Red Hat Ansible Automation Platform 2.4

Deploying the Red Hat Ansible Automation Platform operator on OpenShift Container Platform

Install and configure Ansible Automation Platform operator on OpenShift Container Platform

Last Updated: 2024-10-10

Red Hat Ansible Automation Platform 2.4 Deploying the Red Hat Ansible Automation Platform operator on OpenShift Container Platform

Install and configure Ansible Automation Platform operator on OpenShift Container Platform

Abstract

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

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, migration and upgrade requirements for deploying the Ansible Automation Platform Operator on OpenShift Container Platform.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

If you have a suggestion to improve this documentation, or find an error, you can contact technical support at https://access.redhat.com to open a request.

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

Red Hat Ansible Automation Platform is supported on both Red Hat Enterprise Linux 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 2.4 is available on OpenShift Container Platform 4.9 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 {OperatorPlatform} 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 custom resources first, then automation hub custom resources;
- Automation hub custom resources first, then automation controller custom resources;
- Automation controller custom resources;
- Automation hub 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 ON RED HAT OPENSHIFT CONTAINER PLATFORM

Prerequisites

- You have installed the Red Hat Ansible Automation Platform catalog in OperatorHub.
- You have created a **StorageClass** object for your platform and a persistent volume claim (PVC) with **ReadWriteMany** access mode. See Dynamic provisioning for details.
- To run Red Hat OpenShift Container Platform clusters on Amazon Web Services (AWS) with **ReadWriteMany** access mode, you must add NFS or other storage.
  - For information about 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 begins. When installation finishes, a modal appears notifying you that the Ansible Automation Platform Operator is installed in the specified namespace.

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

IMPORTANT

You can only install a single instance of the Ansible Automation Platform Operator into a single namespace. Installing multiple instances in the same namespace can lead to improper operation for both operator instances.

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.

Automation controller configuration can be done through the automation controller extra_settings or directly in the user interface after deployment. However, it is important to note that configurations made in extra_settings take precedence over settings made in the user interface.

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.
- For Controller, a default StorageClass must be configured on the cluster for the operator to dynamically create needed PVCs. This is not necessary if an external PostgreSQL database is configured.
- For Hub a StorageClass that supports ReadWriteMany must be available on the cluster to dynamically created the PVC needed for the content, redis and api pods. If it is not the default StorageClass on the cluster, you can specify it when creating your AutomationHub object.

3.2. INSTALLING THE AUTOMATION CONTROLLER OPERATOR

Use this procedure to install the automation controller operator.

Procedure

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. Creating your automation controller form-view

Use this procedure to create your automation controller using the form-view.

Procedure

1. Ensure **Form view** is selected. It should be selected by default.

2. Enter the name of the new controller.
3. Optional: Add any labels necessary.
4. Click **Advanced configuration**.
5. Enter **Hostname** of the instance. The hostname is optional. The default hostname will be generated based upon the deployment name you have selected.
6. Enter the **Admin account username**.
7. Enter the **Admin email address**.
8. Under the **Admin password secret** drop-down menu, select the secret.
9. Under **Database configuration secret** drop-down menu, select the secret.
10. Under **Old Database configuration secret** drop-down menu, select the secret.
11. Under **Secret key secret** drop-down menu, select the secret.
12. Under **Broadcast Websocket Secret** drop-down menu, select the secret.
13. Enter any **Service Account Annotations** necessary.

3.2.2. Configuring your controller image pull policy

Use this procedure to configure the image pull policy on your automation controller.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Go to **Operators → Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Automation Controller** tab.
5. For new instances, click **Create AutomationController**.
    a. For existing instances, you can edit the YAML view by clicking the ⋮ icon and then **Edit AutomationController**.
6. Click **advanced Configuration**. Under **Image Pull Policy**, click on the radio button to select
    - **Always**
    - **Never**
    - **IfNotPresent**
7. To display the option under **Image Pull Secrets**, click the arrow.
    a. Click **+** beside **Add Image Pull Secret** and enter a value.
8. To display fields under the **Web container resource requirements** drop-down list, click the arrow.

   a. Under **Limits**, and **Requests**, enter values for **CPU cores**, **Memory**, and **Storage**.

9. To display fields under the **Task container resource requirements** drop-down list, click the arrow.
   a. Under **Limits**, and **Requests**, enter values for **CPU cores**, **Memory**, and **Storage**.
10. To display fields under the **EE Control Plane container resource requirements** drop-down list, click the arrow.
    a. Under **Limits**, and **Requests**, enter values for **CPU cores**, **Memory**, and **Storage**.
11. To display fields under the **PostgreSQL init container resource requirements (when using a managed service)** drop-down list, click the arrow.
    a. Under **Limits**, and **Requests**, enter values for **CPU cores**, **Memory**, and **Storage**.
12. To display fields under the **Redis container resource requirements** drop-down list, click the arrow.
    a. Under **Limits**, and **Requests**, enter values for **CPU cores**, **Memory**, and **Storage**.
13. To display fields under the **PostgreSQL container resource requirements (when using a managed instance)*** drop-down list, click the arrow.
    a. Under **Limits**, and **Requests**, enter values for **CPU cores**, **Memory**, and **Storage**.
14. To display the **PostgreSQL container storage requirements (when using a managed instance)** drop-down list, click the arrow.
    a. Under **Limits**, and **Requests**, enter values for **CPU cores**, **Memory**, and **Storage**.
15. Under Replicas, enter the number of instance replicas.
16. Under **Remove used secrets on instance removal**, select **true** or **false**. The default is false.
17. Under **Preload instance with data upon creation**, select **true** or **false**. The default is true.

3.2.3. Configuring your controller LDAP security

Use this procedure to configure LDAP security for your automation controller.

Procedure

1. If you do not have a **ldap_cacert_secret**, you can create one with the following command:

```
$ oc create secret generic <resourcename>-custom-certs \
    --from-file=ldap-ca.crt=<PATH/TO/YOUR/CA/PEM/FILE>  \ (1)
```

(1) Modify this to point to where your CA cert is stored.

This will create a secret that looks like this:

```
$ oc get secret/mycerts -o yaml
apiVersion: v1
data:
  ldap-ca.crt: <mysecret> 1
kind: Secret
metadata:
  name: mycerts
  namespace: awx
type: Opaque
```

1 Automation controller looks for the data field **ldap-ca.crt** in the specified secret when using the **ldap_cacert_secret**.

2. Under **LDAP Certificate Authority Trust Bundle** click the drop-down menu and select your **ldap_cacert_secret**.
3. Under **LDAP Password Secret**, click the drop-down menu and select a secret.
4. Under **EE Images Pull Credentials Secret**, click the drop-down menu and select a secret.
5. Under **Bundle Cacert Secret**, click the drop-down menu and select a secret.
6. Under **Service Type**, click the drop-down menu and select
   - **ClusterIP**
   - **LoadBalancer**
   - **NodePort**

3.2.4. Configuring 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**.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators → Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Automation Controller** tab.
5. For new instances, click **Create AutomationController**.
   a. For existing instances, you can edit the YAML view by clicking the ⋮ icon and then **Edit AutomationController**.
6. Click **Advanced configuration**.
7. Under **Ingress type**, click the drop-down menu and select **Route**.
8. Under **Route DNS host**, enter a common host name that the route answers to.
9. Under **Route TLS termination mechanism**, click the drop-down menu and select **Edge** or **Passthrough**. For most instances **Edge** should be selected.

10. Under **Route TLS credential secret**, click the drop-down menu and select a secret from the list.

11. Under **Enable persistence for */var/lib/projects* directory** select either true or false by moving the slider.

3.2.5. Configuring the Ingress type for your automation controller operator

The Ansible Automation Platform Operator installation form allows you to further configure your automation controller operator ingress under **Advanced configuration**.

Procedure

1. Log in to Red Hat OpenShift Container Platform.

2. Navigate to **Operators → Installed Operators**.

3. Select your Ansible Automation Platform Operator deployment.

4. Select the **Automation Controller** tab.

5. For new instances, click **Create AutomationController**.

    a. For existing instances, you can edit the YAML view by clicking the ⋮ icon and then **Edit AutomationController**.

6. Click **Advanced configuration**.

7. Under **Ingress type**, click the drop-down menu and select **Ingress**.

8. Under **Ingress annotations**, enter any annotations to add to the ingress.

9. Under **Ingress TLS secret**, click the drop-down menu and select a secret from the list.

After 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.

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 controller 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.

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 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 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 2.4 supports PostgreSQL 13.

Procedure

The external postgres instance credentials and connection information must be stored in a secret, which is then 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 want 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: automationcontroller.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>
   ```

3.5. 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.

Automation hub configuration can be done through the automation hub pulp_settings or directly in the user interface after deployment. However, it is important to note that configurations made in pulp_settings take precedence over settings made in the user interface. Hub settings should always be set as lowercase on the Hub custom resource specification.

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 Ansible Automation Platform Operator in Operator Hub.

4.2. INSTALLING THE AUTOMATION HUB OPERATOR

Use this procedure to install the automation hub operator.

Procedure

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

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.

The process for configuring object storage on the **AutomationHub** CR is similar for Amazon S3 and Azure Blob Storage.

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

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).

4.2.1.2. Configuring object storage on Amazon S3

Red Hat supports Amazon Simple Storage Service (S3) for automation hub. You can configure it when deploying the **AutomationHub** custom resource (CR), or you can configure it for an existing instance.

Prerequisites

- Create an Amazon S3 bucket to store the objects.
- Note the name of the S3 bucket.

Procedure

1. Create a Kubernetes secret containing the AWS credentials and connection details, and the name of your Amazon S3 bucket. The following example creates a secret called **test-s3**:

```
$ oc -n $HUB_NAMESPACE apply -f- <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: 'test-s3'
stringData:
  s3-access-key-id: $S3_ACCESS_KEY_ID
  s3-secret-access-key: $S3_SECRET_ACCESS_KEY
  s3-bucket-name: $S3_BUCKET_NAME
  s3-region: $S3_REGION
EOF
```

2. Add the secret to the automation hub custom resource (CR) **spec**:

```
spec:
  object_storage_s3_secret: test-s3
```

3. If you are applying this secret to an existing instance, restart the API pods for the change to take effect. **<hub-name>** is the name of your hub instance.

```
$ oc -n $HUB_NAMESPACE delete pod -l app.kubernetes.io/name=<hub-name>-api
```

4.2.1.3. Configuring object storage on Azure Blob

Red Hat supports Azure Blob Storage for automation hub. You can configure it when deploying the **AutomationHub** custom resource (CR), or you can configure it for an existing instance.

Prerequisites

- Create an Azure Storage blob container to store the objects.
- Note the name of the blob container.

Procedure

1. Create a Kubernetes secret containing the credentials and connection details for your Azure account, and the name of your Azure Storage blob container. The following example creates a secret called **test-azure**:

   ```
   $ oc -n $HUB_NAMESPACE apply -f- <<EOF
   apiVersion: v1
   kind: Secret
   metadata:
     name: 'test-azure'
   stringData:
     azure-account-name: $AZURE_ACCOUNT_NAME
     azure-account-key: $AZURE_ACCOUNT_KEY
     azure-container: $AZURE_CONTAINER
     azure-container-path: $AZURE_CONTAINER_PATH
     azure-connection-string: $AZURE_CONNECTION_STRING
   EOF
   ```

2. Add the secret to the automation hub custom resource (CR) **spec**:

   ```
   spec:
     object_storage_azure_secret: test-azure
   ```

3. If you are applying this secret to an existing instance, restart the API pods for the change to take effect. **<hub-name>** is the name of your hub instance.

```
$ oc -n $HUB_NAMESPACE delete pod -l app.kubernetes.io/name=<hub-name>-api
```

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**.

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators → Installed Operators**.

3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Automation Hub** tab.
5. For new instances, click **Create AutomationHub**.
    a. For existing instances, you can edit the YAML view by clicking the ⋮ icon and then **Edit AutomationHub**.
6. Click **Advanced configuration**.
7. Under **Ingress type**, click the drop-down menu and select **Route**.
8. Under **Route DNS host**, enter a common host name that the route answers to.
9. Under **Route TLS termination mechanism**, click the drop-down menu and select **Edge** or **Passthrough**.
10. Under **Route TLS credential secret**, click the drop-down menu and select a secret from the list.

4.2.3. Configuring the Ingress type for your automation hub operator

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

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operators → Installed Operators**.
3. Select your Ansible Automation Platform Operator deployment.
4. Select the **Automation Hub** tab.
5. For new instances, click **Create AutomationHub**.
    a. For existing instances, you can edit the YAML view by clicking the ⋮ icon and then **Edit AutomationHub**.
6. Click **Advanced Configuration**.
7. Under **Ingress type**, click the drop-down menu and select **Ingress**.
8. Under **Ingress annotations**, enter any annotations to add to the ingress.
9. 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.

4.3. CONFIGURING LDAP AUTHENTICATION FOR ANSIBLE AUTOMATION HUB ON OPENSHIFT CONTAINER PLATFORM

Configure LDAP authentication settings for Ansible Automation Platform on OpenShift Container Platform in the spec section of your Hub instance configuration file.

Procedure

- Use the following example to configure LDAP in your automation hub instance. For any blank fields, enter ``.

```
spec:
  pulp_settings:
    auth_ldap_user_attr_map:
      email: "mail"
      first_name: "givenName"
      last_name: "sn"
    auth_ldap_group_search_base_dn: 'cn=groups,cn=accounts,dc=example,dc=com'
    auth_ldap_bind_dn: ' '
    auth_ldap_bind_password: ' '
    auth_ldap_group_search_filter: (objectClass=posixGroup)
    auth_ldap_user_search_scope: SUBTREE
    auth_ldap_server_uri: 'ldap://ldapserver:389'
    authentication_backend_preset: ldap
    auth_ldap_mirror_groups: 'True'
    auth_ldap_user_search_base_dn: 'cn=users,cn=accounts,dc=example,dc=com'
    auth_ldap_bind_password: 'ldappassword'
    auth_ldap_user_search_filter: (uid=%(user)s)
    auth_ldap_group_search_scope: SUBTREE
    auth_ldap_user_flags_by_group: '@json {"is_superuser": "cn=tower-
admin,cn=groups,cn=accounts,dc=example,dc=com"}'
```

NOTE

Do not leave any fields empty. For fields with no variable, enter `` to indicate a default value.

4.4. ACCESSING THE AUTOMATION HUB USER INTERFACE

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

Procedure

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.

4.5. 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 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 2.4 supports PostgreSQL 13.

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

4.5.1. Enabling the hstore extension for the automation hub PostgreSQL database

From Ansible Automation Platform 2.4, the database migration script uses **hstore** fields to store information, therefore the **hstore** extension to the automation hub PostgreSQL database must be enabled.

This process is automatic when using the Ansible Automation Platform installer and a managed PostgreSQL server.

If the PostgreSQL database is external, you must enable the **hstore** extension to the automation hub PostreSQL database manually before automation hub installation.

If the **hstore** extension is not enabled before automation hub installation, a failure is raised during database migration.

Procedure

1. Check if the extension is available on the PostgreSQL server (automation hub database).

   ```
   $ psql -d <automation hub database> -c "SELECT * FROM pg_available_extensions WHERE name='hstore'"
   ```

   Where the default value for **<automation hub database>** is **automationhub**.

   Example output with hstore available:

   ```
   name  | default_version | installed_version |comment
   ------+-----------------+-------------------+--------------------------------------------------
    hstore | 1.7           |                   | data type for storing sets of (key, value) pairs
   (1 row)
   ```

   Example output with hstore not available:

   ```
    name | default_version | installed_version | comment
   ------+-----------------+-------------------+---------
   (0 rows)
   ```

2. On a RHEL based server, the **hstore** extension is included in the **postgresql-contrib** RPM package, which is not installed automatically when installing the PostgreSQL server RPM package.
   To install the RPM package, use the following command:

   ```
   dnf install postgresql-contrib
   ```

3. Create the **hstore** PostgreSQL extension on the automation hub database with the following command:

   ```
   $ psql -d <automation hub database> -c "CREATE EXTENSION hstore;"
   ```

The output of which is:

```
CREATE EXTENSION
```

4. In the following output, the **installed_version** field contains the **hstore** extension used, indicating that **hstore** is enabled.

```
name | default_version | installed_version | comment
-----+-----------------+-------------------+------------------------------------------------------
hstore |     1.7      |       1.7         | data type for storing sets of (key, value) pairs
(1 row)
```

4.6. 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>
```

4.7. ADDITIONAL CONFIGURATIONS

A collection download count can help you understand collection usage. To add a collection download count to automation hub, set the following configuration:

```
spec:
  pulp_settings:
    ansible_collect_download_count: true
```

When **ansible_collect_download_count** is enabled, automation hub will display a download count by the collection.

4.8. 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 RED HAT 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

Use this procedure to subscribe a namespace to an operator.

IMPORTANT

You can only subscribe a single instance of the Ansible Automation Platform Operator into a single namespace. Subscribing multiple instances in the same namespace can lead to improper operation for both operator instances.

Procedure

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-2.4'
  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. DEPLOYING EVENT-DRIVEN ANSIBLE CONTROLLER WITH RED HAT ANSIBLE AUTOMATION PLATFORM OPERATOR ON RED HAT OPENSHIFT CONTAINER PLATFORM

Event-Driven Ansible controller is the interface for event-driven automation and introduces automated resolution of IT requests. This component helps you connect to sources of events and acts on those events using rulebooks. When you deploy Event-Driven Ansible controller, you can automate decision making, use numerous event sources, implement event-driven automation within and across multiple IT use cases, and achieve more efficient service delivery.

Use the following instructions to install Event-Driven Ansible with your Ansible Automation Platform Operator on OpenShift Container Platform.

Prerequisites

- You have installed Ansible Automation Platform Operator on OpenShift Container Platform.
- You have installed and configured automation controller.

Procedure

1. Select **Operators → Installed Operators**.
2. Locate and select your installation of Ansible Automation Platform.
3. Under the **Details** tab, locate the **EDA** modal and click **Create instance**.
4. Click **Form view**, and in the **Name** field, enter the name you want for your new Event-Driven Ansible controller deployment.

   IMPORTANT

   If you have installed other Ansible Automation Platform components in your current OpenShift Container Platform namespace, ensure that you provide a unique name for your Event-Driven Ansible controller when you create your Event-Driven Ansible custom resource. Otherwise, naming conflicts can occur and impact Event-Driven Ansible controller deployment.

5. Specify your controller URL in the **Automation Server URL** field.
   If you deployed automation controller in Openshift as well, you can find the URL in the navigation panel under **Networking → Routes**.

   NOTE

   This is the only required customization, but you can customize other options using the UI form or directly in the YAML configuration tab, if desired.

IMPORTANT

To ensure that you can run concurrent Event-Driven Ansible activations efficiently, you must set your maximum number of activations in proportion to the resources available on your cluster. You can do this by adjusting your Event-Driven Ansible settings in the YAML view.

When you activate an Event-Driven Ansible rulebook under standard conditions, it uses approximately 250 MB of memory. However, the actual memory consumption can vary significantly based on the complexity of your rules and the volume and size of the events processed. In scenarios where a large number of events are anticipated or the rulebook complexity is high, conduct a preliminary assessment of resource usage in a staging environment. This ensures that your maximum number of activations is based on the capacity of your resources.

6. Click **YAML view** to update your YAML key values.
7. Copy and paste the following string at the end of the **spec** key value section:

```
extra_settings:
  - setting: EDA_MAX_RUNNING_ACTIVATIONS
    value: '12'
```

8. Click **Create**. This deploys Event-Driven Ansible controller in the namespace you specified. After a couple minutes when the installation is marked as **Successful**, you can find the URL for the Event-Driven Ansible UI on the **Routes** page in the OpenShift UI.
9. From the navigation panel, select **Networking → Routes** to find the new Route URL that has been created for you.
   Routes are listed according to the name of your custom resource.
10. Click the new URL under the **Location** column to navigate to Event-Driven Ansible in the browser.
11. From the navigation panel, select **Workloads → Secrets** and locate the Admin Password k8s secret that was created for you, unless you specified a custom one.
    Secrets are listed according to the name of your custom resource and appended with **-admin-password.**

NOTE

You can use the password value in the secret to log in to the Event-Driven Ansible controller UI. The default user is **admin**.

CHAPTER 7. 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.

7.1. 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.

7.2. 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: Ansible Automation Platform
  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.

7.3. 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 response 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 automation hub
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:  'automation-hub'
        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: 'automation-hub'
      protocol: openid-connect
      name: audience mapper
      protocolMapper: oidc-audience-mapper
roles:
  - name: "hubadmin"
    description: "An administrator role for automation hub"
```

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.

7.4. 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.

7.5. INSTALLING THE ANSIBLE AUTOMATION PLATFORM OPERATOR

Procedure

1. Log in to Red Hat OpenShift Container Platform.
2. Navigate to **Operator → Operator Hub**.
3. Search for the Ansible Automation Platform Operator.
4. Select the Ansible Automation Platform Operator project.
5. Click on the Operator tile.
6. Click **Install**.
7. 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**.
8. Click **Install**.
9. When the Operator has been installed, click **View Operator**.

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

Use this procedure to create a connection secret for Red Hat Single Sign-On.

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.

7.7. INSTALLING AUTOMATION HUB USING THE ANSIBLE AUTOMATION PLATFORM OPERATOR

Use the following procedure to install automation hub using the Ansible Automation Platform Operator.

Procedure

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

```
apiVersion: automationhub.ansible.com/v1beta1
kind: AutomationHub
metadata:
  name: private-ah                  1
  namespace: aap
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
  replicas: 1                       3
```

```
web_replicas: N
task_replicas: N
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.

3 Scale replicas up or down for each deployment by using the **web_replicas** or **task_replicas** respectively, where N represents the number of replicas you want to create. Alternatively, you can scale all pods across both deployments by using **replicas**. See Scaling the Web and Task Pods independently for 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.

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

7.8. 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.

7.9. 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.

7.10. 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.

CHAPTER 8. MIGRATING RED HAT ANSIBLE AUTOMATION PLATFORM TO RED HAT ANSIBLE AUTOMATION PLATFORM OPERATOR

Migrating your Red Hat Ansible Automation Platform deployment to the Ansible Automation Platform Operator allows you to take advantage of the benefits provided by a Kubernetes native operator, including simplified upgrades and full lifecycle support for your Red Hat Ansible Automation Platform deployments.

Use these procedures to migrate any of the following deployments to the Ansible Automation Platform Operator:

- A VM-based installation of Ansible Tower 3.8.6, automation controller, or automation hub
- An Openshift instance of Ansible Tower 3.8.6 (Ansible Automation Platform 1.2)

8.1. MIGRATION CONSIDERATIONS

If you are upgrading from Ansible Automation Platform 1.2 on OpenShift Container Platform 3 to Ansible Automation Platform 2.x on OpenShift Container Platform 4, you must provision a fresh OpenShift Container Platform version 4 cluster and then migrate the Ansible Automation Platform to the new cluster.

8.2. PREPARING FOR MIGRATION

Before migrating your current Ansible Automation Platform deployment to Ansible Automation Platform Operator, you need to back up your existing data, create k8s secrets for your secret key and postgresql configuration.

NOTE

If you are migrating both automation controller and automation hub instances, repeat the steps in Creating a secret key secret and Creating a postgresql configuration secret for both and then proceed to Migrating data to the Ansible Automation Platform Operator .

8.2.1. Migrating to Ansible Automation Platform Operator

Prerequisites

To migrate Ansible Automation Platform deployment to Ansible Automation Platform Operator, you must have the following:

- Secret key secret
- Postgresql configuration
- Role-based Access Control for the namespaces on the new OpenShift cluster
- The new OpenShift cluster must be able to connect to the previous PostgreSQL database

NOTE

You can store the secret key information in the inventory file before the initial Red Hat Ansible Automation Platform installation. If you are unable to remember your secret key or have trouble locating your inventory file, contact Ansible support through the Red Hat Customer portal.

Before migrating your data from Ansible Automation Platform 2.x or earlier, you must back up your data for loss prevention. To backup your data, do the following:

Procedure

1. Log in to your current deployment project.
2. Run **setup.sh** to create a backup of your current data or deployment:
   For on-prem deployments of version 2.x or earlier:

   ```
   $ ./setup.sh -b
   ```

   For OpenShift deployments before version 2.0 (non-operator deployments):

   ```
   ./setup_openshift.sh -b
   ```

8.2.2. Creating a secret key secret

To migrate your data to Ansible Automation Platform Operator on OpenShift Container Platform, you must create a secret key that matches the secret key defined in the inventory file during your initial installation. Otherwise, the migrated data will remain encrypted and unusable after migration.

Procedure

1. Locate the old secret key in the inventory file you used to deploy Ansible Automation Platform in your previous installation.
2. Create a yaml file for your secret key:

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: <resourcename>-admin-password
     namespace: <target-namespace>
   stringData:
     password: mysuperlongpassword
   type: Opaque
   ```

   NOTE

   If admin_password_secret is not provided, the operator will look for a secret named <resourcename>-admin-password for the admin password. If it is not present, the operator will generate a password and create a Secret from it named <resourcename>-admin-password.

3. Apply the secret key yaml to the cluster:

```
oc apply -f <secret-key.yml>
```

8.2.3. Creating a postgresql configuration secret

For migration to be successful, you must provide access to the database for your existing deployment.

Procedure

1. Create a yaml file for your postgresql configuration secret:

```
apiVersion: v1
kind: Secret
metadata:
  name: <resourcename>-old-postgres-configuration
  namespace: <target namespace>
stringData:
  host: "<external ip or url resolvable by the cluster>"
  port: "<external port, this usually defaults to 5432>"
  database: "<desired database name>"
  username: "<username to connect as>"
  password: "<password to connect with>"
type: Opaque
```

2. Apply the postgresql configuration yaml to the cluster:

```
oc apply -f <old-postgres-configuration.yml>
```

8.2.4. Verifying network connectivity

To ensure successful migration of your data, verify that you have network connectivity from your new operator deployment to your old deployment database.

Prerequisites

Take note of the host and port information from your existing deployment. This information is located in the postgres.py file located in the conf.d directory.

Procedure

1. Create a yaml file to verify the connection between your new deployment and your old deployment database:

```
apiVersion: v1
kind: Pod
metadata:
    name: dbchecker
spec:
  containers:
    - name: dbchecker
      image: registry.redhat.io/rhel8/postgresql-13:latest
      command: ["sleep"]
      args: ["600"]
```

2. Apply the connection checker yaml file to your new project deployment:

```
oc project ansible-automation-platform
oc apply -f connection_checker.yaml
```

3. Verify that the connection checker pod is running:

```
oc get pods
```

4. Connect to a pod shell:

```
oc rsh dbchecker
```

5. After the shell session opens in the pod, verify that the new project can connect to your old project cluster:

```
pg_isready -h <old-host-address> -p <old-port-number> -U awx
```

Example

```
<old-host-address>:<old-port-number> - accepting connections
```

8.3. MIGRATING DATA TO THE ANSIBLE AUTOMATION PLATFORM OPERATOR

After you have set your secret key, postgresql credentials, verified network connectivity and installed the Ansible Automation Platform Operator, you must create a custom resource controller object before you can migrate your data.

8.3.1. Creating an AutomationController object

Use the following steps to create an AutomationController custom resource object.

Procedure

1. Log in to **Red Hat OpenShift Container Platform**.
2. Navigate to **Operators → Installed Operators**.
3. Select the Ansible Automation Platform Operator installed on your project namespace.
4. Select the **Automation Controller** tab.
5. Click **Create AutomationController**.
6. Enter a name for the new deployment.
7. In **Advanced configurations**, do the following:
    a. From the **Admin Password Secret** list, select your secret key secret.
    b. From the **Database Configuration Secret** list, select the postgres configuration secret.

8. Click **Create**.

8.3.2. Creating an AutomationHub object

Use the following steps to create an AutomationHub custom resource object.

Procedure

1. Log in to **Red Hat OpenShift Container Platform**
2. Navigate to **Operators** → **Installed Operators**.
3. Select the Ansible Automation Platform Operator installed on your project namespace.
4. Select the **Automation Hub** tab.
5. Click **Create AutomationHub**.
6. Enter a name for the new deployment.
7. In **Advanced configurations**, select your secret key secret and postgres configuration secret.
8. Click **Create**.

8.4. POST MIGRATION CLEANUP

After your data migration is complete, you must delete any Instance Groups that are no longer required.

Procedure

1. Log in to Red Hat Ansible Automation Platform as the administrator with the password you created during migration.

   NOTE

   Note: If you did not create an administrator password during migration, 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 Red Hat Ansible Automation Platform password field.

2. Select **Administration** → **Instance Groups**.
3. Select all Instance Groups except controlplane and default.
4. Click **Delete**.

CHAPTER 9. UPGRADING RED HAT ANSIBLE AUTOMATION PLATFORM OPERATOR ON OPENSHIFT CONTAINER PLATFORM

The Ansible Automation Platform Operator simplifies the installation, upgrade and deployment of new Red Hat Ansible Automation Platform instances in your OpenShift Container Platform environment.

9.1. UPGRADE CONSIDERATIONS

Red Hat Ansible Automation Platform version 2.0 was the first release of the Ansible Automation Platform Operator. If you are upgrading from version 2.0, continue to the Upgrading the Ansible Automation Platform Operator procedure.

If you are using a version of OpenShift Container Platform that is not supported by the version of Red Hat Ansible Automation Platform to which you are upgrading, you must upgrade your OpenShift Container Platform cluster to a supported version before upgrading.

Refer to the Red Hat Ansible Automation Platform Life Cycle to determine the OpenShift Container Platform version needed.

For information about upgrading your cluster, refer to Updating clusters.

9.2. PREREQUISITES

To upgrade to a newer version of Ansible Automation Platform Operator, it is recommended that you do the following:

- Create AutomationControllerBackup and AutomationHubBackup objects. For help with this see Creating Red Hat Ansible Automation Platform backup resources
- Review the release notes for the new Ansible Automation Platform version to which you are upgrading and any intermediate versions.

9.3. UPGRADING THE ANSIBLE AUTOMATION PLATFORM OPERATOR

To upgrade to the latest version of Ansible Automation Platform Operator on OpenShift Container Platform, do the following:

Prodedure

1. Log in to OpenShift Container Platform.
2. Navigate to **Operators → Installed Operators**.
3. Select the **Subscriptions** tab.
4. Under **Upgrade status**, click **Upgrade Available**.
5. Click **Preview InstallPlan**.
6. Click **Approve**.

CHAPTER 10. ADDING EXECUTION NODES TO RED HAT ANSIBLE AUTOMATION PLATFORM OPERATOR

You can enable the Ansible Automation Platform Operator with execution nodes by downloading and installing the install bundle.

Prerequisites

- An automation controller instance.
- The receptor collection package is installed.
- AAP Repository **ansible-automation-platform-2.4-for-rhel-{RHEL-RELEASE-NUMBER}-x86_64-rpms** is enabled.

Procedure

1. Log in to Red Hat Ansible Automation Platform.
2. In the navigation panel, select **Administration** → **Instances**.
3. Click **Add**.
4. Input the Execution Node domain name or IP in the **Host Name** field.
5. Optional: Input the port number in the **Listener Port** field.
6. Click **Save**.
7. Click the download icon next to **Install Bundle**. This starts a download, take note of where you save the file
8. Untar the gz file.

   NOTE

   To run the **install_receptor.yml** playbook you need to install the receptor collection from Ansible Galaxy: **Ansible-galaxy collection install -r requirements.yml**

9. Update the playbook with your user name and SSH private key file. Note that **ansible_host** pre-populates with the hostname you input earlier.

```
all:
  hosts:
    remote-execution:
      ansible_host: example_host_name # Same with configured in AAP WebUI
      ansible_user: <username> #user provided
      Ansible_ssh_private_key_file: ~/.ssh/id_example
```

10. Open your terminal, and navigate to the directory where you saved the playbook.
11. To install the bundle run:

```
ansible-playbook install_receptor.yml -i inventory.yml
```

12. When installed you can now upgrade your execution node by downloading and re-running the playbook for the instance you created.

Verification

To verify receptor service status run the following command:

```
sudo systemctl status receptor.service
```

Make sure the service is in **active (running)** state

To verify if your playbook runs correctly on your new node run the following command:

```
watch podman ps
```

Additional resources

- For more information about managing instance groups see the Managing Instance Groups section of the Automation Controller User Guide.

CHAPTER 11. ANSIBLE AUTOMATION PLATFORM RESOURCE OPERATOR

11.1. RESOURCE OPERATOR OVERVIEW

Resource Operator is a custom resource (CR) that you can deploy after you have created your automation controller deployment. With Resource Operator you can define projects, job templates, and inventories through the use of YAML files. These YAML files are then used by automation controller to create these resources. You can create the YAML through the **Form view** that prompts you for keys and values for your YAML code. Alternatively, to work with YAML directly, you can select **YAML view**.

There are currently two custom resources provided by the Resource Operator:

- AnsibleJob: launches a job in the automation controller instance specified in the Kubernetes secret (automation controller host URL, token).
- JobTemplate: creates a job template in the automation controller instance specified.

11.2. USING RESOURCE OPERATOR

The Resource Operator itself does not do anything until the user creates an object. As soon as the user creates an **AutomationControllerProject** or **AnsibleJob** resource, the Resource Operator will start processing that object.

Prerequisites

- Install the Kubernetes-based cluster of your choice.
- Deploy automation controller using the **automation-controller-operator**.

After installing the **automation-controller-resource-operator** in your cluster, you must create a Kubernetes (k8s) secret with the connection information for your automation controller instance. Then you can use Resource Operator to create a k8s resource to manage your automation controller instance.

11.3. CONNECTING RESOURCE OPERATOR TO AUTOMATION CONTROLLER

To connect Resource Operator with automation controller you need to create a k8s secret with the connection information for your automation controller instance.

Procedure

To create an OAuth2 token for your user in the automation controller UI:

1. In the navigation panel, select **Access → Users**.
2. Select the username you want to create a token for.
3. Click on **Tokens**, then click **Add**.
4. You can leave **Applications** empty. Add a description and select **Read** or **Write** for the **Scope**.

Alternatively, you can create a OAuth2 token at the command-line by using the **create_oauth2_token** manage command:

```
$ controller-manage create_oauth2_token --user example_user
New OAuth2 token for example_user: j89ia8OO79te6IAZ97L7E8bMgXCON2
```

NOTE

Make sure you provide a valid user when creating tokens. Otherwise, you will get an error message that you tried to issue the command without specifying a user, or supplying a username that does not exist.

11.4. CREATING A AUTOMATION CONTROLLER CONNECTION SECRET FOR RESOURCE OPERATOR

To make your connection information available to the Resource Operator, create a k8s secret with the token and host value.

Procedure

1. The following is an example of the YAML for the connection secret. Save the following example to a file, for example, **automation-controller-connection-secret.yml**.

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: controller-access
     type: Opaque
   stringData:
     token: <generated-token>
     host: https://my-controller-host.example.com/
   ```

2. Edit the file with your host and token value.

3. Apply it to your cluster by running the **kubectl create** command:

```
kubectl create -f controller-connection-secret.yml
```

11.5. CREATING AN ANSIBLEJOB

Launch an automation job on automation controller by creating an AnsibleJob resource.

Procedure

1. Specify the connection secret and job template you want to launch.

   ```
   apiVersion: tower.ansible.com/v1alpha1
   kind: AnsibleJob
   metadata:
     generateName: demo-job-1 # generate a unique suffix per 'kubectl create'
   ```

```
spec:
  connection_secret: controller-access
  job_template_name: Demo Job Template
```

2. Configure features such as, inventory, extra variables, and time to live for the job.

```
spec:
  connection_secret: controller-access
  job_template_name: Demo Job Template
  inventory: Demo Inventory                    # Inventory prompt on launch needs to be enabled
  runner_image: quay.io/ansible/controller-resource-runner
  runner_version: latest
  job_ttl: 100
  extra_vars:                                  # Extra variables prompt on launch needs to be enabled
     test_var: test
  job_tags: "provision,install,configuration"  # Specify tags to run
  skip_tags: "configuration,restart"           # Skip tasks with a given tag
```

NOTE

You must enable prompt on launch for inventories and extra variables if you are configuring those. To enable **Prompt on launch**, within the automation controller UI: From the **Resources → Templates** page, select your template and select the **Prompt on launch** checkbox next to **Inventory** and **Variables** sections.

3. Launch a workflow job template with an AnsibleJob object by specifying the **workflow_template_name** instead of **job_template_name**:

```
apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleJob
metadata:
  generateName: demo-job-1 # generate a unique suffix per 'kubectl create'
spec:
  connection_secret: controller-access
  workflow_template_name: Demo Workflow Template
```

11.6. CREATING A JOBTEMPLATE

- Create a job template on automation controller by creating a JobTemplate resource:

```
apiVersion: tower.ansible.com/v1alpha1
kind: JobTemplate
metadata:
  name: jobtemplate-4
spec:
  connection_secret: controller-access
  job_template_name: ExampleJobTemplate4
  job_template_project: Demo Project
  job_template_playbook: hello_world.yml
  job_template_inventory: Demo Inventory
```