Red Hat Ansible Automation Platform 2.1

Red Hat Ansible Automation Platform Operator Installation Guide

This guide provides procedures and reference information for the supported installation scenarios for the Red Hat Ansible Automation Platform operator on OpenShift Container Platform

Last Updated: 2023-09-27

Red Hat Ansible Automation Platform 2.1 Red Hat Ansible Automation Platform Operator Installation Guide

This guide provides procedures and reference information for the supported installation scenarios for the Red Hat Ansible Automation Platform operator on OpenShift Container Platform

Legal Notice

Abstract

Providing Feedback: If you have a suggestion to improve this documentation, or find an error, please contact technical support at to create an issue on the Ansible Automation Platform Jira project using the Docs component.

Table of Contents

PREFACE

Thank you for your interest in Red Hat Ansible Automation Platform. Ansible Automation Platform is a commercial offering that helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments.

This guide helps you to understand the installation requirements and processes behind installing the Ansible Automation Platform operator on OpenShift Container Platform.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. PLANNING YOUR RED HAT ANSIBLE AUTOMATION PLATFORM OPERATOR INSTALLATION ON RED HAT OPENSHIFT CONTAINER PLATFORM

Red Hat Ansible Automation Platform is supported on both Red Hat Enterprise Linux 8 and Red Hat Openshift.

OpenShift operators help install and automate day-2 operations of complex, distributed software on Red Hat OpenShift Container Platform. The Ansible Automation Platform Operator enables you to deploy and manage Ansible Automation Platform components on Red Hat OpenShift Container Platform.

You can use this section to help plan your Red Hat Ansible Automation Platform installation on your Red Hat OpenShift Container Platform environment. Before installing, review the supported installation scenarios to determine which meets your requirements.

1.1. ABOUT ANSIBLE AUTOMATION PLATFORM OPERATOR

The Ansible Automation Platform Operator provides cloud-native, push-button deployment of new Ansible Automation Platform instances in your OpenShift environment. The Ansible Automation Platform Operator includes resource types to deploy and manage instances of Automation controller and Private Automation hub. It also includes automation controller job resources for defining and launching jobs inside your automation controller deployments.

Deploying Ansible Automation Platform instances with a Kubernetes native operator offers several advantages over launching instances from a playbook deployed on Red Hat OpenShift Container Platform, including upgrades and full lifecycle support for your Red Hat Ansible Automation Platform deployments.

You can install the Ansible Automation Platform Operator from the Red Hat Operators catalog in OperatorHub.

1.2. OPENSHIFT CONTAINER PLATFORM VERSION COMPATIBILITY

The Ansible Automation Platform Operator to install Ansible Automation Platform 2.1 is available on OpenShift Container Platform 4.7 and later versions.

Additional resources

- See the Red Hat Ansible Automation Platform Life Cycle for the most current compatibility details.

1.3. SUPPORTED INSTALLATION SCENARIOS FOR RED HAT OPENSHIFT CONTAINER PLATFORM

You can use the OperatorHub on the Red Hat OpenShift Container Platform web console to install Ansible Automation Platform Operator.

Alternatively, you can install Ansible Automation Platform Operator from the OpenShift Container Platform command-line interface (CLI), **oc**.

Follow one of the workflows below to install the Ansible Automation Platform Operator and use it to install the components of Ansible Automation Platform that you require.

- Automation controller and customer resources first, then automation hub and customer resources;
- Automation hub and customer resources first, then automation controller and customer resources;
- Automation controller and customer resources;
- Automation hub and custom resources.

1.4. CUSTOM RESOURCES

You can define custom resources for each primary installation workflows.

1.5. ADDITIONAL RESOURCES

- See Understanding OperatorHub to learn more about OpenShift Container Platform OperatorHub.

CHAPTER 2. INSTALLING THE RED HAT ANSIBLE AUTOMATION PLATFORM OPERATOR ON RED HAT OPENSHIFT CONTAINER PLATFORM

Prerequisites

- You have installed the Red Hat Ansible Automation Platform catalog in Operator Hub.
- You have created a **StorageClass** object for your platform and a persistant volume claim (PVC) with **ReadWriteMany** access mode. See Dyamic Provisioning for details.
- To run Red Hat OpenShift Container Platform clusters on Amazon Web Services with **ReadWriteMany** access mode, you must add NFS or other storage.
  - For information on the AWS Elastic Block Store (EBS) or to use the **aws-ebs** storage class, see Persistent storage using AWS Elastic Block Store .
  - To use multi-attach **ReadWriteMany** access mode for AWS EBS, see Attaching a volume to multiple instances with Amazon EBS Multi-Attach.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators → OperatorHub**.
3. Search for the Red Hat Ansible Automation Platform operator and click **Install**.
4. Select an **Update Channel**:
   - **stable-2.x**: installs a namespace-scoped operator, which limits deployments of automation hub and automation controller instances to the namespace the operator is installed in. This is suitable for most cases. The stable-2.x channel does not require administrator privileges and utilizes fewer resources because it only monitors a single namespace.
   - **stable-2.x-cluster-scoped**: deploys automation hub and automation controller across multiple namespaces in the cluster and requires administrator privileges for all namespaces in the cluster.
5. Select **Installation Mode**, **Installed Namespace**, and **Approval Strategy**.
6. Click **Install**.

The installation process will begin. When installation is complete, a modal will appear notifying you that the Red Hat Ansible Automation Platform operator is installed in the specified namespace.

- Click **View Operator** to view your newly installed Red Hat Ansible Automation Platform operator.

CHAPTER 3. INSTALLING AND CONFIGURING AUTOMATION CONTROLLER ON RED HAT OPENSHIFT CONTAINER PLATFORM WEB CONSOLE

You can use these instructions to install the automation controller operator on Red Hat OpenShift Container Platform, specify custom resources, and deploy Ansible Automation Platform with an external database.

NOTE

When an instance of automation controller is removed, the associated PVCs are not automatically deleted. This can cause issues during migration if the new deployment has the same name as the previous one. Therefore, it is recommended that you manually remove old PVCs before deploying a new automation controller instance in the same namespace. See Finding and deleting PVCs for more information.

3.1. PREREQUISITES

- You have installed the Red Hat Ansible Automation Platform catalog in Operator Hub.

3.2. INSTALLING THE AUTOMATION CONTROLLER OPERATOR

1. Navigate to **Operators → Installed Operators**, then click on the **Ansible Automation Platform** operator.
2. Locate the **Automation controller** tab, then click **Create instance**.

You can proceed with configuring the instance using either the Form View or YAML view.

3.2.1. Configure your automation controller operator route options

The Red Hat Ansible Automation Platform operator installation form allows you to further configure your automation controller operator route options under **Advanced configuration**.

1. Click **Advanced configuration**.
2. Under **Ingress type**, click the drop-down menu and select **Route**.
3. Under **Route DNS host**, enter a common host name that the route answers to.
4. Under **Route TLS termination mechanism**, click the drop-down menu and select **Edge** or **Passthrough**.
5. Under **Route TLS credential secret**, click the drop-down menu and select a secret from the list.

3.2.2. Configure the Ingress type for your automation hub operator

The Red Hat Ansible Automation Platform operator installation form allows you to further configure your automation hub operator Ingress under **Advanced configuration**.

Procedure

1. Click **Advanced Configuration**.

2. Under **Ingress type**, click the drop-down menu and select **Ingress**.
3. Under **Ingress annotations**, enter any annotations to add to the ingress.
4. Under **Ingress TLS secret**, click the drop-down menu and select a secret from the list.

After you have configured your automation hub operator, click **Create** at the bottom of the form view. Red Hat OpenShift Container Platform will now create the pods. This may take a few minutes.

You can view the progress by navigating to **Workloads → Pods** and locating the newly created instance.

Verification

Verify that the following operator pods provided by the Ansible Automation Platform Operator installation from automation hub are running:

Operator manager controllers	automation controller	automation hub
The operator manager controllers for each of the 3 operators, include the following: • automation-controller-operator-controller-manager • automation-hub-operator-controller-manager • resource-operator-controller-manager	After deploying automation controller, you will see the addition of these pods: • controller • controller-postgres	After deploying automation hub, you will see the addition of these pods: • hub-api • hub-content • hub-postgres • hub-redis • hub-worker

NOTE

A missing pod can indicate the need for a pull secret. Pull secrets are required for protected or private image registries. See Using image pull secrets for more information. You can diagnose this issue further by running **oc describe pod <pod-name>** to see if there is an ImagePullBackOff error on that pod.

Once you have configured your automation controller operator, click **Create** at the bottom of the form view. Red Hat OpenShift Container Platform will now create the pods. This may take a few minutes.

- View progress by navigating to **Workloads → Pods** and locating the newly created instance.

3.3. CONFIGURING AN EXTERNAL DATABASE FOR AUTOMATION CONTROLLER ON RED HAT ANSIBLE AUTOMATION PLATFORM OPERATOR

For users who prefer to deploy Ansible Automation Platform with an external database, they can do so by configuring a secret with instance credentials and connection information, then applying it to their cluster using the **oc create** command.

By default, the Red Hat Ansible Automation Platform operator automatically creates and configures a managed PostgreSQL pod in the same namespace as your Ansible Automation Platform deployment. You can deploy Ansible Automation Platform with an external database instead of the managed PostgreSQL pod that the Red Hat Ansible Automation Platform operator automatically creates.

Using an external database lets you share and reuse resources and manually manage backups, upgrades, and performance optimizations.

NOTE

The same external database (PostgreSQL instance) can be used for both automation hub and automation controller as long as the database names are different. In other words, you can have multiple databases with different names inside a single PostgreSQL instance.

The following section outlines the steps to configure an external database for your automation controller on a Ansible Automation Platform operator.

Prerequisite

The external database must be a PostgreSQL database that is the version supported by the current release of Ansible Automation Platform.

NOTE

Ansible Automation Platform 2.0 and 2.1 supports PostgreSQL 12.

Procedure

The external postgres instance credentials and connection information will need to be stored in a secret, which will then be set on the automation controller spec.

1. Create a **postgres_configuration_secret** .yaml file, following the template below:

```
apiVersion: v1
kind: Secret
metadata:
  name: external-postgres-configuration
  namespace: <target_namespace> (1)
stringData:
  host: "<external_ip_or_url_resolvable_by_the_cluster>" (2)
  port: "<external_port>" (3)
  database: "<desired_database_name>"
  username: "<username_to_connect_as>"
  password: "<password_to_connect_with>" (4)
  sslmode: "prefer" (5)
  type: "unmanaged"
type: Opaque
```

(1) Namespace to create the secret in. This should be the same namespace you wish to deploy to.

(2) The resolvable hostname for your database node.

3. External port defaults to **5432**.
4. Value for variable **password** should not contain single or double quotes (', ") or backslashes (\) to avoid any issues during deployment, backup or restoration.
5. The variable **sslmode** is valid for **external** databases only. The allowed values are: **prefer**, **disable**, **allow**, **require**, **verify-ca**, and **verify-full**.

2. Apply **external-postgres-configuration-secret.yml** to your cluster using the **oc create** command.

```
$ oc create -f external-postgres-configuration-secret.yml
```

3. When creating your **AutomationController** custom resource object, specify the secret on your spec, following the example below:

```
apiVersion: awx.ansible.com/v1beta1
kind: AutomationController
metadata:
  name: controller-dev
spec:
  postgres_configuration_secret: external-postgres-configuration
```

3.4. FINDING AND DELETING PVCS

A persistent volume claim (PVC) is a storage volume used to store data that automation hub and automation controller applications use. These PVCs are independent from the applications and remain even when the application is deleted. If you are confident that you no longer need a PVC, or have backed it up elsewhere, you can manually delete them.

Procedure

1. List the existing PVCs in your deployment namespace:

```
oc get pvc -n <namespace>
```

2. Identify the PVC associated with your previous deployment by comparing the old deployment name and the PVC name.
3. Delete the old PVC:

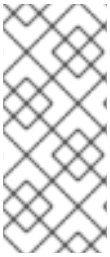
```
oc delete pvc -n <namespace> <pvc-name>
```

== Additional resources

- For more information on running operators on OpenShift Container Platform, navigate to the OpenShift Container Platform product documentation and click the *Operators - Working with Operators in OpenShift Container Platform* guide.

CHAPTER 4. INSTALLING AND CONFIGURING AUTOMATION HUB ON RED HAT OPENSHIFT CONTAINER PLATFORM WEB CONSOLE

You can use these instructions to install the automation hub operator on Red Hat OpenShift Container Platform, specify custom resources, and deploy Ansible Automation Platform with an external database.



NOTE

When an instance of automation hub is removed, the PVCs are not automatically deleted. This can cause issues during migration if the new deployment has the same name as the previous one. Therefore, it is recommended that you manually remove old PVCs before deploying a new automation hub instance in the same namespace. See Finding and deleting PVCs for more information.

4.1. PREREQUISITES

- You have installed the Red Hat Ansible Automation Platform operator in Operator Hub.

4.2. INSTALLING THE AUTOMATION HUB OPERATOR

1. Navigate to **Operators → Installed Operators**.
2. Locate the **Automation hub** entry, then click **Create instance**.

4.2.1. Storage options for Ansible Automation Platform Operator installation on Red Hat OpenShift Container Platform

If you are using file-based storage and your installation scenario includes automation hub, ensure that you change the **ReadWriteOnce** default storage option for Ansible Automation Platform Operator to **ReadWriteMany**.

Automation hub requires **ReadWriteMany** file-based storage, Azure Blob storage, or Amazon S3-compliant storage for operation so that multiple pods can access shared content, such as collections.

In addition, OpenShift Data Foundation provides a **ReadWriteMany** or S3-compliant implementation. Also, you can set up NFS storage configuration to support **ReadWriteMany**. This, however, introduces the NFS server as a potential, single point of failure.

Additional resources

- Persistent storage using NFS in the OpenShift Container Platform *Storage* guide
- IBM's How do I create a storage class for NFS dynamic storage provisioning in an OpenShift environment?

4.2.1.1. Provisioning OCP storage with ReadWriteMany access mode

To ensure successful installation of Ansible Automation Platform Operator, you must provision your storage type for automation hub initially to **ReadWriteMany** access mode.

Procedure

1. Click Provisioning to update the access mode.
2. In the first step, update the **accessModes** from the default **ReadWriteOnce** to **ReadWriteMany**.
3. Complete the additional steps in this section to create the persistent volume claim (PVC).

4.2.2. Configure your automation hub operator route options

The Red Hat Ansible Automation Platform operator installation form allows you to further configure your automation hub operator route options under **Advanced configuration**.

1. Click **Advanced configuration**.
2. Under **Ingress type**, click the drop-down menu and select **Route**.
3. Under **Route DNS host**, enter a common host name that the route answers to.
4. Under **Route TLS termination mechanism**, click the drop-down menu and select **Edge** or **Passthrough**.
5. Under **Route TLS credential secret**, click the drop-down menu and select a secret from the list.

4.2.3. Configure the Ingress type for your automation hub operator

The Red Hat Ansible Automation Platform operator installation form allows you to further configure your automation hub operator Ingress under **Advanced configuration**.

Procedure

1. Click **Advanced Configuration**.
2. Under **Ingress type**, click the drop-down menu and select **Ingress**.
3. Under **Ingress annotations**, enter any annotations to add to the ingress.
4. Under **Ingress TLS secret**, click the drop-down menu and select a secret from the list.

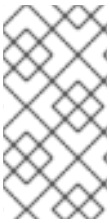
After you have configured your automation hub operator, click **Create** at the bottom of the form view. Red Hat OpenShift Container Platform will now create the pods. This may take a few minutes.

You can view the progress by navigating to **Workloads → Pods** and locating the newly created instance.

Verification

Verify that the following operator pods provided by the Ansible Automation Platform Operator installation from automation hub are running:

Operator manager controllers	automation controller	automation hub
The operator manager controllers for each of the 3 operators, include the following: • automation-controller-operator-controller-manager • automation-hub-operator-controller-manager • resource-operator-controller-manager	After deploying automation controller, you will see the addition of these pods: • controller • controller-postgres	After deploying automation hub, you will see the addition of these pods: • hub-api • hub-content • hub-postgres • hub-redis • hub-worker



NOTE

A missing pod can indicate the need for a pull secret. Pull secrets are required for protected or private image registries. See Using image pull secrets for more information. You can diagnose this issue further by running **oc describe pod <pod-name>** to see if there is an ImagePullBackOff error on that pod.

Once you have configured your automation hub operator, click **Create** at the bottom of the form view. Red Hat OpenShift Container Platform will now create the pods. This may take a few minutes.

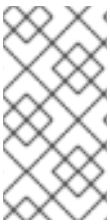
- View progress by navigating to **Workloads → Pods** and locating the newly created instance.

4.3. ACCESSING THE AUTOMATION HUB USER INTERFACE

You can access the automation hub interface once all pods have successfully launched.

1. Navigate to **Networking → Routes**.
2. Under **Location**, click on the URL for your automation hub instance.

The automation hub user interface launches where you can sign in with the administrator credentials specified during the operator configuration process.



NOTE

If you did not specify an administrator password during configuration, one was automatically created for you. To locate this password, go to your project, select **Workloads → Secrets** and open controller-admin-password. From there you can copy the password and paste it into the Automation hub password field.

4.4. CONFIGURING AN EXTERNAL DATABASE FOR AUTOMATION HUB ON RED HAT ANSIBLE AUTOMATION PLATFORM OPERATOR

For users who prefer to deploy Ansible Automation Platform with an external database, they can do so by configuring a secret with instance credentials and connection information, then applying it to their cluster using the **oc create** command.

By default, the Red Hat Ansible Automation Platform operator automatically creates and configures a managed PostgreSQL pod in the same namespace as your Ansible Automation Platform deployment.

You can choose to use an external database instead if you prefer to use a dedicated node to ensure dedicated resources or to manually manage backups, upgrades, or performance tweaks.

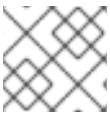
NOTE

The same external database (PostgreSQL instance) can be used for both automation hub and automation controller as long as the database names are different. In other words, you can have multiple databases with different names inside a single PostgreSQL instance.

The following section outlines the steps to configure an external database for your automation hub on a Ansible Automation Platform operator.

Prerequisite

The external database must be a PostgreSQL database that is the version supported by the current release of Ansible Automation Platform.



NOTE

Ansible Automation Platform 2.0 and 2.1 supports PostgreSQL 12.

Procedure

The external postgres instance credentials and connection information will need to be stored in a secret, which will then be set on the automation hub spec.

1. Create a **postgres_configuration_secret** .yaml file, following the template below:

```
apiVersion: v1
kind: Secret
metadata:
  name: external-postgres-configuration
  namespace: <target_namespace> (1)
stringData:
  host: "<external_ip_or_url_resolvable_by_the_cluster>" (2)
  port: "<external_port>" (3)
  database: "<desired_database_name>"
  username: "<username_to_connect_as>"
  password: "<password_to_connect_with>" (4)
  sslmode: "prefer" (5)
  type: "unmanaged"
type: Opaque
```

(1) Namespace to create the secret in. This should be the same namespace you wish to deploy to.

2 The resolvable hostname for your database node.

3 External port defaults to **5432**.

4 Value for variable **password** should not contain single or double quotes (', ") or backslashes (\) to avoid any issues during deployment, backup or restoration.

5 The variable **sslmode** is valid for **external** databases only. The allowed values are: **prefer**, **disable**, **allow**, **require**, **verify-ca**, and **verify-full**.

2. Apply **external-postgres-configuration-secret.yml** to your cluster using the **oc create** command.

```
$ oc create -f external-postgres-configuration-secret.yml
```

3. When creating your **AutomationHub** custom resource object, specify the secret on your spec, following the example below:

```
apiVersion: awx.ansible.com/v1beta1
kind: AutomationHub
metadata:
  name: hub-dev
spec:
  postgres_configuration_secret: external-postgres-configuration
```

4.5. FINDING AND DELETING PVCS

A persistent volume claim (PVC) is a storage volume used to store data that automation hub and automation controller applications use. These PVCs are independent from the applications and remain even when the application is deleted. If you are confident that you no longer need a PVC, or have backed it up elsewhere, you can manually delete them.

Procedure

1. List the existing PVCs in your deployment namespace:

```
oc get pvc -n <namespace>
```

2. Identify the PVC associated with your previous deployment by comparing the old deployment name and the PVC name.

3. Delete the old PVC:

```
oc delete pvc -n <namespace> <pvc-name>
```

== Additional resources

- For more information on running operators on OpenShift Container Platform, navigate to the OpenShift Container Platform product documentation and click the *Operators - Working with Operators in OpenShift Container Platform* guide.

CHAPTER 5. INSTALLING ANSIBLE AUTOMATION PLATFORM OPERATOR FROM THE OPENSHIFT CONTAINER PLATFORM CLI

Use these instructions to install the Ansible Automation Platform Operator on Red Hat OpenShift Container Platform from the OpenShift Container Platform command-line interface (CLI) using the **oc** command.

5.1. PREREQUISITES

- Access to Red Hat OpenShift Container Platform using an account with operator installation permissions.
- The OpenShift Container Platform CLI **oc** command is installed on your local system. Refer to Installing the OpenShift CLI in the Red Hat OpenShift Container Platform product documentation for further information.

5.2. SUBSCRIBING A NAMESPACE TO AN OPERATOR USING THE OPENSHIFT CONTAINER PLATFORM CLI

1. Create a project for the operator

```
oc new-project ansible-automation-platform
```

2. Create a file called **sub.yaml**.
3. Add the following YAML code to the **sub.yaml** file.

```
---
apiVersion: v1
kind: Namespace
metadata:
  labels:
    openshift.io/cluster-monitoring: "true"
  name: ansible-automation-platform
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: ansible-automation-platform-operator
  namespace: ansible-automation-platform
spec:
  targetNamespaces:
    - ansible-automation-platform
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ansible-automation-platform
  namespace: ansible-automation-platform
spec:
  channel: 'stable-2.1'
  installPlanApproval: Automatic
```

```
  name: ansible-automation-platform-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
---
apiVersion: automationcontroller.ansible.com/v1beta1
kind: AutomationController
metadata:
  name: example
  namespace: ansible-automation-platform
spec:
  replicas: 1
```

This file creates a **Subscription** object called ***ansible-automation-platform*** that subscribes the **ansible-automation-platform** namespace to the **ansible-automation-platform-operator** operator.

It then creates an **AutomationController** object called ***example*** in the **ansible-automation-platform** namespace.

To change the Automation controller name from ***example***, edit the *name* field in the **kind: AutomationController** section of **sub.yaml** and replace ***<automation_controller_name>*** with the name you want to use:

```
apiVersion: automationcontroller.ansible.com/v1beta1
kind: AutomationController
metadata:
  name: <automation_controller_name>
  namespace: ansible-automation-platform
```

4. Run the **oc apply** command to create the objects specified in the **sub.yaml** file:

```
oc apply -f sub.yaml
```

To verify that the namespace has been successfully subscribed to the **ansible-automation-platform-operator** operator, run the **oc get subs** command:

```
$ oc get subs -n ansible-automation-platform
```

For further information about subscribing namespaces to operators, see Installing from OperatorHub using the CLI in the Red Hat OpenShift Container Platform *Operators* guide.

You can use the OpenShift Container Platform CLI to fetch the web address and the password of the Automation controller that you created.

5.3. FETCHING AUTOMATION CONTROLLER LOGIN DETAILS FROM THE OPENSHIFT CONTAINER PLATFORM CLI

To login to the Automation controller, you need the web address and the password.

5.3.1. Fetching the automation controller web address

A Red Hat OpenShift Container Platform route exposes a service at a host name, so that external clients can reach it by name. When you created the automation controller instance, a route was created for it. The route inherits the name that you assigned to the automation controller object in the YAML file.

Use the following command to fetch the routes:

```
oc get routes -n <controller_namespace>
```

In the following example, the ***example*** automation controller is running in the ***ansible-automation-platform*** namespace.

```
$ oc get routes -n ansible-automation-platform

NAME      HOST/PORT                                         PATH   SERVICES        PORT   TERMINATION
WILDCARD
example   example-ansible-automation-platform.apps-crc.testing          example-service   http
edge/Redirect   None
```

The address for the automation controller instance is **example-ansible-automation-platform.apps-crc.testing**.

5.3.2. Fetching the automation controller password

The YAML block for the automation controller instance in **sub.yaml** assigns values to the *name* and *admin_user* keys. Use these values in the following command to fetch the password for the automation controller instance.

```
oc get secret/<controller_name>-<admin_user>-password -o yaml
```

The default value for *admin_user* is ***admin***. Modify the command if you changed the admin username in **sub.yaml**.

The following example retrieves the password for an automation controller object called ***example***:

```
oc get secret/example-admin-password -o yaml
```

The password for the automation controller instance is listed in the **metadata** field in the output:

```
$ oc get secret/example-admin-password -o yaml

apiVersion: v1
data:
  password: ODzLODzLODzLODzLODzLODzLODzLODzLODzLODzLODzL
kind: Secret
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: '{"apiVersion":"v1","kind":"Secret","metadata":
{"labels":{"app.kubernetes.io/component":"automationcontroller","app.kubernetes.io/managed-
by":"automationcontroller-
operator","app.kubernetes.io/name":"example","app.kubernetes.io/operator-
version":"","app.kubernetes.io/part-of":"example"},"name":"example-admin-
password","namespace":"ansible-automation-platform"},"stringData":
{"password":"88TG88TG88TG88TG88TG88TG88TG88TG"}}'
  creationTimestamp: "2021-11-03T00:02:24Z"
  labels:
    app.kubernetes.io/component: automationcontroller
    app.kubernetes.io/managed-by: automationcontroller-operator
    app.kubernetes.io/name: example
```

```
    app.kubernetes.io/operator-version: ""
    app.kubernetes.io/part-of: example
  name: example-admin-password
  namespace: ansible-automation-platform
  resourceVersion: "185185"
  uid: 39393939-5252-4242-b929-665f665f665f
```

For this example, the password is **88TG88TG88TG88TG88TG88TG88TG88TG**.

5.4. ADDITIONAL RESOURCES

- For more information on running operators on OpenShift Container Platform, navigate to the OpenShift Container Platform product documentation and click the *Operators - Working with Operators in OpenShift Container Platform* guide.

CHAPTER 6. USING RED HAT SINGLE SIGN-ON OPERATOR WITH AUTOMATION HUB

Private automation hub uses Red Hat Single Sign-On for authentication.

The Red Hat Single Sign-On Operator creates and manages resources. Use this Operator to create custom resources to automate Red Hat Single Sign-On administration in Openshift.

- When installing Ansible Automation Platform on *Virtual Machines* (VMs) the installer can automatically install and configure Red Hat Single Sign-On for use with private automation hub.
- When installing Ansible Automation Platform on Red Hat OpenShift Container Platform you must install Single Sign-On separately.

This chapter describes the process to configure Red Hat Single Sign-On and integrate it with private automation hub when Ansible Automation Platform is installed on OpenShift Container Platform.

Prerequisites

- You have access to Red Hat OpenShift Container Platform using an account with operator installation permissions.
- You have installed the catalog containing the Red Hat Ansible Automation Platform operators.
- You have installed the Red Hat Single Sign-On Operator. To install the Red Hat Single Sign-On Operator, follow the procedure in Installing Red Hat Single Sign-On using a custom resource in the Red Hat Single Sign-On documentation.

6.1. CREATING A KEYCLOAK INSTANCE

When the Red Hat Single Sign-On Operator is installed you can create a Keycloak instance for use with Ansible Automation Platform.

From here you provide an external Postgres or one will be created for you.

Procedure

1. Navigate to **Operator → Installed Operators**.
2. Select the **rh-sso** project.
3. Select the **Red Hat Single Sign-On Operator**.
4. On the Red Hat Single Sign-On Operator details page select **Keycloak**.
5. Click **Create instance**.
6. Click **YAML view**.
   The default Keycloak custom resource is as follows:

```
apiVersion: keycloak.org/v1alpha1
kind: Keycloak
metadata:
  name: example-keycloak
  labels:
```

```
  app: sso
   namespace: aap
spec:
  externalAccess:
 enabled: true
  instances: 1
```

7. Click **Create**

8. When deployment is complete, you can use this credential to login to the administrative console.

9. You can find the credentials for the administrator in the **credential-<custom-resource>** (example keycloak) secret in the namespace.

6.2. CREATING A KEYCLOAK REALM FOR ANSIBLE AUTOMATION PLATFORM

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

Procedure

1. Navigate to **Operator → Installed Operators**.

2. Select the **Red Hat Single Sign-On Operator** project.

3. Select the **Keycloak Realm** tab and click **Create Keycloak Realm**.

4. On the **Keycloak Realm** form, select **YAML view**. Edit the YAML file as follows:

```
kind: KeycloakRealm
apiVersion: keycloak.org/v1alpha1
metadata:
  name: ansible-automation-platform-keycloakrealm
  namespace: rh-sso
  labels:
    app: sso
    realm: ansible-automation-platform
spec:
  realm:
    id: ansible-automation-platform
    realm: ansible-automation-platform
    enabled: true
    displayName: {PlatformNameShort}
  instanceSelector:
    matchLabels:
      app: sso
```

Field	Description
metadata.name	Set a unique value in metadata for the name of the configuration resource (CR).

metadata.namespace	Set a unique value in metadata for the name of the configuration resource (CR).
metadata.labels.app	Set labels to a unique value. This is used when creating the client CR.
metadata.labels.realm	Set labels to a unique value. This is used when creating the client CR.
spec.realm.id	Set the realm name and id. These must be the same.
spec.realm.realm	Set the realm name and id. These must be the same.
spec.realm.displayname	Set the name to display.

5. Click **Create** and wait for the process to complete.

6.3. CREATING A KEYCLOAK CLIENT

Keycloak clients authenticate hub users with Red Hat Single Sign-On. When a user authenticates the request goes through the Keycloak client. When Single Sign-On validates or issues the **OAuth** token, the client provides the resonse to automation hub and the user can log in.

Procedure

1. Navigate to **Operator → Installed Operators**.
2. Select the Red Hat Single Sign-On Operator project.
3. Select the **Keycloak Client** tab and click **Create Keycloak Client**.
4. On the Keycloak Realm form, select **YAML view**.
5. Replace the default YAML file with the following:

```
kind: KeycloakClient
apiVersion: keycloak.org/v1alpha1
metadata:
  name: automation-hub-client-secret
  labels:
    app: sso
    realm: ansible-automation-platform
  namespace: rh-sso
spec:
  realmSelector:
    matchLabels:
      app: sso
      realm: ansible-automation-platform
  client:
```

```
name: Automation Hub
clientId: automation-hub
secret: <client-secret>                    (1)
clientAuthenticatorType: client-secret
description: Client for {HubNameStart}
attributes:
  user.info.response.signature.alg: RS256
  request.object.signature.alg: RS256
directAccessGrantsEnabled: true
publicClient: true
protocol: openid-connect
standardFlowEnabled: true
protocolMappers:
  - config:
      access.token.claim: "true"
      claim.name: "family_name"
      id.token.claim: "true"
      jsonType.label: String
      user.attribute: lastName
      userinfo.token.claim: "true"
    consentRequired: false
    name: family name
    protocol: openid-connect
    protocolMapper: oidc-usermodel-property-mapper
  - config:
      userinfo.token.claim: "true"
      user.attribute: email
      id.token.claim: "true"
      access.token.claim: "true"
      claim.name: email
      jsonType.label: String
    name: email
    protocol: openid-connect
    protocolMapper: oidc-usermodel-property-mapper
    consentRequired: false
  - config:
      multivalued: "true"
      access.token.claim: "true"
      claim.name: "resource_access.${client_id}.roles"
      jsonType.label: String
    name: client roles
    protocol: openid-connect
    protocolMapper: oidc-usermodel-client-role-mapper
    consentRequired: false
  - config:
      userinfo.token.claim: "true"
      user.attribute: firstName
      id.token.claim: "true"
      access.token.claim: "true"
      claim.name: given_name
      jsonType.label: String
    name: given name
    protocol: openid-connect
    protocolMapper: oidc-usermodel-property-mapper
    consentRequired: false
  - config:
```

```
      id.token.claim: "true"
      access.token.claim: "true"
      userinfo.token.claim: "true"
    name: full name
    protocol: openid-connect
    protocolMapper: oidc-full-name-mapper
    consentRequired: false
  - config:
      userinfo.token.claim: "true"
      user.attribute: username
      id.token.claim: "true"
      access.token.claim: "true"
      claim.name: preferred_username
      jsonType.label: String
    name: <username>
    protocol: openid-connect
    protocolMapper: oidc-usermodel-property-mapper
    consentRequired: false
  - config:
      access.token.claim: "true"
      claim.name: "group"
      full.path: "true"
      id.token.claim: "true"
      userinfo.token.claim: "true"
    consentRequired: false
    name: group
    protocol: openid-connect
    protocolMapper: oidc-group-membership-mapper
  - config:
      multivalued: 'true'
      id.token.claim: 'true'
      access.token.claim: 'true'
      userinfo.token.claim: 'true'
      usermodel.clientRoleMapping.clientId:  '{HubName}'
      claim.name: client_roles
      jsonType.label: String
    name: client_roles
    protocolMapper: oidc-usermodel-client-role-mapper
    protocol: openid-connect
  - config:
      id.token.claim: "true"
      access.token.claim: "true"
      included.client.audience: '{HubName}'
    protocol: openid-connect
    name: audience mapper
    protocolMapper: oidc-audience-mapper
roles:
  - name: "hubadmin"
    description: "An administrator role for {HubNameStart}"
```

1 Replace this with a unique value.

6. Click **Create** and wait for the process to complete.

When automation hub is deployed, you must update the client with the “Valid Redirect URIs” and “Web

Origins" as described in Updating the Red Hat Single Sign-On client Additionally, the client comes pre-configured with token mappers, however, if your authentication provider does not provide group data to Red Hat SSO, then the group mapping must be updated to reflect how that information is passed. This is commonly by user attribute.

6.4. CREATING A KEYCLOAK USER

This procedure creates a Keycloak user, with the **hubadmin** role, that can log in to automation hub with Super Administration privileges.

Procedure

1. Navigate to **Operator → Installed Operators**.
2. Select the Red Hat Single Sign-On Operator project.
3. Select the **Keycloak Realm** tab and click **Create Keycloak User**.
4. On the **Keycloak User** form, select **YAML view**.
5. Replace the default YAML file with the following:

```
apiVersion: keycloak.org/v1alpha1
kind: KeycloakUser
metadata:
  name: hubadmin-user
  labels:
    app: sso
    realm: ansible-automation-platform
  namespace: rh-sso
spec:
  realmSelector:
    matchLabels:
      app: sso
      realm: ansible-automation-platform
  user:
    username: hub_admin
    firstName: Hub
    lastName: Admin
    email: hub_admin@example.com
    enabled: true
    emailVerified: false
    credentials:
      - type: password
        value: <ch8ngeme>
    clientRoles:
      automation-hub:
        - hubadmin
```

6. Click **Create** and wait for the process to complete.

When a user is created, the Operator creates a Secret containing both the username and password using the following naming pattern: **credential-<realm name>-<username>-<namespace>**. In this example the credential is called **credential-ansible-automation-platform-hub-admin-rh-sso**. When a

user is created the Operator does not update the user's password. Password changes are not reflected in the Secret.

6.5. INSTALLING THE ANSIBLE AUTOMATION PLATFORM OPERATOR

Procedure

1. Navigate to **Operator → Operator Hub** and search for the Ansible Automation Platform Operator.
2. Select the Ansible Automation Platform Operator project.
3. Click on the Operator tile.
4. Click **Install**.
5. Select a Project to install the Operator into. Red Hat recommends using the Operator recommended Namespace name.
    a. If you want to install the Operator into a project other than the recommended one, select **Create Project** from the drop down menu.
    b. Enter the Project name.
    c. Click **Create**.
6. Click **Install**.
7. When the Operator has been installed, click **View Operator**.

6.6. CREATING A RED HAT SINGLE SIGN-ON CONNECTION SECRET

Procedure

1. Navigate to **https://<sso_host>/auth/realms/ansible-automation-platform**.
2. Copy the **public_key** value.
3. In the OpenShift Web UI, navigate to **Workloads → Secrets**.
4. Select the **ansible-automation-platform** project.
5. Click **Create**, and select **From YAML**.
6. Edit the following YAML to create the secret

```
apiVersion: v1
kind: Secret
metadata:
  name: automation-hub-sso                    (1)
  namespace: ansible-automation-platform
type: Opaque
stringData:
  keycloak_host: "keycloak-rh-sso.apps-crc.testing"
  keycloak_port: "443"
```

```
keycloak_protocol: "https"
keycloak_realm: "ansible-automation-platform"
keycloak_admin_role: "hubadmin"
social_auth_keycloak_key: "automation-hub"
social_auth_keycloak_secret: "client-secret"  2
social_auth_keycloak_public_key: >-  3
```

1. This name is used in the next step when creating the automation hub instance.
2. If the secret was changed when creating the Keycloak client for automation hub be sure to change this value to match.
3. Enter the value of the **public_key** copied in Installing the Ansible Automation Platform Operator.

7. Click **Create** and wait for the process to complete.

6.7. INSTALLING AUTOMATION HUB USING THE OPERATOR

Use the following procedure to install automation hub using the operator.

Procedure

1. Navigate to **Operator → Installed Operators**.
2. Select the Ansible Automation Platform.
3. Select the Automation hub tab and click **Create Automation hub**.
4. Select **YAML view**. The YAML should be similar to:

```
apiVersion: automationhub.ansible.com/v1beta1
kind: AutomationHub
metadata:
  name: private-ah  1
  namespace: ansible-automation-platform
spec:
  sso_secret: automation-hub-sso  2
  pulp_settings:
    verify_ssl: false
  route_tls_termination_mechanism: Edge
  ingress_type: Route
  loadbalancer_port: 80
  file_storage_size: 100Gi
  image_pull_policy: IfNotPresent
  web:
    replicas: 1
  file_storage_access_mode: ReadWriteMany
  content:
    log_level: INFO
    replicas: 2
  postgres_storage_requirements:
    limits:
      storage: 50Gi
```

```
    requests:
      storage: 8Gi
  api:
    log_level: INFO
    replicas: 1
  postgres_resource_requirements:
    limits:
      cpu: 1000m
      memory: 8Gi
    requests:
      cpu: 500m
      memory: 2Gi
  loadbalancer_protocol: http
  resource_manager:
    replicas: 1
  worker:
    replicas: 2
```

1. Set metadata.name to the name to use for the instance.
2. Set spec.sso_secret to the name of the secret created in Creating a Secret to hold the Red Hat Single Sign On connection details.

NOTE

This YAML turns off SSL verification (**ssl_verify: false**). If you are not using self-signed certificates for OpenShift this setting can be removed.

5. Click **Create** and wait for the process to complete.

6.8. DETERMINING THE AUTOMATION HUB ROUTE

Use the following procedure to determine the hub route.

Procedure

1. Navigate to **Networking → Routes**.
2. Select the project you used for the install.
3. Copy the location of the **private-ah-web-svc** service. The name of the service is different if you used a different name when creating the automation hub instance. This is used later to update the Red Hat Single Sign-On client.

6.9. UPDATING THE RED HAT SINGLE SIGN-ON CLIENT

When automation hub is installed and you know the URL of the instance, you must update the Red Hat Single Sign-On to set the Valid Redirect URIs and Web Origins settings.

Procedure

1. Navigate to **Operator → Installed Operators**.

2. Select the RH-SSO project.
3. Click **Red Hat Single Sign-On Operator**.
4. Select **Keycloak Client**.
5. Click on the automation-hub-client-secret client.
6. Select **YAML**.
7. Update the Client YAML to add the Valid Redirect URIs and Web Origins settings.

```
redirectUris:
  - 'https://private-ah-ansible-automation-platform.apps-crc.testing/*'
webOrigins:
  - 'https://private-ah-ansible-automation-platform.apps-crc.testing'
```

Field	Description
redirectURIs	This is the location determined in Determine Automation Hub Route. Be sure to add the /* to the end of the **redirectUris** setting.
webOrigins	This is the location determined in Determine Automation Hub Route.

NOTE

Ensure the indentation is correct when entering these settings.

8. Click **Save**.

To verify connectivity

1. Navigate to the automation hub route.
2. Enter the **hub_admin** user credentials and sign in.
3. Red Hat Single Sign-On processes the authentication and redirects back to automation hub.

6.10. ADDITIONAL RESOURCES

- For more information on running operators on OpenShift Container Platform, see Working with Operators in OpenShift Container Platform in the OpenShift Container Platform product documentation.