx-
controller"
namespace: "knime" from "": no matches for kind "ServiceMonitor" in version
"monitoring.coreos.com/v1"
ensure CRDs are installed first
```

How Grafana Dashboards are saved in KNIME Business Hub

Grafana Dashboards for KNIME Business Hub are saved as Kubernetes ConfigMap resources within the cluster (which contain the JSON configuration of the dashboard). This ensures that accidentally deleting a dashboard from the Grafana web console will not result in a permanent loss of the dashboard. It also ensures that changes cannot be saved directly to the dashboard - however, it is possible to create a copy of the dashboard and edit it freely.

```
## Search for all Grafana dashboard ConfigMap resources in the cluster.
## Most dashboards are prefixed with `grafana-dashboard`.
kubectl get configmaps -A | grep grafana
```

If you would like to create a custom dashboard for KNIME Business Hub, it is recommended to save it as a ConfigMap resource in the cluster to ensure it is persisted if the Grafana pod is restarted. See Grafana Docs: JSON model for more information on how to retrieve the JSON data for a custom dashboard created in the Grafana web console. Also refer to existing Grafana Dashboard ConfigMap resources as a reference, and note that the `grafana_dashboard="1"` label is required for Grafana to recognize your custom ConfigMap.

KNIME Business Hub API documentation

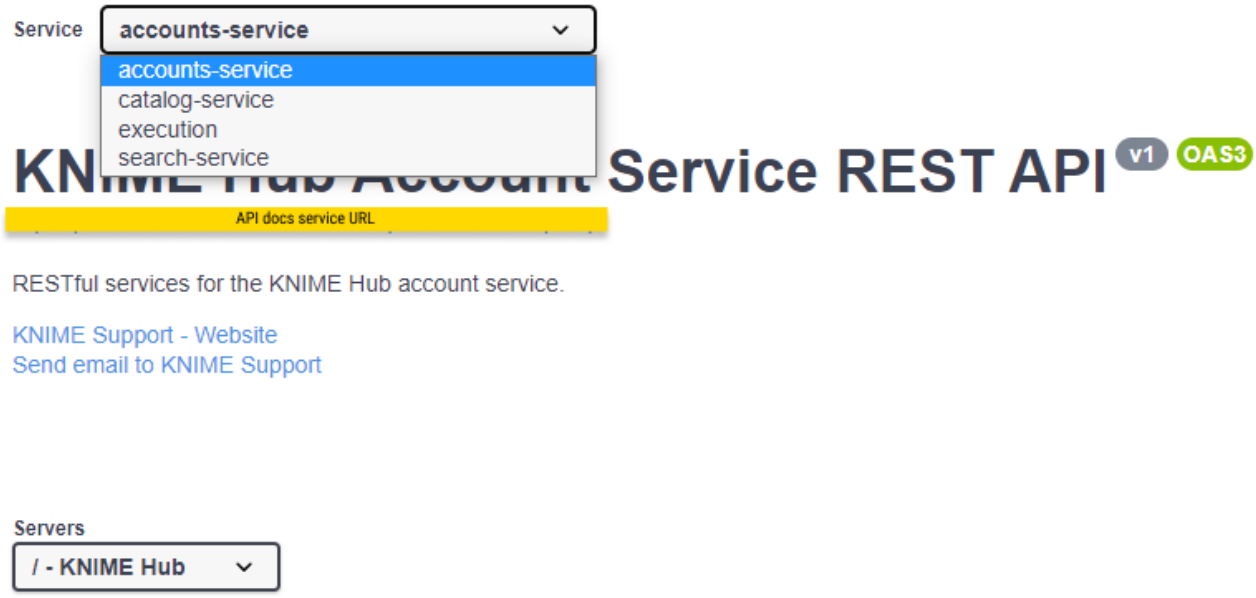
Most KNIME Business Hub functionalities are also available via REST API allowing you to perform several actions.

You can access the API documentation by navigating to the following URL:

```
api.<base-url>/api-doc
```

where `<base-url>` is your Business Hub instance URL, e.g. `hub.example.com`.

Here you can select from the drop-down menu the service you want to use.



Support Bundles and Troubleshooting

When generating a support bundle, ***no data leaves the cluster***.

If necessary, you can download the support bundle and send it to KNIME for the purpose of troubleshooting. Contact us by sending an email to support@knime.com. Under extreme circumstances, the KNIME team may forward the support bundle to the Replicated support team for additional help.

When generating a support bundle, a limited amount of information will be automatically redacted (IPv4 addresses, connection strings, etc.). You can configure additional redactions and/or manually redact information prior to sending the bundle. See the **Configuring redaction in support bundles** section for more details.

KNIME Business Hub is capable of generating support bundles in a standard format, even when the admin console isn't working. This ensures that users are able to provide all of the necessary information for KNIME to be able to identify the problem and prescribe a solution.

Generating a support bundle (GUI)

In order to help troubleshoot an installation, or to simply inspect the logs of the cluster in a user-friendly format, you will need to generate a support bundle.

Simply open the KOTS Admin Console, navigate to the **Troubleshoot** pane, and click the **Generate a support bundle** button to generate a support bundle.

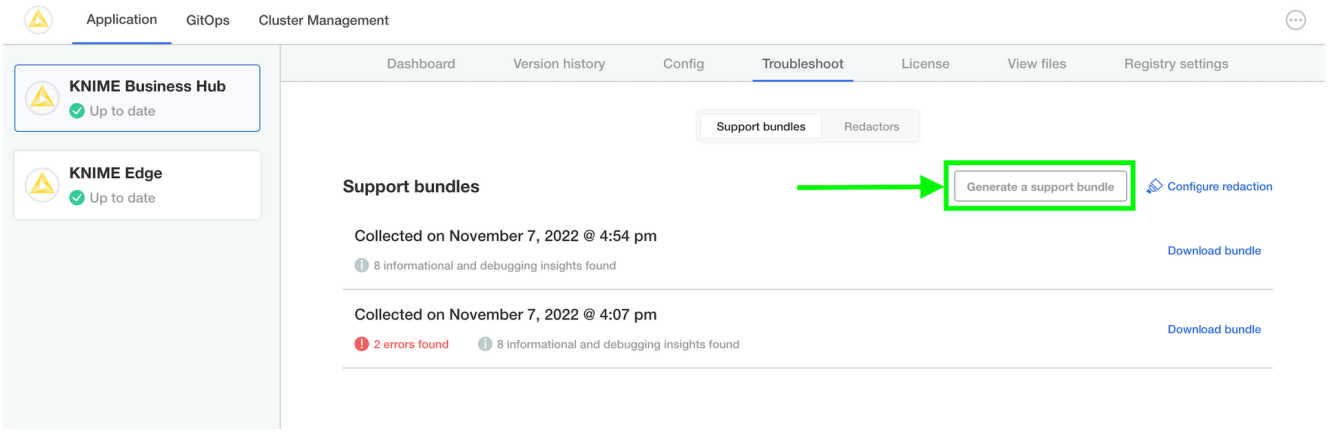


Figure 68. Generate a support bundle

All generated support bundles will display in the list above. Click the **Download bundle** button to download the bundle(s) you want to share with KNIME, and please see the **Configuring redaction in support bundles** section for information on how to redact confidential/personal information before sending.

Generating a support bundle (CLI)

See Replicated documentation for instructions on how to generate a support bundle via the Replicated CLI.

Configuring redaction in support bundles

When generating a support bundle, a limited amount of information will be automatically redacted (IPv4 addresses, connection strings, etc.) but it is not guaranteed to be a comprehensive set of redactions. You may have additional information in your logs or configuration that you do not wish to share with the KNIME engineering team.

One option is to unzip the generated .zip support bundle and manually review/redact information prior to sending the bundle to KNIME. However, there is a lot of information to review and the redaction of certain information can be automated fairly easily. The ideal option is to configure automated redactions via Redactor resources, which will automatically redact information for all future support bundles.

In order to configure automated redactors, first open the KOTS Admin Console. Navigate to the **Troubleshoot** pane and click **Configure Redaction**.

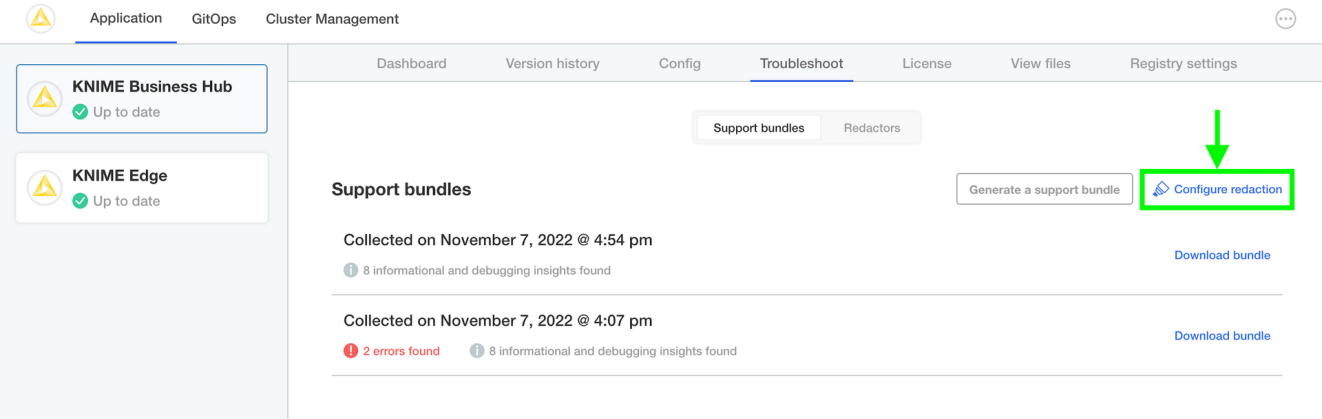


Figure 69. Configure Redaction

If you have configured your own custom redactions that you feel would be valuable to other users of KNIME Business Hub, please feel encouraged to share the configuration with KNIME so that it can be considered & potentially added to future releases.

See this link and this link for more information.

Inspecting support bundles

There are quite a number of of files generated in a support bundle. Not necessarily every file

is useful for every problem. However, by collecting the same information in the same way each time, KNIME can ensure the best quality support possible for customers.

It is possible to inspect a support bundle entirely in the admin console. See below for an example screenshot.

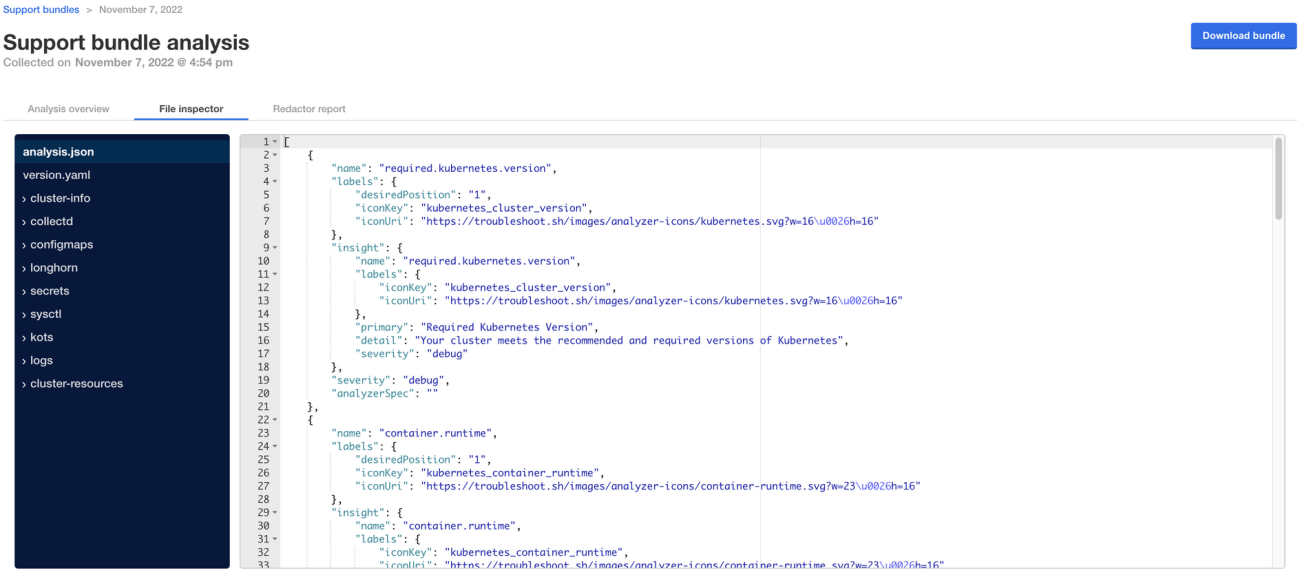


Figure 70. Inspect a support bundle in the admin console

Here are the most important folders/files and their purposes:

Path	Purpose	Example (may have properties omitted)

`./analysis.json`	• Collects the highest-level insights possible for the installation.
	• Often times, the issue and/or resolution may be identified in this file by inspecting the `[].name.insight.detail` property.
		<pre>[
		  {
		    "name":
		"kotsadm.status",
		    "insight": {
		      "name":
		"kotsadm.status",
		      "primary": "kotsadm
		Status",
		      "detail": "At least
		1 replica of the Admin
		Console API is running
		and ready",
		      "severity": "debug"
		    },
		    "severity": "debug",
		    "analyzerSpec": ""
		  }
		]</pre>

`./logs`	• Contains logs of individual pods. • Execution Context logs are stored in `./logs/execution-contexts`.	(typical application logs)

Path	Purpose	Example (may have properties omitted)
`./cluster-resources`	• Contains the configuration of each visible resource in the cluster. • For example, to see all pods in the cluster, navigate to the `./cluster-resources/pods` directory which contains one file per namespace in the cluster.	`{ "kind": "PodList", "apiVersion": "v1", "metadata": { "resourceVersion": "1686941" }, "items": [ ... ] }`

Maintenance operations

Restart a node

You might need to reboot a node if you are performing maintenance on the operating system level of the node, e.g. after a kernel update, rebooting the node will apply the changes.

Before rebooting a node on a cluster managed by kURL please call the shutdown script on the node:

```
/opt/ekco/shutdown.sh
```

See more documentation on rebooting nodes here.

Otherwise, after a VM restart old pods might be in `Failed` or `Shutdown` state.

In case that happens, delete the failed pods after the restart with the following command:

```
kubectl delete pod --field-selector=status.phase==Failed --all-namespaces
```

Backup and restore with Velero Snapshots and Kotsadm

Snapshot backups and restore features are available into Replicated deployments via Velero, a tool for backing up Kubernetes cluster resources and persistent volumes.

One-time snapshots as well as automated scheduled snapshots can be managed from the *Snapshots* panel within your Kotsadm dashboard at https://<base-url>:8800/app/knime-hub.

Snapshot creation and restoration are disruptive processes. KNIME applications, and Replicated admin access may be unavailable during an active backup or restore operation.

Creating snapshot backups

1. First, configure storage for your backups. Navigate to the *Snapshots* tab of your Kotsadm dashboard. Click the 'Settings' button to edit backup settings where you'll be able to add a storage target.

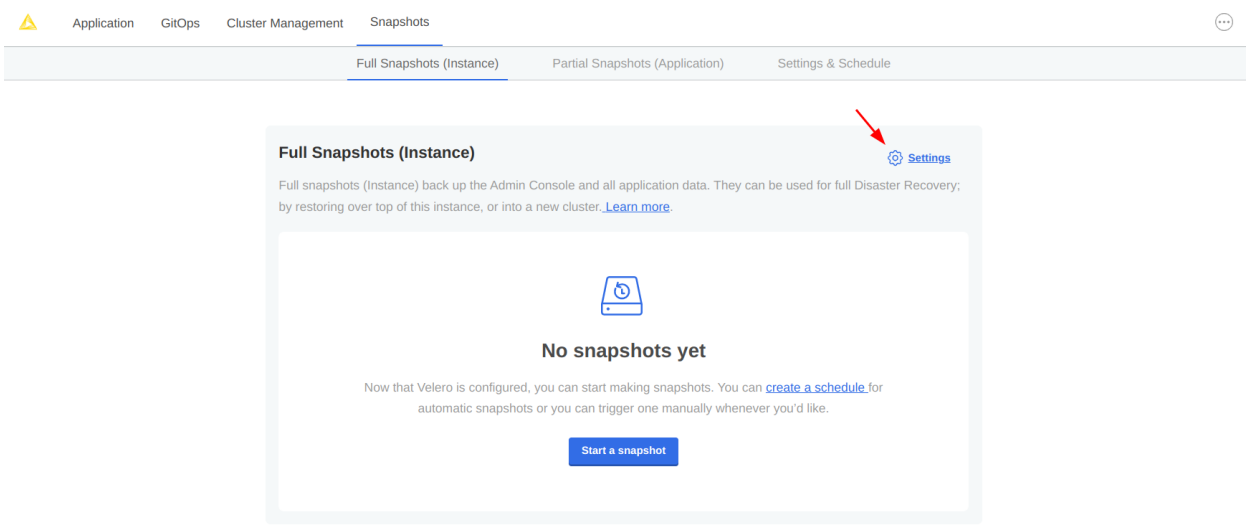


Figure 71. Snapshots tab with settings link

2. Velero supports local storage (not recommended), Amazon S3, Azure Blob Store, Google Cloud Storage, and S3 API compatible storage endpoints such as MinIO. Select your preferred snapshot storage type from the 'Destination' drop-down menu, and fill in the required fields with parameters specific to your storage endpoint. Click the 'Update storage settings' button and wait for Velero to verify backup storage access.

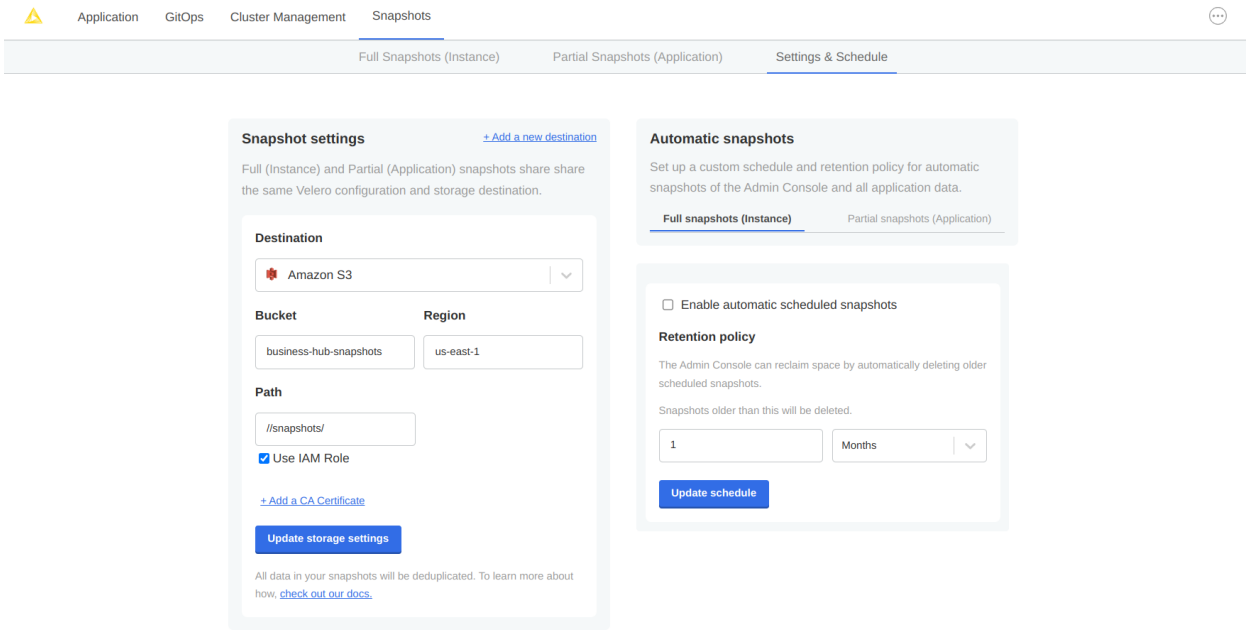


Figure 72. Snapshots destination settings for AWS S3 storage

3. With a valid backup storage configured, you can create a Snapshot of your KNIME deployment by clicking the *Full Snapshots* tab, and then the *Start a snapshot* button. This may take a few minutes to complete.
4. Once your snapshot is complete, from the same *Full Snapshots* screen, you can click the 'Settings' button to manage snapshot retention, or configure automatic snapshots by checking the *Enable automatic scheduled snapshots* box and setting a schedule using a CRON expression.

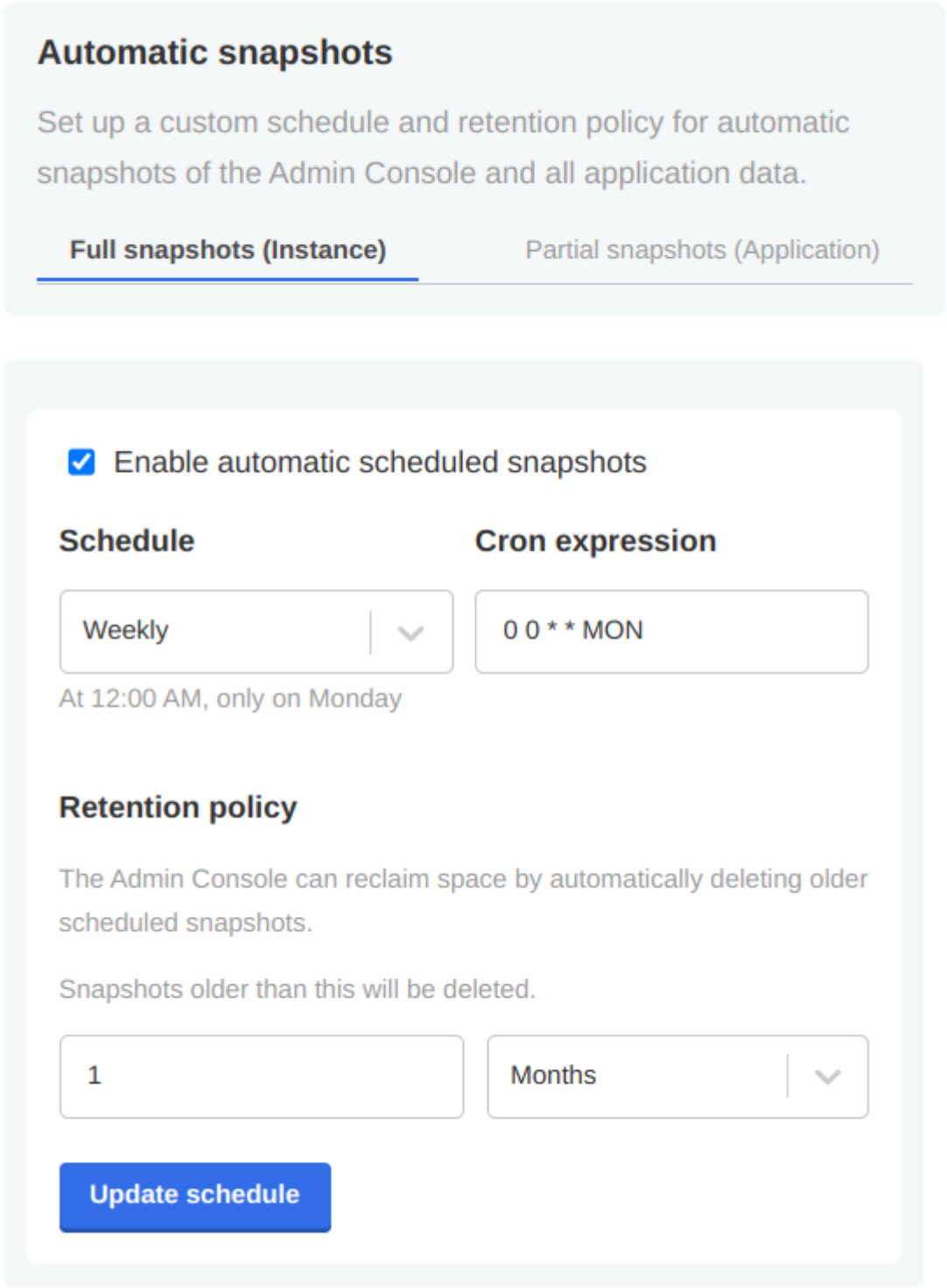


Figure 73. Example automatic snapshot scheduled to run at 12:00am weekly with a 1 month retention policy.

Backup Troubleshooting

Velero is installed into the embedded `Kurl` Kubernetes cluster with default settings and resource allocations.

As the number of objects or overall size of data to be backed up increases, it may eventually occur that the CPU and memory resources allocated for Velero processes are no longer sufficient to successfully complete the backup.

In the event that backup failures are encountered, it is recommended to increase the CPU and memory allocation **directly** to the Velero's node agent process via `kubectl`.

```
$ kubectl patch daemonset node-agent -n velero --patch \
'{"spec":{"template":{"spec":{"containers":[{"name": "node-agent", "resources":
{"limits":{"cpu": "2", "memory": "2048Mi"}, "requests": {"cpu": "1", "memory":
"512Mi"}}}]}}}}'
```

The CPU and memory resources and limit values can be adjusted as needed to find sufficient values for backup process. Typically, only the **limit** values will need to be increased.

At this time, the resource allocation override to Velero will **revert** after a `Kurl` upgrade has been performed. Please ensure any changes to the Velero node agent are reapplied after any `Kurl` cluster-level upgrades.

Restoring a snapshot

1. Navigate to the list of available snapshot restore points from your Kotsadm dashboard by browsing to *Snapshots* → *Full Snapshots*. From this screen, identify the snapshot instance you would like to use, and take note of the instance ID.

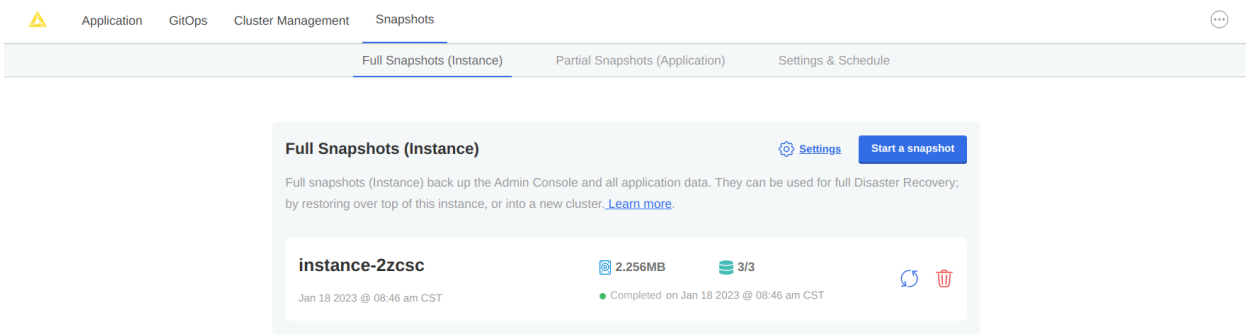


Figure 74. In this example, there is only one snapshot available and its ID is `instance-2xcsc`

A list of snapshots can also be retrieved by command line:

```
$ kubectl kots get backups
NAME              STATUS       ERRORS    WARNINGS    STARTED
COMPLETED                        EXPIRES
instance-2zcsc    Completed    0         0           2023-01-18 14:46:26 +0000 UTC
2023-01-18 14:46:53 +0000 UTC    29d
```

2. Now, restore the snapshot using a single CLI command.

```
$ kubectl kots restore --from-backup {Snapshot ID}
```

3. Next, redeploy the Hub instance from the KOTS Admin Console and trigger the restart of all executors by performing any change to each execution context in the Hub UI, e.g. decreasing/increasing the memory.
4. Finally, assuming the restore completed without errors, you can verify your Hub installation is functioning as expected.

Changelog (KNIME Business Hub 1.12)

You can find the changelog and release notes for older version of KNIME Business Hub releases in the KNIME Business Hub Release Notes document.

KNIME Business Hub 1.12.4

(released March 28, 2025)

Security fix

This release resolves a security vulnerability in all prior versions of KNIME Business Hub.

On March 24, 2025, a high-severity vulnerability in the widely used ingress-nginx component for Kubernetes was publicly disclosed. By sending specially crafted HTTP requests from within the cluster to the ingress-nginx controller, attackers could achieve remote code execution. Since ingress-nginx holds access to all cluster credentials, this vulnerability could lead to a full cluster takeover.

For full details, refer to the KNIME Security Advisory: CVE-2025-2787.

KNIME Business Hub 1.12.3

(released March 20, 2025)

Security fix

This release addresses a security vulnerability affecting all previous versions of KNIME Business Hub. We've identified that the existing Kubernetes secret configuration could potentially allow parties with specific credential knowledge to interact with job-related data on accessible KNIME Business Hub installations.

For more details, see the KNIME Security Advisory: CVE-2025-2402.

KNIME Business Hub 1.12.2

(released November 26, 2024)

Important changes (please read carefully)

- KNIME Business Hub 1.12.2 addresses default memory requests across multiple services. This helps lowering the minimum hardware requirements, so that Business Hub can again be deployed on machines with only 32GB of memory (while allowing an executor with at most 12GB RAM). Customers with larger machines do not need to go to this version.
- This release introduces the ability to configure hardware resource requests for the additional services in the KOTS admin console. At the same time, default memory requests have been reduced for these services:
- Search-sync service: 2560MB → 1024MB
- Artemis: 2048MB → 1024MB
- Elasticsearch: 2048MB → 1024MB
- Memory requests for istio-proxy sidecar containers have been reduced from 128MB to 64MB per container
- This setting will only be applied after relevant pods have been restarted via `kubectl rollout restart deployment -n hub`. Doing so is not strictly necessary, but will save around 2.5GB of memory combined

KNIME Business Hub 1.12.1

(released September 11, 2024)

Important changes (please read carefully)

This release fixes wrong default values for CPU requests for various services. These values were introduced in KNIME Business Hub 1.12.0, and led to resource issues on smaller clusters.

For instance, the default memory requests for core services have been raised to 24GB, leaving only 8GB for an executor on a 32GB machine. This will be made configurable in an upcoming release in order to allow increasing execution resources again.

KNIME Business Hub 1.12.0

(released September 9, 2024)

Important installation notes

During the update executors of the Hub will restart. Some downtime in execution is expected.

Important changes (please read carefully)

- The Secret Store feature is now also available for Business Hub Basic Edition licenses (See documentation)

Improvements

- Scalability options for selected Hub services: IT now has the flexibility to configure resource usage and set limits for scaling with parameters such as minimum/maximum replicas, target CPU utilization, and memory resources for various Hub services like Account, Catalog, Execution Rest Interface, Search, and Websocket Proxy (See documentation)
- New monitoring dashboards: Administrators now have access to metrics from Kubernetes, KNIME Business Hub services, and tools like Keycloak and Minio to understand KNIME Business Hub operations better and troubleshoot performance issues more effectively (See documentation)
- Trigger deployments: Workflows can now be executed by events that occur across multiple teams and spaces, including those in newly created spaces. Global administrators can even set workflows to execute based on events across all spaces in the organization. This update is particularly helpful to administrators who want to deploy team or organization-wide governance workflows (See documentation)

Important Bugfixes

- To avoid missing image problems coming up in airgapped installations. Garbage collection in the internal image registry shipped with Hub is now disabled by default.

KNIME AG
Talacker 50
8001 Zurich, Switzerland
www.knime.com
info@knime.com