LDAP integration for the private automation hub server

Facts:

- Users in the IdM server can't be used directly for authentication in the private automation hub server.
- The private automation hub server can use the central authentication solution.

Restrictions:

Central authentication only works in standalone mode, it can't be used for clustered deployments, so it's not recommended for production deployments. The current support statement taken from https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html/installing_and_configuring_central_authentication_for_the_ansible_automation_platform/assembly-central-auth-hub is:

IMPORTANT

The installer in this guide will install central authentication for a basic standalone deployment. Standalone mode only runs one central authentication server instance, and thus will not be usable for clustered deployments. Standalone mode can be useful to test drive and play with the features of {CentralAuth}, but it is not recommended that you use standalone mode in production as you will only have a single point of failure.

To install central authentication in a different deployment mode, please see this guide for more deployment options.

Requirements:

- Other server that runs Java, with Java 8 JDK, `zip` or `gzip` and `tar` commands.
- 512 MB of RAM.
- 1 GB of disk space.

Installation

The installer is the same for the automation controller and private automation hub servers.

1. Download the installer to the workstation machine and extract the contents.

2. Edit the `inventory` file.
    a. Delete the line for the automation controller, and specify the fully qualified domain name of the private automation hub, its database and the central authentication server.

Section of the inventory	**FQDN**
`[automationcontroller]`	
`[automationhub]`	`hub.lab.example.com`
`[db]`	`db.lab.example.com`
`[sso]`	`hub2.lab.example.com`

    b. Set the value for the following variables related to private automation hub and its database:

Variable	**Value**
`automationhub_admin_password`	`redhat`
`automationhub_pg_host`	`db.lab.example.com`
`automationhub_pg_port`	`5432`
`automationhub_pg_password`	`redhat`

    c. Set the sso related variables:

Variable	**Value**
`sso_keystore_password`	`redhat`
`sso_console_admin_password`	`redhat`

3. When modified, the uncommented content of the inventory file displays as follows

```
[automationcontroller]
[automationcontroller:vars]
peers=execution_nodes
[execution_nodes]
```

```
[automationhub]
hub.lab.example.com
[database]
db.lab.example.com
[servicescatalog_workers]
[sso]
hub2.lab.example.com
[all:vars]
admin_password=''
pg_host=''
pg_port=''
pg_database='awx'
pg_username='awx'
pg_password=''
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL
registry_url='registry.redhat.io'
registry_username=''
registry_password=''
receptor_listener_port=27199
automationhub_admin_password='redhat'
automationhub_pg_host='db.lab.example.com'
automationhub_pg_port='5432'
automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password='redhat'
automationhub_pg_sslmode='prefer'
sso_keystore_password='redhat'
sso_console_admin_password='redhat'
```

4. As the `root` user, run the `setup.sh` installation script.

```
]# ./setup.sh -e ignore_preflight_errors=true
```

The `ansible-automation-platform-2.1-for-rhel-8-x86_64.rpms` repo is not required for the sso server, but it's required a subscription with the following repos:

- `jb-eap-7.3-for-rhel-8-x86_64.rpms`
- `rh-sso-7.4-for-rhel-8-x86_64.rpms`

The installation takes ~8 min.

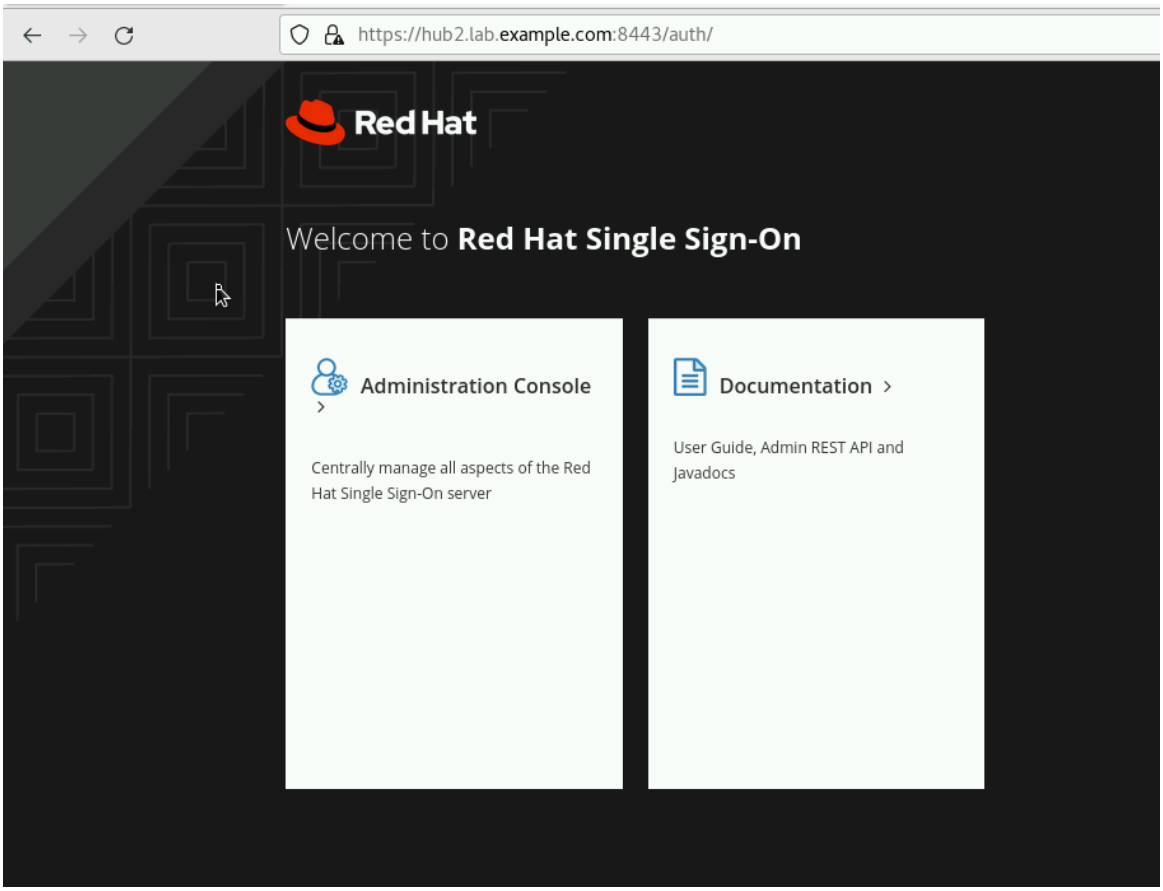
IdM

For testing purposes I have previously created the following users/groups in the IdM server:

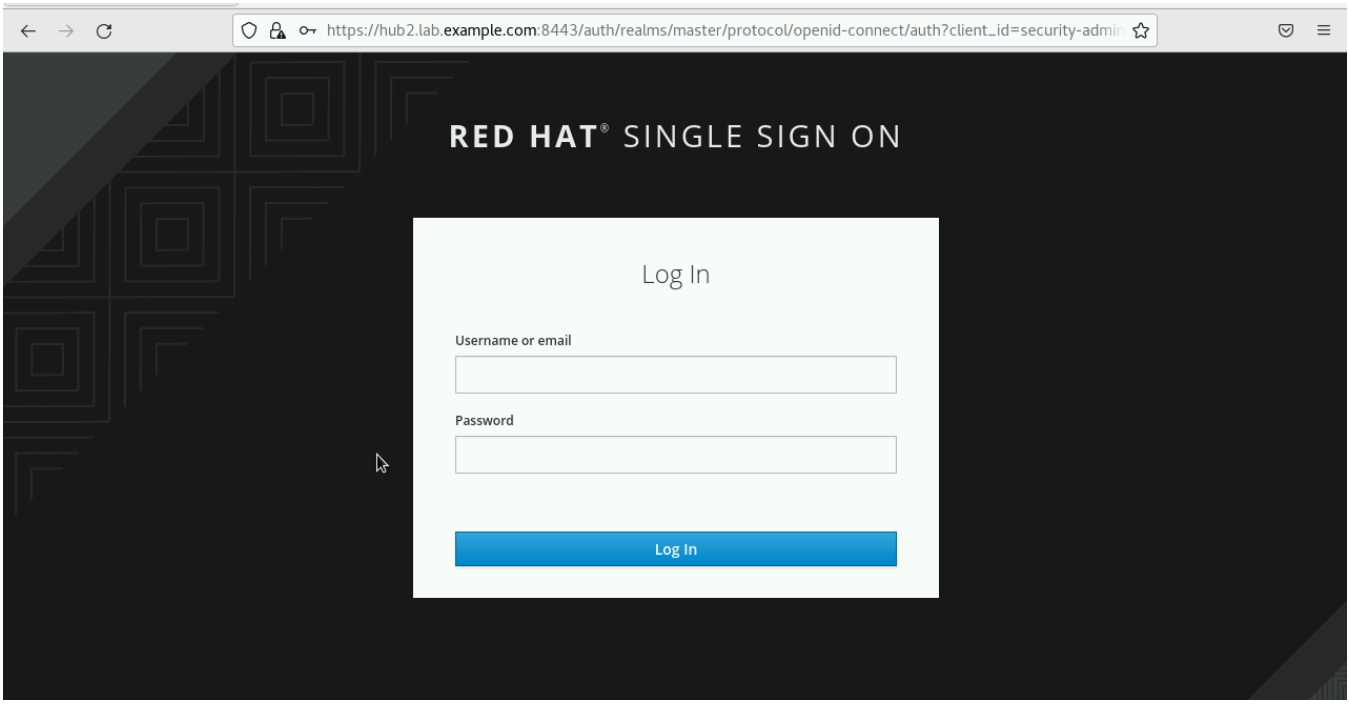
User login	First name	Last name	Group
hubadmin	Automation Hub	Administrator	hubadministrators
containerdev1	Container1	Developer	container_devs
containerdev2	Container2	Developer	
collectionop1	Collection1	Operator	collections_ops
collectionop2	Collection2	Operator	

Central authentication

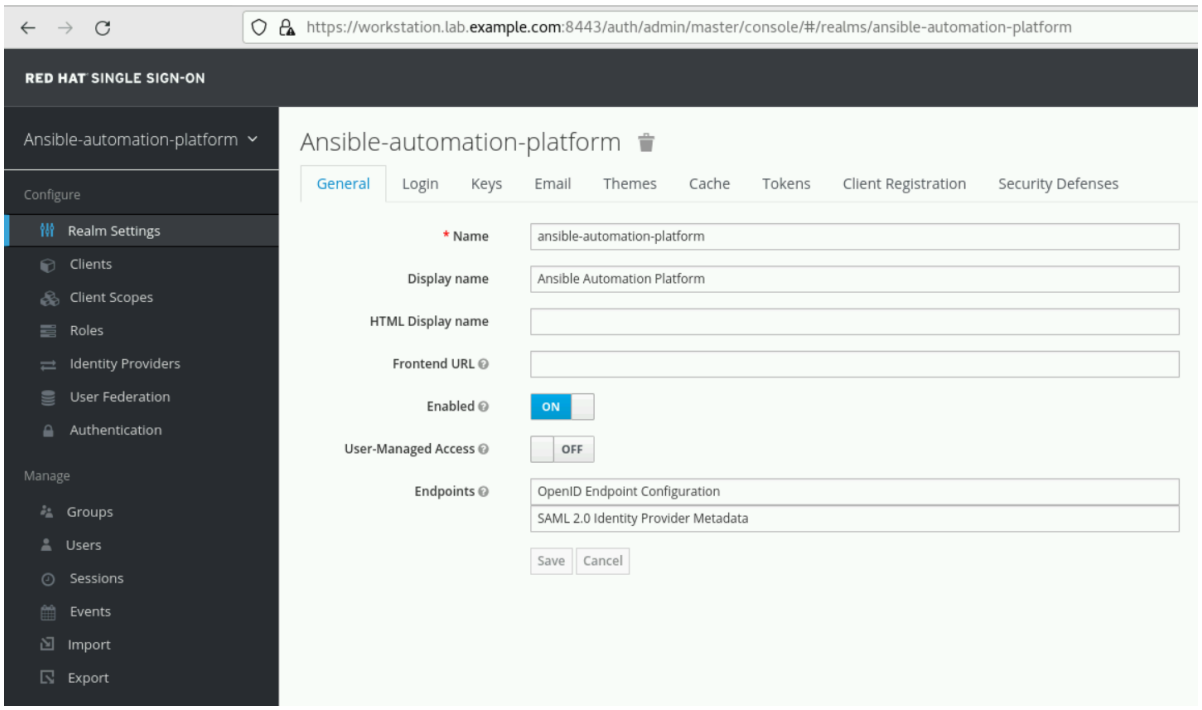
1. After the installation finishes successfully, connect to https://hub2.lab.example.com:8443 (the central authentication web UI). The web browser generates a warning message regarding to the self-signed certificate for hub2, accept the risk and continue.



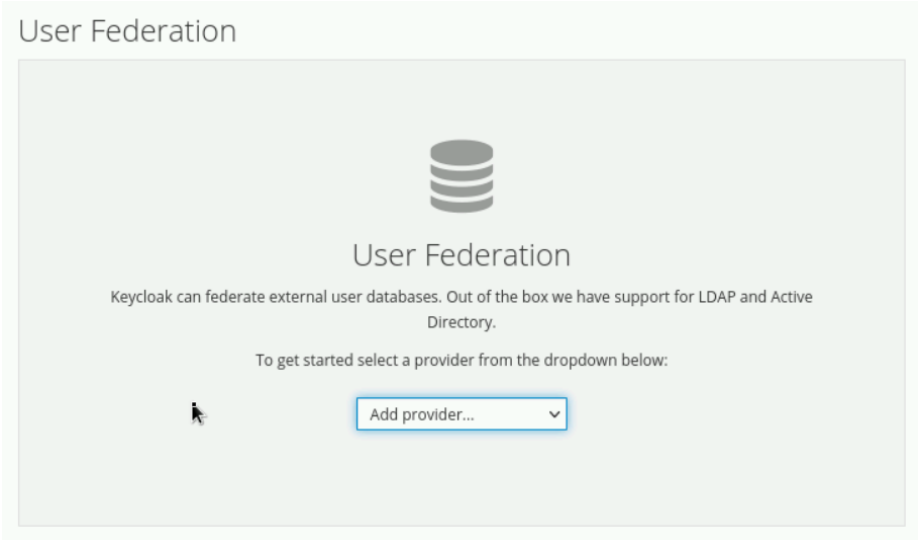
2. Click **Administration Console**, then the following screen is displayed:



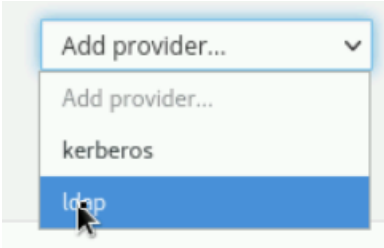
3. Log in the web UI as `admin` using `redhat` as the password.



4. Click **Configure > User Federation** then display the menu **Add provider**.



5. Select **ldap**.



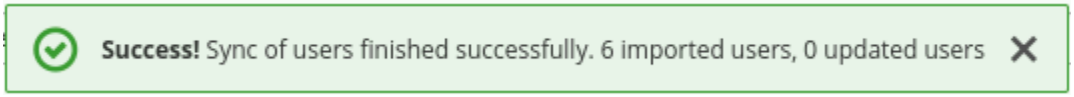
6. There are several values that can be set. Change the values for the following settings and those that are not in the following table, leave them as they are:

Required setting	Value
Console Display Name	`ldap aap`
Edit Mode	`WRITABLE`
Sync Registrations	`ON`
Vendor	`Red Hat Directory Server`
Connection URL	`ldaps://utility.lab.example.com:636`
Users DN	`cn=users,cn=accounts,dc=lab,dc=example,dc=com`
Bind DN	`uid=admin,cn=users,cn=accounts,dc=lab,dc=example,dc=com`
Bind Credential	`redhat321`

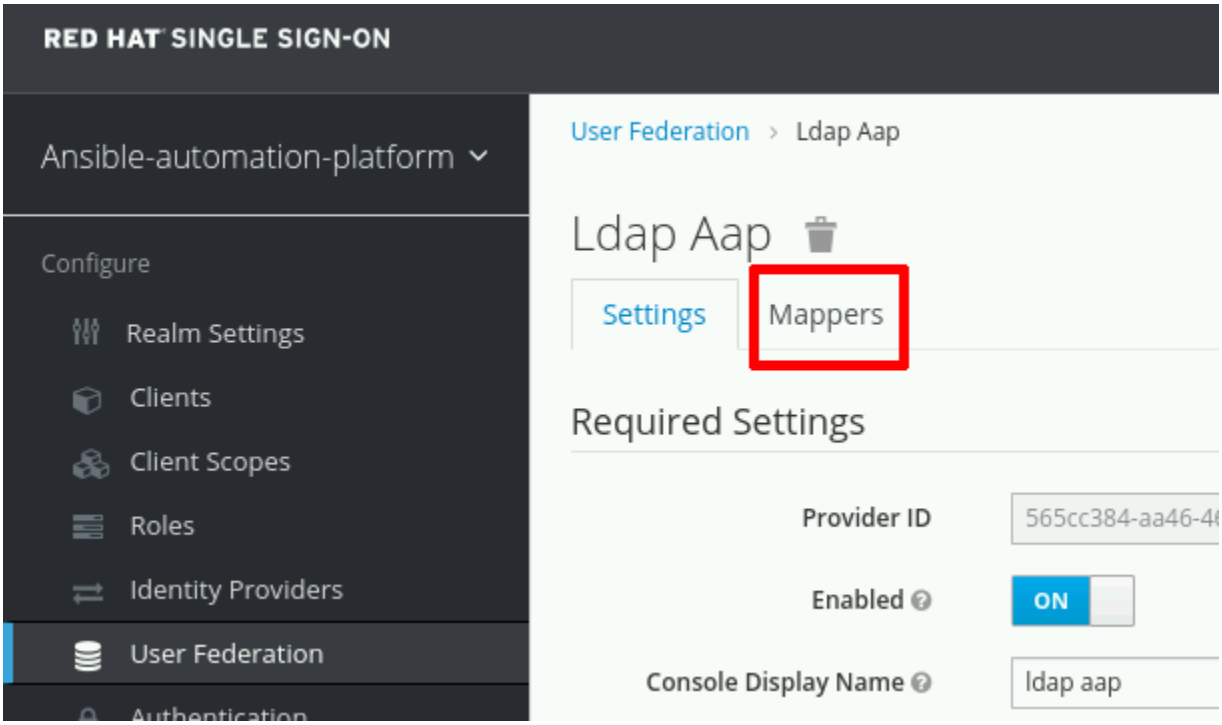
7. Optional: Click **Test connection** and **Test authentication** to validate the connection url and the bind credentials.

8. Click **Save**. New buttons appear at the button of the page and a tab called **Mappers**.

9. Click on the new button **Synchronize all users**.



10. Click in the **Mappers** tab:



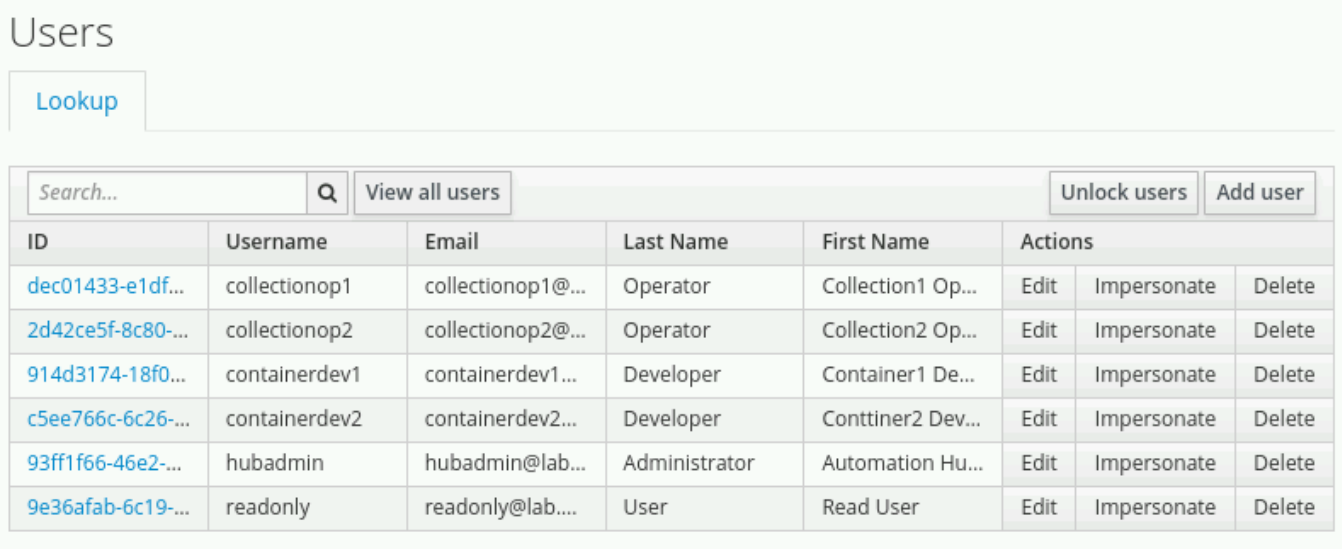
11. Click **Create** to add a new user federation mapper. Fill as follow:

Required setting	Value
Name	`IdM Groups`
Mapper Type	`group-ldap-mapper`
LDAP Groups DN	`cn=groups,cn=accounts,dc=lab,dc=example,dc=com`

12. Click **Save**.

13. Click **Sync LDAP Groups To Keycloak**.

14. Click **Manage > Users.** Then click **View all users**, all the IdM users must appear:

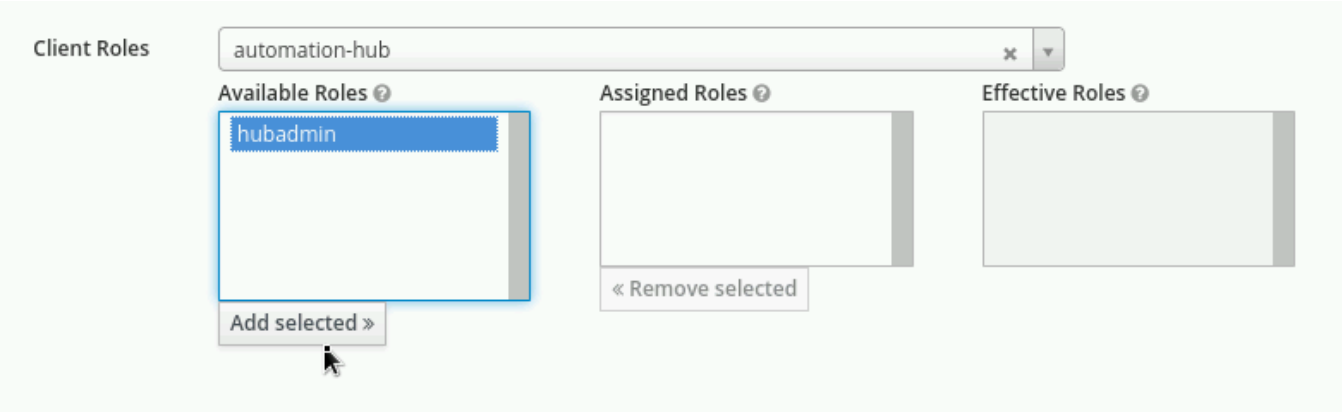


15. Click in the ID link for the `hubadmin` user. We have to indicate to the private automation hub that `hubadmin` is the new administrator.

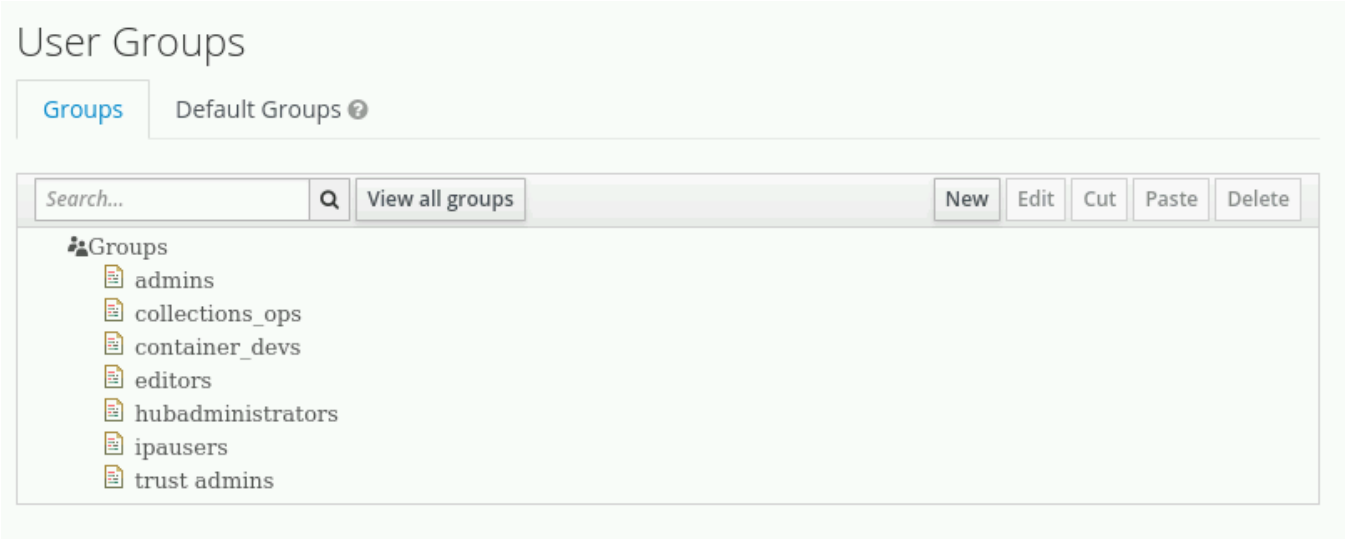
16. Click the **Role Mappings** tab.

17. In **Client Roles** start writing `automation-hub` (it's not visible if you don't start typing it, just write '`au`') then select `automation-hub`.

18. Select the only available role: `hubadmin` and click **Add selected** to add it the the list of assigned roles for this user.

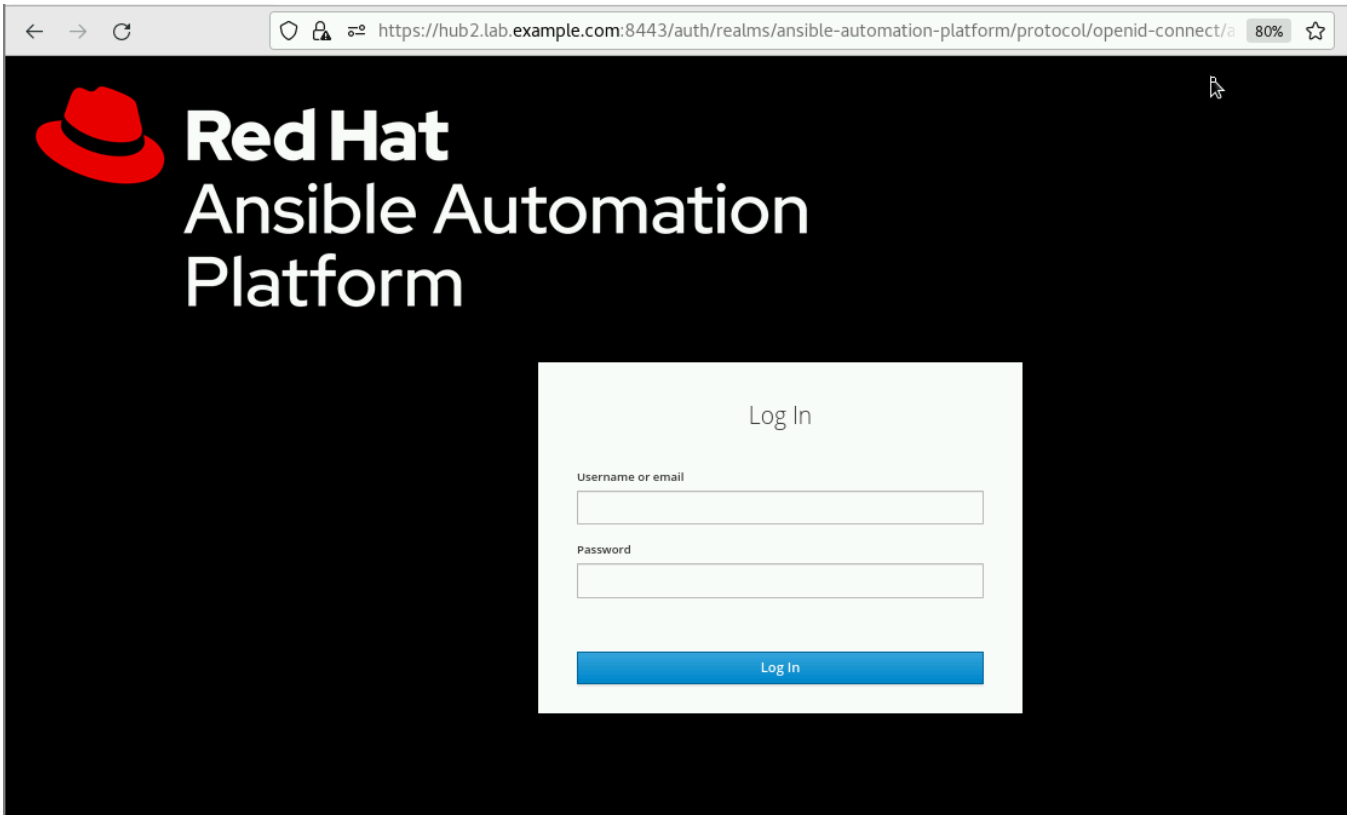


19. Click **Manage > Groups** to verify that the existing groups in the IdM server were well recognized here:

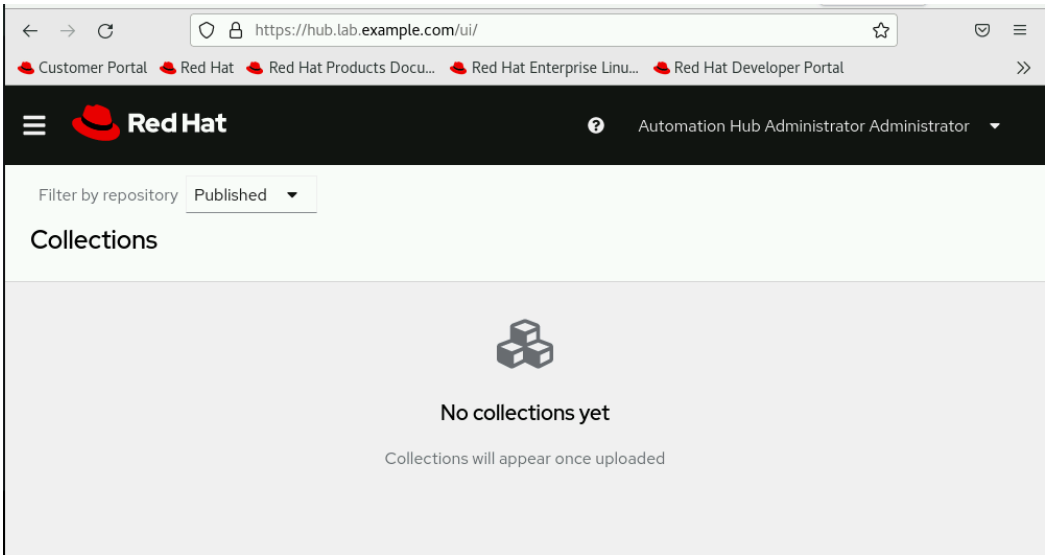


Private automation hub

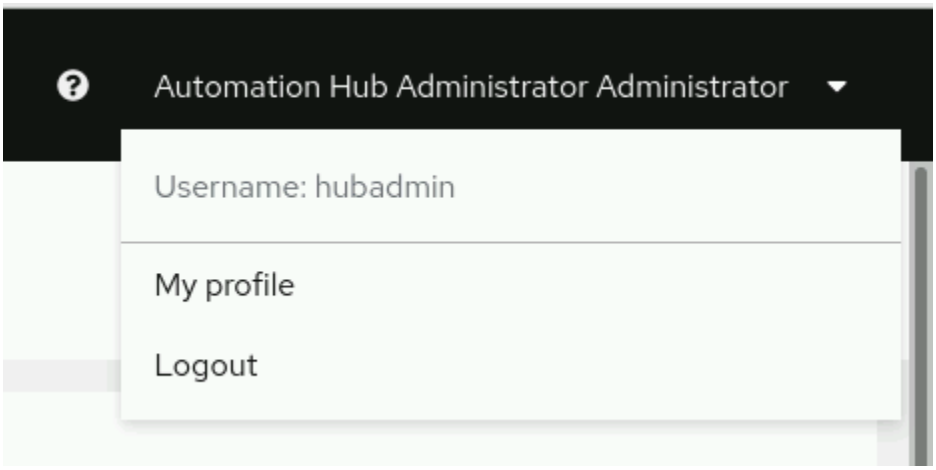
Try to access to private automation hub as usual, browsing to https://hub.lab.example.com. You will be redirected to `hub2` server:



1. Log in the web UI as `hubadmin` using `redhat` as the password. The `hub2` server is used only to authenticate, once your user is recognized, you are redirected to hub server:



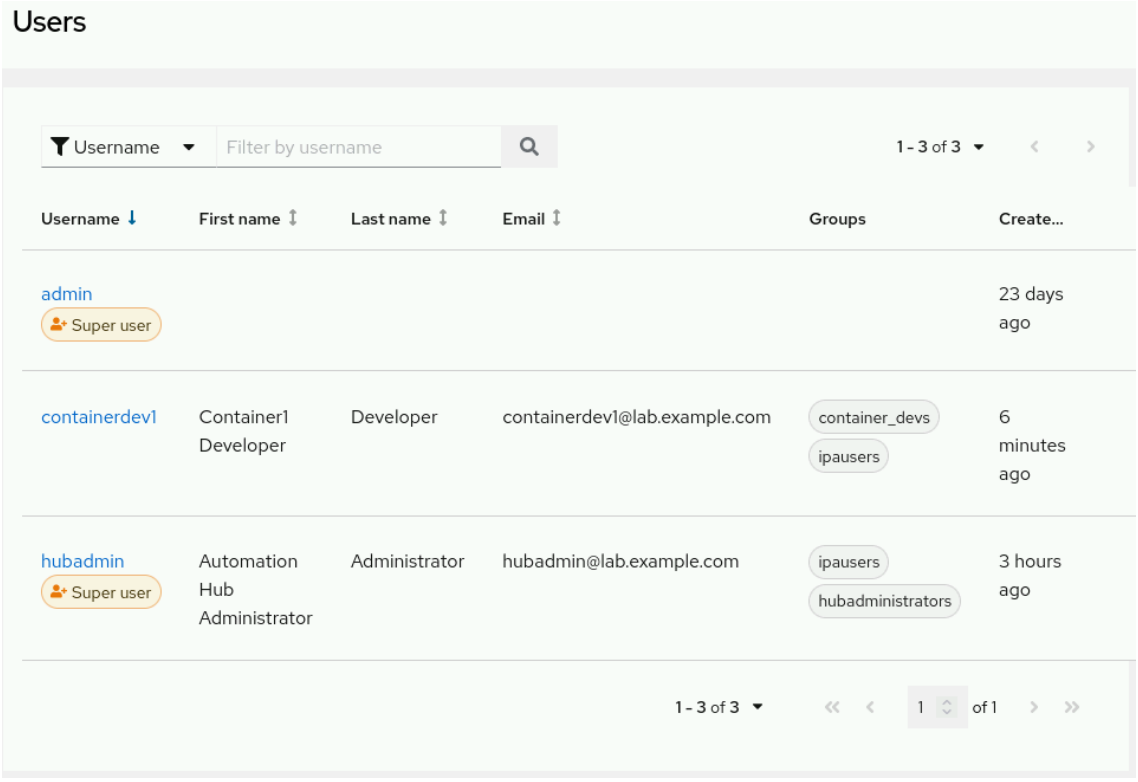
2. Click **Automation Hub Administrator Administrator > My profile** to verify that `hubadmin` is **Super user** user type.



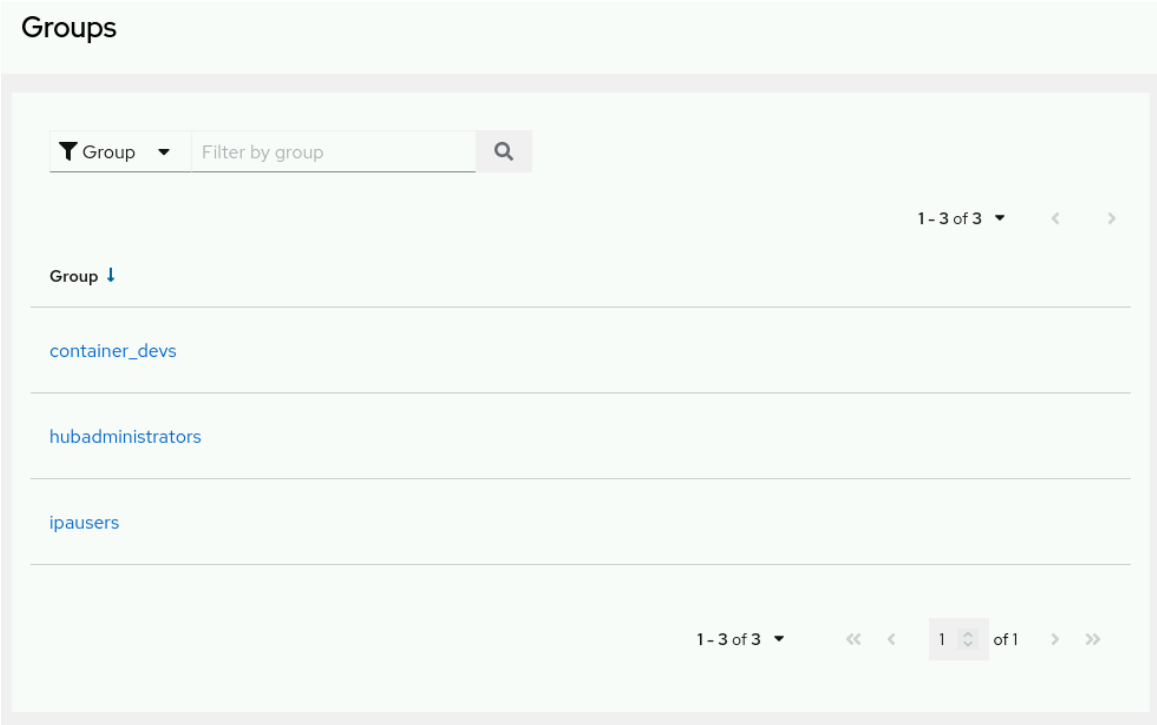
3. Click **Access > Users**. Note that there is no option to create a new user, even when you are a Super user. This is because all the users/groups management is now in the sso server.

4. Log out from the private automation hub web UI and log in as `containerdev1` user. Don't do anything here. This is only for private automation hub recognizes the user and its corresponding groups. Log out and log in as the `hubadmin` user again.

5. Click **Access > Users**. This time we also had the `containerdev1` user in the list of users.



6. Click **Access > Groups** to list the current groups in the private automation hub. As with users, you cannot handle group creation / deletion from here, but you can assign permissions.

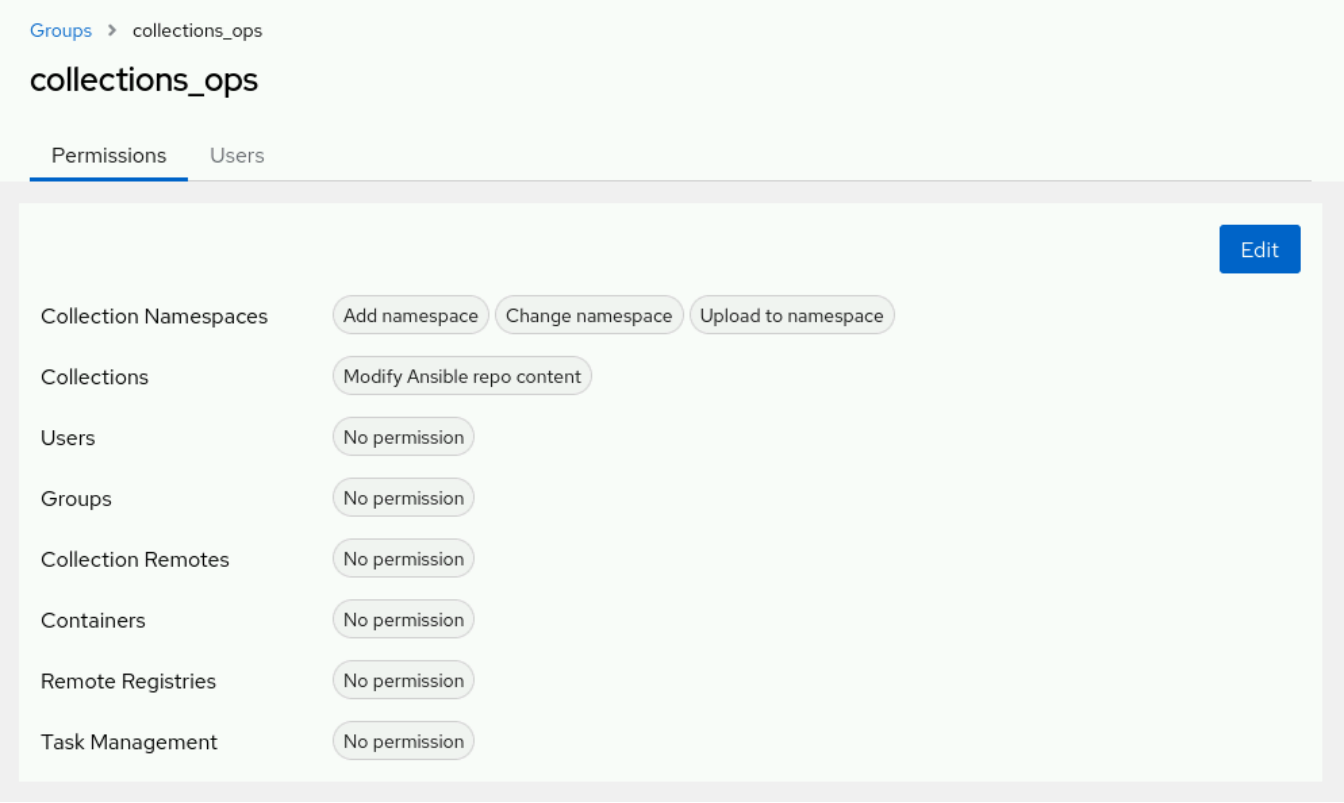


7. Click the link for **container_devs** group.

8. Click **Edit**. In the **Containers** object list, select all the permissions for containers. Then click **Save**.

9. Log out and access now as the user `collectionop1`. Don't do anything here, log out and log in as the `hubadmin` user again.

10. Assign the permission to **Add namespace**, **Change namespace**, **Upload namespace** and **Modify Ansible repo content** to the **collections_ops** group.



11. To verify the permissions for the **collections_ops group**, log out and access now with the `collectionop2` user.

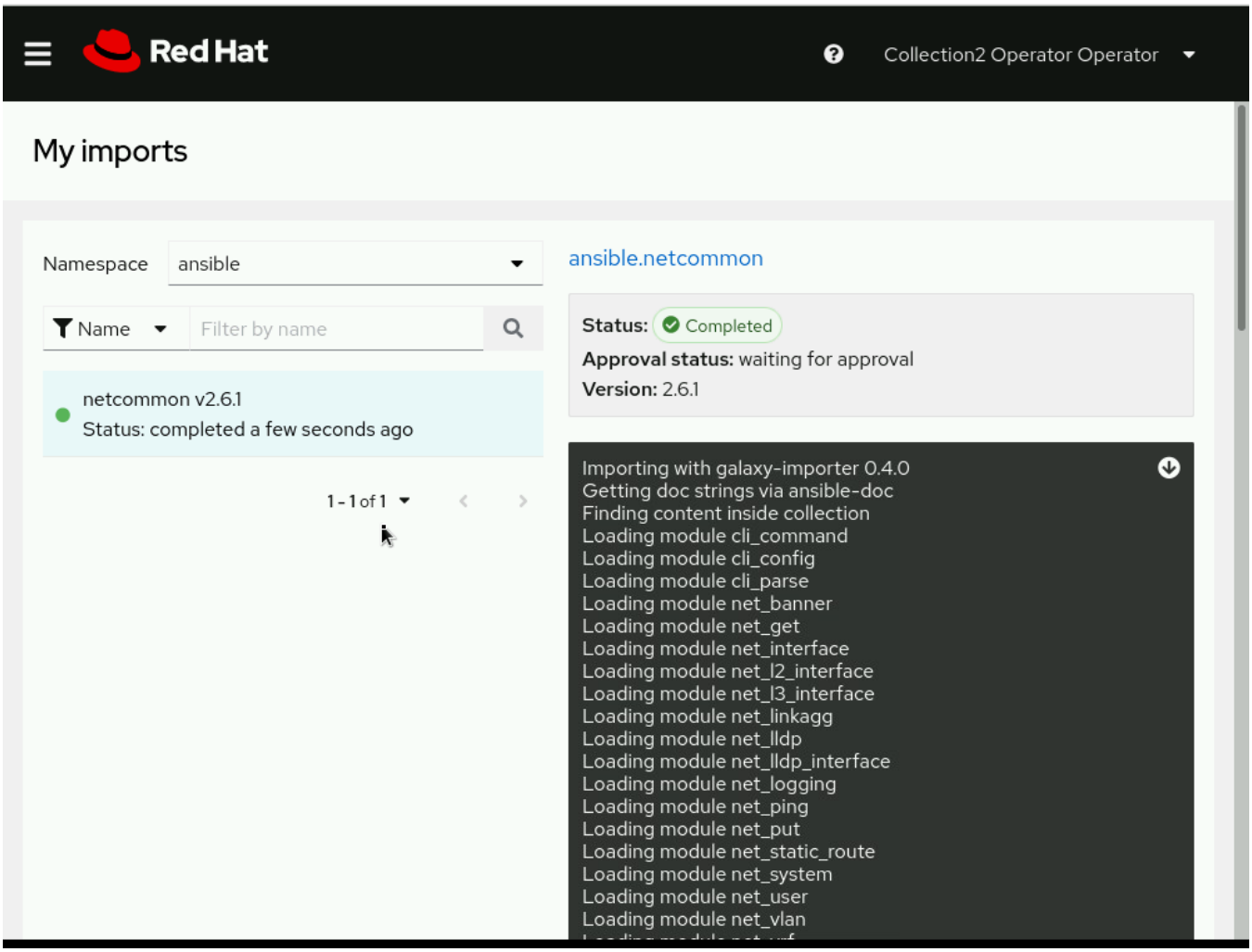
12. Click **Collections > Namespaces**, then click **Create**.

13. Fill in the following and click **Create**.

Name	`ansible`
Namespace owners	`collections_ops`

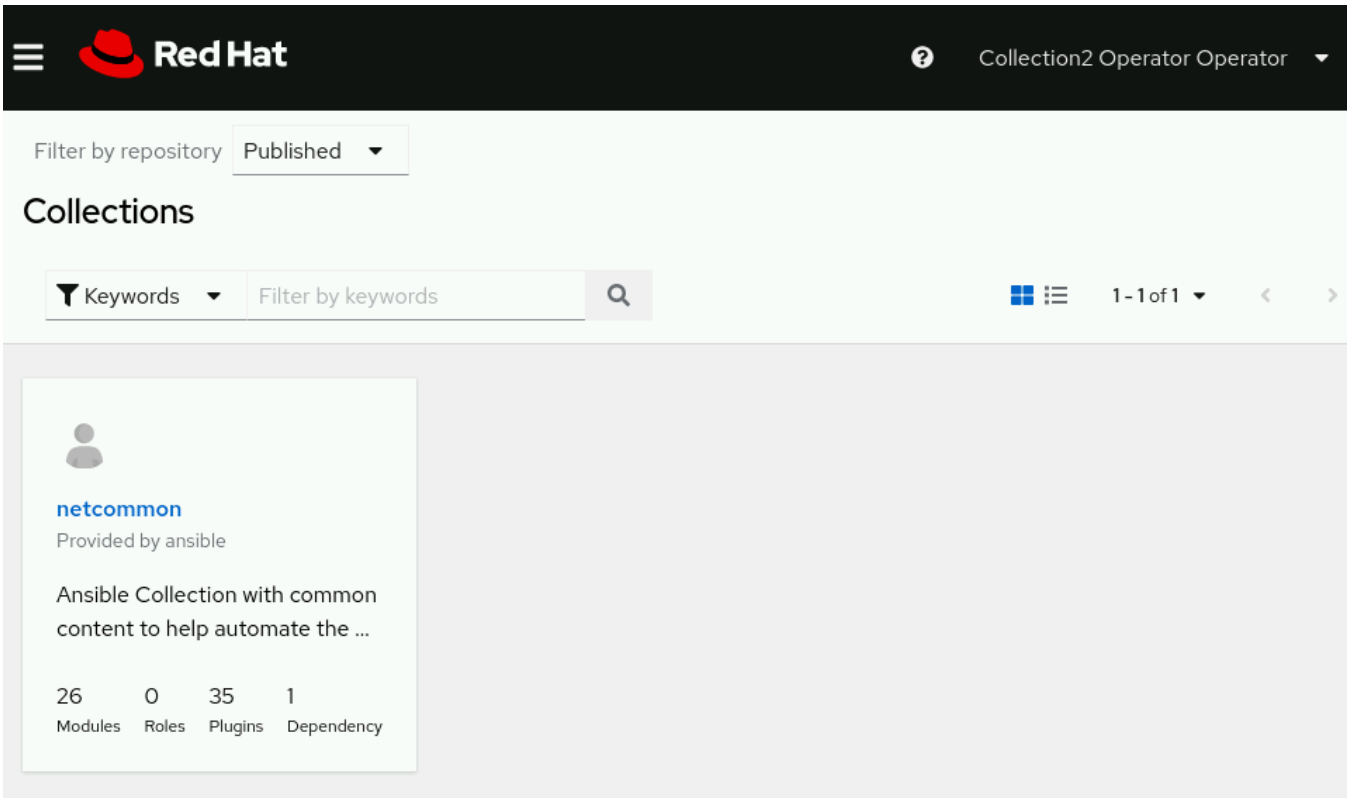
14. Click **Upload collection**.

15. Click select file, and select the collection file.



16. Click **Collections > Approval**, then click in **Approve** for the collection.

17. Click **Collections > Collections** to verify that the collection is inthe private automation hub.



Note that this was done with the `collectionop2`, that was logued for the first time in the private automation hub server, but that user has the privileges to create the namespace because the user belongs to the `collections_ops` group in the IdM server.

In the same way, we can verify permissions for the other users.

Replacing the sso CA certificate

You can configure the sso server to use a valid certificate, but unlike hub, controller and db, for sso it is not installed when running the installation script.

Requirements:

- The CA certificate if using a corporate or enterprise CA.
- The signed certificate for the sso server.
- The associated private key for the signed certificate.

In the `/home/student/certs` directory, I have placed the following files:

- `classroom-ca.pem`
- `hub2.lab.example.com.crt`
- `hub2.lab.example.com.key`

When the sso server is installed, a self-signed certificate for it is created by default. We can check it in the `ansible-automation-platform.jks` keystore inside the `/etc/opt/rh/rh-sso7/keycloak/standalone/` directory, using the `keytool` command with the `redhat` password:

```
[root@hub2 ~]# keytool -list -v -keystore
/etc/opt/rh/rh-sso7/keycloak/standalone/ansible-automation-platform.jks
```

Procedure:

1. Delete the current self-signed certificate for the keystore.

```
[root@hub2 ~]# keytool -delete -alias ansible-automation-platform -keystore
/etc/opt/rh/rh-sso7/keycloak/standalone/ansible-automation-platform.jks
```

2. Create a keystore in `p12` format using the private key and the signed certificate.

```
[root@hub2 ~]# cd /home/student/
[root@hub2 student]# openssl pkcs12 -export -out hub2key.p12 -inkey
certs/hub2.lab.example.com.key -in certs/hub2.lab.example.com.crt -name hub2-certificate
```

   The **`hub2key.p12`** is then created in the work directory.

3. Import the keystore to java keystore using the `keytool` command

```
[root@hub2 student]# keytool -importkeystore -srckeystore hub2key.p12 -destkeystore
/etc/opt/rh/rh-sso7/keycloak/standalone/ansible-automation-platform.jks -srcstoretype
pkcs12
```

4. Add also the CA to the java keystore

```
[root@hub2 student]# keytool -import -file certs/classroom-ca.pem -keystore
/etc/opt/rh/rh-sso7/keycloak/standalone/ansible-automation-platform.jks
```

5. You can verify both entries in the java keystore.

```
[root@hub2 student]# keytool -list -v -keystore
/etc/opt/rh/rh-sso7/keycloak/standalone/ansible-automation-platform.jks|grep
"Certificate\[" -A2
Enter keystore password:  redhat
Certificate[1]:
Owner: CN=hub2.lab.example.com, OU=Training, O="Red Hat, Inc.", L=Raleigh, ST=NC, C=US
Issuer: CN=GLS Training Classroom Certificate Authority, OU=Training, O="Red Hat, Inc.",
L=Raleigh, ST=NC, C=US
```

```
--
Certificate[2]:
Owner: CN=GLS Training Classroom Certificate Authority, OU=Training, O="Red Hat, Inc.",
L=Raleigh, ST=NC, C=US
Issuer: CN=GLS Training Classroom Certificate Authority, OU=Training, O="Red Hat, Inc.",
L=Raleigh, ST=NC, C=US
[root@hub2 student]#
```

6. Restart the `rh-sso7` service

```
[root@hub2 student]# systemctl restart rh-sso7
```

7. The users can now connect to the sso web UI and no warning message regarding a self-signed certificate must appear.