Red Hat

Secure

Ansible Automation Platform 2.6

May 12, 2026

Contents

1 Secure

Configure central authentication for Ansible Automation Platform

Configure authentication methods such as LDAP or SAML to simplify the user login experience. Providing the correct connection details for your chosen identity provider helps ensure seamless and secure access to Ansible Automation Platform.

Configuring authentication involves the following procedures:

- Selecting an authentication type, where you select the type of authenticator plugin you want to configure, including the authentication details for the authentication type selected.
- Mapping, where you define mapping rule types and triggers to control access to the system, and mapping order where you can define the mapping precedence.

IMPORTANT:

- During an upgrade to Ansible Automation Platform 2.6, platform gateway uses a new central authentication service.
- After the upgrade, local users that used to exist in automation controller can be automatically converted into local platform gateway users. Other types of authentication from automation controller, such as LDAP, SAML, or OIDC, are migrated to platform gateway but platform gateway might need additional configuration before those users are ready for use.

Local user passwords are not automatically synchronized between automation controller and platform gateway after an upgrade. Platform gateway uses the following process to authenticate a local user for the first time:

- Platform gateway attempts to authenticate the user with the platform gateway password.
- If the attempt fails, platform gateway authenticates the user with the automation controller password.
- On successful authentication, platform gateway updates the user's password in its database.
- The user is authenticated directly by platform gateway on subsequent logins.

Prerequisites

- A running installation of Ansible Automation Platform 2.6 or later
- A running instance of your authentication source

- Administrator rights to the Ansible Automation Platform
- Any connection information needed to connect Ansible Automation Platform 2.6 to your source (see individual authentication types for details).

Pluggable authentication

Authentication verifies a user's identity to Red Hat Ansible Automation Platform. While users can authenticate through a username and password, configuring external sources like LDAP, SAML, or OIDC enables a single sign-on (SSO) experience using existing enterprise credentials.

NOTE:
When you log out of Ansible Automation Platform, only your session with the platform ends. Your session with the external Single Sign-On (SSO) provider stays active. To switch to a different account with the same provider, you must log out of the SSO provider's website directly. This ensures that you can successfully sign in with a new account.

Ansible Automation Platform uses a pluggable authentication system with a configuration wizard that provides a common, simplified method of configuring different types of authenticators such as LDAP and SAML. The pluggable system also allows you to configure multiple authenticators of the same type.

In the pluggable system we have a couple of concepts:

Authenticator Plugin

A plugin allows Ansible Automation Platform to connect to a source system, such as, LDAP, or SAML. Ansible Automation Platform includes a variety of authenticator plugins. Authenticator plugins are similar to Ansible collections, in that all of the required code is in a package and can be versioned independently if needed.

Authenticator

An authenticator is an instantiation of an authenticator plugin and allows users from the specified source to log in. For example, the LDAP authenticator plugin defines a required LDAP server setting. When you instantiate an authenticator from the LDAP authentication plugin, you must provide the authenticator the LDAP server URL it needs to connect to.

Authenticator Map

Authenticator maps are applied to authenticators and tell Ansible Automation Platform what permissions to give a user logging into the system.

Configure authentication

Configure authentication by connecting an identity provider to Ansible Automation Platform and then configuring it to fine tune user access.

Configuring authentication involves the following procedures:

- Selecting an authentication type, where you select the type of authenticator plugin you want to configure, including the authentication details for the authentication type selected.
- Mapping, where you define mapping rule types and triggers to control access to the system, and mapping order where you can define the mapping precedence.

> **NOTE:**
> Mapping order is only available if you have defined one or more authenticator maps.

Related information
Select an authentication type
Define authentication mapping rules and triggers
Adjust the mapping order

Enable and disable the local authenticator

As a platform administrator, you can enable or disable authenticators. However, disabling your local authenticator can have significant impacts and should only be done under specific circumstances. Before you disable your local authenticator, you must consider the following:

Before you begin

- You have at least one other authenticator method configured.
- You have at least one administrator account that can authenticate using your alternate authenticator.

> **CAUTION:**
> Disabling the local authenticator without an alternative authentication in place can result in a locked environment.

Local account inaccessibility

Disabling the local authenticator prevents all local accounts, including the default `admin` account from logging in.

Potential inaccessibility

Disabling the local authenticator without having at least one other configured authenticator can render the Ansible Automation Platform environment completely inaccessible.

Dependency on enterprise authentication provider

If the local authenticator is disabled and an issue occurs with the configured enterprise authentication provider, the platform will become inaccessible until the enterprise authentication provider issue is resolved.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Ensure that at least one other authenticator type is configured and enabled.
3. Select your **Local Authenticator**.
4. Toggle the **Enabled** switch to the off position to disable the local authenticator.

If the local authenticator is disabled without another authentication method configured, or if an issue arises with your configured enterprise authentication provider, making the Ansible Automation Platform inaccessible, you can re-enable the local authenticator from the command line as follows:

1. List the available authenticators and retrieve the ID of your local authenticator by running the following command:

```
aap-gateway-api authenticators --list
```

2. Enable the local authenticator using its ID:

```
aap-gateway-manage authenticators --enable :id
```

where: `:id` is the ID of the local authenticator obtained from the previous step.

Adjust the mapping order

Adjust the execution order of your authenticator maps to control authorization rule precedence. As later maps override earlier ones, setting the correct sequence helps ensure users receive the intended permissions and team memberships.

For example, if the first authenticator map is of type `is_superuser` and the trigger is set to **never**, any user logging into the system would never be granted the `is_superuser` flag.

And, if the second map is of type `is_superuser` and the trigger is based on the user having a specific group, any user logging in would initially be denied the `is_superuser` permission. However, any user with the specified group would subsequently be granted the `is_superuser` permission by the second rule.

The order of rules is important beyond whether you want to process organizations, teams or roles first. They can also be used to refine access and careful consideration is needed to avoid login issues.

For example:

- Authenticator map A denies all users access to the system
- Authenticator map B allows the user `john` access to the system

When the mapping order is set to A, B; the first map denies access for all users, including `john`. The second map subsequently allows `john` access to the system and the result is that `john` is granted access and is able to log in to the platform.

However, when the mapping order is changed to B, A; the first map allows `john` access to the system. The second map subsequently denies all users access to the system (including `john`) and the result is that `john` is denied access and is unable to log in to the platform.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. In the list view, select the authenticator name displayed in the **Name** column.
3. Select the **Mapping** tab from the **Details** page of your authenticator.
4. Click **Manage mappings**.
5. Adjust the mapping order by dragging and dropping the mappings up or down in the list using the icon.

> **NOTE:**
> The mapping precedence is determined by the order in which the mappings are listed.

6. After your authenticator maps are in the correct order, click **Apply**.

Define authentication mapping rules and triggers

Authentication map types can be used with any type of authenticator. Each map has a trigger that defines when the map should be evaluated as true.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. In the list view, select the authenticator name displayed in the **Name** column.
3. Select the **Mapping** tab from the **Details** page of your authenticator.

4. Click **Create mapping**.
5. Select a map type from the **Authentication mapping** list. See Authenticator map types for detailed descriptions of the different map types. Choices include:
    - Allow
    - Organization
    - Team
    - Role
    - Is Superuser
6. Enter a unique rule **Name** to identify the rule.
7. Select a **Trigger** from the list. See Authenticator map triggers for more details. Choices include:
    - **Always**
    - **Never**
    - **Group**
    - **Attribute**
8. Click **Create mapping**.
9. Repeat this procedure to create additional mapping rules and triggers for the authenticator.
10. Proceed to Adjust the Mapping order to optionally reorder the mappings for your authenticator.

> **NOTE:**
> The mapping order setting is only available if there is more than one authenticator map defined.

Select an authentication type

On the **Authentication Methods** page you can select the type of authenticator plugin you want to configure.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.

3. Enter a unique **Name** for the authenticator. The name is required, must be unique across all authenticators, and must not be longer than 512 characters. This becomes the unique identifier generated for the authenticator.

> **NOTE:**
> Changing the name does not update the unique identifier of the authenticator. For example, if you create an authenticator with the name `My Authenticator` and later change it to `My LDAP Authenticator` you will not be able to create another authenticator with the name `My Authenticator` because the unique identifier is still in use.

4. Select the authenticator type from the **Authentication type** list. See Configuring an authentication type for the complete list of authentication plugins available.
5. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type. See the relevant sections in Configuring an authentication type for the required details.

   For all authentication types you can enter a **Name**, **Additional Authenticator Fields** and **Create Objects**.
6. Enable or disable **Enabled** to specify if the authenticator should be enabled or disabled. If enabled, users are able to login from the authenticator. If disabled, users will not be allowed to login from the authenticator.
7. Enable or disable **Create Object** to specify whether the authenticator should create teams and organizations in the system when a user logs in.

   Enabled

   Teams and organizations defined in the authenticator maps are created and the users added to them.

   Disabled

   Organizations and teams defined in the authenticator maps will not be created automatically in the system. However, if they already exist, for example, created by a superuser, users who trigger the maps are granted access to them.
8. Enable or disable **Remove Users**. If enabled, any access previously granted to a user is removed when they authenticate from this source. If disabled, permissions are only added or removed from the user based on the results of this authenticator's authenticator mappings.

   For example, assume a user has been granted the `is_superuser` permission in the system. And that user will log in to an authenticator whose maps will not formulate an opinion as to whether or not the user should be a superuser. If **Remove Users** is enabled, the `is_superuser` permission will be removed from the user, the authenticator maps will not have an opinion as to whether it should be there or not so, after login the user will not have the `is_superuser` permission.

   If **Remove Users** is disabled, the `is_superuser` permission *will not* be removed from the user. The authenticator maps will not have an opinion as to whether it should be there or not so after login the user *will* have the `is_superuser` permission.
9. Click **Create Authentication Method** and proceed to Define authentication mapping rules and triggers.

Next steps

IMPORTANT:
Before you enable an external authenticator, verify that your identity provider enforces email verification and restricts self-service email changes. The platform accepts email claims from identity providers without verifying email ownership.

Ansible Automation Platform uses email addresses to link external identities to existing platform accounts. If an identity provider permits unverified email addresses or unrestricted email changes, this can result in unintended account linking.

When you select an identity provider:

- Confirm that the provider verifies email addresses during user registration.
- Confirm that the provider requires administrator approval for email changes, or that email changes trigger re-verification.
- If the provider does not meet these requirements, use a different provider or implement compensating controls.

For more information about how the platform links external identities to accounts, see User association and attribute synchronization.

Configuration notes for all authentication types

Managing authentication configuration includes updating callback URLs for OAuth and SSO authenticators and configuring timeout values for password-based authenticators.

Address these configuration requirements after setting up authenticators to maintain proper integration with external identity providers.

Managing authentication configuration helps you to:

- **Maintain OAuth and SSO authentication in Ansible Automation Platform**: Update callback URLs in your identity providers to redirect authentication flows from automation controller to platform gateway.
- **Prevent authentication request failures**: Configure layered timeout values for password-based authenticators to ensure each upstream timeout exceeds the sum of its downstream timeouts.
- **Align timeout values with your environment**: Set authenticator-specific timeout values that match the performance characteristics of your external authentication servers.

Update callback URLs for OAuth and SSO providers

After architectural changes in Ansible Automation Platform 2.6, you must manually update the `callback_url` in your IdPs (GitHub, Entra ID, etc.). This redirect now points to the platform gateway instead of the controller and is dynamically generated for each authenticator.

Before you begin

- You have administrative access to the configuration settings of your external IdP.

IMPORTANT:
After upgrading, your authentication method is disabled. As a platform administrator ensure that you enable the authenticator.

The `callback_url` is normally auto-generated by Ansible Automation Platform once the authentication method is configured. You must copy this generated URL from within Ansible Automation Platform and then paste it into your IdP's settings.

Procedure

1. Go to your authenticator's configuration details within the Ansible Automation Platform UI to locate the `callback_url`.

   For more information, see Displaying authenticator details.
2. Identify and copy the auto-generated **Callback URL**, **Redirect URL**, or **Reply URL** that Ansible Automation Platform provides for your specific authentication method.
3. Update your IdP's configuration by pasting the copied redirect URL from Ansible Automation Platform into your IdP's configuration where necessary.

Result

Verify that the `callback_url` has been updated correctly and authentication is working:

- Log in to Ansible Ansible Automation Platform as an administrator account and enable the authentication method, if you have not already.
- Log in to Ansible Automation Platform using the newly configured or updated OAuth or SSO provider. A successful login indicates correct configuration.

Configure authenticator timeouts

Configure layered timeout settings for password-based authenticators, such as LDAP, RADIUS, and TACACS+. Properly aligning these upstream and downstream timeouts helps ensure that your authentication requests do not fail.

The system processes authentication requests through a chain of services, each with its own timeout setting:

- **Envoy timeout**: The total time a request can take before the initial entry point (Envoy) terminates the connection. This is the highest-level timeout.
- **gRPC timeout**: A downstream timeout that bounds the time spent communicating with the internal authentication service.
- **Authenticator timeout**: The lowest-level timeout, which defines how long an individual authenticator (LDAP, RADIUS, TACACS+) waits for a response from its third-party server.

Set timeout values

Adjust your individual authenticator timeout values to align with the specific performance needs of your authentication servers. Fine-tuning these settings helps ensure stable and reliable authentication for your Ansible Automation Platform environment.

- Configure authenticator timeouts: Adjust the timeout setting for each authenticator to a value that aligns with the expected response time of your external server.
    - For LDAP, set the `OPT_NETWORK_TIMEOUT` in seconds. For example, `OPT_NETWORK_TIMEOUT` : 30 sets an LDAP timeout of 30 seconds. For more information, see Configuring LDAP authentication.
    - For TACACS+ authentication, if you want to change the timeout you have to do it through the platform gateway API.
    - For RADIUS authentication, the timeout is not changeable and is set to 5 seconds.

Configure an authentication type to simplify user logins

Ansible Automation Platform provides multiple authenticator plugins that you can configure to simplify the login experience for your organization.

Configure local authentication

As a platform administrator, you can configure local system authentication. With local authentication, users and their passwords are checked against local system accounts.

NOTE:
A local authenticator is automatically created by the Ansible Automation Platform installation process, and is configured with the specified admin credentials in the inventory file before installation. After successful installation, you can log in to the Ansible Automation Platform using those credentials.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this Local configuration. The configuration name is required, must be unique across all authenticators, and must not be longer than 512 characters.
4. Select **Local** from the **Authentication type** list.
5. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

   NOTE:
   Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

6. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
7. To enable this authentication method upon creation, select **Enabled**.
8. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
9. Click **Next**.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Configure LDAP authentication

As a platform administrator, you can configure LDAP as the source for account authentication information for Ansible Automation Platform users.

NOTE:
If the LDAP server you want to connect to has a certificate that is self-signed or signed by an internal certificate authority (CA), the CA certificate must be added to the system's trusted CAs. Otherwise, connection to the LDAP server will result in an error that the certificate issuer is not recognized.

If you are upgrading Ansible Automation Platform and your LDAP authentication relies on a certificate added to the system's truststore, the LDAP certificate configuration is not automatically migrated to platform gateway. You must manually configure the LDAP certificate after upgrading.

- **For upgrades from Ansible Automation Platform 2.4 to Ansible Automation Platform 2.6**:
  - The migration of all authenticator configurations from automation controller to platform gateway are automated. This includes moving third-party authentication configuration and sensitive configuration data, such as SAML private keys or OAuth secret keys, from automation controller to platform gateway. However, if you are using custom LDAP certificates you still need to complete the following procedure for these certificates.
  - The `is_superuser` and `is_system_auditor` flags in your `LDAP AUTH_LDAP_USER_FLAGS_BY_GROUP` settings are successfully migrated to the new platform gateway. However, the `is_active flag` is not available in platform gateway and therefore is not migrated. Instead you can use a deny rule to prevent access to the system by users.
- **For upgrades from Ansible Automation Platform 2.5 to Ansible Automation Platform 2.6**: Authenticator configurations are not automatically migrated from automation controller. If you configured authentication in Ansible Automation Platform 2.5, those settings remain as currently configured after upgrading to 2.6. If you used a custom certificate in 2.5 for LDAP you need to migrate that as well.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **LDAP** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. In the **LDAP Server URI** field, enter or modify the list of LDAP servers to which you want to connect. This field supports multiple addresses.

6. In the **LDAP Bind DN text** field, enter the Distinguished Name (DN) to specify the user that the Ansible Automation Platform uses to connect to the LDAP server as shown in the following example:

```
cn=josie,cn=users,dc=website,dc=com
```

7. In the **LDAP Bind Password** text field, enter the password to use for the binding user.
8. Select a group type from the **LDAP Group Type** list.

The group type defines the class name of the group, which manages the groups associated with users in your LDAP directory and is returned by the search specified in Step 14 of this procedure. The group type, along with group parameters and the group search, is used to find and assign groups to users during log in, and can also be evaluated during the mapping process. The following table lists the available group types, along with their descriptions and the necessary parameters for each. By default, LDAP groups will be mapped to Django groups by taking the first value of the cn attribute. You can specify a different attribute with `name_attr`. For example, `name_attr='cn'`.

Available LDAP group types

LDAP Group Type	Description	Initializer method (*init*)
`PosixGroupType`	Handles the `posixGroup` object class. This checks for both primary group and group membership.	`name_attr='cn'`
`MemberDNGroupType`	Handles the grouping mechanisms wherein the group object contains a list of its member DNs.	`member_attr, name_attr='cn'`
`GroupOfNamesType`	Handles the `groupOfNames` object class. Equivalent to `MemberDNGroupType('member')`.	`name_attr='cn'`
`GroupOfUniqueNamesType`	Handles the `groupOfUniqueNames` object class. Equivalent to `MemberDNGroupType('uniqueMember')`.	`name_attr='cn'`
`ActiveDirectoryGroupType`	Handles the Active Directory groups. Equivalent to	`name_attr='cn'`

LDAP Group Type	Description	Initializer method (*init*)
	`MemberDNGroupType('member')`.	
`OrganizationalRoleGroupType`	Handles the `organizationalRole` object class. Equivalent to `MemberDNGroupType('roleOccupant')`.	`name_attr='cn'`
`NestedGroupOfNamesType`	Handles the `groupOfNames` object class. Equivalent to `NestedMemberDNGroupType('member')`.	`member_attr, name_attr='cn'`
`NestedGroupOfUniqueNamesType`	Handles the `groupOfUniqueNames` object class. Equivalent to `NestedMemberDNGroupType('uniqueMember')`.	`name_attr='cn'`
`NestedActiveDirectoryGroupType`	Handles the Active Directory groups. Equivalent to `NestedMemberDNGroupType('member')`.	`name_attr='cn'`
`NestedOrganizationalRoleGroupType`	Handles the `organizationalRole` object class. Equivalent to `NestedMemberDNGroupType('roleOccupant')`.	`name_attr='cn'`

NOTE:
The group types that are supported by Ansible Automation Platform use the underlying django-auth-ldap library. To specify the parameters for the selected group type, see Step 14 of this procedure.

9. You can use **LDAP User DN Template** as an alternative to user search. This approach is more efficient for user lookups than searching if it is usable in your organizational environment. Enter the name of the template as shown in the following example:

```
uid=%(user)s,cn=users,cn=accounts,dc=example,dc=com
```

where: `uid` is the user identifier, `cn` is the common name and `dc` is the domain component.

> **NOTE:**
> If this setting has a value it will be used instead of the **LDAP User Search** setting.

10. **LDAP Start TLS** is disabled by default. StartTLS allows your LDAP connection to be upgraded from an unencrypted connection to a secure connection using Transport Layer Security (TLS). To enable StartTLS when the LDAP connection is not using SSL, set the switch to **On**.
11. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

12. Enter any **LDAP Connection Options** to set for the LDAP connection. By default, LDAP referrals are disabled to prevent certain LDAP queries from hanging with Active Directory. Option names should be strings as shown in the following example:

```
OPT_REFERRALS: 0
OPT_NETWORK_TIMEOUT: 30
```

See the python-LDAP Reference for possible options and values that can be set.

13. Depending on the selected **LDAP Group Type**, different parameters are available in the **LDAP Group Type Parameters** field to account for this. `LDAP_GROUP_TYPE_PARAMS` is a dictionary, which is converted to `kwargs` and passed to the **LDAP Group Type** class selected. There are two common parameters used by group types: `name_attr` and `member_attr`. Where `name_attr` defaults to `cn` and `member_attr` defaults to `member`:

```
{"name_attr": "cn", "member_attr": "member"}
```

To determine the parameters that a specific **LDAP Group Type** requires, refer to the django_auth_ldap documentation on the classes `init` parameters.

14. In the **LDAP Group Search** field, specify which groups should be searched and how to search them as shown in the following example:

```
[
"dc=example,dc=com",
"SCOPE_SUBTREE",
"(objectClass=group)"
]
```

15. In the **LDAP User Attribute Map** field, enter user attributes to map LDAP fields to your Ansible Automation Platform users, for example, `email` or `first_name` as shown in the following example:

```
{
"first_name": "givenname",
"last_name": "sn",
"email": "mail"
}
```

16. In the **LDAP User Search** field, enter where to search for users during authentication as shown in the following example:

```
[
"ou=users,dc=website,dc=com",
"SCOPE_SUBTREE",
"(cn=%(user)s)"
]
```

If the **LDAP User DN Template** is not set, the Ansible Automation Platform authenticates to LDAP using the **Bind DN Template** and **LDAP Bind Password**. After authentication, an LDAP search is performed to locate the user specified by this field. If the user is found, Ansible Automation Platform validates the provided password against the user found by the LDAP search. Multiple search queries are supported for users with `LDAPUnion` by entering multiple search terms as shown in the following example:

```
[
    [
        "ou=users,dc=example,dc=com",
        "SCOPE_SUBTREE",
        "uid=%(user)s"
    ],
     [
        "ou=employees,dc=subdivision,dc=com",
        "SCOPE_SUBTREE",
        "uid=%(user)s"
     ]
]
```

If non-unique users are found during multiple searches, those users will not be able to log in to Ansible Automation Platform. Based on the example provided, if a user with `uid=jdoe` was found in both the `ou=users,dc=example,dc=com` and `ou=employees,dc=subdivision,dc=com`, neither `jdoe` user would be able to log in. All other unique users that are found in either branch would still be able to log in.

NOTE:
If the field **LDAP User DN Template** is populated, it takes precedence over the **LDAP User Search** field and only the template will be used to authenticate users.

17. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
18. To enable this authentication method upon creation, select **Enabled**.
19. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
20. Click **Create Authentication Method**.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Import a certificate authority in automation controller for LDAPS integration

You can authenticate to the automation controller server by using LDAP. However, if you change to using LDAPS (LDAP over SSL/TLS) to authenticate and the TLS certificate is not trusted by platform gateway, it fails with an error such as:

```
2025-08-26 16:40:56,141 WARNING  django_auth_ldap Caught LDAPError while
authenticating: SERVER_DOWN({'result': -1, 'desc': "Can't contact LDAP server",
'ctrls': [], 'info': 'error:0A000086:SSL routines::certificate verify failed (self-
signed certificate)'})
```

To get Ansible Automation Platform to trust the certificate coming from LDAP, perform the following procedure on all platform gateway instances.

Procedure

1. Place a copy of the LDAP servers certificate in the directory, `/etc/pki/ca-trust/source/anchors/`.
2. Run the command:

```
update-ca-trust
```

Configure SAML authentication

SAML allows the exchange of authentication and authorization data between an Identity Provider (IdP) and a Service Provider (SP). Ansible Automation Platform is a SAML SP that you can configure to talk with one or more SAML IdPs to authenticate users.

Before you begin

Before you configure SAML authentication in Ansible Automation Platform, be sure you do the following:

- Configure a SAML Identity Provider (IdP).
- Pre-configure the SAML IdP with the settings required for integration with Ansible Automation Platform. For example, in Microsoft Entra ID you can configure the following:
  - **Identifier (Entity ID):** This can be any value that you want, but it needs to match the one configured in your Ansible Automation Platform.
  - **Reply URL (Assertion Consumer Service (ACS) URL):** This URL is auto generated when the SAML method is configured in Ansible Automation Platform. Copy that value from Ansible Automation Platform and paste in your IdP settings.

- Gather the user attributes for your SAML IdP application. Different IdPs might use different attribute names and formats. Refer to documentation for your specific IdP for the exact attribute names and the expected values.
- Generate a private key and public certificate using the following command:

```
$ openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -sha256
-days 3650 -nodes
```

Based on groups and attributes optionally provided by the SAML IdP, users can be placed into teams and organizations in Ansible Automation Platform based on the authenticator maps tied to this authenticator. This mapping ensures that when a user logs in through SAML, Ansible Automation Platform can correctly identify the user and assign the proper attributes such as given name, surname, email, and group membership.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this SAML configuration.
4. Select **SAML** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. Enter the application-defined unique identifier used as the audience of the SAML service provider configuration in the **SAML Service Provider Entity ID** field. This is usually the base URL of your service provider, but the actual value depends on the Entity ID expected by your IdP.
6. Include the certificate content in the **SAML Service Provider Public Certificate** field. This information is contained in the cert.pem you created as a prerequisite and must include the `--BEGIN CERTIFICATE--` and `--END CERTIFICATE--`.
7. Include the private key content in the **SAML Service Provider Private Key** field. This information is contained in the key.pem you created as a prerequisite and must include the `--BEGIN PRIVATE KEY--` and `--END PRIVATE KEY--`.
8. Enter the URL to redirect the user to for login initiation in the **IdP Login URL** field. This is the login URL from your SAML IdP application.
9. Enter the public cert used for secrets coming from the **IdP in the IdP Public Cert** field. This is the SAML certificate available for download from IdP.

NOTE:
The IdP in the IdP Public Cert field should contain the entire certificate, including the `--BEGIN--` and `--END CERTIFICATE--`. You must manually enter the prefix and suffix if the IdP does not include it.

10. Enter the entity ID returned in the assertion in the **Entity ID**. This is the identifier from your IdP SAML application. You can find this value in the SAML metadata provided by your IdP.

11. Enter user details in the **Groups**, **User Email**, **Username**, **User Last Name** and **User First Name**.
12. Enter a permanent ID for the user in the **User Permanent ID** field. This field is required.

> **NOTE:**
> Additional attributes might be available through your SAML IdP. Those values must be included in either the **Additional Authenticators Fields** or the **SAML IDP to extra_data attribute mapping** field. Refer to those steps for details.

13. The **SAML Assertion Consumer Service (ACS) URL** field registers the service as a service provider (SP) with each identity provider (IdP) you have configured. Leave this field blank. After you save this authentication method, it is auto generated. This field must match the **Reply URL** setting in your IdP.
14. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator. For example, to ensure all SAML IdP attributes other than Email, Username, Last Name, First Name are included for mapping, enter the following:

```
GET_ALL_EXTRA_DATA: true
```

Alternatively, you can include a list of SAML IdP attributes in the **SAML IDP to extra_data attribute mapping** field.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

15. In the **SAML Service Provider Organization Info** field, provide the URL, display name, and the name of your app.

```
{
  "en-US": {
    "url": "http://www.example.com",
    "displayname": "Example",
    "name": "example"
  }
}
```

16. In the **SAML Service Provider Technical Contact** field, give the name and email address of the technical contact for your service provider.

```
{
"givenName": "Some User",
"emailAddress": "suser@example.com"
}
```

17. In the **SAML Service Provider Support Contact** field, give the name and email address of the support contact for your service provider.

```
{
"givenName": "Some User",
"emailAddress": "suser@example.com"
}
```

18. Optional: Provide extra configuration data in the **SAML Service Provider extra configuration data** field. For example, you can choose to enable signing requests for added security:

```
{
"sign_request": True,
}
```

This field is the equivalent to the `SOCIAL_AUTH_SAML_SP_EXTRA` in the API. For more information, see OneLogin's SAML Python Toolkit to learn about the valid service provider extra (SP_EXTRA) parameters.

NOTE: References to the OneLogin SAML Python Toolkit are for formatting examples only. Some configuration entries in the OneLogin documentation might not apply to Ansible Automation Platform.

19. Optional: Provide security settings in the **SAML Security Config** field. This field is the equivalent to the `SOCIAL_AUTH_SAML_SECURITY_CONFIG` field in the API.

```
 // Indicates whether the <samlp:AuthnRequest> messages sent by this SP // will
be signed. [Metadata of the SP will offer this info]

"authnRequestsSigned": false,



// Indicates a requirement for the <samlp:Response>, <samlp:LogoutRequest> //
and <samlp:LogoutResponse> elements received by this SP to be signed.

"wantMessagesSigned": false,



// Indicates a requirement for the <saml:Assertion> elements received by //
this SP to be signed. [Metadata of the SP will offer this info]

"wantAssertionsSigned": false,



// Authentication context.

// Set to false and no AuthContext will be sent in the AuthNRequest,

// Set true or don't present this parameter and you will get an AuthContext
'exact' 'urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport'

// Set an array with the possible auth context values: array
('urn:oasis:names:tc:SAML:2.0:ac:classes:Password',
'urn:oasis:names:tc:SAML:2.0:ac:classes:X509'),

"requestedAuthnContext": true,
```

For more information and additional options, see OneLogin's SAML Python Toolkit.

20. Optional: In the **SAML IDP to extra_data attribute mapping** field, enter values to map IDP attributes to extra_data attributes. These values will include additional user information beyond standard attributes such as Email or Username to be mapped. For example:

```
 - Department

- UserType

- Organization
```

For more information on the values you can include, see advanced SAML settings.

IMPORTANT:
Ensure to include all relevant values so that everything gets mapped correctly for your configuration. Alternatively, you can include the `GET_ALL_EXTRA_DATA: true` in the **Additional Authenticator Fields** to allow mapping of all available SAML IdP attributes.

21. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
22. To enable this authentication method upon creation, select **Enabled**.

23. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
24. Click **Create Authentication Method**.

> **IMPORTANT:**
> You can configure an HTTPS redirect for SAML in operator-based deployments to simplify login for your users.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong.

Configure transparent SAML logins

For transparent logins to work, you must first get IdP-initiated logins to work.

- Set the `RelayState` on the IdP to "IdP".

Configure TACACS+ authentication

Terminal Access Controller Access-Control System Plus (TACACS+) provides centralized AAA services (authentication, authorization, and accounting). You can configure Ansible Automation Platform to use TACACS+ as a source for remote authentication and access control.

> **NOTE:**
> This feature is deprecated and will be removed in a future release.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **TACACS+** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. Enter the following information:

    - Hostname of TACACS+ Server: Provide the hostname or IP address of the TACACS+ server with which to authenticate. If you leave this field blank, TACACS+ authentication is disabled.

- TACACS+ Authentication Protocol: The protocol used by the TACACS+ client. The options are ascii or pap.
- Shared secret for authenticating to TACACS+ server: The secret key for TACACS+ authentication server.

6. The **TACACS+ client address sending enabled** is disabled by default. To enable client address sending, select the checkbox.
7. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

8. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
9. To enable this authentication method upon creation, select **Enabled**.
10. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
11. Click **Create Authentication Method**.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Configure Microsoft Entra ID authentication

To set up enterprise authentication for Microsoft Entra ID, formerly known as Microsoft Azure Active Directory (AD), follow these steps:

1. **Configure your Ansible Automation Platform** to use Microsoft Entra ID authentication using the steps in this procedure.
2. **Register Ansible Automation Platform** in Microsoft Entra ID by following the Quickstart: Register an application with the Microsoft identity platform. This process provides you with an Application (client) ID and Application secret.
3. **Add the redirect URL in Microsoft Entra ID**. After completing the configuration wizard for Microsoft Entra ID authentication in your platform, copy the URL displayed in the **Azure AD OAuth2 Callback URL** field. Then, go to your registered enterprise application in Azure and add this URL as a **Redirect URL** (also referred to as a **Callback URL** in Ansible Automation Platform) as described in How to add a redirect URI to your application. This step is required for the login flow to work correctly.

The attributes provided by Microsoft Entra ID are not set in the Ansible Automation Platform configuration for this authentication type. Instead, the social_core azuread backend provides the translation of claims provided by Microsoft Entra ID. The user attributes that allow Ansible Automation Platform to correctly identify the user and assign the proper attributes such as given name, surname, email, and username include the following:

Ansible Automation Platform attribute	Microsoft Entra ID parameter
authenticator_uid	upn
Username	name
First Name	given_name
Last Name	family_name
Email	email (falling back to upn)

To set up enterprise authentication for Microsoft Azure Active Directory (AD), you need to obtain an OAuth2 key and secret by registering your organization-owned application from Azure at: Quickstart: Register an application with the Microsoft identity platform.

Each key and secret must belong to a unique application and cannot be shared or reused between different authentication backends. To register the application, you must supply it with your webpage URL, which is the Callback URL shown in the Authenticator details for your authenticator configuration. See Displaying authenticator details for instructions on accessing this information.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **Azuread** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. Click **Edit**, copy and paste Microsoft's **Application (Client) ID** to the **OIDC Key** field.
6. If your Microsoft Entra ID is configured to provide user group information within a groups claim, ensure that the platform is configured with a **Groups Claim** name that matches your Microsoft Entra ID configuration. This allows the platform to correctly identify and associate groups for users logging in through Microsoft Entra ID.

NOTE:
Groups coming from Microsoft Entra ID can be identified using either unique IDs or group names. When creating group mappings for a Microsoft Entra ID authenticator, you can use either the unique ID or the group name.

By default, Microsoft Entra ID uses groups as the default group claim name. So, be sure to either set the value to the default or to any custom override you have set in your IdP. The current default is set to preserve the existing behavior unless explicitly changed.

7. Following instructions for registering your application with the Microsoft identity platform, supply the key (shown at one time only) to the client for authentication.
8. Copy and paste the secret key created for your Microsoft Entra ID/Microsoft Azure AD application to the **OIDC Secret** field.
9. Optional: By default, the name attribute from Microsoft Entra ID is used as the username field within Ansible Automation Platform. If you want to override that default value and use another attribute from Microsoft Entra ID, enter that attribute in the username field.
10. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

NOTE:
Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

11. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
12. To enable this authentication method upon creation, select **Enabled**.
13. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
14. Click **Create Authentication Method**.

Result

To verify that the authentication is configured correctly, log out of Ansible Automation Platform and check that the login screen displays the logo of your authentication chosen method to enable logging in with those credentials.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Related information
What is the Microsoft identity platform?

Configure Google OAuth2 authentication

To set up social authentication for Google, you must obtain an OAuth2 key and secret for a web application. To do this, you must first create a project and set it up with Google.

For instructions, see Setting up OAuth 2.0 in the Google API Console Help documentation.

If you have already completed the setup process, you can access those credentials by going to the Credentials section of the Google API Manager Console. The OAuth2 key (Client ID) and secret (Client secret) are used to supply the required fields in the UI.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **Google OAuth** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. The **Google OAuth2 Key** and **Google OAuth2 Secret** fields are pre-populated.

   If not, use the credentials Google supplied during the web application setup process. Save these settings for use in the following steps.
6. Copy and paste Google's Client ID into the **Google OAuth2 Key** field.
7. Copy and paste Google's Client secret into the **Google OAuth2 Secret** field.
8. Optional: Enter information for the following fields using the tooltip provided for instructions and required format:

   - **Access Token URL**
   - **Access Token Method**
   - **Authorization URL**
   - **Revoke Token Method**
   - **Revoke Token URL**
   - **OIDC JWT Algorithm(s)**
   - **OIDC JWT**
9. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

NOTE:
Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

10. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
11. To enable this authentication method upon creation, select **Enabled**.
12. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
13. Click **Create Authentication Method**.

Result

To verify that the authentication is configured correctly, log out of Ansible Automation Platform and check that the login screen displays the logo of your authentication chosen method to enable logging in with those credentials.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Configure generic OIDC authentication

OpenID Connect (OIDC) uses OAuth 2.0 to verify identity and obtain user info. Unlike SAML's provider-to-provider trust, OIDC relies on the HTTPS channel to secure tokens. To set up OIDC with Ansible Automation Platform, consult your IdP's documentation for the required credentials.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **Generic OIDC** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. Enter the following information:
    - **OIDC Provider URL**: The URL for your OIDC provider.
    - **OIDC Key**: The client ID from your third-party IdP.
    - **OIDC Secret**: The client secret from your IdP.
6. Optional: Select the HTTP method to be used when requesting an access token from the **Access Token Method** list. The default method is **POST**.
7. Optionally enter information for the following fields using the tooltip provided for instructions and required format:
    - **Access Token Method** - The default method is **POST**.

- **Access Token URL**
- **Access Token Method**
- **Authorization URL**
- **Callback URL** - The OIDC **Callback URL** field registers the service as a service provider (SP) with each OIDC provider you have configured. Leave this field blank. After you save this authentication method, it is auto generated. Configure your IdP to allow redirects to this URL as part of the authentication flow.
- **ID Key**
- **ID Token Issuer**
- **JWKS URI**
- **OIDC Public Key**
- **Revoke Token Method** - The default method is **GET**.
- **Revoke Token URL**
- **Response Type**
- **Token Endpoint Auth Method**
- **Userinfo URL**
- **Username Key**

8. Use the **Verify OIDC Provider Certificate** to enable or disable the OIDC provider SSL/TLS certificate verification.
9. Use the **Redirect** State to enable or disable the state parameter in the redirect URI. Enable this to prevent Cross Site Request Forgery (CSRF) attacks.
10. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

11. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
12. To enable this authentication method upon creation, select **Enabled**.
13. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
14. Click **Create Authentication Method**.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Troubleshoot Generic OIDC Single Sign-On authentication failures

Authentication fails if the `OIDC JWT Algorithm` setting is not explicitly defined. The authentication code requires a list of acceptable algorithms, which it does not retrieve automatically from the OpenID Connect (OIDC) configuration endpoint.

Configure JWT_Algorithms manually

To resolve the authentication failure, manually provide the list of supported algorithms in the platform gateway configuration.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Select your OIDC authenticator from the list.
3. Click **Edit authentication** and locate the **OIDC JWT Algorithm(s)** field.
4. Enter the list of supported algorithms as a YAML list or a JSON array. These algorithms are typically available from your IdP's OpenID Connect (OIDC) discovery endpoint.

Example

```
[
    "PS384",
    "ES384",
    "RS384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
]
```

5. Save your changes. The system uses these specified algorithms for token verification, resolving any authentication failures related to their absence.

Enable debugging for enterprise authentication

To further diagnose authentication issues, enable debug logging in platform gateway.

Procedure

1. Change the logging configuration in the platform gateway's `settings.py` file.
2. Set the logging level for the `ansible_base` logger to `DEBUG`:

```
LOGGING['loggers']['ansible_base']['level'] = 'DEBUG'
```

After this change, detailed `AuthTokenError` messages are displayed in the logs, providing specific information about the cause of the failure.

Troubleshoot Generic OIDC scope mismatches

Authentication fails when the Identity Provider (IdP) does not support the default scopes automatically appended by the system.

To prevent the system from appending this default scope, you must add a setting to your authenticator configuration.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Select your OIDC authenticator from the list.
3. Click **Edit authentication**.
4. In the **Additional Authenticator Fields** section, add the following attribute and value. This input box supports either YAML or JSON. Ensure you add this key-value pair on a new line if there are other fields present:

```
IGNORE_DEFAULT_SCOPE: True
```

5. Save your changes. The authenticator now only uses the scopes you explicitly defined, resolving any authentication failures related to unsupported scopes.

Configure keycloak authentication

You can configure Ansible Automation Platform to integrate Keycloak to manage user authentication.

NOTE:
When using this authenticator some specific setup in your Keycloak instance is required. Refer to the Python Keycloak reference for more details.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **Keycloak** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. Enter the location where the user's token can be retrieved in the **Keycloak Access Token URL** field.
6. Optional: Enter the redirect location the user is taken to during the login flow in the **Keycloak Provider URL** field.
7. Enter the Client ID from your Keycloak installation in the **Keycloak OIDC Key** field.
8. Enter the RS256 public key provided by your Keycloak realm in the **Keycloak Public Key** field.
9. Enter the OIDC secret (Client Secret) from your Keycloak installation in the **Keycloak OIDC Secret** field.
10. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

    NOTE:
    Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

11. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
12. To enable this authentication method upon creation, select **Enabled**.
13. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
14. Click **Create Authentication Method**.

If you receive an `jwt.exceptions.InvalidAudienceError: Audience doesn't match` error, you must re-enable the audience by doing the following:

1. From the navigation for your Keycloak configuration, select **Client scopes > YOUR-CLIENT-ID-dedicated > Add mapper > Audience**.
2. Pick a name for the mapper.
3. Select the **Client ID** corresponding to your client in `Included Client Audience`.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Configure GitHub authentication

You can connect GitHub identities to Ansible Automation Platform using OAuth. To set up GitHub authentication, you need to obtain an OAuth2 key and secret by registering your organization-owned application from GitHub by using the registering the new application with GitHub.

The OAuth2 key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI. To register the application, you must supply it with your webpage URL, which is the Callback URL shown in the Authenticator details for your authenticator configuration. See Displaying authenticator details for instructions on accessing this information.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **GitHub** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. When the application is registered, GitHub displays the **Client ID** and **Client Secret**:
    a. Copy and paste the GitHub Client ID into the GitHub OAuth2 Key field.
    b. Copy and paste the GitHub Client Secret into the GitHub OAuth2 Secret field.
6. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

7. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
8. To enable this authentication method upon creation, select **Enabled**.

9. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
10. Click **Create Authentication Method**.

Result

To verify that the authentication is configured correctly, log out of Ansible Automation Platform and check that the login screen displays the logo of your authentication chosen method to enable logging in with those credentials.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Configure GitHub organization authentication

When defining account authentication with either an organization or a team within an organization, you should use the specific organization and team settings. Account authentication can be limited by an organization and by a team within an organization.

You can also choose to permit all by specifying non-organization or non-team based settings. You can limit users who can log in to the platform by limiting only those in an organization or on a team within an organization.

To set up social authentication for a GitHub organization, you must obtain an OAuth2 key and secret for a web application by using the registering the new application with GitHub.

The OAuth2 key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI. To register the application, you must supply it with your webpage URL, which is the Callback URL shown in the Authenticator details for your authenticator configuration. See Displaying authenticator details for instructions on accessing this information.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **GitHub organization** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. When the application is registered, GitHub displays the **Client ID** and **Client Secret**:
    a. Copy and paste the GitHub Client ID into the GitHub OAuth2 Key field.
    b. Copy and paste the GitHub Client Secret into the GitHub OAuth2 Secret field.

6. Enter the name of your GitHub organization, as used in your organization's URL, for example, `https://github.com/<yourorg>/` in the **GitHub OAuth Organization Name** field.
7. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

8. Enter the authorization scope for users in the **GitHub OAuth2 Scope** field. The default is `read:org`.
9. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
10. To enable this authentication method upon creation, select **Enabled**.
11. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
12. Click **Create Authentication Method**.

Result

To verify that the authentication is configured correctly, log out of Ansible Automation Platform and check that the login screen displays the logo of your authentication chosen method to enable logging in with those credentials.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Configure GitHub team authentication

To set up social authentication for a GitHub team, you must obtain an OAuth2 key and secret for a web application by using the instructions provided in registering the new application with GitHub.

The OAuth2 key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI. To register the application, you must supply it with your webpage URL, which is the **Callback URL** shown in the Authenticator details for your authenticator configuration. See Displaying authenticator details for instructions on accessing this information.

Each key and secret must belong to a unique application and cannot be shared or reused between different authentication backends. The OAuth2 key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **GitHub team** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. When the application is registered, GitHub displays the **Client ID** and **Client Secret**:
    a. Copy and paste the GitHub Client ID into the GitHub OAuth2 Key field.
    b. Copy and paste the GitHub Client Secret into the GitHub OAuth2 Secret field.
6. Copy and paste GitHub's team ID in the **GitHub OAuth2 Team ID** field.
7. Enter the authorization scope for users in the GitHub OAuth2 Scope field. The default is `read:org`.
8. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

9. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
10. To enable this authentication method upon creation, select **Enabled**.
11. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
12. Click **Create Authentication Method**.

    To verify that the authentication is configured correctly, log out of Ansible Automation Platform and check that the login screen displays the logo of your authentication chosen method to enable logging in with those credentials.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Configure GitHub enterprise authentication

To set up social authentication for a GitHub enterprise, you must obtain a GitHub Enterprise URL, an API URL, OAuth2 key and secret for a web application.

To obtain the URLs, see the GitHub Enterprise administration documentation.

The OAuth2 key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI. To register the application, you must supply it with your webpage URL, which is the **Callback URL** shown in the Authenticator details for your authenticator configuration. See Displaying authenticator details for instructions on accessing this information.

Each key and secret must belong to a unique application and cannot be shared or reused between different authentication backends. The OAuth2 key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **GitHub enterprise** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. When the application is registered, GitHub displays the **Client ID** and **Client Secret**:
    a. Copy and paste the GitHub Client ID into the GitHub OAuth2 Key field.
    b. Copy and paste the GitHub Client Secret into the GitHub OAuth2 Secret field.
6. In the **Base URL** field, enter the hostname of the GitHub Enterprise instance, for example, `https://github.example.com`.
7. In the **Github OAuth2 Enterprise API URL** field, enter the API URL of the GitHub Enterprise instance, for example, `https://github.example.com/api/v3`.
8. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

9. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
10. To enable this authentication method upon creation, select **Enabled**.
11. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
12. Click **Create Authentication Method**.

Result

To verify that the authentication is configured correctly, log out of Ansible Automation Platform and check that the login screen displays the logo of your authentication chosen method to enable logging in with those credentials.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Configure GitHub enterprise organization authentication

To set up social authentication for a GitHub enterprise organization, you must obtain a GitHub enterprise organization URL, an Organization API URL, an Organization OAuth2 key and secret for a web application.

To obtain the URLs, see the GitHub Enterprise administration documentation.

The OAuth2 key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI. To register the application, you must supply it with your webpage URL, which is the **Callback URL** shown in the Authenticator details for your authenticator configuration. See Displaying authenticator details for instructions on accessing this information.

Each key and secret must belong to a unique application and cannot be shared or reused between different authentication backends. The OAuth2 key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.
3. Enter a **Name** for this authentication configuration.
4. Select **GitHub enterprise organization** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. When you register the application, GitHub displays the **Client ID** and **Client Secret**:
    a. Copy and paste the GitHub Client ID into the GitHub OAuth2 Key field.
    b. Copy and paste the GitHub Client Secret into the GitHub OAuth2 Secret field.
6. In the **Base URL** field, enter the hostname of the GitHub Enterprise instance, for example, `https://github.example.com`.
7. In the **Github OAuth2 Enterprise API URL** field, enter the API URL of the GitHub Enterprise instance, for example, `https://github.example.com/api/v3`.
8. Enter the name of your GitHub enterprise organization, as used in your organization's URL, for example, `https://github.com/<yourorg>/` in the **GitHub OAuth2 Enterprise Org Name** field.
9. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

NOTE:
Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

10. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
11. To enable this authentication method upon creation, select **Enabled**.
12. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
13. Click **Create Authentication Method**.

    To verify that the authentication is configured correctly, log out of Ansible Automation Platform and check that the login screen displays the logo of your authentication chosen method to enable logging in with those credentials.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to See "Map external authenticators to Ansible Automation Platform" on page 156.

Configure GitHub enterprise team authentication

To set up social authentication for a GitHub enterprise team, you must obtain a GitHub Enterprise Organization URL, an Organization API URL, an Organization OAuth2 key and secret for a web application.

To obtain the URLs, see the GitHub Enterprise administration documentation.

To obtain the key and secret, you must first register your enterprise organization-owned application at `https://github.com/organizations/<yourorg>/settings/applications`.

The OAuth2 key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI. To register the application, you must supply it with your webpage URL, which is the **Callback URL** shown in the Authenticator details for your authenticator configuration. See Displaying authenticator details for instructions on accessing this information.

Each key and secret must belong to a unique application and cannot be shared or reused between different authentication backends. The OAuth2key (Client ID) and secret (Client Secret) are used to supply the required fields in the UI.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. Click **Create authentication**.

3. Enter a **Name** for this authentication configuration.
4. Select **GitHub enterprise team** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. When the application is registered, GitHub displays the **Client ID** and **Client Secret**:
    a. Copy and paste the GitHub Client ID into the GitHub OAuth2 Key field.
    b. Copy and paste the GitHub Client Secret into the GitHub OAuth2 Secret field.
6. In the **Base URL** field, enter the hostname of the GitHub Enterprise instance, for example, `https://github.orgexample.com`.
7. In the **Github OAuth2 Enterprise API URL** field, enter the API URL of the GitHub Enterprise instance, for example, `https://github.example.com/api/v3`.
8. Enter the OAuth2 key (Client ID) from your GitHub developer application in the **GitHub OAuth2** team ID field.
9. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

10. To automatically create organizations, users, and teams upon successful login, select **Create objects**.
11. To enable this authentication method upon creation, select **Enabled**.
12. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users**.
13. Click **Create Authentication Method**.

Result

To verify that the authentication is configured correctly, log out of Ansible Automation Platform and check that the login screen displays the logo of your authentication chosen method to enable logging in with those credentials.

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or to what groups they belong, continue to Mapping.

Configure RADIUS authentication

You can configure Ansible Automation Platform to centrally use RADIUS as a source for authentication information.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods** .
2. Click **Create authentication** .
3. Enter a **Name** for this authentication configuration.
4. Select **Radius** from the **Authentication type** list. The **Authentication details** section automatically updates to show the fields relevant to the selected authentication type.
5. Enter the host or IP of the RADIUS server in the **RADIUS Server** field. If you leave this field blank, RADIUS authentication is disabled.
6. Enter the **Shared secret** for authenticating to RADIUS server.
7. Optional: Enter any **Additional Authenticator Fields** that this authenticator can take. These fields are not validated and are passed directly back to the authenticator.

> **NOTE:**
> Values defined in this field override the dedicated fields provided in the UI. Any values not defined here are not provided to the authenticator.

8. To automatically create organizations, users, and teams upon successful login, select **Create objects** .
9. To enable this authentication method upon creation, select **Enabled** .
10. To remove a user for any groups they were previously added to when they authenticate from this source, select **Remove users** .
11. Click **Create Authentication Method** .

Next steps

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (like username and email address) or to what groups they belong, continue to Mapping .

User and external authentication mapping

Ansible Automation Platform manages user accounts and synchronizes attributes by centralizing user identification around a matching email address. You can sign in with existing accounts from different sources while maintaining a consistent user profile and access permissions.

WARNING:
The platform accepts email claims from identity providers without verifying email ownership. Before you configure an external authenticator, verify that your identity provider enforces email verification and restricts self-service email changes.

When you log in to the platform for the first time with an authenticator, such as Local, GitHub, SAML, or LDAP, the platform evaluates the username and email address. If a single email match exists, the platform links the external identity to that existing account.

- Subsequent logins with the same authenticator and external Unique Identifier (UID) directly sign the user into their linked account.
- If a user's external UID changes, the system re-triggers the email-based linking logic. If the new UID's email matches the existing account, the new authenticator is linked. If the email does not match or is not provided, a new user account might be created.
- If a user's external email changes, the platform does not automatically update the email address in the existing account, but the user can still sign in and a new account with the new email is created for the user.

If a user has a hashed account, such as `bob-hash`, due to a username collision from a previous version, that association is honored for that authenticator. However, for new authentications from other identity providers, the platform maps to the user's primary account, such as `bob`, provided a single matching email exists. This consolidates user identities and prevents the creation of new hashed accounts. If users should have been previously merged, you can delete the user-<hash> account from Ansible Automation Platform and on a subsequent login, the users are merged based on emails as described above.

IMPORTANT:

- **Authenticators without email**: If an authenticator such as RADIUS or TACACS+ does not return an email address, a new account is created on first sign-in. To ensure consistent future access, manually add an email to the account after creation.
- **Multiple users with the same email**: If an email from an authenticator matches multiple existing platform accounts, the sign-in process fails.
- **LDAP usernames**: The platform treats LDAP usernames as case-insensitive. It converts the username to lowercase and stores it in the database.
- `associated_authenticators` field: The `associated_authenticators` field in the API supports multiple UIDs per user.

Email address modification restrictions

The platform restricts email address modifications because it uses email addresses as the primary criterion for linking external identities to existing accounts.

Non-admin users cannot modify their Ansible Automation Platform email address. Only administrators manage email address changes to prevent unauthorized account linking.

The following users can modify email addresses:

- Platform administrators (superusers)
- Organization administrators, on deployments where **Organization admins can manage users and teams** is enabled

If a user's email address requires updating, contact a platform administrator to make the change.

Related information
Override email modification restrictions

Override email modification restrictions

If your organization requires non-admin users to modify their own email addresses, you can enable the `ALLOW_USER_EMAIL_SELF_EDIT` setting for each Ansible Automation Platform component.

IMPORTANT:
Enabling `ALLOW_USER_EMAIL_SELF_EDIT` re-introduces the risk of account pre-hijacking through email address manipulation. Only enable this setting if your organization has compensating controls in place.

This setting is deprecated and will be removed in a future version of Ansible Automation Platform.

Each component manages its own settings independently. Apply the setting to each component based on your deployment topology.

Procedure

1. Configure the setting for your deployment type:
   - **RPM deployments**: Create or edit the override file for the relevant component, then restart its service:
     - automation controller: */etc/tower/conf.d/custom.py*
     - platform gateway: */etc/ansible-automation-platform/gateway/settings.py*

- automation hub: */etc/pulp/settings.py*
- Event-Driven Ansible: */etc/ansible-automation-platform/eda/settings.yaml*

- **Containerized deployments**: Add the setting through the `extra_settings` variable for each component in your installer inventory or group variables:
  - `gateway_extra_settings` – platform gateway
  - `controller_extra_settings` – automation controller
  - `eda_extra_settings` – Event-Driven Ansible
  - `hub_extra_settings` – automation hub

  Use the following format:

```
controller_extra_settings:
  - setting: ALLOW_USER_EMAIL_SELF_EDIT
    value: true
```

- **OpenShift (Operator) deployments**: Add the setting under `spec.extra_settings` on the `AnsibleAutomationPlatform` custom resource:

```
spec:
  extra_settings:
    - setting: ALLOW_USER_EMAIL_SELF_EDIT
      value: "true"
```

  The operator writes these settings into a ConfigMap and mounts it as a settings file inside the pod. The platform applies changes automatically after you update the custom resource.

Audit email address modifications

Use the **detect_changed_emails** management command to identify users whose email addresses have been modified. The command analyzes audit data from the activity stream and compares it against current user records.

The command does not attribute changes to specific causes. It provides data for you to investigate changes that might require further action.

Procedure

1. Run the following command to list email changes recorded in the activity stream:

```
$ aap-gateway-manage detect_changed_emails
```

2. Optional: To run a full security audit that includes authenticator linkage checks, duplicate email detection, and high-risk scoring, add the `--audit` flag:

```
$ aap-gateway-manage detect_changed_emails --audit
```

3. Review the output and investigate any accounts where the email address does not match the expected owner based on your organization's directory, the account has elevated RBAC permissions, the email change occurred shortly before a new identity provider login, or the account has both local and external authenticators linked.

Map external authenticators to Ansible Automation Platform

To control which users are allowed into the Ansible Automation Platform server, and placed into Ansible Automation Platform organizations or teams based on their attributes (such as username and email address) or what groups they belong to, you can configure authenticator maps.

Use authenticator maps to add conditions that must be met before a user is given or denied access to a resource type. Authenticator maps are associated with an authenticator and are given an order. The maps are processed in order when the user logs in. These are similar to firewall rules or mail filters.

Understand authenticator mapping

Review how authenticator maps evaluate trigger conditions sequentially to govern user authorization. Understanding this process helps ensure that Ansible Automation Platform assigns the correct resource permissions when users log in.

Authentication

Validates a user's identity, typically through a username and password or a trust system.

Authorization

Determines what an authenticated user can do once they are authenticated.

In Ansible Automation Platform, authenticators manage authentication, validating users and returning details such as their username, given name, email, and group memberships (for example, LDAP groups). Authorization comes from the authenticator's associated maps.

During the authentication process, after a user has authenticated, the authorization system starts with a default set of permissions in memory. Then sequentially, the authenticator maps are processed and adjust permissions based on their trigger conditions. When all the authenticator's maps are processed, the in-memory representation of the user's permissions are reconciled with their existing permissions.

For example, here is a simplified in-memory representation of the default permissions as follows:

```
Access allowed = True

Superuser permission = Undefined

Admin of teams = None
```

And, you might have maps that need to be processed are processed in the following order:

1. **Allow** rule set to never
2. **Allow** rule based on group
3. **Superuser** rule based on user attributes
4. **Team** admin rule based on user group

The first **Allow** map, set to never, denies access to the system and the in-memory representation looks like:

```
Access allowed = False

Superuser permission = Undefined

Admin of teams = None
```

However, if the user matches the second **Allow** map (the group-based allow), the permissions change to the following:

```
Access allowed = True

Superuser permission = Undefined

Admin of teams = None
```

Where the user is subsequently granted access to Ansible Automation Platform because they have the required groups.

Next, the **Superuser** map checks user attributes. If no match is found, it does not revoke existing permissions by default. Therefore, the permissions remain the same as the results from the previous map:

```
Access allowed = True
Superuser permission = Skipped
Admin of teams = None
```

To revoke superuser access, you can select the **Revoke** option on the **Superuser** map. That way, when the user does not meet the attribute criteria, the permissions update to False such as the following:

```
Access allowed = True
Superuser permission = False
Admin of teams = None
```

The final **Team** map checks the user's groups coming from the authenticator for admin access on the team "My Team". If the user has the required group, the permissions update to the following:

```
Access allowed = True
Superuser permission = False
Admin of teams = "My Team"
```

If the user lacks the required group, permissions remain unchanged unless the **Revoke** option has been selected on the map, in which case permissions update to the following:

```
Access allowed = True
Superuser permission = False
Admin of teams = Revoke admin of "My Team"
```

After processing all maps in the order defined, the final permissions reconcile, updating the user's access based on the map rules.

In summary, authenticators validate users and delegate system authorization to the authenticator maps. Authenticator maps are executed in order creating an in-memory representation of the users' permissions which get reconciled with the actual permissions after all maps are executed.

By default, authenticator maps return either **ALLOW** or **SKIPPED**.

ALLOW

Means that a match is detected and the platform should grant the user access to the corresponding role or permission (such as, superuser, or team member).

SKIPPED

Means that the user did not match the trigger in the map and, the platform skips processing this map and continues to check the remaining maps. This is useful if you want to grant users additional permissions in the system without having to change the authenticator maps.

However, when the **Revoke** option is selected, **SKIPPED** becomes **DENY** and users who do not meet the required trigger criteria are denied access to the corresponding role or permission. This ensures that only users with matching trigger conditions are granted access.

Authenticator map types

Ansible Automation Platform supports the following rule types:

Allow

Determine if the user is allowed to log in to the system.

Organization

Determine if a user should be put into an organization.

Team

Determine if the user should be a member of a team.

Role

Determine if the user is a member of a role (for example, *System Auditor*).

Is Superuser

Determine if the user is a superuser in the system.

These authentication map types can be used with any type of authenticator.

Authenticator map triggers

Each map has a trigger that defines when the map should be evaluated as true. Trigger types include the following:

Always

The trigger should always be fired.

Never

The trigger should never be fired.

Group

The map is true or false based on a user having, not having, or having multiple groups in the source system.

When defining a group trigger, the authentication mapping expands to include the following selections:

- **Operation:** This field includes conditional settings that trigger the handling of the rule based on the specified **Groups** criteria. The choices include **and** and **or**. For example, if you select **and** the user logging in must be a member of all of the groups specified in the **Groups** field for this trigger to be true. Alternatively, if you select **or** the user logging in must be a member of any of the specified **Groups** in order for the trigger to fire.

> **NOTE:**
> If you are only keying off one group it does not matter if you select **"and"** or **"or"**.

- **Groups:** This is a list of one or more groups coming from the authentication system that the user must be a member of. The first time you create a **Groups** entry, you must manually enter the values. Once entered, that selection will be available from the **Groups** list.
  See the **Operation** field to determine the behavior of the trigger if more than one group is specified in the trigger.

> **NOTE:**
> You must enter group identifiers in lowercase. For example, `cn=johnsmith,dc=example,dc=com` instead of `CN=JohnSmith,DC=Example,DC=COM`.

Attribute

The map is true or false based on a users attributes coming from the source system.

When defining an attribute trigger, the authentication mapping expands to include the following selections:

- **Operation:** This field includes conditional settings that trigger the handling of the rule based on the specified **Attribute** criteria. In version 2.6 this field indicates what will happen if the source system returns a list of attributes instead of a single value. For example, if the source system returns multiple emails for a user and **Operation** was set to **and**, all of the given emails must match the **Comparison** for the trigger to be *True*. If **Operation** was set to **or**, any of the returned emails will set the trigger to *True* if they match the **Comparison** in the trigger.

> **NOTE:**
> If you want to experiment with multiple attribute maps you can do that through the API but the UI form will remove multi-attribute maps if the authenticator is saved through the UI. When adding multiple attributes to a map, the **Operation** will also apply to the attributes.

- **Attribute:** The name of the attribute coming from the source system this trigger will be evaluated against. For example, if you wanted the trigger to fire based on the user's

surname and the last name field in the source system was called `users_last_name` you would enter the value "users_last_name" in this field.

- **Comparison:** Tells the trigger how to evaluate the value of the users. **Attribute** in the source system compared to the **Value** specified on the trigger. Available options are: **contains**, **matches**, **ends with**, **in**, or **equals**. Below is a breakdown of each **Comparison** type:
    - **contains**: The specified character sequence in **Value** is contained within the attributes value returned from the source. For example, given an attribute value of "John" from the source the contains **Comparison** would set the trigger to *True* if the trigger **Value** was set to "Jo" and *False* if the trigger **Value** was "Joy".
    - **matches**: The **Value** on the trigger is treated as a python regular expression and does a regular expression match (with case ignore on) between the specified **Value** and the value returned from the source system. For example, if the trigger's **Value** was "Jo" the trigger would return *True* if the value from the source was "John" or "Joanne" or any other value which matched the regular expression "Jo". The trigger would return *False* if the sources value for the attribute was "Dan" because "Dan" does not match the regular expression "Jo".
    - **ends with**: The trigger will see if the value provided by the source ends with the specified **Value** of the trigger. For example, if the source provided a value of "John" the trigger would be *True* if its **Value** was set to "n" or "on". The trigger would be *False* if its **Value** was set to "z" because the value "John" coming from the source does not end with the value "z" specified by the trigger.
    - **equal**: The trigger will see if the value provided by the source is equal to (in its entirety) the specified **Value** of the trigger. For example, if the source returned the value "John", the trigger would be *True* if its **Value** was set to "John". Any value other than "John" returned from the source would set this trigger to *False*.
    - **in**: The **in** condition checks if the value matches one of several values. When **in** is specified as the **Comparison**, the **Value** field can be a comma-separated list. For example, if a trigger had a **Value** of "John,Donna" the trigger would be *True* if the attribute coming from the source had either the value "John" or "Donna". Otherwise, the trigger would be *False*.
    - **Value**: The value that a users attribute will be matched against based on the **Comparison** field. See examples in the **Comparison** definition.

      NOTE:
      If the **Comparison** type is **in**, this field can be a comma-separated list (without spaces).

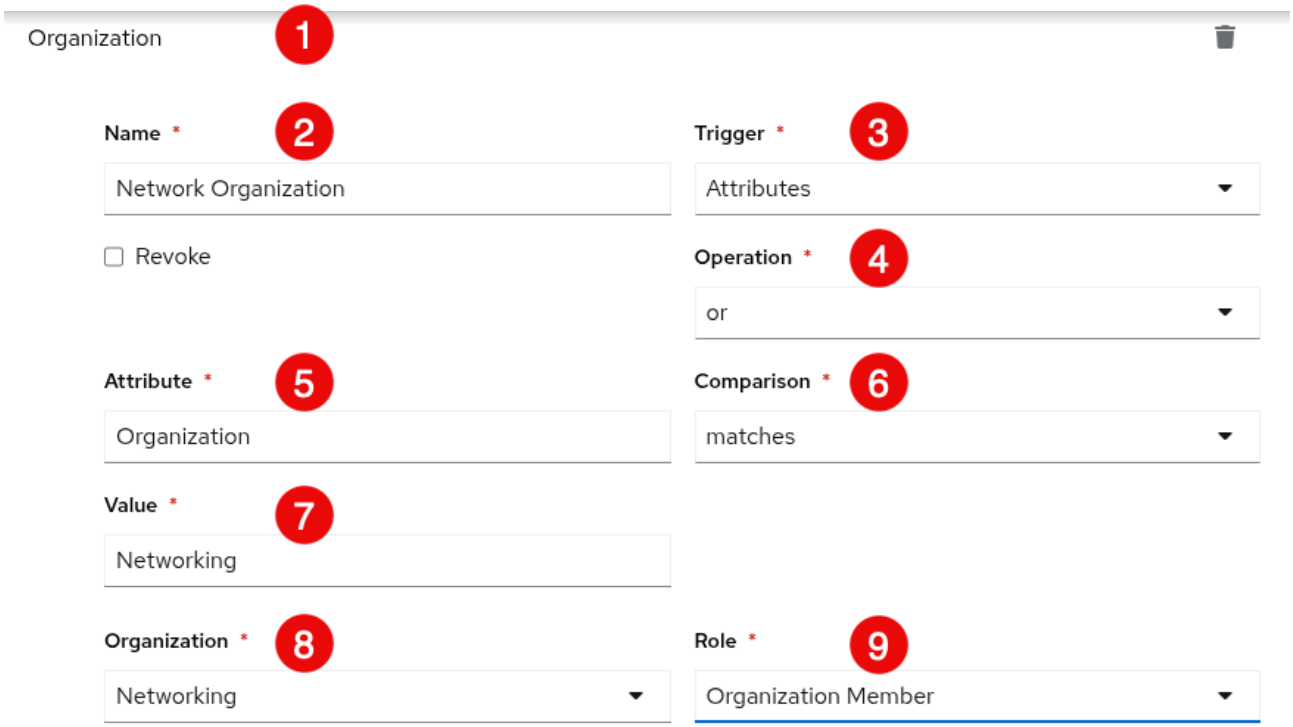
Related information
Authenticator map examples
Regular expression match (re.match)

Authenticator map examples

Use the following examples to explore the different conditions, such as groups and attribute values you can implement to control user access to the platform.

Add users to an organization based on an attribute

In this example, you will add a user to the **Networking** organization if they have an `Organization` attribute with the value of `Networking`:



1. The **Organization** title of the page indicates that you are configuring settings permissions on an organization.
2. `Network Organization` is entered in this field and is the unique, descriptive name for this map configuration.
3. **Attributes** is selected from the **Trigger** list to configure authentication based on an attribute from the source system, which in this example is `Organization`.
4. The operation is defined as `or` meaning that at least one condition must be true for authentication to succeed.
5. The **Attribute** coming from the source system is Organization.
6. The **Comparison** value is set to `matches` which means that when a user has the attribute **Value** of `Networking`, they are added to the **Networking** organization.
7. The attribute **Value** coming from the source system is `Networking`.
8. The name of the **Organization** to which you are adding members is `Networking`.

9. Users are added to the **Networking** organization with the `Organization Member` role.

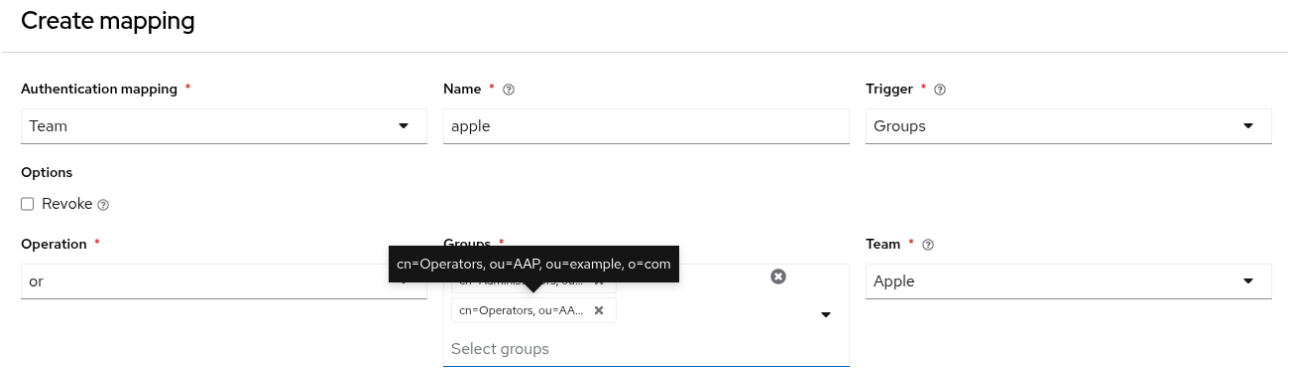
Add users to a team based on the users group

In this example, you will add user to the `Apple` team if they have either of the following groups:

```
cn=administrators,ou=aap,ou=example,o=com
```

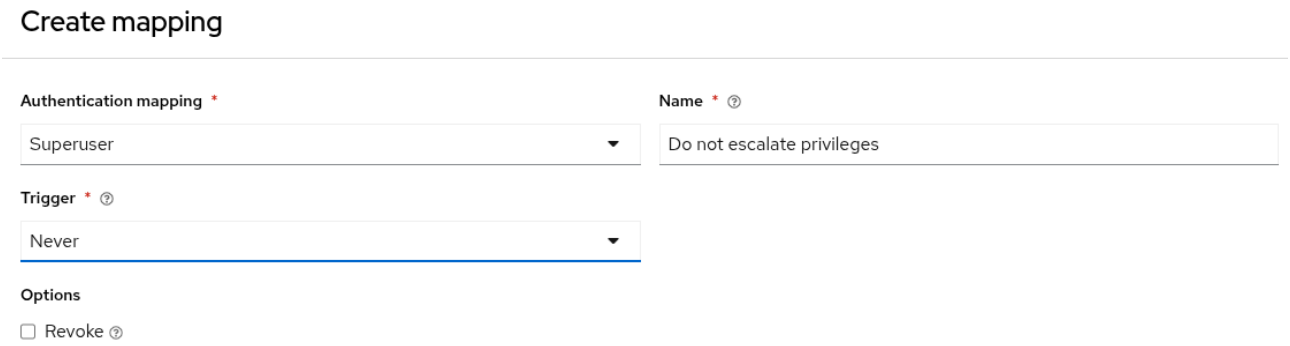
or

```
cn=operators,ou=aap,ou=example,co=com
```



Do not escalate privileges

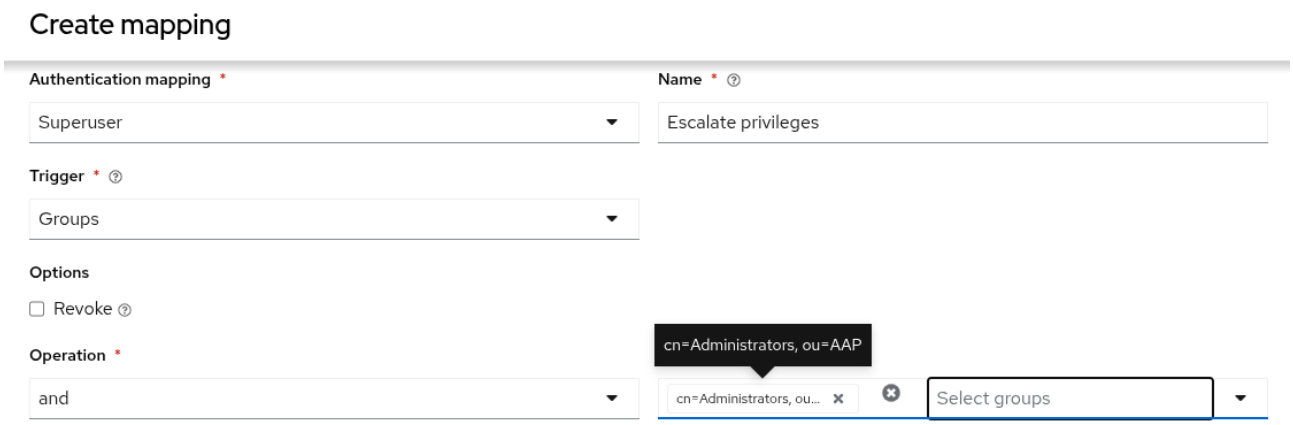
In this example, you never escalate users to a superuser. But note, this rule does not revoke a user's superuser permission because the revoke option is not set.



Escalate privileges based on a user having a group

In this example, you escalate user privileges to superuser if they belong to the following group:

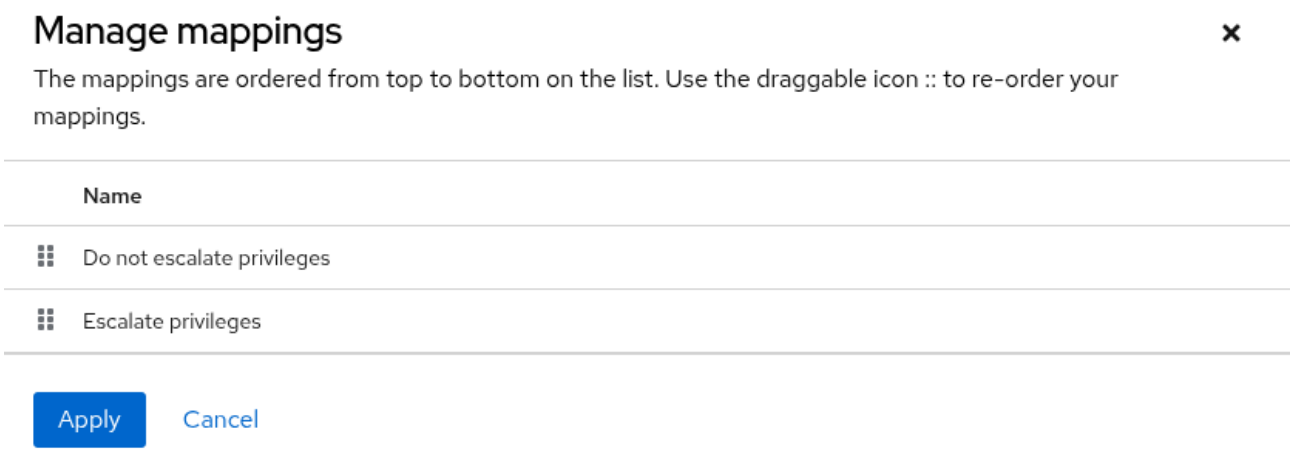
```
cn=administrators,ou=aap
```



Using mapping order to create exceptions

Since maps are executed in order, it is possible to create exceptions. Expanding on the previous example for *Do not escalate privileges*, you can add another rule with a higher order, such as, *Escalate privileges*.

The first rule (*Do not escalate privileges*) prevents any user from being escalated to a superuser, but the second rule (*Escalate privileges*) alters that decision to grant superuser privileges to a user if they are in the `Administrators` group.



Control user access to the system with allow mapping

With allow mapping, you can control which users have access to the system by defining the conditions that must be met.

Procedure

1. After configuring the authentication details for your authentication method, select the **Mapping** tab.

2. Select **Allow** from the **Add authentication mapping** list.
3. Enter a unique rule **Name** to identify the rule.
4. Select a **Trigger** from the list. See Authenticator map triggers for more information about map triggers.
5. Select **Revoke** to deny user access to the system when the trigger conditions are not matched.
6. Click **Next**.

Next steps

1. You can manage the authentication mappings order by clicking **Manage mappings**.
2. Drag and drop the mapping up or down in the list.

> **NOTE:**
> The mapping precedence is determined by the order in which the mappings are listed.

3. Click **Apply**.

Map users to organizations

You can control which users are placed into which Ansible Automation Platform organizations based on attributes such as their username and email address or based on groups provided from an authenticator.

When organization mapping is positively evaluated, a specified organization is created, if it does not exist if the authenticator tied to the map is allowed to create objects.

Procedure

1. After configuring the authentication details for your authentication method, select the **Mapping** tab.
2. Select **Organization** from the **Add authentication mapping** list.
3. Enter a unique rule **Name** to identify the rule.
4. Select a **Trigger** from the list. See Authenticator map triggers for more information about map triggers.
5. Select **Revoke** to remove the user's access to the selected organization role when the trigger conditions are not matched.
6. Select the **Organization** to which matching users are added or blocked.
7. Select a **Role** to be applied or removed for matching users (for example, **Organization Admin** or **Organization Member**).
8. Click **Next**.

Next steps

1. You can manage the authentication mappings order by clicking **Manage mappings**.
2. Drag and drop the mapping up or down in the list.

> **NOTE:**
> The mapping precedence is determined by the order in which the mappings are listed.

3. Click **Apply**.

Map users to teams

Team mapping is the mapping of team members (users) from authenticators.

You can define the options for each team's membership. For each team, you can specify which users are automatically added as members of the team and also which users can administer the team.

You can specify Team mappings separately for each account authentication.

When Team mapping is positively evaluated, a specified team and its organization are created, if they do not exist if the related authenticator is allowed to create objects.

> **IMPORTANT:**
> When configuring team mappings with an Attribute trigger, use the `or` operation. The `and` operation requires every single value in a list to match the comparison criteria for the trigger to be successful. This is rarely the intended behavior, as you typically want a match on at least one value in the list.

Procedure

1. After configuring the authentication details for your authentication method, select the **Mapping** tab.
2. Select **Team** from the **Add authentication mapping** list.
3. Enter a unique rule **Name** to identify the rule.
4. Select a **Trigger** from the list. See Authenticator map triggers for more information about map triggers.
5. Select **Revoke** to remove the user's access to the selected organization role and deny user access to the system when the trigger conditions are not matched.
6. Select the **Team** and **Organization** to which matching users are added or blocked.
7. Select a **Role** to be applied or removed for matching users (for example, **Team Admin** or **Team Member**).

8. Click **Next**.

Next steps

1. You can manage the authentication mappings order by clicking **Manage mappings**.
2. Drag and drop the mapping up or down in the list.

> **NOTE:**
> The mapping precedence is determined by the order in which the mappings are listed.

3. Click **Apply**.

Map users to roles

Role mapping is the mapping of a user either to a global role, such as Platform Auditor, or team or organization role.

When a Team or Organization is specified together with the appropriate Role, the behavior is the same with Organization mapping or Team mapping.

You can specify role mapping separately for each account authentication.

Procedure

1. After configuring the authentication details for your authentication method, select the **Mapping** tab.
2. Select **Role** from the **Add authentication mapping** list.
3. Enter a unique rule **Name** to identify the rule.
4. Select a **Trigger** from the list. See Authenticator map triggers for more information about map triggers.
5. Select **Revoke** to remove the role for the user when none of the trigger conditions are matched.
6. Select a **Role** to be applied or removed for matching users.
7. Click **Next**.

Next steps

1. You can manage the authentication mappings order by clicking **Manage mappings**.
2. Drag and drop the mapping up or down in the list.

> **NOTE:**
> The mapping precedence is determined by the order in which the mappings are listed.

3. Click **Apply**.

Map users to the superuser role

Superuser mapping is the mapping of a user to the superuser role, such as System Administrator.

Procedure

1. After configuring the authentication details for your authentication method, select the **Mapping** tab.
2. Select **Superuser** from the **Add authentication mapping** list.
3. Enter a unique rule **Name** to identify the rule.
4. Select a **Trigger** from the list. See Authenticator map triggers for more information about map triggers.
5. Select **Revoke** to remove the superuser role from the user when none of the trigger conditions are matched.
6. Click **Next**.

Next steps

1. You can manage the authentication mappings order by clicking **Manage mappings**.
2. Drag and drop the mapping up or down in the list.

> **NOTE:**
> The mapping precedence is determined by the order in which the mappings are listed.

3. Click **Apply**.

Review authenticator map results

Platform administrators can review authenticator map evaluation results through the user API endpoint.

The results show how the maps are processed during the user's login:

```
api/gateway/v1/users/X
```

Locate and manage authentication configurations

After you have configured your authentication settings, you can view a list of authenticators, search, sort and view the details for each authenticator configured on the system.

Authentication list view

On the **Authentication Methods** page, you can view and manage the configured authentication methods for your organization.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.

   The **Authentication Methods** page is displayed.
2. Click **Create authentication** and follow the steps for creating an authentication method in Configuring an authentication type. Otherwise, proceed to step 3.
3. From the menu bar, you can sort the list of authentication methods by using the arrows in the menu bar for **Order**, **Name** and **Authentication type**.
4. Click the toggles to **Enable** or **Disable** authenticators.

Search for an authenticator

You can search for a previously configured authenticator from the Authentication list view.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. In the search bar, enter an appropriate keyword for the authentication method you want to search for and click the arrow icon.
3. If you do not find what you are looking for, you can narrow your search. From the filter list, select **Name** or **Authentication type** depending on the search term you want to use.
4. Scroll through the list of search results and select the authenticator you want to review.

Display authenticator details

After you locate the authenticator you want to review, you can display the configuration details:

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. In the list view, select the authenticator name displayed in the **Name** column.

   The authenticator **Details** page is displayed.
3. From the **Details** page, you can review the configuration settings applied to the authenticator.

Edit an authenticator

You can change the settings of previously configured authenticators from the **Authentication** list view.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.
2. In the list view, you can either:
   a. Select the **Edit** icon next to authenticator you want to modify, or
   b. Select the authenticator name displayed in the **Name** column and click **Edit authenticator** from the **Details** page.
3. Change the authentication details or mapping configurations as required.
4. Click **Save**.

Delete an authenticator

Delete unused authenticators to prevent users from logging in through outdated identity providers. Remove these configurations to ensure your Ansible Automation Platform environment remains secure and organized.

You can change the settings of previously configured authenticators from the **Authentication** list view.

Procedure

1. From the navigation panel, select **Access Management > Authentication Methods**.

2. In the list view, select the checkbox next to the authenticator you want to delete.
3. Select **Delete authentication** from the ⋮ list.

> **NOTE:**
> You can delete many authenticators by selecting the checkbox next to each authenticator you want to remove, and clicking **Delete selected authentication** from the ⋮ list on the menu bar.

Configure Google Cloud for increased authentication performance

Increase the default port limit on your Google Cloud Platform (GCP) Cloud NAT gateway to prevent authentication and performance issues during high traffic. This helps ensure stable connectivity for Ansible Automation Platform deployments on OpenShift (version 4.17 and above).

The default setting for the Cloud NAT gateway's **Minimum ports per VM instance** in OpenShift installations on GCP (version 4.17 and above) is 64. This low port limit can be quickly exhausted when platform gateway handles concurrent external network connections, such as Single Sign-On (SSO) requests. When the limit is reached, it prevents new outgoing connections, causing authentication failures or severe performance degradation.

Increase the minimum ports

To address this limitation, manually increase the **Minimum ports per VM instance** setting for the Cloud NAT gateway associated with the worker nodes.

Use the Google Cloud Console to apply this workaround.

Procedure

1. Go to the Cloud NAT service.
2. Locate and select the NAT gateway configured for your OpenShift cluster's worker nodes.
3. Increase the default value of 64 for the **Minimum ports per VM instance** setting to a higher value to accommodate your anticipated traffic volume.

   Increasing this limit ensures enough available ports for external communication, reducing the likelihood of performance issues during high-volume authentication and external communication tasks.

Configure access to external applications with tokens

Token-based authentication permits authentication of third-party tools and services with the platform through integrated OAuth 2 token support. Ansible Automation Platform utilizes both OAuth Tokens and Personal Access Tokens (PATs).

OAuth Tokens

OAuth Tokens are tied to specific applications and allow applications to access data without disclosing user login information.

Personal Access Tokens

PATs are personal to a user and not tied to a specific application. They are created directly by a user for their own use.

The default expiration for access tokens has been updated from 1000 years to 1 year. This change ensures frequent token rotation for increased credential security.

NOTE:
Access tokens in controller 2.4 and previous versions of the platform gateway were valid for 1000 years. Any existing tokens created before the 2.5.20250604 patch release will retain a 1000 year expiration.

You can customize this setting to meet your specific requirements by modifying the expiration time in your `settings.py` file as follows:

```
OAUTH2_PROVIDER__ACCESS_TOKEN_EXPIRE_SECONDS = 31536000
```

Related information
The OAuth 2.0 Authorization Framework
Token and session management

Manage OAuth applications

Create and configure token-based authentication for external applications such as ServiceNow and Jenkins. With token-based authentication, external applications can easily integrate with Ansible Automation Platform.

IMPORTANT:
Automation controller OAuth applications on the platform UI are not supported for 2.4 to 2.5 migration.

As a platform administrator, you can configure a custom external application URL within the platform, providing seamless integration with external services. This functionality is currently available as a Technology Preview. Once configured, the external application URL is displayed in the platform UI navigation panel, providing users with easy access to the application. This feature streamlines workflows by ensuring quick access to external services from within the platform UI.

NOTE:
Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

With OAuth 2 you can use tokens to share data with an application without disclosing login information. You can configure these tokens as read-only.

You can create an application that is representative of the external application you are integrating with, then use it to create tokens for the application to use on behalf of its users.

Associate these tokens with an application resource to manage all tokens issued for a particular application. By separating the issue of tokens under **OAuth Applications**, you can revoke all tokens based on the application without having to revoke all tokens in the system.

Related information
OAuth applications and tokens do not exist after migrating from Ansible Automation Platform 2.4 to 2.5
Technology Preview Features - Scope of Support

Get started with OAuth Applications

You can access the **OAuth Applications** page from the navigation panel by selecting **Access Management > OAuth Applications**. From there you can view, create, sort and search for applications currently managed by Ansible Automation Platform and automation controller.

If no applications exist, you can create one by clicking **Create OAuth application**.

Application functions

Several OAuth 2 utilities are available for authorization, token refresh, and revoke. You can specify the following grant types when creating an application:

Password

This grant type is ideal for users who have native access to the web application and must be used when the client is the resource owner.

Authorization code

This grant type should be used when access tokens must be issued directly to an external application or service.

NOTE:
You can only use the authorization code type to acquire an access token when using an application. When integrating an external web application with Ansible Automation Platform, that web application might need to create OAuth2 tokens on behalf of users in that other web application. Creating an application in the platform with the authorization code grant type is the preferred way to do this because:

- This allows an external application to obtain a token from Ansible Automation Platform for a user, using their credentials.
- Compartmentalized tokens issued for a particular application enables those tokens to be easily managed. For example, revoking *all* tokens associated with that application without having to revoke all tokens in the system.

Request an access token after expiration

The default expiration for access tokens is 1 year.

The best way to set up application integrations by using the **Authorization code** grant type is to allowlist the origins for those cross-site requests. More generally, you must allowlist the service or application you are integrating with the platform, for which you want to provide access tokens.

To do this, have your administrator add this allowlist to their local Ansible Automation Platform settings file:

```
CORS_ORIGIN_ALLOW_ALL = True
CORS_ALLOWED_ORIGIN_REGEXES = [
    r"http://django-oauth-toolkit.herokuapp.com*",
    r"http://www.example.com*"
]
```

Where `http://django-oauth-toolkit.herokuapp.com` and `http://www.example.com` are applications requiring tokens with which to access the platform.

OAuth2 application and token migration (2.4 to 2.6)

During the upgrade from Ansible Automation Platform 2.4 to 2.6, there are important changes to how OAuth2 applications and tokens are managed. Ansible Automation Platform now uses platform gateway OAuth applications and deprecates automation controller OAuth applications.

- **Automation controller OAuth applications**: You can view and edit existing automation controller applications, but new ones can no longer be created. These legacy applications continue to function, but they might be removed in a future release. Plan to migrate to platform gateway OAuth applications.
- **Automation controller tokens**: Automation controller personal access tokens (PATs), are also deprecated. Guide users to move to platform gateway PATs.
- **Platform gateway OAuth applications and tokens**: Platform applications and tokens offer an updated interface and are the standard for future use. Move to these applications and tokens.

Manage `OAUTH2_PROVIDER` settings

The `OAUTH2_PROVIDER` settings from automation controller are managed by platform gateway after upgrading from 2.4. to 2.6. The default token expiration values might differ between automation controller and platform gateway.

- The default access token expiration is updated from 1,000 years to 1 year. This change increases credential security through more frequent token rotation.
- Platform gateway's default OAUTH2_PROVIDER settings are:

```
{
  "ACCESS_TOKEN_EXPIRE_SECONDS": 31536000000,
  "REFRESH_TOKEN_EXPIRE_SECONDS": 2628000,
  "AUTHORIZATION_CODE_EXPIRE_SECONDS": 600
}
```

  If you previously set a token expiration shorter than one year, you must manually update the platform gateway settings to match your required configuration.

Create a new application

When integrating an external web application with Ansible Automation Platform, the web application might need to create OAuth2 tokens on behalf of users of the web application.

Creating an application with the Authorization Code grant type is the preferred way to do this for the following reasons:

- External applications can obtain a token for users, using their credentials.
- Compartmentalized tokens issued for a particular application, enables those tokens to be easily managed. For example, revoking all tokens associated with that application.

Procedure

1. From the navigation panel, select **Access Management > OAuth Applications**.
2. Click **Create OAuth application**. The **Create Application** page opens.
3. Enter the following details:

Name

(required) Enter a name for the application you want to create.

URL

(optional) Enter the URL of the external application. This link is added to the navigation panel for easy access. This setting is currently offered as a Technology Preview only.

Description

(optional) Include a short description for your application.

Organization

(required) Select an organization with which this application is associated.

Authorization grant type

(required) Select one of the grant types to use for the user to get tokens for this application. For more information, see Application functions for more information about grant types.

Client Type

(required) Select the level of security of the client device.

Redirect URIS

Provide a list of allowed URIs, separated by spaces. You need this if you specified the grant type to be **Authorization code**.

4. Click **Create OAuth application**, or click **Cancel** to abandon your changes.

The **Client ID** and **Client Secret** display in a window. This will be the only time the client secret will be shown.

> **NOTE:**
> The **Client Secret** is only created when the **Client type** is set to **Confidential**.

5. Click the copy icon and save the client ID and client secret to integrate an external application with Ansible Automation Platform.

Associate tokens with applications

You can view a list of users that have tokens to access an application by selecting the **Tokens** tab in the **OAuth Applications** details page.

> **NOTE:**
> You can only create OAuth 2 Tokens for your own user, which means you can only configure or view tokens from your own user profile.

When authentication tokens have been configured, you can select the application to which the token is associated and the level of access that the token has.

Procedure

1. From the navigation panel, select **Access Management > Users**.
2. Select the username for your user profile to configure OAuth 2 tokens.
3. Select the **Tokens** tab.

   When no tokens are present, the **Tokens** screen prompts you to add them.
4. Click **Create token** to open the **Create Token** window.
5. Enter the following details:

   Application

   Enter the name of the application with which you want to associate your token. Alternatively, you can search for it by clicking **Browse**. This opens a separate window that enables you to choose from the available options. Select **Name** from the filter list to filter by name if the list is extensive.

   > **NOTE:**
   > To create a Personal Access Token (PAT) that is not linked to any application, leave the Application field blank.

   Description

   (optional) Provide a short description for your token.

   Scope

   (required) Specify the level of access you want this token to have. The scope of an OAuth 2 token can be set as one of the following:

   - **Write**: Allows requests sent with this token to add, edit and delete resources in the system.
   - **Read**: Limits actions to read only. Note that the write scope includes read scope.
6. Click **Create token**, or click **Cancel** to abandon your changes.

The Token information is displayed with **Token** and **Refresh Token** information, and the expiration date of the token. This will be the only time the token and refresh token will be shown. You can view the token association and token information from the list view.

7. Click the copy icon and save the token and refresh token for future use.

Result

You can verify that the application now shows the user with the appropriate token by using the **Tokens** tab on the Applications details page.

1. From the navigation panel, select **Access Management > OAuth Applications**.
2. Select the application you want to verify from the **Applications** list view.
3. Select the **Tokens** tab.
   Your token should be displayed in the list of tokens associated with the application you chose.

Related information
Token and session management

Application token functions

The `refresh` and `revoke` functions associated with tokens, for tokens at the `/o/` endpoints can currently only be carried out with application tokens.

Refresh an existing access token

The following example shows an existing access token with a refresh token provided:

```
{
    "id": 35,
    "type": "access_token",
    ...
    "user": 1,
    "token": "omMFLk7UKpB36WN2Qma9H3gbwEBSOc",
    "refresh_token": "AL0NK9TTpv0qp54dGbC4VUZtsZ9r8z",
    "application": 6,
    "expires": "2017-12-06T03:46:17.087022Z",
    "scope": "read write"
}
```

The `/o/token/` endpoint is used for refreshing the access token:

```
curl -X POST \

    -d "grant_type=refresh_token&refresh_token=AL0NK9TTpv0qp54dGbC4VUZtsZ9r8z" \

    -u
"gwSPoasWSdNkMDtBN3Hu2WYQpPWCO9SwUEsKK22l:fI6ZpfocHYBGfm1tP92r0yIgCyfRdDQt0Tos9L8a4f
NsJjQQMwp9569eIaUBsaVDgt2eiwOGe0bg5m5vCSstClZmtdy359RVx2rQK5YlIWyPlrolpt2LEpVeKXWaiy
bo" \

    http://<gateway>/o/token/ -i
```

Where `refresh_token` is provided by `refresh_token` field of the preceding access token.

The authentication information is of format `<client_id>:<client_secret>`, where `client_id` and `client_secret` are the corresponding fields of the underlying related application of the access token.

NOTE:
The special OAuth 2 endpoints only support using the `x-www-form-urlencoded` **Content-type**, so as a result, none of the `/o/*` endpoints accept `application/json`.

On success, a response displays in JSON format containing the new (refreshed) access token with the same scope information as the previous one:

```
HTTP/1.1 200 OK

Server: nginx/1.12.2

Date: Tue, 05 Dec 2017 17:54:06 GMT

Content-Type: application/json

Content-Length: 169

Connection: keep-alive

Content-Language: en

Vary: Accept-Language, Cookie

Pragma: no-cache

Cache-Control: no-store

Strict-Transport-Security: max-age=15768000


{"access_token": "NDInWxGJI4iZgqpsreujjbvzCfJqgR", "token_type": "Bearer",
"expires_in": 315360000000, "refresh_token": "DqOrmz8bx3srlHkZNKmDpqA86bnQkT",
"scope": "read write"}
```

The refresh operation replaces the existing token by deleting the original and then immediately creating a new token with the same scope and related application as the original one.

Verify that the new token is present and the old one is deleted in the `api/gateway/v1/tokens/` endpoint.

Revoke an access token

You can revoke an access token by deleting the token in the platform UI, or by using the `/o/revoke-token/` endpoint.

Revoking an access token by this method is the same as deleting the token resource object, but it enables you to delete a token by providing its token value, and the associated `client_id` (and `client_secret` if the application is `confidential`). For example:

```
curl -X POST -d "token=rQONsve372fQwuc2pn76k3IHDCYpi7" \

-u
"gwSPoasWSdNkMDtBN3Hu2WYQpPWCO9SwUEsKK22l:fI6ZpfocHYBGfm1tP92r0yIgCyfRdDQt0Tos9L8a4f
NsJjQQMwp9569eIaUBsaVDgt2eiwOGe0bg5m5vCSstClZmtdy359RVx2rQK5YlIWyPlrolpt2LEpVeKXWaiy
bo" \

http://<gateway>/o/revoke_token/ -i
```

NOTE:

- The special OAuth 2 endpoints only support using the `x-www-form-urlencoded` **Content-type**, so as a result, none of the `/o/*` endpoints accept `application/json`.
- The **Allow External Users to Create Oauth2 Tokens** (`ALLOW_OAUTH2_FOR_EXTERNAL_USERS` in the API) setting is disabled by default. External users refer to users authenticated externally with a service such as LDAP, or any of the other SSO services. This setting ensures external users cannot create their own tokens. If you enable then disable it, any tokens created by external users in the meantime will still exist, and are not automatically revoked. This setting can be configured from the **Settings > Platform gateway** menu.

You can also revoke OAuth2 tokens by using the `manage` utility, see Revoke oauth2 tokens.

On success, a response of `200 OK` is displayed. Verify the deletion by checking whether the token is present in the `api/gateway/v1/tokens/` endpoint.

Create, revoke, or clear tokens

Ansible Automation Platform supports the following commands for OAuth2 token management:

- `create_oauth2_token`

- `revoke_oauth2_tokens`
- `cleartokens`
- `clearsessions`

`create_oauth2_token`

Use the following command to create OAuth2 tokens (specify the username for `example_user`):

```
$ aap-gateway-manage create_oauth2_token --user example_user



  New OAuth2 token for example_user: j89ia8OO79te6IAZ97L7E8bMgXCON2
```

Ensure that you give a valid user when creating tokens. Otherwise, an error message that you attempted to issue the command without specifying a user, or supplied a username that does not exist, is displayed.

`revoke_oauth2_tokens`

Use this command to revoke OAuth2 tokens, both application tokens and personal access tokens (PAT). It revokes all application tokens (but not their associated refresh tokens), and revokes all personal access tokens. However, you can also specify a user for whom to revoke all tokens.

To revoke all existing OAuth2 tokens use the following command:

```
$ aap-gateway-manage revoke_oauth2_tokens
```

To revoke all OAuth2 tokens and their refresh tokens use the following command:

```
$ aap-gateway-manage revoke_oauth2_tokens --revoke_refresh
```

To revoke all OAuth2 tokens for the user with `id=example_user` (specify the username for `example_user`):

```
$ aap-gateway-manage revoke_oauth2_tokens --user example_user
```

To revoke all OAuth2 tokens and refresh token for the user with `id=example_user`:

```
$ aap-gateway-manage revoke_oauth2_tokens --user example_user --revoke_refresh
```

`cleartokens`

Use the cleartokens command to delete all sessions that have expired.

For more information, see cleartokens in Django's Oauth Toolkit documentation.

`clearsessions`

Use the `cleartokens` command to delete all sessions that have expired.

For more information, see Clearing the session store in Django's Oauth Toolkit documentation.

For more information about OAuth2 token management in the UI, see the Applications.

Personal Access Token migration

After upgrading to Ansible Automation Platform 2.6, Personal Access Tokens (PATs) from a 2.4 automation controller remain functional. They are visible in the platform gateway UI and you can use them with both automation controller and platform gateway APIs.

Managing automation controller tokens

After the upgrade, you can perform the following actions with your automation controller tokens:

- **Platform gateway UI**: You can edit or delete the tokens, but you cannot create or refresh them.
- **Automation controller** API: You can create, edit, delete, or refresh the tokens.

Tokens are labeled in the UI to indicate if they are automation controller only or platform gateway. Platform gateway tokens are unaffected by these requirements, other than being rendered in the UI with a **platform** type.

Security settings for OAuth2 tokens and external users

Configure your platform settings to enable OAuth2 token creation for external users. Allowing users authenticated via LDAP, SAML, or SSO to generate tokens prevents 403 Forbidden errors and helps ensure they maintain programmatic API access.

This default behavior is a deliberate security measure. Ansible Automation Platform prioritizes centralized control over token generation, which encourages administrators to select the appropriate method for enabling OAuth 2.0 user token generation for external authentication providers.

It is important to understand that an OAuth2 token is created within Ansible Automation Platform, and Ansible Automation Platform itself manages its lifecycle, including its expiration. This lifecycle is independent of the user's session with their external Identity Provider (IdP). For example, if a user generates an Ansible Automation Platform token and their account is later disabled in the external IdP, the Ansible Automation Platform token remains valid until it expires or is manually revoked. Being aware of this interaction is crucial for a secure configuration, as it highlights the need for compensating controls if you enable token creation for external users.

Enable OAuth2 token creation for external users

To enable external users to create OAuth2 tokens, change the appropriate setting in your Ansible Automation Platform environment. Ensure the implementation of compensating security controls after enabling this setting.

Procedure

1. From the navigation panel, go to **Settings > Platform gateway**.
2. Click **Edit platform gateway** settings.
3. Change the **Allow external users to create OAuth2 tokens** setting to **Enabled**.
4. Click **Save platform gateway settings**.

Next steps

Implement the recommended security controls as described in *Implementing security controls for external user OAuth2 tokens*.

Implement security controls for external user OAuth2 tokens

After enabling OAuth2 token creation for external users, implement the following compensating controls to keep a strong security posture.

- **Limit token lifetime**: Configure a shorter duration for OAuth2 tokens to reduce the window of exposure.
    - In your Ansible Automation Platform settings, adjust the `OAUTH2_ACCESS_TOKEN_EXPIRE_SECONDS value`. A value of 28800 (8 hours) is recommended, limiting token validity to a standard workday.
- **Enforce strict role-based access control (RBAC)**: Grant users only the minimum necessary permissions.

    - Assign users who create tokens to **Teams** with highly restrictive roles. Avoid granting broad permissions that could lead to privilege escalation.

- **Establish a clear offboarding process**: Integrate token revocation into your organizational offboarding procedures. Your HR and IT offboarding processes must include a step for an Ansible Automation Platform administrator to revoke all active tokens for a departing user. Tokens can be manually revoked from the user's profile under the **Tokens** tab.
- **Audit and monitor**: Regularly review token-related activities for legitimacy in the **Activity Stream**.

Replace legacy automation controller tokens

When Event-Driven Ansible controller is deployed on Ansible Automation Platform, you can create a Red Hat Ansible Automation Platform credential to connect to automation controller through the use of an automation controller URL and a username and password.

After it has been created, you can attach the Red Hat Ansible Automation Platform credential to a rulebook and use it to run rulebook activations. These credentials provide a simple way to configure communication between automation controller and Event-Driven Ansible controller, enabling your rulebook activations to launch job templates.

NOTE:
If you previously used controller tokens to connect automation controller and Event-Driven Ansible controller, these tokens are now deprecated. To delete deprecated controller tokens and the rulebook activations associated with them, complete the following procedures starting with replacing controller tokens before proceeding with setting up a Red Hat Ansible Automation Platform credential.

Related information
Replace controller tokens

Replace deprecated controller tokens

To use Event-Driven Ansible controller, you must replace legacy controller tokens configured in your environment with Red Hat Ansible Automation Platform credentials because controller tokens have been deprecated.

Delete rulebook activations with controller tokens

Delete rulebook activations that rely on deprecated controller tokens. This mandatory step prevents conflicts before migrating to the new, required Red Hat Ansible Automation Platform credentials.

Procedure

1. Log in to the Ansible Automation Platform Dashboard.
2. From the top navigation panel, select **Automation Decisions > Rulebook Activations**.
3. Select the rulebook activations that have controller tokens.
4. Select the **More Actions** icon ⋮ next to the **Rulebook Activation enabled/disabled** toggle.
5. Select **Delete rulebook activation**.
6. In the window, select **Yes, I confirm that I want to delete these X rulebook activations**.
7. Select **Delete rulebook activations**.

Delete controller tokens

Before configuring the new Red Hat Ansible Automation Platform credentials, delete all existing controller tokens, which are now deprecated and will conflict with the new Red Hat Ansible Automation Platform credentials.

Before you begin

- You have deleted all rulebook activations that use controller tokens.

Procedure

1. Log in to the Ansible Automation Platform Dashboard.
2. From the top navigation panel, select your profile.
3. Click **User details**.
4. Select the **Tokens** tab.
5. Delete all of your previous controller tokens.

Next steps

After deleting the controller tokens and rulebook activations, proceed withSee "Replace legacy automation controller tokens" on page 184.

Manage access with role-based access control

Role-based access control (RBAC) restricts user access based on the user's role within the organization they are assigned to in Ansible Automation Platform. The roles in RBAC refer to the levels of access that users have to Ansible Automation Platform components and resources.

You can control what users can do with the components of Ansible Automation Platform at a broad or granular level depending on your RBAC policy. You can choose whether the user is a system administrator or normal user and align roles and access permissions with their positions within the organization.

You can define roles with multiple permissions that can then be assigned to resources, teams, and users. The permissions that make up a role govern what the assigned role allows. Permissions are allocated with only the access needed for a user to perform the tasks appropriate for their role.

IMPORTANT:
When managing users, teams, and organizations, use the Unified UI or the platform gateway API to ensure real-time synchronization across all platform components, including Event-Driven Ansible controller. If you use the legacy automation controller API, changes can take up to 15 minutes to propagate to Event-Driven Ansible controller, which can result in authentication errors for new users or teams.

Structure groups and resources with organizations

Administrators use organizations to group resources. Assigning a team or user to an organization grants access to all its contents. This simplifies management, as members automatically gain access to new resources added to the organization without needing individual permissions.

After you have created an organization, Ansible Automation Platform displays the organization details. You can then manage resources such as access and execution environments for the organization.

Ansible Automation Platform automatically creates a default organization. If you have a self-support level license, you have only the default organization available and must not delete it.

Create an organization

Ansible Automation Platform automatically creates a default organization. If you have a self-support level license, you have only the default organization available and cannot delete it.

Procedure

1. From the navigation panel, select **Access Management > Organizations**.
2. Click **Create organization**.
3. Enter the **Name** and give a **Description** for your organization.

> **NOTE:**
> If automation controller is enabled on the platform, continue with Step 4. Otherwise, proceed to Step 6.

4. Select the name of the **Execution environment** or search for one that members of this organization can use to run automation.
5. Enter the name of the **Instance Groups** on which to run this organization.
6. Optional: Enter the **Galaxy credentials** or search from a list of existing ones.
7. Select the **Max hosts** for this organization. The default is 0. When this value is 0, it signifies no limit. If you try to add a host to an organization that has reached or exceeded its cap on hosts, an error message displays:

```
You have already reached the maximum number of 1 hosts allowed for your
organization. Contact your System Administrator for assistance.
```

8. Click **Next**.
9. If you selected more than 1 instance group, you can manage the order by dragging and dropping the instance group up or down in the list and clicking **Confirm**.

> **NOTE:**
> The execution precedence is determined by the order in which the instance groups are listed.

10. Click **Next** and verify the organization settings.
11. Click **Finish**.

View the Organizations list

The **Organizations** page displays the existing organizations for your installation. From here, you can search for a specific organization, filter the list of organizations, or change the sort order for the list.

Procedure

1. From the navigation panel, select **Access Management > Organizations**.

2. In the Search bar, enter an appropriate keyword for the organization you want to search for and click the arrow icon.
3. From the menu bar, you can sort the list of organizations by using the arrows for **Name** to toggle your sorting preference.
4. You can also sort the list by selecting **Name**, **Created** or **Last modified** from the **Sort** list.
5. You can view organization details by clicking an organization **Name** on the **Organizations** page.

Manage access to organizations

You can manage access to an organization by selecting an organization from the **Organizations** list view and selecting the associated tabs for providing access to users, administrators, or teams.

Related information
Assign a user to an organization
Assign an administrator to an organization
Assigning a team to an organization

Assign a user to an organization

You can give a user with access to an organization, and therefore the resources within the organization, by assigning them to the organization and managing the organization roles associated with the user.

You can view a list of users associated with an organization, along with the roles each user is directly assigned, in the organization's **Users** tab. When you manage a user's organization roles in the **Users** tab, you can also see how the user was assigned their roles, whether indirectly, through association with a team, or through direct user assignment by an administrator.

NOTE:
If a user is assigned a "team member" role, this likely indicates that they have an indirectly-assigned role. To see a user's indirectly-assigned roles, click the pencil icon to view and manage roles, and then click the link labeled **View indirectly-assigned organization roles** in the page banner.

To assign a user to an organization, the user must already exist. For more information, see Creating a user. To assign roles to a user, the role must already exist. See Creating a role for more information.

Procedure

1. From the navigation panel, select **Access Management > Organizations**.

2. From the **Organizations** list view, select the organization to which you want to add a user.
3. Click the **Users** tab, then click **Assign Users** to add users.
4. Select one or more users from the list by clicking the checkbox next to the name to add them as members.
5. Click **Next**.
6. Select the roles you want the selected user to have. Scroll down for a complete list of roles.

> **NOTE:**
> Ensure that you are selecting the desired role within the correct component context, because resources like projects and credentials can be associated with both Automation Execution (automation controller) and Automation Decisions (Event-Driven Ansible).

7. Click **Next** to review the roles settings.
8. Click **Finish** to apply the roles to the selected users, and to add them as members. The **Add roles** dialog displays the updated roles assigned for each user.

> **NOTE:**
> A user with roles associated with an organization loses those roles if they are removed from the organization.

9. To remove a particular user from the organization, select **Remove user** from the **More Actions** ⋮ list next to the user. This launches a confirmation dialog, asking you to confirm the removal. Note that removing a user from an organization will also remove all organization roles that the user is indirectly assigned from that specific organization.
10. To manage roles for users in an organization, click the ⚙ icon next to the user and select **Manage roles**. You can manage organization roles that are directly assigned to a user by selecting or clearing the checkboxes. Double-check the component column to ensure you are selecting the desired role in the correct component context.

> **NOTE:**
> From this screen, you can view, but not manage, indirectly-assigned roles that a user has inherited from a team assignment. To view indirectly-assigned roles, along with the team assignment they originated from, click **View indirectly-assigned organization roles** link in the banner beneath the page heading. To manage roles indirectly assigned to a user through a team assignment, manage that team's role assignments or remove the user from that team.

Assign an administrator to an organization

Assign existing users as administrators to an organization so they can manage its membership and settings. This allows designated administrators to create new users and teams and grant permissions.

Procedure

1. From the navigation panel, select **Access Management > Organizations**.
2. From the Organizations list view, select the organization to which you want to add a user, administrator, or team.
3. Click the **Administrators** tab.
4. Click **Add administrators**.
5. Select the users from the list by clicking the checkbox next to the name to assign the administrator role to them for this organization.
6. Click **Add administrators**.
7. To remove a particular administrator from the organization, select **Remove administrator** from the **More actions** ⋮ list next to the administrator name. This launches a confirmation dialog asking you to confirm the removal.

> **NOTE:**
> If the user has been added as a member to this organization, they will continue to be a member of this organization. However, if they were added to the organization when the administrator assignment was made, they are removed from the organization.

Assigning a team to an organization

You can give a team access to an organization, and to the resources within that organization, by assigning roles to the team in the organization's **Teams** tab. All users who are part of a team assigned to the organization will inherit the team's organization role assignments.

To assign roles to a team, the team must already exist in the organization. For more information, see Creating a team. To assign roles for a team, the role must already exist. See Creating a role for more information.

Procedure

1. From the navigation panel, select **Access Management > Organizations**.
2. From the Organizations list view, select the organization to which you want to assign team access.
3. Click the **Teams** tab. If no teams exist, click **Create team** to create a team and assign it to this organization.

4. Click **Assign roles**.
5. Select the roles you want the selected team to have. Scroll down for a complete list of roles.

> **NOTE:**
> Ensure that you are selecting the desired role within the correct component context, because resources like projects and credentials can be associated with both Automation Execution (automation controller) and Automation Decisions (Event-Driven Ansible).

6. Click **Next** to review the roles settings.
7. Click **Finish** to apply the roles to the selected teams. The Assign roles dialog displays the updated roles assigned for each team.
8. Click **Close**.

> **NOTE:**
> A team with associated roles retains them if they are reassigned to another organization.

9. To manage roles for teams in an organization, click the ⚙ icon next to the user and select **Manage roles**.

Delete an organization

Before you can delete an organization, you must be an Organization administrator or System administrator. When you delete an organization, the organization, team, users and resources are permanently removed from Ansible Automation Platform.

> **NOTE:**
> When you try to delete items that are used by other resources, a message is displayed warning you that the deletion might impact other resources and prompts you to confirm the deletion. Some screens contain items that are invalid or have been deleted previously, and will fail to run.

Procedure

1. From the navigation panel, select **Access Management > Organizations**.
2. Click the ⋮ icon next to the organization you want removed and select **Delete organization**.
3. Select the confirmation checkbox and click **Delete organizations** to proceed with the deletion. Otherwise, click **Cancel**.

NOTE:
You can delete multiple organizations by selecting the checkbox next to each organization you want to remove, and selecting **Delete selected organizations** from the **More actions** ⋮ list on the menu bar.

Assign notifiers and execution environments to organizations

When automation controller is enabled on the platform, you can review any notifier integrations you have set up and manage their settings within the organization resource.

Procedure

1. From the navigation panel, select **Access Management > Organizations**.
2. From the **Organizations** list view, select the organization whose notifications you want to manage.
3. Select the **Notification** tab.
4. Use the toggles to enable or disable the notifications to use with your particular organization.
5. If no notifiers have been set up, select **Automation Execution > Administration > Notifiers** from the navigation panel.

Work with execution environments

When automation controller is enabled on the platform, you can review any execution environments you have set up and manage their settings within the organization resource.

For more information about execution environments, see Define, create, and build execution environments.

Procedure

1. From the navigation panel, select **Access Management > Organizations**.
2. From the Organizations list view, select the organization whose execution environments you want to manage.
3. Select the **Execution Environments** tab.

4. If no execution environments are available, click **Create execution environment** to create one. Alternatively, you can create an execution environment from the navigation panel by selecting **Automation Execution > Infrastructure > Execution Environments**.
5. Click **Create execution environment**.

> **NOTE:**
> After creating a new execution environments, return to **Access Management > Organizations** and select the organization in which you created the execution environment to update the list on that tab.

6. Select the execution environments to use with your particular organization.

Bulk-assign roles to users with teams

As an administrator, you can use teams to bulk-assign roles to users that need to share the same access.

A team is a subdivision of an organization that groups users and roles together for specific resources. Teams offer a means to implement role-based access control schemes and delegate responsibilities across organizations by allowing you to grant access to users in bulk. For example, you can grant resource access to a team, and therefore to all the users in the team, rather than granting access to each individual user on the team.

You can create as many teams as needed for your organization. Teams can only be assigned to one organization while an organization can be made up of multiple teams. Each team can be assigned roles, the same way roles are assigned for users. Teams can also scalably assign ownership for credentials, preventing multiple interface click-throughs to assign the same credentials to the same user.

View the Teams list

The **Teams** page displays the existing teams for your installation. From here, you can search for a specific team, filter the list of teams by team name or organization, or change the sort order for the list.

Procedure

1. From the navigation panel, select **Access Management > Teams**.
2. In the **Search** bar, enter an appropriate keyword for the team you want to search for and click the arrow icon.

3. From the menu bar, you can sort the list of teams by using the arrows for **Name** and **Organization** to toggle your sorting preference.
4. You can view team details by clicking a team **Name** on the **Teams** page.
5. You can view organization details by clicking the link in the **Organization** column.

Create a team

Manage teams by creating them, assigning an organization, and adding users or administrators. Team members automatically inherit all assigned roles and permissions. Users must exist in the system before they can be added to a team.

Procedure

1. From the navigation panel, select **Access Management > Teams**.
2. Click **Create team**.
3. Enter a **Name** and optionally give a **Description** for the team.
4. Select an **Organization** to be associated with this team.

> **NOTE:**
> Each team can only be assigned to one organization.

5. Click **Create team**. The **Details** page opens, where you can review and edit your team information and access.

Assign users to a team

To assign a user to a team, the user must already have been created. For more information, see Creating a user. Assigning a user to a team adds them as a member only. Use the **Roles** tab to assign a role that gives users on the team resource access.

New user memberships to a team must be added at the platform level.

Procedure

1. From the navigation panel, select **Access Management > Teams**.
2. Select the team to which you want to add users.
3. Select the **Users** tab.
4. Select one or more users from the list by clicking the checkbox next to the name to add them as members of this team.

5. Click **Add users**.

Remove users from a team

You can remove a user from a team from the Team list view.

Procedure

1. From the navigation panel, select **Access Management > Teams**.
2. Select the team from which you want to remove users.
3. Select the **Users** tab.
4. Click the **Remove user** icon next to the user you want to remove as a member of the team.
5. You can delete multiple users by selecting the checkbox next to each user you want to remove, and selecting **Remove selected users** from the **More actions** ⋮ list.

> **NOTE:**
> If the user is a Team administrator, you can remove their membership to the team from the **Administrators** tab.

6. A confirmation dialog asking you to confirm the removal will appear. Confirm the removal. Note that removing a user from a team removes all of that team's role assignments from the user.

Assign administrators to a team

Assign existing users as administrators to a team so they can manage its membership and settings. This allows designated administrators to create new users and grant permissions within the team.

Procedure

1. From the navigation panel, select **Access Management > Teams**.
2. Select the team to which you want to add an administrator.
3. Select the **Administrators** tab and click **Add administrator(s)**.
4. Select one or more users from the list by clicking the checkbox next to the name to add them as administrators of this team.
5. Click **Add administrators**.

Assign roles to a team

You can grant a team granular access to specific resources such as inventories, projects, and job templates by assigning the team roles associated with those particular resources. You can also set permissions at the level of the organization from the Organizations view.

NOTE:
Teams cannot be assigned to an organization through role assignment, nor can teams be assigned organization roles from the Teams view. Refer to the steps provided in Adding a team to an organization for detailed instructions on assigning a team to an organization.

Procedure

1. From the navigation panel, select **Access Management > Teams**.
2. Select the team **Name** to which you want to add roles.
3. Select the **Roles** tab and click **Add roles**.

NOTE:
Ensure that you are selecting the desired role within the correct component context, because resources like projects and credentials can be associated with both Automation Execution (automation controller) and Automation Decisions (Event-Driven Ansible).

4. Select a **Resource type** and click **Next**.
5. Select the resources that you want to give the team role-based access to and click **Next**.
6. Select the roles to apply to the resources and click **Next**.

NOTE:
If you are selecting more than one role in this step, consider creating a custom role that includes all the permissions for this resource type to give the team the correct access.

7. Review the settings and click **Finish**.
8. The **Add roles** dialog displays indicating whether the role assignments were successfully applied. Click **Close** to close the dialog.

Remove roles from a team

Remove roles from a team to revoke access to specific Ansible Automation Platform resources. Updating these permissions helps ensure that team members only keep the necessary access for their tasks within the correct component context.

Procedure

1. From the navigation panel, select **Access Management > Teams**.
2. Select the team **Name** from which you want to remove roles.
3. Select the **Roles** tab.
4. To remove a single role, click the minus icon next to the resource and confirm removal on the dialog that is displayed.

> **NOTE:**
> Ensure that you are selecting the desired role within the correct component context, because resources like projects and credentials can be associated with both Automation Execution (automation controller) and Automation Decisions (Event-Driven Ansible).

5. To remove roles in bulk, select the checkbox next to each resource you want to remove and click **Delete selected roles** from the **More Actions** ⋮ list on the menu bar, then confirm removal and click **Delete role**.

Delete a team

Before you can delete a team, you must have team permissions. When you delete a team, the roles that users inherited from that team are revoked.

Procedure

1. From the navigation panel, select **Access Management > Teams**.
2. To remove a single team, click the minus icon **-** next to the team and confirm removal on the dialog that is displayed.
3. To remove teams in bulk, select the checkbox next to each team that you want to remove, then click the **More Actions** ⋮ icon and select **Delete team**.

View, create, or assign roles to users

A user is an individual or entity that can log in to the platform and perform tasks. Users are fundamental units to which roles can be assigned, either directly by an administrator or indirectly through a team.

WARNING:
Ansible Automation Platform automatically creates a default system administrator user so they can log in and set up Ansible Automation Platformfor their organization. Do not delete this user.

The containerized installer uses this account to register services with platform gateway. Deleting the default system administrator user causes installation and upgrade operations to fail.

If you have already deleted the default system administrator user, you must set the `gateway_admin_user` variable in your installer inventory file to specify an alternative system administrator account before running the installation program.

You can sort or search the User list by **Username**, **First name**, **Last name**, or **Email**. Click the arrows in the header to toggle your sorting preference. You can view **User type** and **Email** beside the user name on the Users page.

Related information
Platform gateway variables

View the Users list

The **Users** page displays the existing users for your installation. From here, you can search for a specific user, filter the list of users, or change the sort order for the list.

When user accounts have been migrated to Ansible Automation Platform 2.6 during the upgrade process, these accounts are also displayed in the **Users** list view. You can see whether these users have administrator privileges by editing the account. See Editing a user for instructions.

Procedure

1. From the navigation panel, select **Access Management > Users**.
2. In the **Search** bar, enter an appropriate keyword for the user you want to search for and click the arrow icon.
3. From the menu bar, you can sort the list of users by using the arrows for **Username**, **Email**, **First name**, **Last name** or **Last login** to toggle your sorting preference.
4. You can view user details by selecting a **Username** from the **Users** list view.

Create a user

You can create three types of users in Ansible Automation Platform:

Normal user

Normal users have read and write access limited to the resources (such as inventory, projects, and job templates) for which that user has been granted the appropriate roles and privileges. Normal users are the default type of user when no other **User type** is specified.

Ansible Automation Platform Administrator

An administrator (also known as a Superuser) has full system administration privileges, with full read and write privileges over the entire installation. An administrator is typically responsible for managing all aspects of and delegating responsibilities for day-to-day work to various users.

Ansible Automation Platform Auditor

Auditors have read-only capability for all objects within the environment.

Procedure

1. From the navigation panel, select **Access Management > Users**.
2. Click **Create user**.
3. Enter the details about your new user in the fields on the **Create user** page. Fields marked with an asterisk (*) are required.
4. Normal users are the default when no **User type** is specified. To define a user as an administrator or auditor, select a **User type** from the drop-down menu.

> **NOTE:**
> If you are modifying your own password, log out and log back in for the change to take effect.

5. Select the **Organization** to be assigned for this user. For information about creating a new organization, see Creating an organization.
6. Click **Create user**.

   When the user is successfully created, the **User** details screen opens. From here, you can review and change the user's teams, roles, tokens and other membership details.

> **NOTE:**
> If the user is not newly-created, the details screen displays the user's last login activity.

Next steps

If you log in as yourself, and view the details of your user profile, you can manage tokens from your user profile by selecting the **Tokens** tab. For more information, see Adding a token.

Edit a user

You can change user account properties after creation, including password, user type, and organizational assignment. Only platform administrators and organization administrators can change a user's email address.

To see whether a user had service level auditor privileges, you must refer to the API.

NOTE:
After upgrading to 2.6, users previously designated as automation controller administrators are labeled as platform administrators in the **User type** column in the Users list view. Automation hub administrators are labeled as **Normal** in the **User Type** column.

Procedure

1. From the navigation panel, select **Access Management > Users**.
2. Click the **Pencil** icon next to the user you want to edit and select **Edit user**.
3. The **Edit** user page is displayed where you can change user details such as **Password**, **Email**, **User type**, and **Organization**.
4. After your changes are complete, click **Save user**.

Delete a user

Before you can delete a user, you must have normal user or system administrator permissions. When you delete a user account, the name and email of the user are permanently removed from Ansible Automation Platform.

Procedure

1. From the navigation panel, select **Access Management > Users**.
2. To delete a single user, select the **More Actions** ⋮ icon next to the user you want to remove and select **Delete user**.
3. To bulk delete users, select the checkbox next to each user you want to remove, and then from the **More Actions** ⋮ list, click **Delete users**.

Assign roles to a user

You can grant users granular access to specific resources such as inventories, projects, or job templates by assigning users roles.

You can view and manage roles that were assigned directly to a user by an administrator in the user's **Roles** tab.

You can view roles that a user inherited from a team assignment in the **View indirectly assigned roles** link in the page banner. You cannot directly manage an indirectly-assigned role. You can only manage indirectly-assigned roles by editing the team's role assignments, or by removing the user from the team.

NOTE:
Users cannot be assigned to an organization through role assignment, nor can you assign users organization roles from this screen. Refer to the steps provided in Adding a user to an organization for detailed instructions on assigning a user to an organization.

Roles are labeled with their associated Ansible Automation Platform component and function. These components align with Ansible Automation Platform services and the side navigation structure in the user interface. Component labels can be understood as follows:

- **Automation Execution** refers to automation controller
- **Automation Decisions** refers to Event-Driven Ansible
- **Automation Content** refers to automation hub

When assigning roles, ensure that you are selecting the required resource in the correct component context, because resources such as projects and credentials can be associated with both Automation Execution and Automation Decisions.

Procedure

1. From the navigation panel, select **Access Management > Users**.
2. From the **Users** list view, click the user to which you want to add roles.
3. Select the **Roles** tab to display the set of roles assigned to this user. These provide the ability to read, change, and administer resources.
4. To add new roles, click **Add roles**.

NOTE:
Ensure that you are selecting the desired role within the correct component context, because resources like projects and credentials can be associated with both Automation Execution (automation controller) and Automation Decisions (Event-Driven Ansible).

5. Select a Resource type and click **Next**.

6. Select the resources that you want to give role-based access to and click **Next**.
7. Select the roles that will be applied to the resources and click **Next**.

> **NOTE:**
> If you are selecting more than one role, consider creating a custom role that includes all the permissions for this resource type so you can give your users the appropriate level of access.

8. Review the settings and click **Finish**. The **Add roles** dialog displays indicating whether the role assignments were successfully applied. Click **Close** to close the dialog.

Remove roles from a user

You can remove a user's roles by editing the user information in the **Roles** tab.

Procedure

1. From the navigation panel, select **Access Management > Users**.
2. Select the user name whose role access you want to remove.
3. Select the **Roles** tab.

> **NOTE:**
> Ensure that you are selecting the desired role within the correct component context, because resources like projects and credentials can be associated with both Automation Execution (automation controller) and Automation Decisions (Event-Driven Ansible).

4. To remove a single role, click the **-** icon next to the role and confirm removal on the dialog that is displayed.
5. To remove multiple roles, select the checkbox next to each role you want to remove and click **Remove selected roles** from the **More actions** ⋮ list on the menu bar. On the dialog that is displayed, confirm removal of the selected roles and click **Remove role**.

Manage user access to resources

Manage user access to Ansible Automation Platform resources via directly assigned or team-inherited roles. Resources vary by function, such as job templates and projects for automation execution, or decision environments and rulebook activations for automation decisions.

Provide team access to a resource

You can grant users access based on their team membership. When you add a user as a member of a team, they inherit access to the roles and resources defined for that team.

NOTE:
Direct team access cannot be granted to **Automation Content > Remote Registries** resources.

Procedure

1. From the navigation panel, click the name of the resource that you want to give a team access to. For example, **Automation Execution > Templates**.
2. On the details page, select the **Team Access** tab.
3. Click **Assign Teams**.
4. Click the checkbox beside the team to assign that team access to your chosen resource and click **Next**.
5. Select the roles you want applied to the team for the chosen resource and click **Next**.
6. Review the settings and click **Finish**. The Assign Teams dialog displays indicating whether the role assignments were successfully applied.
7. You can remove resource access for a team by selecting the **Remove team** icon next to the team. This launches a confirmation dialog, asking you to confirm the removal.

Provide direct user access to a resource

You can directly grant users access to resources, and edit their access after it has been granted.

NOTE:
Direct user access cannot be granted to **Automation Content > Remote Registries** resources.

Procedure

1. From the navigation panel, select a resource that you want to give a team access to. For example, **Automation Execution > Templates**.
2. Select the **User access** tab.
3. Click **Assign users**.

4. Click the checkbox beside the user to assign that user to your chosen resource and click **Next**.
5. Select the roles you want applied to the user for the chosen resource and click **Next**.
6. Review the settings and click **Finish**. The Assign Roles dialog displays indicating whether the role assignments were successfully applied.
7. You can edit a user's access to a resource from the **User Access** tab by clicking the pencil icon next to the user's name and adding or removing directly-assigned roles.
8. You can remove resource access for a user by selecting the **Remove role** icon next to the user. This launches a confirmation dialog asking you to confirm the removal.

View, create, and assign roles to grant user access to resources

Assign roles to teams or users to grant them targeted access to Red Hat Ansible Automation Platform resources. Defining these permissions allows you to safely govern who can view, modify, or execute tasks on resources like projects and inventories.

Roles define permissions for a specific resource, centralizing all access to that resource through the role itself. This design makes roles reusable units that enable administrators to share defined behaviors among many resources or with different users.

As an administrator, you have the option of using default predefined roles, or you can create roles based on your organization's needs.

Display roles

You can display the roles assigned to each component resource from the **Access Management** menu.

Roles are labeled with their associated Ansible Automation Platform component and function. These components align with Ansible Automation Platform services and the side navigation structure in the user interface. Component labels can be understood as follows:

- **Automation Execution** refers to automation controller
- **Automation Decisions** refers to Event-Driven Ansible
- **Automation Content** refers to automation hub

Roles created at the level of the organization can be associated with multiple components because they group together permissions from automation controller (Automation Execution) and Event-Driven Ansible (Automation Decisions). Only organization roles can span multiple components.

A similar role entity for Automation Content is a "system" role, which gives access to all of the specified resource types in Automation Content.

Procedure

1. From the navigation panel, select **Access Management > Roles**.
2. From the table header, you can sort the list of roles by using the arrows for **Name**, **Description**, **Component**, **Resource Type**, and **Role Creation**, or by making sort selections in the **Sort** list.
3. You can filter the list of roles by selecting **Name**, **Editable**, or **Component** from the filter list and clicking the arrow.

Create a role

Ansible Automation Platform services provide a set of predefined roles with permissions enough for standard automation tasks. It is also possible to configure custom roles that define access permissions to a resource.

If the default predefined roles for a resource type do not give the necessary permissions, you can create custom roles for an organization. Creating a custom role reduces complexity by consolidating all required permissions into a single assignment per resource or resource type, eliminating the need to assign multiple roles to a user or team.

Procedure

1. From the navigation panel, select **Access Management > Roles**.
2. Click **Create role**.
3. Provide a **Name** and a short **Description** for the role. The name and description should be unique and specific to the role's intended use and permissions to give context when assigning the role.
4. Select a **Resource Type**. Ensure that you are selecting the required resource in the correct component context, because resources such as projects and credentials can be associated with both Automation Execution and Automation Decisions.
5. Select the **Permissions** you want assigned to this role from the drop-down menu.
6. Click **Create role** to create your new role.

Edit a role

Default roles are predefined in the platform and cannot be changed; however, you can modify custom roles from the **Roles** list view. The **Role Creation** column in the **Roles** list view indicates whether a role is a default role that cannot be changed, or a custom role that can be modified.

Procedure

1. From the navigation panel, select **Access Management > Roles**.
2. Click the **Edit role** icon next to the role you want and modify the role settings as needed.
3. Click **Save role** to save your changes.

Delete a role

You cannot delete default roles; however, you can delete custom roles from the **Roles** list view.

Procedure

1. From the navigation panel, select **Access Management > Roles**.
2. Click the **More Actions** icon ⋮ next to the role you want and select **Delete role**.
3. To delete roles in bulk, select the roles you want to delete from the **Roles** list view, click the **More Actions** icon ⋮, and select **Delete roles**.

Configure an external secret management system for automation

Configure machine and cloud credentials to allow your automation to securely access external services and machines. Encrypting and storing sensitive values like SSH keys and API tokens in the database helps ensure your authentication details remain protected.

With external credentials backed by credential plugins, you can map credential fields (such as a password or an SSH Private key) to values stored in a `secret management system` instead of providing them to automation controller directly.

Automation controller provides a secret management system that include integrations for:

- AWS Secrets Manager Lookup
- Centrify Vault Credential Provider Lookup
- *CyberArk Central Credential Provider* Lookup (CCP)
- CyberArk Conjur Secrets Manager Lookup
- HashiCorp Vault *Key-Value* Store (KV)
- HashiCorp Vault SSH Secrets Engine
- Microsoft Azure *Key Management System* (KMS)
- Thycotic DevOps Secrets Vault

- Thycotic Secret Server
- GitHub app token lookup

These external secret values are fetched before running a playbook that needs them.

Related information
Configure credentials to authenticate remote systems and services

Configuring and linking secret lookups

automation controller can be configured to retrieve secrets from third-party secret management systems, such as HashiCorp Vault, AWS Secrets Manager, CyberArk Conjur, and others.

Learn how to configure automation controller to retrieve secrets from third-party systems by linking credential fields to external credentials that contain the necessary information to authenticate and retrieve secrets from these systems.

When pulling a secret from a third-party system, you are linking credential fields to external systems. To link a credential field to a value stored in an external system, select the external credential corresponding to that system and provide `metadata` to look up the required value. The metadata input fields are part of the external credential type definition of the source credential.

Automation controller provides a credential plugin interface for developers, integrators, system administrators, and power-users with the ability to add new external credential types to extend it to support other secret management systems.

Use the following procedure to use automation controller to configure and use each of the supported third-party secret management systems.

Procedure

1. Create an external credential for authenticating with the secret management system. At minimum, give a name for the external credential and select one of the following for the **Credential type** field:
    - AWS Secrets Manager Lookup
    - Centrify Vault Credential Provider Lookup
    - CyberArk Central Credential Provider (CCP) Lookup
    - CyberArk Conjur Secrets Manager Lookup
    - HashiCorp Vault Secret Lookup
    - HashiCorp Vault Signed SSH
    - Microsoft Azure Key Vault
    - Thycotic DevOps Secrets Vault

- Thycotic Secret Server
- Configuring a GitHub App Installation Access Token Lookup
  In this example, the *Demo Credential* is the target credential.

2. For any of the fields that follow the **Type Details** area that you want to link to the external credential, click the key icon in the input field to link one or more input fields to the external credential along with metadata for locating the secret in the external system.
3. Select the input source to use to retrieve your secret information.
4. Select the credential you want to link to, and click **Next**. This takes you to the **Metadata** tab of the input source. This example shows the Metadata prompt for HashiVault Secret Lookup. Metadata is specific to the input source you select.

   For more information, see the Metadata for credential input sources table.
5. Click **Test** to verify connection to the secret management system. If the lookup is unsuccessful, an error message displays.
6. Click **OK**. You return to the **Details** screen of your target credential.
7. Repeat these steps, starting with Step 3 to complete the remaining input fields for the target credential. By linking the information in this manner, automation controller retrieves sensitive information, such as username, password, keys, certificates, and tokens from the third-party management systems and populates the remaining fields of the target credential form with that data.
8. If necessary, supply any information manually for those fields that do not use linking as a way of retrieving sensitive information. For more information about each of the fields, see the appropriate Credential types.
9. Click **Save**.

Related information
Credential plugins

Metadata for credential input sources

Learn how to apply the information required for the **Metadata** tab of the input source.

AWS Secrets Manager Lookup

Metadata	Description
AWS Secrets Manager Region (required)	Location of the region where the secrets manager is.
AWS Secret Name (required)	Give the AWS secret name that generated by the AWS access key.

Centrify Vault Credential Provider Lookup

Metadata	Description
Account name (required)	Name of the system account or domain associated with Centrify Vault.
System Name	Specify the name used by the Centrify portal.

CyberArk Central Credential Provider Lookup

Metadata	Description
Object Query (Required)	Lookup query for the object.
Object Query Format	Select `Exact` for a specific secret name, or `Regexp` for a secret that has a dynamically generated name.
Object Property	Specifies the name of the property to return. For example, `UserName` or `Address` other than the default of `Content`.
Reason	If required for the object's policy, supply a reason for checking out the secret, as CyberArk logs those.

CyberArk Conjur Secrets Lookup

Metadata	Description
Secret Identifier	The identifier for the secret.
Secret Version	Specify a version of the secret, if necessary, otherwise, leave it empty to use the latest version.

HashiVault Secret Lookup

Metadata	Description
Name of Secret Backend	Specify the name of the KV backend to use. Leave it blank to use the first path segment of the **Path to Secret** field instead.
Path to Secret (required)	Specify the path to where the secret information is stored; for example, `/path/username`.
Key Name (required)	Specify the name of the key to look up the secret information.
Secret Version (V2 Only)	Specify a version if necessary, otherwise, leave it empty to use the latest version.

HashiCorp Signed SSH

Metadata	Description
Unsigned Public Key (required)	Specify the public key of the certificate you want to have signed. It needs to be present in the authorized keys file of the target hosts.
Path to Secret (required)	Specify the path to where the secret information is stored; for example, `/path/username`.
Role Name (required)	A role is a collection of SSH settings and parameters that are stored in Hashi vault. Typically, you can specify some with different privileges or timeouts, for example. So you could have a role that is permitted to get a certificate signed for root, and other less privileged ones, for example.
Valid Principals	Specify a user (or users) other than the default, that you are requesting vault to authorize the cert for the stored key. Hashi vault has a default user for whom it signs, for example, ec2-user.

Microsoft Azure KMS

Metadata	Description
Secret Name (required)	The name of the secret as it is referenced in Microsoft Azure's Key vault app.
Secret Version	Specify a version of the secret, if necessary, otherwise, leave it empty to use the latest version.

Thycotic DevOps Secrets Vault

Metadata	Description
Secret Path (required)	Specify the path to where the secret information is stored, for example, /path/username.

Thycotic Secret Server

Metadata	Description
Secret ID (required)	The identifier for the secret.
Secret Field	Specify the field to be used from the secret.

Integrate third-party secret management systems

Integrating third-party secret management systems with Ansible Automation Platform lets you store credentials in your existing infrastructure instead of providing them directly to the platform.

Integrating third-party secret management helps you to:

- Use existing secret management infrastructure: Store automation credentials in your enterprise secret management system instead of providing them directly to the platform.
- Maintain centralized credential storage: Manage automation credentials alongside other organizational credentials in a single secret management system.
- Choose the system that fits your needs: Select from multiple supported secret management systems including CyberArk, HashiCorp Vault, Microsoft Azure, and others.

How Ansible Automation Platform supports third-party secret management

Ansible Automation Platform integrates with external secret management systems. Map credential fields to values stored in your secret management system. The platform fetches these external secret values before running a playbook that needs them.

AWS Secrets Manager lookup

This plugin enables Amazon Web Services to be used as a credential input source to pull secrets from the Amazon Web Services Secrets Manager. The AWS Secrets Manager provides similar service to Microsoft Azure Key Vault, and the AWS collection provides a lookup plugin for it.

When AWS Secrets Manager lookup is selected for **Credential type**, give the following metadata to configure your lookup:

- **AWS Access Key** (required): give the access key used for communicating with AWS key management system
- **AWS Secret Key** (required): give the secret as obtained by the AWS IAM console

Centrify Vault Credential Provider Lookup

Centrify Vault Credential Provider Lookup enables automation controller to retrieve secrets from Centrify Vault when executing jobs. This integration uses the Centrify Vault API to look up secrets at job runtime.

You need the Centrify Vault web service running to store secrets for this integration to work. When you select **Centrify Vault Credential Provider Lookup** for **Credential Type**, give the following metadata to configure your lookup:

- **Centrify Tenant URL** (required): give the URL used for communicating with Centrify's secret management system
- **Centrify API User** (required): give the username
- **Centrify API Password** (required): give the password
- **OAuth2 Application ID** : specify the identifier given associated with the OAuth2 client
- **OAuth2 Scope** : specify the scope of the OAuth2 client

CyberArk Central Credential Provider (CCP) Lookup

You can use automation controller to look up credentials stored in CyberArk Central Credential Provider (CCP) and use them in your automation tasks. This integration allows you to securely retrieve secrets from CyberArk CCP during job execution.

The CyberArk Central Credential Provider web service must be running to store secrets for this integration to work. When you select **CyberArk Central Credential Provider Lookup** for **Credential Type**, give the following metadata to configure your lookup:

- **CyberArk CCP URL** (required): give the URL used for communicating with CyberArk CCP's secret management system. It must include the URL scheme such as http or https.
- Optional: **Web Service ID**: specify the identifier for the web service. Leaving this blank defaults to AIMWebService.
- **Application ID** (required): specify the identifier given by CyberArk CCP services.
- **Client Key**: paste the client key if provided by CyberArk.
- **Client Certificate**: include the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines when pasting the certificate, if provided by CyberArk.
- **Verify SSL Certificates**: this option is only available when the URL uses HTTPS. Check this option to verify that the server's SSL/TLS certificate is valid and trusted. For environments that use internal or private CA's, leave this option unchecked to disable verification.

CyberArk Conjur Secrets Manager Lookup

Learn how to configure a CyberArk Conjur Secrets Manager Lookup credential in automation controller.

When you select **CyberArk Conjur Secrets Manager Lookup** for **Credential Type**, give the following metadata to configure your lookup:

- **Conjur URL** (required): give the URL used for communicating with CyberArk Conjur's secret management system. This must include the URL scheme, such as http or https.
- **API Key** (required): give the key given by your Conjur admin
- **Account** (required): the organization's account name
- **Username** (required): the authenticated user for this service
- **Public Key Certificate**: include the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines when pasting the public key, if provided by CyberArk

HashiCorp Vault Secret Lookup

The HashiCorp Vault secret lookup credential type allows you to retrieve secrets from a HashiCorp Vault server during playbook execution. This integration supports various authentication methods, including Token, AppRole, LDAP, Userpass, Kubernetes, and TLS Certificates.

When you select **HashiCorp Vault Secret Lookup** for **Credential Type**, give the following metadata to configure your lookup:

- **Server URL** (required): give the URL used for communicating with HashiCorp Vault's secret management system.
- **Token**: specify the access token used to authenticate HashiCorp's server.
- **CA Certificate**: specify the CA certificate used to verify HashiCorp's server.
- **AppRole role_id**: specify the ID if using AppRole for authentication.
- **AppRole secret_id**: specify the corresponding secret ID for AppRole authentication.
- **Client Certificate**: specify a PEM-encoded client certificate when using the TLS authentication method, including any required intermediate certificates expected by Hashicorp Vault.
- **Client Certificate Key**: specify a PEM-encoded certificate private key when using the TLS authentication method.
- **TLS Authentication Role**: specify the role or certificate name in Hashicorp Vault that corresponds to your client certificate when using the TLS authentication method. If it is not provided, Hashicorp Vault attempts to match the certificate automatically.
- **Namespace name**: specify the Namespace name (Hashicorp Vault enterprise only).
- **Kubernetes role**: specify the role name when using Kubernetes authentication.
- **Username**: enter the username of the user to be used to authenticate this service.
- **Password**: enter the password associated with the user to be used to authenticate this service.
- **Path to Auth**: specify a path if other than the default path of `/approle`.
- **API Version** (required): select v1 for static lookups and v2 for versioned lookups.

LDAP authentication requires LDAP to be configured in HashiCorp's Vault UI and a policy added to the user. Cubbyhole is the name of the default secret mount. If you have proper permissions, you can create other mounts and write key values to those.

To test the lookup, create another credential that uses Hashicorp Vault lookup.

Related information
Vault documentation for LDAP auth method
Vault documentation for AppRole auth method
Vault documentation for userpass auth method
Vault documentation for Kubernetes auth method
Vault documentation for TLS certificates auth method

HashiCorp Vault Signed SSH

You can use HashiCorp Vault Signed SSH to securely manage SSH credentials for accessing managed nodes in automation controller.

When you select **HashiCorp Vault Signed SSH** for **Credential Type**, give the following metadata to configure your lookup:

- **Server URL** (required): give the URL used for communicating with HashiCorp Signed SSH's secret management system.
- **Token**: specify the access token used to authenticate HashiCorp's server.
- **CA Certificate**: specify the CA certificate used to verify HashiCorp's server.
- **AppRole role_id**: specify the ID for AppRole authentication.
- **AppRole secret_id**: specify the corresponding secret ID for AppRole authentication.
- **Client Certificate**: specify a PEM-encoded client certificate when using the TLS authentication method, including any required intermediate certificates expected by Hashicorp Vault.
- **Client Certificate Key**: specify a PEM-encoded certificate private key when using the TLS authentication method.
- **TLS Authentication Role**: specify the role or certificate name in Hashicorp Vault that corresponds to your client certificate when using the TLS authentication method. If it is not provided, Hashicorp Vault attempts to match the certificate automatically.
- **Namespace name**: specify the Namespace name (Hashicorp Vault enterprise only).
- **Kubernetes role**: specify the role name when using Kubernetes authentication.
- **Username**: enter the username of the user to be used to authenticate this service.
- **Password**: enter the password associated with the user to be used to authenticate this service.
- **Path to Auth**: specify a path if other than the default path of `/approle`.

Related information
Vault documentation for AppRole Auth Method
Vault documentation for Kubernetes auth method
Vault documentation for TLS certificates auth method

Microsoft Azure Key Vault

Use the Microsoft Azure Key Vault lookup to retrieve secrets from Microsoft Azure's Key Management System (KMS) within your automation controller environment.

When you select **Microsoft Azure Key Vault** for **Credential Type**, give the following metadata to configure your lookup:

- **Vault URL (DNS Name)** (required): give the URL used for communicating with Microsoft Azure's key management system
- **Client ID** (required): give the identifier as obtained by Microsoft Entra ID
- **Client Secret** (required): give the secret as obtained by Microsoft Entra ID
- **Tenant ID** (required): give the unique identifier associated with an Microsoft Entra ID instance within an Azure subscription
- **Cloud Environment**: select the applicable cloud environment to apply

Thycotic DevOps Secrets Vault

Thycotic DevOps Secrets Vault is a secrets management system that enables secure storage and retrieval of sensitive information such as passwords, API keys, and certificates.

When you select **Thycotic DevOps Secrets Vault** for **Credential Type**, give the following metadata to configure your lookup:

- **Tenant** (required): give the URL used for communicating with Thycotic's secret management system
- **Top-level Domain (TLD)**: give the top-level domain designation, for example .com, .edu, or .org, associated with the secret vault you want to integrate
- **Client ID** (required): give the identifier as obtained by the Thycotic secret management system
- **Client Secret** (required): give the secret as obtained by the Thycotic secret management system

Thycotic Secret Server

Thycotic Secret Server is a secrets management system that enables secure storage and retrieval of sensitive information such as passwords, API keys, and certificates.

Give the following metadata to configure your lookup When you select **Thycotic Secrets Server** as **Credential Type**:

- **Secret Server URL** (required): give the URL used for communicating with the Thycotic Secrets Server management system
- **Username** (required): specify the authenticated user for this service
- **Domain**: give the (application) user domain
- **Password** (required): give the password associated with the user

Configuring a `GitHub App Installation Access Token Lookup`

Learn how to configure a `GitHub App Installation Access Token Lookup` credential in automation controller.

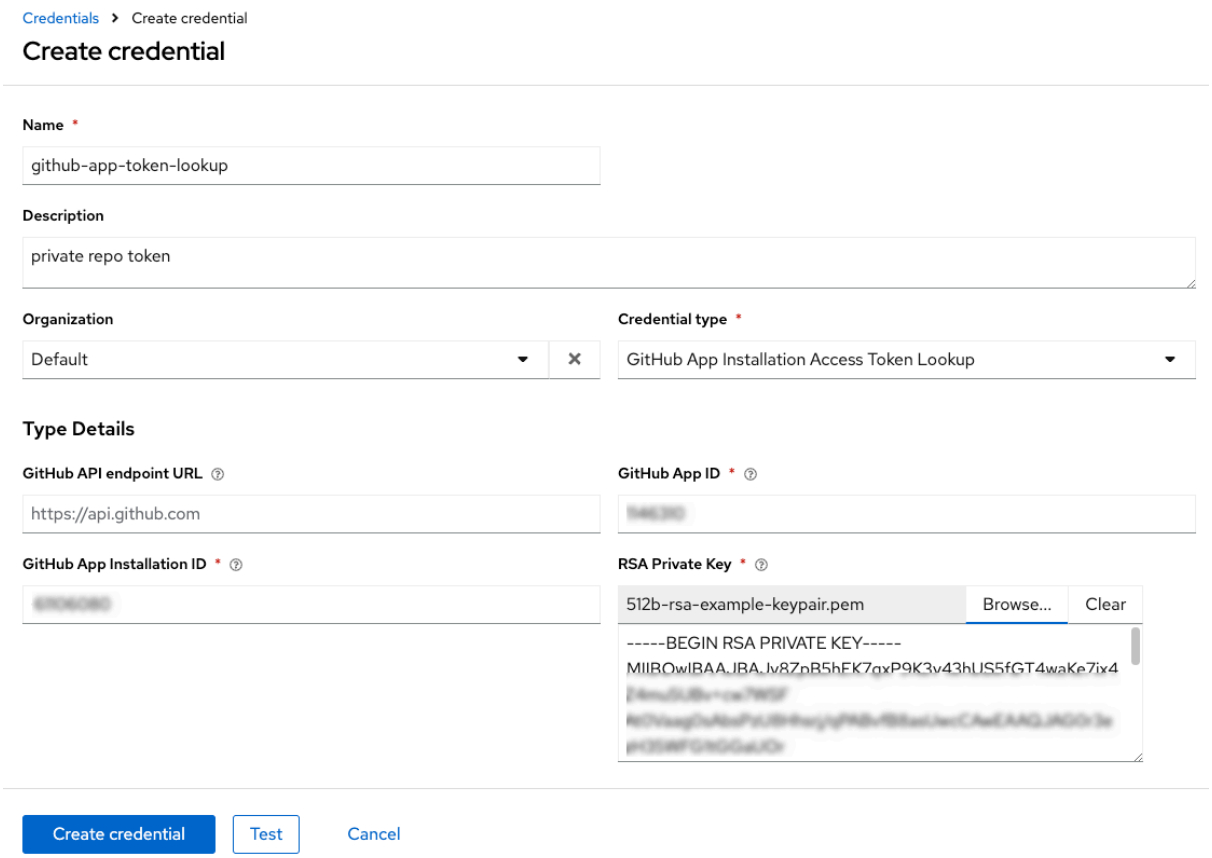
With this plugin you can use a private GitHub App RSA key as a credential input source to pull access tokens from GitHub App installations. Platform gateway uses existing GitHub authorization from organizations' GitHub repositories.

For more information, see Generating an installation access token for a GitHub App.

Procedure

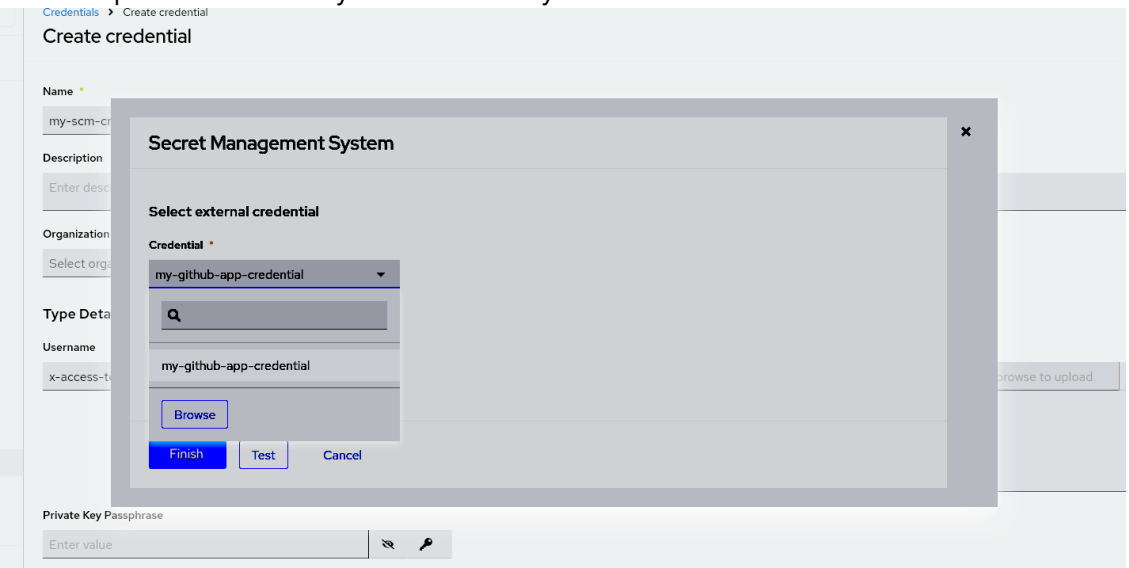
1. Create a lookup credential that stores your secrets. For more information, see See "Create new credentials" on page 1126.
2. Select **GitHub App Installation Access Token Lookup** for **Credential type**, and enter the following attributes to properly configure your lookup:
   - **GitHub App ID**: Enter the App ID provided by your instance of GitHub, this is what is used to authenticate.
   - **GitHub App Installation ID**: Enter the ID of the application into your target organization where the access token is scoped. You must set it up to have access to your target repository.
   - **RSA Private Key**: Enter the generated private key that your GitHub instance generated. You can get it from the GitHub App maintainer within GitHub. For more information, see Managing private keys for GitHub Apps.
3. Click **Create credential** to confirm and save the credential.

   The following is an example of a configured **GitHub App Installation Access Token Lookup** credential:



4. Create a target credential that searches for the lookup credential. To use your lookup in a private repository, select **Source Control** as your **Credential type**. Enter the following attributes to properly configure your target credential:

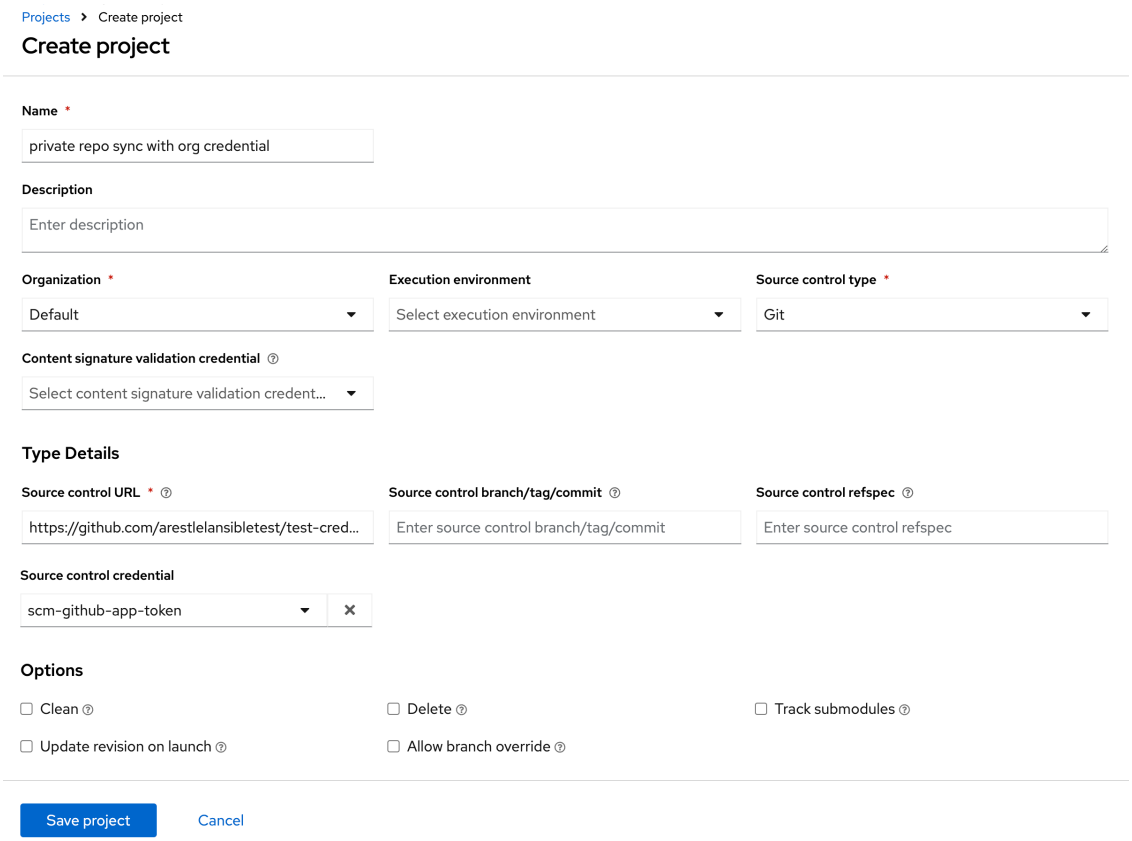
   - **Username**: Enter the username `x-access-token`.
   - **Password**: Click the icon for managing external credentials in the input field. You are prompted to set the input source to use to retrieve your secret information. This is the lookup credential that you have already created.



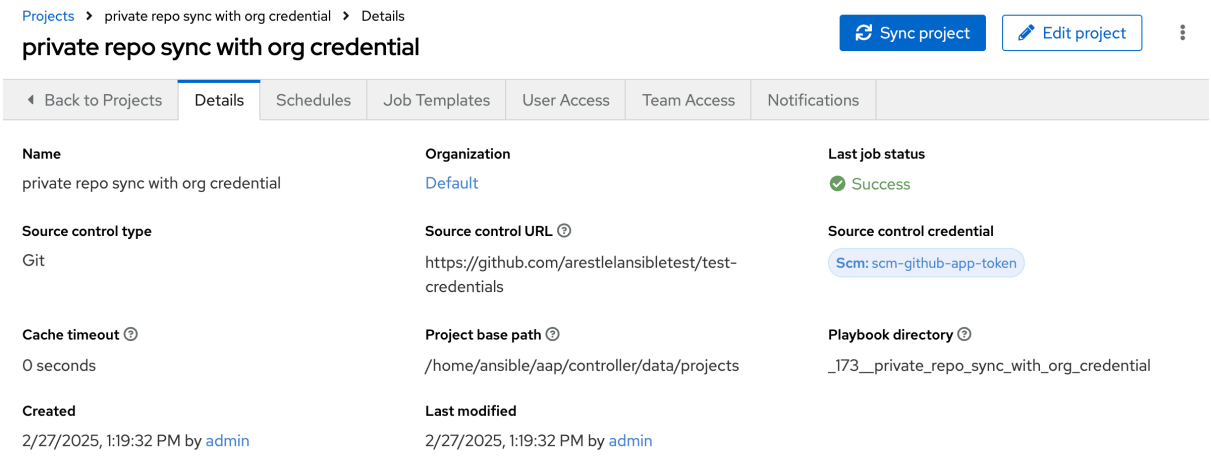
5. Enter an optional description for the metadata requested and click **Finish**.
6. Click **Create credential** to confirm and save the credential.

7. Verify both your lookup credential and your target credential are now available on the **Credentials** list view. To use the target credential in a project, create a project and enter the following information:

    - **Name**: Enter the name for your project.
    - **Organization**: Select the name of the organization from the drop-down menu..
    - **Execution environment**: Optionally select an execution environment, if applicable.
    - **Source control type**: If you are syncing with a private repository, select **Git** for your source control.
      The **Type Details** view opens for additional input. Enter the following information:
    - **Source control URL**: Enter the URL of the private repository you want to access.
      The other related fields pertaining to **branch/tag/commit** and **refspec** are not relevant for use with a lookup credential.
    - **Source control credential**: Select the target credential that you have already created.
      The following is an example of a configured target credential in a project:



8. Click **Create project** and the project sync automatically starts. The project **Details** tab displays the progress of the job:



If your project sync fails, you might have to manually re-enter `https://api.github.com` in the **GitHub API endpoint URL** field from Step 2 and re-run your project sync.

Understand secret handling

Automation controller manages three sets of secrets:

- User passwords for local automation controller users.
- Secrets for automation controller operational use, such as database password or message bus password.
- Secrets for automation use, such as SSH keys, cloud credentials, or external password vault credentials.

NOTE:
You must have 'local' user access for the following users:

- postgres
- awx
- redis
- receptor
- nginx

User passwords for local users

Automation controller hashes local automation controller user passwords with the PBKDF2 algorithm by using a SHA256 hash. Users who authenticate by external account mechanisms, such as LDAP, SAML, and OAuth, do not have any password or secret stored.

Secret handling for operational use

Learn how automation controller handles operational secrets that are required for the service to run properly.

The operational secrets found in automation controller are as follows:

- `/etc/tower/SECRET_KEY`: A secret key used for encrypting automation secrets in the database. If the `SECRET_KEY` changes or is unknown, you cannot access encrypted fields in the database.
- `/etc/tower/tower.{cert,key}`: An SSL/TLS certificate and key for the automation controller web service. The system installs self-signed certificate or key by default; you can give a locally appropriate certificate and key.
- A database password in `/etc/tower/conf.d/postgres.py` and a message bus password in `/etc/tower/conf.d/channels.py`.

The system stores these secrets unencrypted on the automation controller server, because they must be read by the automation controller service at startup in an automated fashion. UNIX permissions protect all secrets, restricting them to root and the automation controller awx service user.

If you need to hide these secrets, the files that these secrets are read from are interpreted by Python. You can adjust these files to retrieve these secrets by some other mechanism anytime a service restarts. This is a customer provided modification that might need to be reapplied after every upgrade. Red Hat Support and Red Hat Consulting have examples of such modifications.

NOTE:
If the secrets system is down, automation controller cannot get the information and can fail in a way that is recoverable once the service is restored. Using some redundancy on that system is highly recommended.

If you believe the `SECRET_KEY` that automation controller generated for you has been compromised and needs to be regenerated, you can run a tool from the installation program that behaves much as the automation controller backup and restore tool.

IMPORTANT:
Ensure that you backup your automation controller database before you generate a new secret key.

To generate a new secret key:

1. Follow the procedure described in the Backing up and Restoring section.
2. Use the inventory from your install (the same inventory with which you run backups and restores), and run the following command:

```
setup.sh -k.
```

A backup copy of the earlier key is saved in `/etc/tower/`.

Secret handling for automation use

Automation controller stores a variety of secrets in the database that are either used for automation or are a result of automation.

These secrets include the following:

- All secret fields of all credential types, including passwords, secret keys, authentication tokens, and secret cloud credentials.
- Secret tokens and passwords for external services defined automation controller settings.
- "password" type survey field entries.

To encrypt secret fields, automation controller uses AES in CBC mode with a 256-bit key for encryption, PKCS7 padding, and HMAC by using SHA256 for authentication.

The encryption or decryption process derives the AES-256 bit encryption key from the `SECRET_KEY`, the field name of the model field and the database assigned auto-incremented record ID. Therefore, if any attribute used in the key generation process changes, the automation controller fails to correctly decrypt the secret.

Automation controller is designed so that:

- The `SECRET_KEY` is never readable in playbooks that automation controller launches.
- These secrets are never readable by automation controller users.
- No secret field values are ever made available by the automation controller REST API.

If a secret value is used in a playbook, it is recommended that you use `no_log` on the task so that it is not accidentally logged.

Internal, external, and managed node connections

Automation controller allows for connections to internal services, external access, and managed nodes.

NOTE:
You must have 'local' user access for the following users:

- postgres
- awx
- redis
- receptor
- nginx

Internal services

Automation controller connects to the following services as part of internal operation:

PostgreSQL database

The connection to the PostgreSQL database is done by password authentication over TCP, either through localhost or remotely (external database). This connection can use PostgreSQL's built-in support for SSL/TLS, as natively configured by the installation program support. SSL/TLS protocols are configured by the default OpenSSL configuration.

A Redis key or value store

The connection to Redis is over a local UNIX socket, restricted to the awx service user.

External access

Automation controller is accessed using standard HTTP/HTTPS on standard ports, provided by Nginx. A self-signed certificate or key is installed by default; you can provide a locally appropriate certificate and key.

SSL/TLS algorithm support is configured in the `/etc/nginx/nginx.conf` configuration file. An "intermediate" profile is used by default, that you can configure. You must reapply changes after each update.

Managed nodes

Automation controller connects to managed machines and services as part of automation. All connections to managed machines are done by standard secure mechanisms, such as SSH, WinRM, or SSL/TLS.

Each of these inherits configuration from the system configuration for the feature in question, such as the system OpenSSL configuration.

Internal services

Automation controller connects to the following services as part of internal operation:

PostgreSQL database

The connection to the PostgreSQL database is done by password authentication over TCP, either through localhost or remotely (external database). This connection can use PostgreSQL's built-in support for SSL/TLS, as natively configured by the installation program support. SSL/TLS protocols are configured by the default OpenSSL configuration.

A Redis key or value store

The connection to Redis is over a local UNIX socket, restricted to the awx service user.

External access

Automation controller is accessed using standard HTTP/HTTPS on standard ports, provided by Nginx. A self-signed certificate or key is installed by default; you can provide a locally appropriate certificate and key.

SSL/TLS algorithm support is configured in the `/etc/nginx/nginx.conf` configuration file. An "intermediate" profile is used by default, that you can configure. You must reapply changes after each update.

Managed nodes

Automation controller connects to managed machines and services as part of automation. All connections to managed machines are done by standard secure mechanisms, such as SSH, WinRM, or SSL/TLS.

Each of these inherits configuration from the system configuration for the feature in question, such as the system OpenSSL configuration.

Configure automation hub tokens

You must create an API token before you can upload or download collections. The automation hub API token authenticates your `ansible-galaxy` client to the Red Hat automation hub server.

NOTE:
Automation hub does not support basic authentication or authenticating through service accounts. You must authenticate using token management.

Your method for creating the API token differs according to the type of automation hub that you are using:

- Automation hub uses offline token management.
- Private automation hub uses API token management.
- If you are using Keycloak to authenticate your private automation hub, follow the procedure for Create the offline token in automation hub.

Related information
Create the offline token in automation hub
Create the API token in private automation hub
Create the offline token in automation hub

Create the offline token in automation hub

In automation hub, you can create an offline token using **Token management**. The offline token is a secret token used to protect your content, so be sure to store it in a secure location.

NOTE:
Your offline token expires after 30 days of inactivity.

Procedure

1. Navigate to Ansible Automation Platform on the Red Hat Hybrid Cloud Console.
2. From the navigation panel, select **Automation Hub > Connect to Hub**.
3. Under **Offline token**, click **Load Token**.
4. Click the **Copy to clipboard** icon to copy the offline token.
5. Paste the token into a file and store in a secure location.

Next steps

The offline token is now available for configuring automation hub as your default collections server or for uploading collections by using the `ansible-galaxy` command line tool.

Related information
Keeping your offline token active

Create the API token in private automation hub

In private automation hub, you can create an API token using API token management. The API token is a secret token used to protect your content, so be sure to store it in a secure location.

Before you begin

- Valid subscription credentials for Red Hat Ansible Automation Platform.

NOTE:
The API token does not expire.

Procedure

1. Log in to your private automation hub.
2. From the navigation panel, select **Automation Content > API token**.
3. Click **Load Token**.
4. To copy the API token, click the **Copy to clipboard** icon.
5. Paste the API token into a file and store in a secure location.

Next steps

The API token is now available for configuring automation hub as your default collections server or uploading collections using the `ansible-galaxy` command line tool.

Configure credentials to authenticate remote systems and services

Credentials authenticate the automation controller user when launching jobs against machines, synchronizing with inventory sources, and importing project content from a version control system.

You can grant users and teams the ability to use these credentials, without exposing the credential to the user. If a user moves to a different team or leaves the organization, you do not have to re-key all of your systems just because that credential was available in automation controller.

NOTE:
Automation controller encrypts passwords and key information in the database and never makes secret information visible through the API.

How credentials work

Credentials in automation controller store the information required to authenticate to remote systems and services. Credentials include usernames and passwords, SSH keys, tokens, and other sensitive data. Automation controller encrypts sensitive credential data in the database to enhance security.

Automation controller uses SSH to connect to remote hosts. To pass the key from automation controller to SSH, the key must be decrypted before it can be written to a named pipe. Automation controller uses that pipe to send the key to SSH, so that the key is never written to disk. If passwords are used, automation controller handles them by responding directly to the password prompt and decrypting the password before writing it to the prompt.

The **Credentials** page shows credentials that are currently available. The default view is collapsed (Compact), showing the credential name, and credential type.

From this screen you can edit , duplicate or delete ⋮ a credential.

NOTE:
It is possible to create duplicate credentials with the same name and without an organization. However, it is not possible to create two duplicate credentials in the same organization.

Example

1. Create two machine credentials with the same name but without an organization.
2. Use the module `ansible.controller.export` to export the credentials.
3. Use the module `ansible.controller.import` in a different automation execution node.
4. Check the imported credentials.
   When you export two duplicate credentials and then import them in a different node, only one credential is imported.

Create new credentials

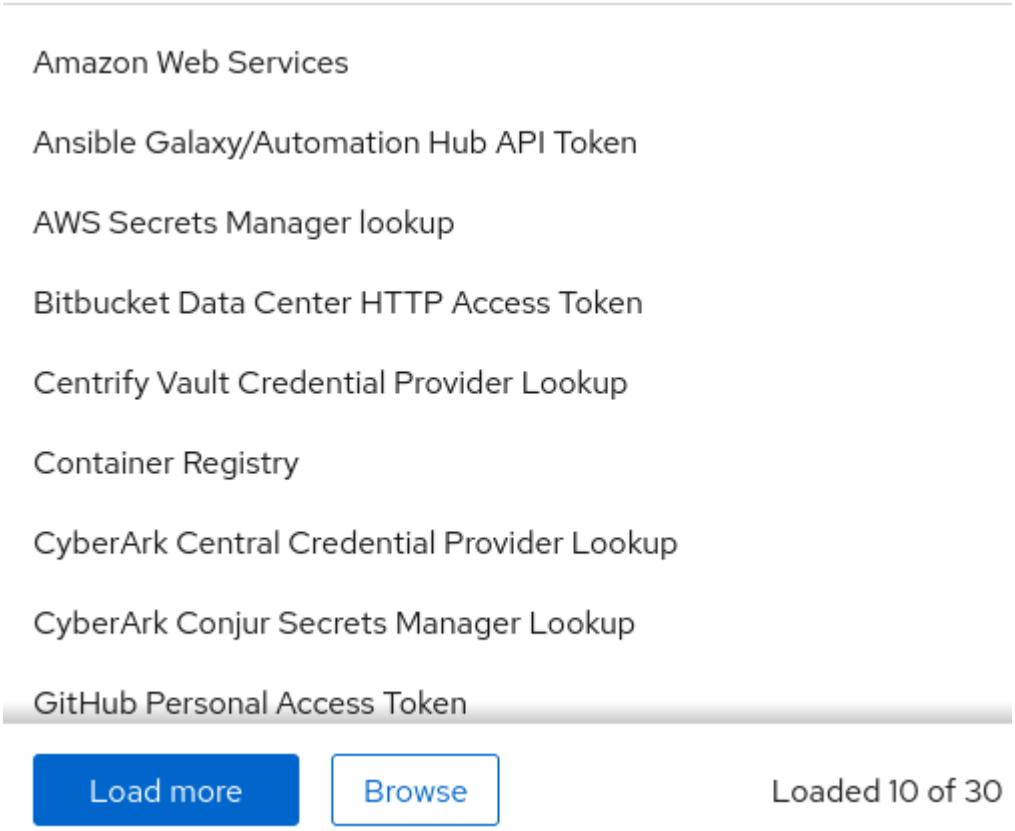
Learn how to create new credentials in Automation controller.

Credentials added to a team are made available to all members of the team. You can also add credentials to individual users.

As part of the initial setup, two credentials are available for your use: Demo Credential and Ansible Galaxy. Use the Ansible Galaxy credential as a template. You can copy this credential, but not edit it. Add more credentials as needed.

Procedure

1. From the navigation panel, select **Automation Execution > Infrastructure > Credentials**.
2. On the **Credentials** page, click **Create credential**.
3. Enter the following information:
   - **Name**: the name for your new credential.
   - (Optional) **Description**: a description for the new credential.
   - Optional **Organization**: The name of the organization with which the credential is associated. The default is **Default**.
   - **Credential type**: enter or select the credential type you want to create.
4. Enter the appropriate details depending on the type of credential selected, as described in Credential types.



5. Click **Create credential**.

Add new users and job templates to existing credentials

You can add new users and job templates to existing credentials in automation controller.

Procedure

1. From the navigation panel, select **Automation Execution > Infrastructure > Credentials**.
2. Select the credential that you want to assign to additional users.
3. Click the **User Access** tab. You can see users and teams associated with this credential and their roles. If no users exist, add them from the **Users** menu. For more information, see Users.
4. Click **Add roles**.
5. Select the user(s) that you want to give access to the credential and click **Next**.
6. From the **Select roles to apply** page, select the roles you want to add to the User.
7. Click **Next**.
8. Review your selections and click **Finish** to add the roles or click **Back** to make changes.

   The **Add roles** window displays stating whether the action was successful.

   If the action is not successful, a warning displays.
9. Click **Close**.
10. The **User Access** page displays the summary information.
11. Select the **Job templates** tab to select a job template to which you want to assign this credential.
12. Chose a job template or select **Create job template** from the **Create template** list to assign the credential to additional job templates.

    For more information about creating new job templates, see Job templates.

Credential types

Ansible Automation Platform supports a number of credential types.

The credential types associated with AWS Secrets Manager, Centrify, CyberArk, HashiCorp Vault, Microsoft Azure Key Vault, and Thycotic are part of the credential plugins capability that enables an external system to lookup your secrets information.

For more information, see Secrets Management System.

Amazon Web Services credential type

Select this credential to enable synchronization of cloud inventory with Amazon Web Services.

Automation controller uses the following environment variables for AWS credentials:

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SECURITY_TOKEN
```

These are fields prompted in the user interface.

Amazon Web Services credentials consist of the AWS **Access Key** and **Secret Key**.

Automation controller provides support for EC2 STS tokens, also known as *Identity and Access Management* (IAM) STS credentials. *Security Token Service* (STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS IAM users.

NOTE:
If the value of your tags in EC2 contain Booleans (`yes/no/true/false`), you must quote them.

WARNING:
To use implicit IAM role credentials, do not attach AWS cloud credentials in automation controller when relying on IAM roles to access the AWS API.

Attaching your AWS cloud credential to your job template forces the use of your AWS credentials, not your IAM role credentials.

Related information
Temporary security credentials in IAM

Access Amazon EC2 credentials in an Ansible Playbook

You can get AWS credential parameters from a job runtime environment:

```
vars:
  aws:
    access_key: '{{ lookup("env", "AWS_ACCESS_KEY_ID") }}'
    secret_key: '{{ lookup("env", "AWS_SECRET_ACCESS_KEY") }}'
    security_token: '{{ lookup("env", "AWS_SECURITY_TOKEN") }}'
```

Ansible Galaxy or automation hub API token credential type

You can create a Ansible Galaxy or private automation hub API token credential in automation controller to access collections published on Ansible Galaxy or an instance of private automation hub.

Select this credential to access Ansible Galaxy or use a collection published on an instance of private automation hub.

Enter the Galaxy server URL on this screen.

Populate the **Galaxy Server URL** field with the contents of the **Server URL** field at Red Hat Hybrid Cloud Console. Populate the **Auth Server URL** field with the contents of the **SSO URL** field at Red Hat Hybrid Cloud Console.

Related information
Use collections with automation hub

AWS secrets manager lookup

Use this credential type so that automation controller can retrieve secrets from AWS Secrets Manager.

This is part of the secret management capability. For more information, see AWS Secrets Manager lookup

BitBucket data center HTTP access token

Bitbucket Data Center is a self-hosted Git repository for collaboration and management. Select this credential type to enable you to use HTTP access tokens in place of passwords for Git over HTTPS.

For further information, see HTTP access tokens in the Bitbucket Data Center documentation..

Centrify Vault Credential Provider Lookup credential type

The Centrify Vault Credential Provider Lookup credential type allows automation controller to retrieve secrets from a Centrify Vault server.

This is considered part of the secret management capability.

For more information, see See "Centrify Vault Credential Provider Lookup" on page 1 111.

Container registry credential type

A Container Registry credential allows automation controller to authenticate to a container registry service to access container images.

Select this credential to enable automation controller to access a collection of container images. For more information, see What is a container registry?.

You must specify a name. The **Authentication URL** field is pre-populated with a default value. You can change the value by specifying the authentication endpoint for a different container registry.

CyberArk Central Credential Provider Lookup credential type

The CyberArk Central Credential Provider (CCP) Lookup credential type allows Automation controller to retrieve credentials from a CyberArk vault by using the Central Credential Provider REST API.

This is considered part of the secret management capability.

For more information, see CyberArk Central Credential Provider (CCP) Lookup.

CyberArk Conjur Secrets Manager Lookup credential type

Automation controller can use CyberArk Conjur Secrets Manager to retrieve secrets for use by the controller and managed nodes.

This is considered part of the secret management capability.

For more information, see CyberArk Conjur Secrets Manager Lookup.

GitHub Personal Access Token credential type

Select this credential to enable you to access GitHub by using a *Personal Access Token* (PAT), which you can get through GitHub.

For more information, see Setting up a GitHub webhook.

GitHub PAT credentials require a value in the **Token** field, which is provided in your GitHub profile settings.

Use this credential to establish an API connection to GitHub for use in webhook listener jobs, to post status updates.

GitLab Personal Access Token credential type

Select this credential to access GitLab by using a *Personal Access Token* (PAT), which you can get through GitLab.

For more information, see Setting up a GitLab webhook.

GitLab PAT credentials require a value in the **Token** field, which is provided in your GitLab profile settings.

Use this credential to establish an API connection to GitLab for use in webhook listener jobs, to post status updates.

Google Compute Engine credential type

Select this credential to enable synchronization of a cloud inventory with Google Compute Engine (GCE).

Automation controller uses the following environment variables for GCE credentials:

```
GCE_EMAIL
GCE_PROJECT
GCE_CREDENTIALS_FILE_PATH
```

These are fields prompted in the user interface:

GCE credentials require the following information:

- **Service Account Email Address**: The email address assigned to the Google Compute Engine **service account**.
- Optional: **Project**: Provide the GCE assigned identification or the unique project ID that you provided at project creation time.
- Optional: **Service Account JSON File**: Upload a GCE service account file. Click **Browse** to browse for the file that has the special account information that can be used by services and applications running on your GCE instance to interact with other Google Cloud APIs. This grants permissions to the service account and virtual machine instances.
- **RSA Private Key**: The PEM file associated with the service account email.

Access Google Compute Engine credentials in an Ansible Playbook

You can access Google Compute Engine (GCE) credentials in an Ansible Playbook by using environment variables.

You can get GCE credential parameters from a job runtime environment:

```
vars:
  gce:
    email: '{{ lookup("env", "GCE_EMAIL") }}'
    project: '{{ lookup("env", "GCE_PROJECT") }}'
    pem_file_path: '{{ lookup("env", "GCE_PEM_FILE_PATH") }}'
```

GPG Public Key credential type

Select this credential type to enable automation controller to verify the integrity of the project when synchronizing from source control.

For more information about how to generate a valid keypair, use the CLI tool to sign content, and how to add the public key to the controller, see Project Signing and Verification.

HashiCorp Vault Secret Lookup credential type

This is considered part of the secret management capability.

For more information, see See "HashiCorp Vault Secret Lookup" on page 1 113.

HashiCorp Vault Signed SSH credential type

The HashiCorp Vault Signed SSH credential type allows automation controller to retrieve signed SSH certificates from a HashiCorp Vault server.

For more information, see *HashiCorp Vault Signed SSH* in Integrate third-party secret management systems.

Red Hat Lightspeed credential type

Select this credential type to enable synchronization of cloud inventory with Red Hat Lightspeed.

Red Hat Lightspeed credentials are the Red Hat Lightspeed **Username** and **Password**, which are the user's Red Hat Customer Portal Account username and password.

The `extra_vars` and `env` injectors for Red Hat Lightspeed are as follows:

```
ManagedCredentialType(
    namespace='insights',
....
....
....

injectors={
        'extra_vars': {
            "scm_username": "{{username}}",
            "scm_password": "{{password}}",
        },
        'env': {
            'INSIGHTS_USER': '{{username}}',
            'INSIGHTS_PASSWORD': '{{password}}',
        },
```

Access machine credentials in an ansible playbook

You can get username and password from Ansible facts:

```
vars:
  machine:
    username: '{{ ansible_user }}'
    password: '{{ ansible_password }}'
```

Machine credential type

Machine credentials enable automation controller to call Ansible on hosts under your management.

You can specify the SSH username, optionally give a password, an SSH key, a key password, or have automation controller prompt the user for their password at deployment time. They define

SSH and user-level privilege escalation access for playbooks, and are used when submitting jobs to run playbooks on a remote host.

The following network connections use **Machine** as the credential type: `httpapi`, `netconf`, and `network_cli`

Machine and SSH credentials do not use environment variables. They pass the username through the ansible `-u` flag, and interactively write the SSH password when the underlying SSH client prompts for it.

Machine credentials require the following inputs:

- **Username**: The username to use for SSH authentication.
- **Password**: The password to use for SSH authentication. This password is stored encrypted in the database, if entered. Alternatively, you can configure automation controller to ask the user for the password at launch time by selecting **Prompt on launch**. In these cases, a dialog opens when the job is launched, promoting the user to enter the password and password confirmation.
- **SSH Private Key**: Copy or drag-and-drop the SSH private key for the machine credential.
- **Private Key Passphrase**: If the SSH Private Key used is protected by a password, you can configure a Key Passphrase for the private key. This password is stored encrypted in the database, if entered. You can also configure automation controller to ask the user for the key passphrase at launch time by selecting **Prompt on launch**. In these cases, a dialog opens when the job is launched, prompting the user to enter the key passphrase and key passphrase confirmation.
- **Privilege Escalation Method**: Specifies the type of escalation privilege to assign to specific users. This is the same as specifying the `--become-method=BECOME_METHOD` parameter, where `BECOME_METHOD` is any of the existing methods, or a custom method you have written. Begin entering the name of the method, and the appropriate name auto-populates.
    - **empty selection**: If a task or play has `become` set to `yes` and is used with an empty selection, then it will default to `sudo`.
    - **sudo**: Performs single commands with superuser (root user) privileges.
    - **su**: Switches to the superuser (root user) account (or to other user accounts).
    - **pbrun**: Requests that an application or command be run in a controlled account and provides for advanced root privilege delegation and keylogging.
    - **pfexec**: Executes commands with predefined process attributes, such as specific user or group IDs.
    - **dzdo**: An enhanced version of sudo that uses RBAC information in Centrify's Active Directory service. For more information, see Centrify's site on DZDO.
    - **pmrun**: Requests that an application is run in a controlled account. See Privilege Manager for Unix 6.0.
    - **runas**: Enables you to run as the current user.

  - **enable**: Switches to elevated permissions on a network device.
  - **doas**: Enables your remote/login user to run commands as another user through the *doas* ("Do as user") utility.
  - **ksu**: Enables your remote/login user to run commands as another user through Kerberos access.
  - **machinectl**: Enables you to manage containers through the `systemd` machine manager.
  - **sesu**: Enables your remote/login user to run commands as another user through the CA Privileged Access Manager.
- **Privilege Escalation Username**: You see this field only if you selected an option for privilege escalation. Enter the username to use with escalation privileges on the remote system.
- **Privilege Escalation Password**: You see this field only if you selected an option for privilege escalation. Enter the password to use to authenticate the user through the selected privilege escalation type on the remote system. This password is stored encrypted in the database. You can also configure automation controller to ask the user for the password at launch time by selecting **Prompt on launch**. In these cases, a dialog opens when the job is launched, promoting the user to enter the password and password confirmation.

NOTE:
You must use sudo password must in combination with SSH passwords or SSH Private Keys, because automation controller must first establish an authenticated SSH connection with the host before invoking `sudo` to change to the sudo user.

WARNING:
Credentials that are used in scheduled jobs must not be configured as **Prompt on launch**.

Microsoft Azure Key Vault credential type

The Microsoft Azure Key Vault credential type allows automation controller to authenticate to Microsoft Azure Key Vault to retrieve secrets for use in playbooks and other automation tasks.

This is considered part of the secret management capability.

For more information, see *Microsoft Azure Key Vault* in Integrate third-party secret management systems.

Microsoft Azure Resource Manager credential type

Select this credential type to enable synchronization of cloud inventory with Microsoft Azure Resource Manager.

Microsoft Azure Resource Manager credentials require the following inputs:

- **Subscription ID**: The Subscription UUID for the Microsoft Azure account.
- **Username**: The username to use to connect to the Microsoft Azure account.
- **Password**: The password to use to connect to the Microsoft Azure account.
- **Client ID**: The Client ID for the Microsoft Azure account.
- **Client Secret**: The Client Secret for the Microsoft Azure account.
- **Tenant ID**: The Tenant ID for the Microsoft Azure account.
- **Azure Cloud Environment**: The variable associated with Azure cloud or Azure stack environments.

These fields are equivalent to the variables in the API.

To pass service principal credentials, define the following variables:

```
AZURE_CLIENT_ID
AZURE_SECRET
AZURE_SUBSCRIPTION_ID
AZURE_TENANT
AZURE_CLOUD_ENVIRONMENT
```

To pass an Active Directory username and password pair, define the following variables:

```
AZURE_AD_USER
AZURE_PASSWORD
AZURE_SUBSCRIPTION_ID
```

You can also pass credentials as parameters to a task within a playbook. The order of precedence is parameters, then environment variables, and finally a file found in your home directory.

To pass credentials as parameters to a task, use the following parameters for service principal credentials:

```
client_id
secret
subscription_id
tenant
azure_cloud_environment
```

Alternatively, pass the following parameters for Active Directory username and password:

```
ad_user
password
subscription_id
```

Access Microsoft Azure Resource Manager credentials in an Ansible Playbook

You can get Microsoft Azure credential parameters from a job runtime environment.

```
vars:
  azure:
    client_id: '{{ lookup("env", "AZURE_CLIENT_ID") }}'
    secret: '{{ lookup("env", "AZURE_SECRET") }}'
    tenant: '{{ lookup("env", "AZURE_TENANT") }}'
    subscription_id: '{{ lookup("env", "AZURE_SUBSCRIPTION_ID") }}'
```

Network credential type

You can create Network credential types in automation controller to manage network devices that use Ansible networking modules.

NOTE:
Select the Network credential type if you are using a *local* connection with *provider* to use Ansible networking modules to connect to and manage networking devices.

When connecting to network devices, the credential type must match the connection type:

- For `local` connections using `provider`, credential type should be **Network**.
- For all other network connections (`httpapi`, `netconf`, and `network_cli`), the credential type should be **Machine**.
  For more information about connection types available for network devices, see Multiple Communication Protocols.

  Automation controller uses the following environment variables for Network credentials:

  ```
  ANSIBLE_NET_USERNAME
  ANSIBLE_NET_PASSWORD
  ```

Provide the following information for network credentials:

- **Username**: The username to use in conjunction with the network device.
- **Password**: The password to use in conjunction with the network device.
- **SSH Private Key**: Copy or drag-and-drop the actual SSH Private Key to be used to authenticate the user to the network through SSH.
- **Private Key Passphrase**: The passphrase for the private key to authenticate the user to the network through SSH.
- **Authorize**: Select this to control whether or not to enter privileged mode.
  - If **Authorize** is checked, enter a password in the **Authorize Password** field to access privileged mode.
    For more information, see Porting Ansible Network Playbooks with New Connection Plugins.

Access network credentials in an Ansible Playbook

When using the **Controller Access Network Credentials** credential type, you can access the username and password parameters in your Ansible Playbook by using the following environment variables:

- `ANSIBLE_NET_USERNAME`
- `ANSIBLE_NET_PASSWORD`

You can get the username and password parameters from a job runtime environment:

```
vars:
  network:
    username: '{{ lookup("env", "ANSIBLE_NET_USERNAME") }}'
    password: '{{ lookup("env", "ANSIBLE_NET_PASSWORD") }}'
```

Multiple communication protocols

Ansible network modules can communicate with network devices by using different protocols.

Because network modules run on the control node instead of on the managed nodes, they can support multiple communication protocols. The communication protocols (XML over SSH, CLI over SSH, or API over HTTPS) selected for each network module depend on the platform and the purpose of the module. Some network modules support only one protocol, while some offer a choice.

The most common protocol is CLI over SSH. You set the communication protocol with the ansible_connection variable:

Value of ansible_connection	Protocol	Requires	Persistent?
`ansible.netcommon.network_cli`	CLI over SSH	network_os setting	yes
`ansible.netcommon.netconf`	XML over SSH	network_os setting	yes
`ansible.netcommon.httpapi`	API over HTTP/HTTPS	network_os setting	yes
`local`	depends on provider	provider setting	no

The `ansible_connection: local` is deprecated. Use one of the persistent connection types listed above instead. With persistent connections, you can define the hosts and credentials only once, rather than in every task. You must also set the `network_os` variable for the specific network platform you are communicating with.

OpenShift or Kubernetes API Bearer Token credential type

Select this credential type to create instance groups that point to a Kubernetes or OpenShift container.

For more information, see Determine where automation runs with instance groups and Control where automation runs with container groups.

Provide the following information for container credentials:

- **OpenShift or Kubernetes API Endpoint** (required): The endpoint used to connect to an OpenShift or Kubernetes container.

- **API authentication bearer token** (required): The token used to authenticate the connection.
- Optional: **Verify SSL**: You can check this option to verify the server's SSL/TLS certificate is valid and trusted. Environments that use internal or private *Certificate Authority* (CA) must leave this option unchecked to disable verification.
- **Certificate Authority data**: Include the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines when pasting the certificate, if provided.

A container group is a type of instance group that has an associated credential that enables connection to an OpenShift cluster. To set up a container group, you must have the following items:

- A namespace you can start into. Although every cluster has a default namespace, you can use a specific namespace.
- A service account that has the roles that enable it to start and manage pods in this namespace.
- If you use execution environments in a private registry, and have a container registry credential associated with them in automation controller, the service account also requires the roles to get, create, and delete secrets in the namespace.
  If you do not want to give these roles to the service account, you can pre-create the `ImagePullSecrets` and specify them on the pod spec for the container group. In this case, the execution environment must not have a Container Registry credential associated, or automation controller attempts to create the secret for you in the namespace.
- A token associated with that service account (OpenShift or Kubernetes Bearer Token)
- A CA certificate associated with the cluster

Creating a service account in an Openshift cluster

Create a service account in an Openshift or Kubernetes cluster to be used to run jobs in a container group through automation controller. After you create the service account, its credentials are provided to automation controller in the form of an Openshift or Kubernetes API bearer token credential.

After you create a service account, use the information in the new service account to configure automation controller.

Procedure

1. To create a service account, download and use the sample service account, `containergroup sa`, and change it as needed to obtain the credentials:

```
 ---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: containergroup-service-account
  namespace: containergroup-namespace
---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: role-containergroup-service-account
  namespace: containergroup-namespace
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
- apiGroups: [""]
  resources: ["pods/log"]
  verbs: ["get"]
- apiGroups: [""]
  resources: ["pods/attach"]
  verbs: ["get", "list", "watch", "create"]
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: role-containergroup-service-account-binding
  namespace: containergroup-namespace
subjects:
- kind: ServiceAccount
  name: containergroup-service-account
  namespace: containergroup-namespace
roleRef:
  kind: Role
  name: role-containergroup-service-account
  apiGroup: rbac.authorization.k8s.io
```

2. Apply the configuration from `containergroup-sa.yml`:

```
oc apply -f containergroup-sa.yml
```

3. Get the secret name associated with the service account:

```
export SA_SECRET=$(oc get sa containergroup-service-account -o json | jq
'.secrets[0].name' | tr -d '"')
```

4. Get the token from the secret:

```
oc get secret $(echo ${SA_SECRET}) -o json | jq '.data.token' | xargs | base64
--decode > containergroup-sa.token
```

5. Get the CA cert:

```
oc get secret $SA_SECRET -o json | jq '.data["ca.crt"]' | xargs | base64 --
decode > containergroup-ca.crt
```

6. Use the contents of `containergroup-sa.token` and `containergroup-ca.crt` to provide the information for the OpenShift or Kubernetes API Bearer Token required for the container group.

OpenStack credential type

Select this credential type to enable synchronization of cloud inventory with OpenStack.

Enter the following information for OpenStack credentials:

- **Username**: The username to use to connect to OpenStack.
- **Password (API Key)**: The password or API key to use to connect to OpenStack.
- **Host (Authentication URL)**: The host to be used for authentication.
- **Project (Tenant Name)**: The Tenant name or Tenant ID used for OpenStack. This value is usually the same as the username.
- Optional: **Project (Domain Name)**: Give the project name associated with your domain.
- Optional: **Domain Name**: Give the FQDN to be used to connect to OpenStack.
- Optional: **Region Name**: Give the region name. For some cloud providers, such as OVH, the region must be specified.

If you are interested in using OpenStack Cloud Credentials, see Associate cloud credentials with a job template, which includes a sample playbook.

Red Hat Ansible Automation Platform credential type

Select this credential to access another automation controller instance.

IMPORTANT:
In Ansible Automation Platform 2.7, the Ansible Automation Platform credential type must use the platform gateway URL. Direct component URLs are no longer supported.

Ansible Automation Platform credentials require the following inputs:

- **Red Hat Ansible Automation Platform**: The base URL or IP address of the other instance to connect to.
- **Username**: The username to use to connect to it.
- **Password**: The password to use to connect to it.
- **Oauth Token**: If username and password are not used, provide an OAuth token to use to authenticate.

The `env` injectors for Ansible Automation Platform are as follows:

```
ManagedCredentialType(

      namespace='controller',

  ....
  ....
  ....


  injectors={

          'env': {
              'TOWER_HOST': '{{host}}',
              'TOWER_USERNAME': '{{username}}',
              'TOWER_PASSWORD': '{{password}}',

              'TOWER_VERIFY_SSL': '{{verify_ssl}}',
              'TOWER_OAUTH_TOKEN': '{{oauth_token}}',

              'CONTROLLER_HOST': '{{host}}',

              'CONTROLLER_USERNAME': '{{username}}',
              'CONTROLLER_PASSWORD': '{{password}}',

              'CONTROLLER_VERIFY_SSL': '{{verify_ssl}}',

              'CONTROLLER_OAUTH_TOKEN': '{{oauth_token}}',

          }
```

Access automation controller credentials in an Ansible Playbook

You can get the host, username, and password parameters from a job runtime environment:

```
vars:
  controller:
    host: '{{ lookup("env", "CONTROLLER_HOST") }}'
    username: '{{ lookup("env", "CONTROLLER_USERNAME") }}'
    password: '{{ lookup("env", "CONTROLLER_PASSWORD") }}'
```

Red Hat Satellite 6 credential type

Select this credential type to enable synchronization of cloud inventory with Red Hat Satellite 6.

Automation controller writes a Satellite configuration file based on fields prompted in the user interface. The absolute path to the file is set in the following environment variable:

`FOREMAN_INI_PATH`

Satellite credentials have the following required inputs:

- **Satellite 6 URL**: The Satellite 6 URL or IP address to connect to.
- **Username**: The username to use to connect to Satellite 6.
- **Password**: The password to use to connect to Satellite 6.

Red Hat Virtualization credential type

Select this credential to enable automation controller to access Ansible's `oVirt4.py` dynamic inventory plugin, which is managed by *Red Hat Virtualization*.

Automation controller uses the following environment variables for Red Hat Virtualization credentials. These are fields in the user interface:

```
OVIRT_URL
OVIRT_USERNAME
OVIRT_PASSWORD
```

Provide the following information for Red Hat Virtualization credentials:

- **Host (Authentication URL)**: The host URL or IP address to connect to. To sync with the inventory, the credential URL needs to include the `ovirt-engine/api` path.
- **Username**: The username to use to connect to oVirt4. This must include the domain profile to succeed, for example `username@ovirt.host.com`.

- **Password**: The password to use to connect to it.
- Optional: **CA File**: Provide an absolute path to the oVirt certificate file (it might end in `.pem`, `.cer` and `.crt` extensions, but preferably `.pem` for consistency)

Source Control credential type

Source Control credentials are used with projects to clone and update local source code repositories from a remote revision control system such as Git or Subversion.

Source Control credentials require the following inputs:

- **Username**: The username to use in conjunction with the source control system.
- **Password**: The password to use in conjunction with the source control system.
- **SCM Private Key**: Copy or drag-and-drop the actual SSH Private Key to be used to authenticate the user to the source control system through SSH.
- **Private Key Passphrase**: If the SSH Private Key used is protected by a passphrase, you can configure a Key Passphrase for the private key.

NOTE:
You cannot configure Source Control credentials as **Prompt on launch**.

If you are using a GitHub account for a Source Control credential and you have *Two Factor Authentication* (2FA) enabled on your account, you must use your Personal Access Token in the password field rather than your account password.

Terraform: Backend configuration

Terraform is a HashiCorp tool used to automate various infrastructure tasks. Select this credential type to enable synchronization with the Terraform inventory source.

The Terraform credential requires the **Backend configuration** attribute which must contain the data from a Terraform backend block. Saving it stores the file path to the backend configuration in an environment variable `TF_BACKEND_CONFIG_FILE` that is made available to any job with the credential attached.

You can paste, drag a file, browse to upload a file, or click the 🔑 icon to populate the field from an external Secret Management System.

Terraform backend configuration requires the following inputs:

- **Name**

- Credential type: Select **Terraform backend configuration**.
- Optional: **Organization**
- Optional: **Description**
- **Backend configuration**: Drag a file here or browse to upload.
  Example configuration for an S3 backend:

```
bucket = "my-terraform-state-bucket"
key = "path/to/terraform-state-file"
region = "us-east-1"
access_key = "my-aws-access-key"
secret_key = "my-aws-secret-access-key"
```

- Optional: **Google Cloud Platform account credentials**

Terraform: HCP Terraform credential type

When using the `hashicorp.terraform` collection, you can use the **HCP Terraform** credential type for HCP Terraform or Terraform Enterprise (TFE/C).

Using this credential type enables users to store TFE/C tokens and host names, and associate them with job or workflow templates. It eliminates the need to set tokens in playbooks or tasks.

Related information
Create a credential for hashicorp.terraform

Thycotic DevOps Secrets Vault credential type

You can use Thycotic DevOps Secrets Vault as a credential type within automation controller.

This is part of the secret management capability.

For more information, see Thycotic DevOps Secrets Vault.

Thycotic secret server credential type

This is considered part of the secret management capability.

For more information, see Thycotic Secret Server.

Ansible Vault credential type

You can use the Ansible Vault credential type to store sensitive data, such as passwords or keys, in encrypted files.

Select this credential type to enable synchronization of inventory with Ansible Vault.

Vault credentials require the **Vault Password** and an optional **Vault Identifier** if applying multi-Vault credentialing.

You can configure automation controller to ask the user for the password at launch time by selecting **Prompt on launch**.

When you select **Prompt on launch**, a dialog opens when the job is launched, prompting the user to enter the password.

WARNING:
Credentials that are used in scheduled jobs must not be configured as **Prompt on launch**.

VMware vCenter credential type

Select this credential type to enable synchronization of inventory with VMware vCenter.

Automation controller uses the following environment variables for VMware vCenter credentials:

```
VMWARE_HOST
VMWARE_USER
VMWARE_PASSWORD
VMWARE_VALIDATE_CERTS
```

These are fields prompted in the user interface.

VMware credentials require the following inputs:

- **vCenter Host**: The vCenter hostname or IP address to connect to.
- **Username**: The username to use to connect to vCenter.
- **Password**: The password to use to connect to vCenter.

NOTE:
If the VMware guest tools are not running on the instance, VMware inventory synchronization does not return an IP address for that instance.

Access VMware vCenter credentials in an Ansible Playbook

You can get VMware vCenter credential parameters from a job runtime environment:

```
vars:
  vmware:
    host: '{{ lookup("env", "VMWARE_HOST") }}'
    username: '{{ lookup("env", "VMWARE_USER") }}'
    password: '{{ lookup("env", "VMWARE_PASSWORD") }}'
```

Use automation controller credentials in a playbook

You can access automation controller credentials in an Ansible Playbook by using environment variables. You can get credential parameters from a job runtime environment:

NOTE:
In Ansible Automation Platform 2.7, the `CONTROLLER_HOST` environment variable contains the platform gateway URL, not the direct automation controller URL. This ensures all API access goes through platform gateway.

The following playbook is an example of how to use automation controller credentials in your playbook.

```
- hosts: all


  vars:
```

```
    machine:
      username: '{{ ansible_user }}'
      password: '{{ ansible_password }}'
    controller:
      host: '{{ lookup("env", "CONTROLLER_HOST") }}'
      username: '{{ lookup("env", "CONTROLLER_USERNAME") }}'
      password: '{{ lookup("env", "CONTROLLER_PASSWORD") }}'
    network:
      username: '{{ lookup("env", "ANSIBLE_NET_USERNAME") }}'
      password: '{{ lookup("env", "ANSIBLE_NET_PASSWORD") }}'
    aws:
      access_key: '{{ lookup("env", "AWS_ACCESS_KEY_ID") }}'
      secret_key: '{{ lookup("env", "AWS_SECRET_ACCESS_KEY") }}'
      security_token: '{{ lookup("env", "AWS_SECURITY_TOKEN") }}'
    vmware:
      host: '{{ lookup("env", "VMWARE_HOST") }}'
      username: '{{ lookup("env", "VMWARE_USER") }}'
      password: '{{ lookup("env", "VMWARE_PASSWORD") }}'
    gce:
      email: '{{ lookup("env", "GCE_EMAIL") }}'
      project: '{{ lookup("env", "GCE_PROJECT") }}'
    azure:
      client_id: '{{ lookup("env", "AZURE_CLIENT_ID") }}'
      secret: '{{ lookup("env", "AZURE_SECRET") }}'
      tenant: '{{ lookup("env", "AZURE_TENANT") }}'
      subscription_id: '{{ lookup("env", "AZURE_SUBSCRIPTION_ID") }}'

  tasks:
    - debug:
        var: machine

    - debug:
        var: controller

    - debug:
        var: network
```

```
    - debug:
        var: aws

    - debug:
        var: vmware

    - debug:
        var: gce

    - shell: 'cat {{ gce.pem_file_path }}'
      delegate_to: localhost

    - debug:
        var: azure
```

Use 'delegate_to' and any lookup variable

```
- command: somecommand
  environment:
    USERNAME: '{{ lookup("env", "USERNAME") }}'
    PASSWORD: '{{ lookup("env", "PASSWORD") }}'
  delegate_to: somehost
```

Custom credential types

As a system administrator, you can define a custom credential type in a standard format by using a YAML or JSON-like definition. You can define a custom credential type that works in ways similar to existing credential types.

For example, a custom credential type can inject an API token for a third-party web service into an environment variable, for your playbook or custom inventory script to consume.

Custom credentials support the following ways of injecting their authentication information:

- Environment variables

- Ansible extra variables
- File-based templating, which means generating `.ini` or `.conf` files that contain credential values

You can attach one SSH and multiple cloud credentials to a job template. Each cloud credential must be of a different type. Only one of each type of credential is permitted. Vault credentials and machine credentials are separate entities.

NOTE:

- When creating a new credential type, you must avoid collisions in the `extra_vars`, `env`, and file namespaces.
- Environment variable or extra variable names must not start with `ANSIBLE_` because they are reserved.
- You must have System administrator (superuser) permissions to be able to create and edit a credential type (`CredentialType`) and to be able to view the `CredentialType.injection` field.

Content sourcing from collections

automation controller supports sourcing content for projects from Ansible collections defined in `requirements.yml` files. This is done by configuring Ansible Galaxy credentials at the organization level to define content sources for collection installations.

A "managed" credential type of `kind=galaxy` represents a content source for fetching collections defined in `requirements.yml` when project updates are run. Examples of content sources are galaxy.ansible.com, console.redhat.com, or on-premise automation hub. This new credential type represents a URL and (optional) authentication details necessary to construct the environment variables when a project update runs `ansible-galaxy collection install` as described in the Ansible documentation, Configuring the ansible-galaxy client. It has fields that map directly to the configuration options exposed to the Ansible Galaxy CLI, for example, per-server.

An endpoint in the API reflects an ordered list of these credentials at the Organization level:

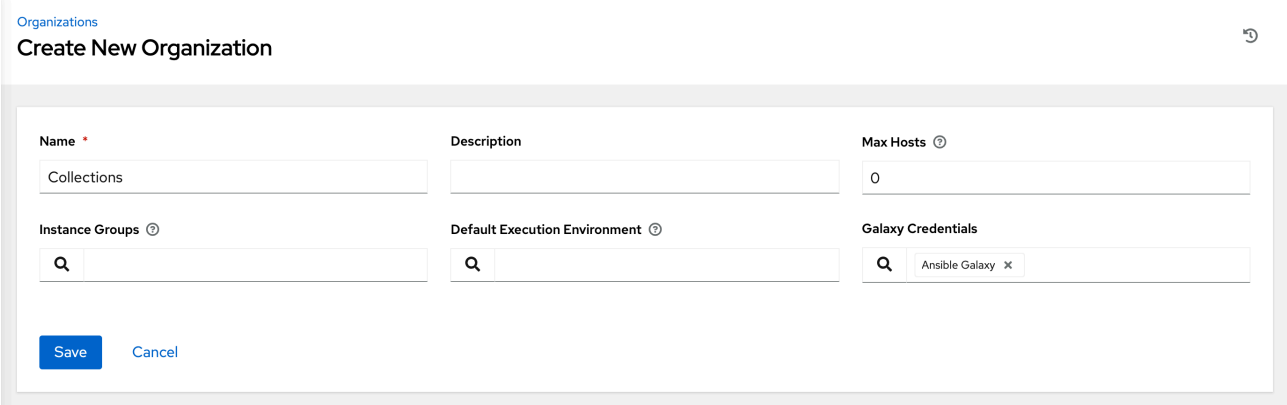
```
/api/v2/organizations/N/galaxy_credentials/
```

When installations of automation controller migrate existing Galaxy-oriented setting values, post-upgrade proper credentials are created and attached to every Organization. After upgrading to the latest version, every organization that existed before upgrade now has a list of one or more "Galaxy" credentials associated with it.

Additionally, post-upgrade, these settings are not visible (or editable) from the `/api/v2/settings/jobs/` endpoint.

Automation controller continues to fetch roles directly from public Galaxy even if `galaxy.ansible.com` is not the first credential in the list for the organization. The global Galaxy settings are no longer configured at the jobs level, but at the organization level in the user interface.

The organization's **Create organization** and **Edit organization** windows have an optional **Galaxy credentials** lookup field for credentials of `kind=galaxy`.



It is important to specify the order of these credentials as order sets precedence for the sync and lookup of the content. For more information, see Creating an organization.

For more information about how to set up a project by using collections, see Using Collections with automation hub.

Backwards-compatible API considerations

When using the automation controller API, there are some considerations to remember regarding backwards compatibility between version 1 (`api/v1/`) and version 2 (`api/v2/`).

Support for version 2 of the API (`api/v2/`) means a one-to-many relationship for job templates to credentials (including multicloud support).

You can filter credentials the v2 API:

```
curl "https://controller.example.org/api/v2/credentials/?
credential_type__namespace=aws"
```

In the V2 Credential Type model, the relationships are defined as follows:

Machine	SSH
Vault	Vault
Network	Sets environment variables, for example `ANSIBLE_NET_AUTHORIZE`

Machine	SSH
SCM	Source Control
Cloud	EC2, AWS
Cloud	Many others
Insights	Insights
Galaxy	galaxy.ansible.com, console.redhat.com
Galaxy	on-premise automation hub

Content verification

Automation controller uses GNU Privacy Guard (GPG) to verify content.

For more information, see The GNU Privacy Handbook.

Add new users and job templates to existing credentials

You can add new users and job templates to existing credentials in automation controller.

Procedure

1. From the navigation panel, select **Automation Execution > Infrastructure > Credentials**.
2. Select the credential that you want to assign to additional users.
3. Click the **User Access** tab. You can see users and teams associated with this credential and their roles. If no users exist, add them from the **Users** menu. For more information, see Users.
4. Click **Add roles**.
5. Select the user(s) that you want to give access to the credential and click **Next**.
6. From the **Select roles to apply** page, select the roles you want to add to the User.
7. Click **Next**.
8. Review your selections and click **Finish** to add the roles or click **Back** to make changes.

   The **Add roles** window displays stating whether the action was successful.

   If the action is not successful, a warning displays.
9. Click **Close**.

10. The **User Access** page displays the summary information.
11. Select the **Job templates** tab to select a job template to which you want to assign this credential.
12. Chose a job template or select **Create job template** from the **Create template** list to assign the credential to additional job templates.

    For more information about creating new job templates, see Job templates.

Getting started with credential types

Explore how automation controller organizes authentication details by using custom credential types. Reviewing existing definitions helps ensure you configure new credentials accurately for your automation tasks.

Procedure

1. From the navigation panel, select **Automation Execution > Infrastructure > Credentials**. If no custom credential types have been created, the **Credential Types** page prompts you to add one.

   If credential types have been created, this page displays a list of existing and available Credential Types.
2. Select the name of a credential or the Edit icon to view more information about a credential type, .
3. On the **Details** tab, each credential type displays its own unique configurations in the **Input Configuration** field and the **Injector Configuration** field, if applicable. Both YAML and JSON formats are supported in the configuration fields.

Configure credentials for Event-Driven Ansible

You can use credentials to store secrets that can be used for authentication purposes with resources, such as decision environments, rulebook activations and projects for Event-Driven Ansible controller, and projects for automation controller.

Credentials authenticate users when launching jobs against machines and importing project content from a version control system.

You can grant users and teams the ability to use these credentials without exposing the credential to the user. If a user moves to a different team or leaves the organization, you do not have to rekey all of your systems just because that credential was previously available.

NOTE:
In the context of automation controller and Event-Driven Ansible controller, you can use both `extra_vars` and credentials to store a variety of information. However, credentials are the preferred method of storing sensitive information such as passwords or API keys because they offer better security and centralized management, whereas `extra_vars` are more suitable for passing dynamic, non-sensitive data.

Set up credentials

Create a credential to securely store sensitive data (like tokens and passwords) required for rulebook activations to connect to source plugins or private registries.

Procedure

1. Log in to the Ansible Automation Platform Dashboard.
2. From the navigation panel, select **Automation Decisions › Infrastructure › Credentials**.
3. Click **Create credential**.
4. Insert the following:

   Name

   Insert the name.

   Description

   This field is optional.

   Organization

   Click the list to select an organization or select **Default**.

   Credential type

   Click the list to select your Credential type.

   NOTE:
   When you select the credential type, the **Type Details** section is displayed with fields that are applicable for the credential type you chose.

5. Complete the fields that are applicable to the credential type you selected.
6. Click **Create credential**.

Next steps

After saving the credential, the credentials details page is displayed. From there or the **Credentials** list view, you can edit or delete it.

Credentials list view

The Credentials list view provides a single pane to monitor all authentication assets, including their type, organization, and usage status across the platform.

From the menu bar, you can search for credentials in the **Name** search field.

You also have the following options in the menu bar:

- **Manage columns** - You can choose how fields are shown in the list view by clicking this option. You have four ways you can arrange your fields:
  - **Column** - Shows the column in the table.
  - **Description** - Shows the column when the item is expanded as a full width description.
  - **Expanded** - Shows the column when the item is expanded as a detail.
  - **Hidden** - Hides the column.
- **List view** or **Card view** - You can choose between these views by clicking the applicable icons.

Edit a credential

Edit credentials to update expired tokens, rotate secrets, or adjust permissions, ensuring secure and continuous access to external systems.

Procedure

1. Edit the credential by using one of these methods:
   - From the **Credentials** list view, click the **Edit credential** icon next to the desired credential.
   - From the **Credentials** list view, select the name of the credential, click **Edit credential**.
2. Edit the appropriate details and click **Save credential**.

Duplicate a credential

Use the **Duplicate credential** feature to rapidly generate a new credential based on an existing one, saving configuration time and reducing potential errors.

Procedure

1. On the Credentials list page, click the name of the credential that you want to duplicate. This takes you to the **Details** tab of the credential.
2. Click **Duplicate credential** in the top right of the Details tab.

> **NOTE:**
> You can also click the **Duplicate credential** icon next to the desired credential on the Credentials list page.

A message is displayed confirming that your selected credential has been duplicated: "<Name of credential> duplicated."

3. Click the **Back to credentials** tab to view the credential you just duplicated.

   The duplicated credential is displayed with the same name as the original credential followed by a time stamp in 24-hour format (for example, **<Name of credential> @ 17:26:30**).
4. Edit the details you prefer for your duplicated credential.
5. Click **Save credential**.

Delete a credential

You can permanently delete unused credentials. You must first ensure they are detached from any event stream before the deletion process can be completed.

Before you begin

1. Ensure that your credential is not currently linked to any rulebook activations.

Procedure

1. Delete the credential by using one of these methods:
   - From the **Credentials** list view, click the **More Actions** icon ⋮ next to the desired credential and click **Delete credential**.
   - From the **Credentials** list view, select the name of the credential, click the **More Actions** icon ⋮ next to **Edit credential**, and click **Delete credential**.
2. In the pop-up window, select **Yes, I confirm that I want to delete this credential**.

NOTE:
If your credential is still in use by other resources in your organization, a warning message is displayed letting you know that the credential cannot be deleted. Also, if your credential is being used in an event stream, you cannot delete it until the event stream is deleted or attached to a different credential. In general, avoid deleting a credential that is in use because it can lead to broken activations.

3. Click **Delete credential**.

Result

You can delete multiple credentials at a time by selecting the checkbox next to each credential, clicking the **More Actions** icon ⋮ in the menu bar, and then clicking **Delete selected credentials**.

Connect to external secret management systems with built-in credentials

Event-Driven Ansible controller includes built-in credentials to sync projects, run rulebooks, execute job templates, fetch container images, and process data through event streams.

These built-in credential types are not editable. So if you want credential types that support authentication with other systems, you can create your own credential types that can be used in your source plugins. Each credential type contains an input configuration and an injector configuration that can be passed to an Ansible rulebook to configure your sources. For more information, see *Create custom credentials for Event-Driven Ansible*.

If you will be executing job templates through automation controller, you can retrieve credential values from external secret management systems listed in *External secret management credential types*.

Related information
Create custom credentials for Event-Driven Ansible
External secret management credential types

External secret management credential types

In addition to the built-in credential types, Event-Driven Ansible supports a variety of external secret management credential types. These credential types allow rulebooks to securely retrieve

sensitive information (API keys, and similar) directly from your organization's centralized secret vault.

The following external credential types are available for use in Event-Driven Ansible controller:

- AWS Secrets Manager
- Azure Key Vault
- Centrify Vault Credential Provider
- CyberArk Central Credential Provider
- CyberArk Conjur Secrets Manager
- HashiCorp Vault Secret
- HashiCorp Vault Signed SSH
- Thycotic DevOps Secrets Vault
- Thycotic Secret Server
- GitHub App Installation Access Token

The process for using these credentials in a rulebook activation is consistent with how they are used in automation controller.

Related information
Configure automation controller
Configure an external secret management system for automation

Create custom credentials for Event-Driven Ansible

Create custom credential types (via JSON/YAML) to define unique security fields and logic, enabling support for proprietary event sources or specialized authentication.

Each credential type displays its own unique configurations in the **Input Configuration** and the **Injector Configuration** fields, if applicable. Both YAML and JSON formats are supported in the configuration fields.

Custom credentials support Ansible extra variables as a means of injecting their authentication information.

You can attach one or more cloud, vault, and Red Hat Ansible Automation Platform credential types to a rulebook activation.

NOTE:

- When creating a new credential type, you must avoid collisions in the `extra_vars`.
- Extra variable names must not start with **EDA_** because they are reserved.
- You must have System administrator (superuser) permissions to be able to create and edit a credential type and to be able to view the **Injector configuration** field.

When you customize your own credential types, they display on the Credential Types page along with a list of built-in credential types.

Input configuration

You can configure the input fields and define which parameters are required when a user creates a credential of this custom type.

The Input configuration has two attributes:

- fields - a collection of properties for a credential type.
- required - a list of required fields.

Fields can have multiple properties, depending on the credential type you select.

Input Configuration Field Properties

Fields	Description	Mandatory (Y/N)
id	Unique id of the field; must be a string type and stores the variable name	Yes
type	Can be string or boolean type	No, default is string
label	Used by the UI when rendering the UI element	Yes
secret	Will be encrypted	No, default false
multiline	If the field contains data from a file the multiline can be set to True	No, default false
help_text	The help text associated with this field	No

Injector configuration

You can use Injector configuration to safely transform and map credential data from input fields so that it can be correctly exposed and consumed by `ansible-rulebook` at runtime.

Event-Driven Ansible supports the following types of injectors:

- Environment variables (`env`) - Used in source plugins for the underlying package or shared library.
- Ansible extra variables (`extra_vars`) - Used for substitution in the rulebook conditions, actions or source plugin parameters.
- File-based templating (`file`) - Used to create file contents from the credential inputs such as certificates and keys, which might be required by source plugins. File injectors provide a way to deliver these certificates and keys to ansible-rulebook at runtime without having to store them in decision environments. As a result, ansible-rulebook creates temporary files and the file names can be accessed using `eda.filename` variables, which are automatically created for you after the files have been created (for instance, "{{eda.filename.my_cert}}").

IMPORTANT:
When creating `extra_vars` in rulebook activations and credential type injectors, avoid using `eda` or `ansible` as key names since that conflicts with internal usage and might cause failure in both rulebook activations and credential type creation.

Injectors enable you to adjust the fields so that they can be injected into a rulebook as one of the above-mentioned injector types, which cannot have duplicate keys at the top level. If you have two sources in a rulebook that both require parameters such as username and password, the injectors, along with the rulebook, help you adapt the arguments for each source.

Related information
Credential injectors
gssapi input credential type

Creating a new credential type

Define a custom credential type by using a YAML or JSON schema. Defining these custom types helps ensure that authentication information is securely injected into automation workflows.

Procedure

1. From the navigation panel, select **Automation Execution > Infrastructure > Credentials**.
2. In the **Credential Types** view, click **Create credential type**.
3. Enter the appropriate details in the **Name** and **Description** field.

NOTE:
When creating a new credential type, do not use reserved variable names that start with `ANSIBLE_` for the **INPUT** and **INJECTOR** names and IDs, as they are invalid for custom credential types.

4. In the **Input configuration** field, specify an input schema that defines a set of ordered fields for that type. The format can be in YAML or JSON:

YAML

```
fields:
  - type: string
    id: username
    label: Username
  - type: string
    id: password
    label: Password
    secret: true
required:
  - username
  - password
```

View more YAML examples at the YAML page.

JSON

```
{
"fields": [
  {
  "type": "string",
  "id": "username",
  "label": "Username"
  },
  {
  "secret": true,
  "type": "string",
  "id": "password",
  "label": "Password"
   }
  ],
 "required": ["username", "password"]
}
```

View more JSON examples at The JSON website.

The following configuration in JSON format shows each field and how they are used:

```
{
  "fields": [{
    "id": "api_token",    # required - a unique name used to reference the
field value

    "label": "API Token", # required - a unique label for the field

    "help_text": "User-facing short text describing the field.",

    "type": ("string" | "boolean")   # defaults to 'string'

    "choices": ["A", "B", "C"]   # (only applicable to `type=string`)

    "format": "ssh_private_key"  # optional, can be used to enforce data
format validity
                                 for SSH private key data (only applicable to
`type=string`)

    "secret": true,       # if true, the field value will be encrypted

    "multiline": false    # if true, the field should be rendered as multi-
line for input entry
                          # (only applicable to `type=string`)
},{
    # field 2...
},{
    # field 3...
}],

"required": ["api_token"]   # optional; one or more fields can be marked as
required
},
```

When `type=string`, fields can optionally specify multiple choice options:

```
{
  "fields": [{
      "id": "api_token",    # required - a unique name used to reference the
field value
      "label": "API Token", # required - a unique label for the field
      "type": "string",
      "choices": ["A", "B", "C"]
  }]
},
```

5. In the **Injector configuration** field, enter environment variables or extra variables that specify the values a credential type can inject. The format can be in YAML or JSON (see examples in the previous step).

   The following configuration in JSON format shows each field and how they are used:

```
{
  "file": {
      "template": "[mycloud]\ntoken={{ api_token }}"
  },
  "env": {
      "THIRD_PARTY_CLOUD_API_TOKEN": "{{ api_token }}"
  },
  "extra_vars": {
      "some_extra_var": "{{ username }}:{{ password }}"
  }
}
```

   Credential Types can also generate temporary files to support `.ini` files or certificate or key data:

```
{
  "file": {
      "template": "[mycloud]\ntoken={{ api_token }}"
  },
  "env": {
      "MY_CLOUD_INI_FILE": "{{ tower.filename }}"
  }
}
```

In this example, automation controller writes a temporary file that has:

```
[mycloud]\ntoken=SOME_TOKEN_VALUE
```

The absolute file path to the generated file is stored in an environment variable named `MY_CLOUD_INI_FILE`.

The following is an example of referencing many files in a custom credential template:

Inputs

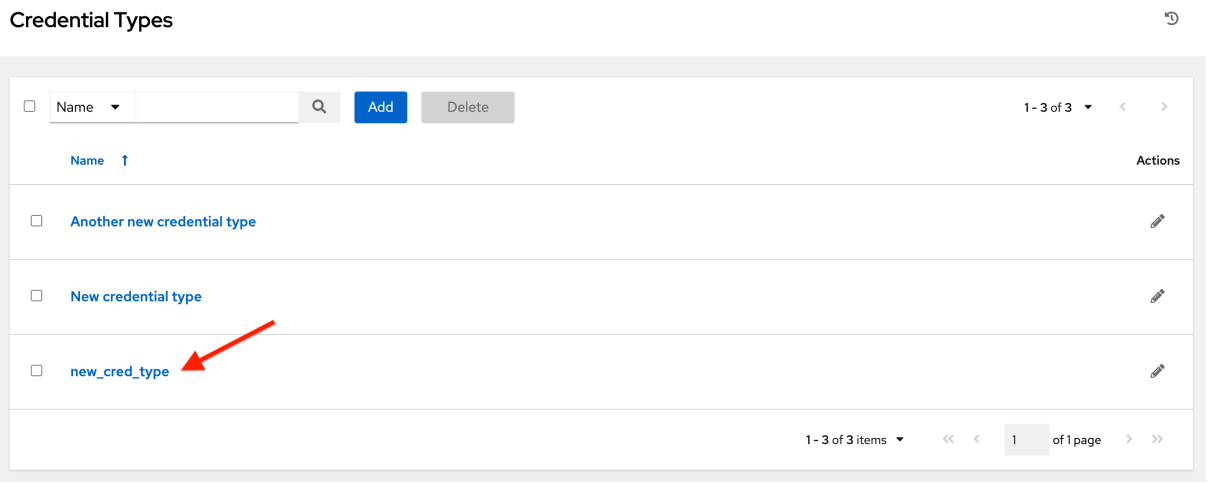
```
{
  "fields": [{
    "id": "cert",
    "label": "Certificate",
    "type": "string"
  },{
    "id": "key",
    "label": "Key",
    "type": "string"
  }]
}
```

Injectors

```
{
  "file": {
    "template.cert_file": "[mycert]\n{{ cert }}",
    "template.key_file": "[mykey]\n{{ key }}"
},
"env": {
    "MY_CERT_INI_FILE": "{{ tower.filename.cert_file }}",
    "MY_KEY_INI_FILE": "{{ tower.filename.key_file }}"
}
}
```

6. Click **Create credential type**.

   Your newly created credential type is displayed on the list of credential types:

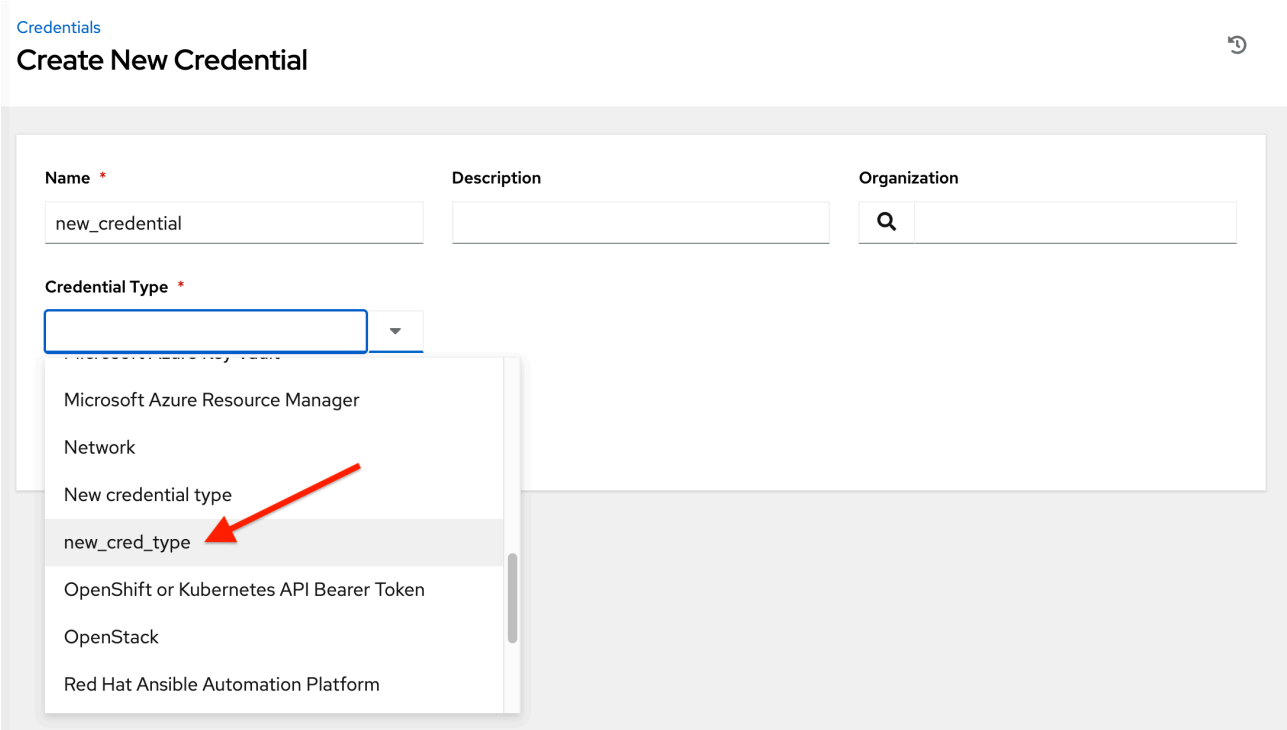


7. Click the Edit icon to modify the credential type options.

> **NOTE:**
> In the **Edit** screen, you can modify the details or delete the credential. If the **Delete** option is disabled, this means that the credential type is being used by a credential, and you must delete the credential type from all the credentials that use it before you can delete it.

Result

- Verify that the newly created credential type can be selected from the **Credential Type** selection window when creating a new credential:



Related information
Creating a credential

Authenticate through the API

Ansible Automation Platform provides a visual dashboard for out-of-the-box control while providing a REST API to integrate with your other tools on a deeper level.

The platform supports several authentication methods to integrate automation into existing tools and processes. This ensures that the right people can access platform resources.

You can use the following authentication methods in the API:

Related information
Session authentication
Basic authentication
OAuth 2 token authentication
Single sign-on authentication

Use session authentication

You can use session authentication when logging in to the platform gateway API or UI to manually create resources, such as inventories, projects, and job templates, and start jobs in the browser.

With this method, you can remain logged in for a prolonged period of time, not just for that HTTP request. For example, when browsing the API or UI in a browser such as Chrome or Mozilla Firefox.

When you log in, a session cookie is created. You can remain logged in when navigating to different pages across the platform.

The following image represents the communication that occurs between the client and server in a session:

Use the curl tool to see the activity that occurs when you log in through platform gateway.

Procedure

1. Use `GET` to go to the `/api/login/` endpoint to get the `csrftoken` cookie:

```
$ curl -k -c - https://<gateway server name>/api/gateway/v1/login/


$YOUR_AAP_URL FALSE / TRUE 1780539778 csrftoken
GODXonA5LyV1uAs8zvcD2k12DQJC74oB
```

2. Extract the `csrftoken` value from the cookie returned in the first step.
3. `POST` to the `/api/login/` endpoint with username, password, and `X-CSRFToken=<token-value>`:

```
curl -X POST -H 'Content-Type: application/x-www-form-urlencoded' \


--referer https://<gateway server name>/api/gateway/v1/login/ \


-H 'X-CSRFToken: <token-value>' \


--data 'username=admin&password=$YOUR_ADMIN_PASSWORD' \


--cookie 'csrftoken=GODXonA5LyV1uAs8zvcD2k12DQJC74oB' \


https://<gateway server name>/api/gateway/v1/login/ -k -D - -o /dev/null
```

4. Access and test the APIs that need authentication, for example `/api/controller/v2/settings/all/`:

NOTE:
latform gateway performs all of these steps when you olog into the UI or API in the browser. You must use this procedure only when authenticating in the browser. For programmatic integration with platform gateway, see OAuth2 token authentication.

```
$ curl -X GET -H 'Cookie: <cookieID>;' https://<gateway server name>/api/
controller/v2/settings/all/ -k
```

Result

The following shows a typical response:

```
Server: nginx

Date: <current date>

Content-Type: text/html; charset=utf-8

Content-Length: 0

Connection: keep-alive

Location: /accounts/profile/

X-API-Session-Cookie-Name: awx_sessionid

Expires: <date>

Cache-Control: max-age=0, no-cache, no-store, must-revalidate, private

Vary: Cookie, Accept-Language, Origin

Session-Timeout: 1800

Content-Language: en

X-API-Total-Time: 0.377s

X-API-Request-Id: 700826696425433fb0c8807cd40c00a0

Access-Control-Expose-Headers: X-API-Request-Id

Set-Cookie: userLoggedIn=true; Path=/

Set-Cookie: current_user=<user cookie data>; Path=/

Set-Cookie: csrftoken=<csrftoken>; Path=/; SameSite=Lax

Set-Cookie: awx_sessionid=<your session id>; expires=<date>; HttpOnly; Max-Age=1800;
Path=/; SameSite=Lax

Strict-Transport-Security: max-age=15768000
```

When a user is successfully authenticated with this method, the server responds with a header called `X-API-Session-Cookie-Name`, indicating the configured name of the session cookie. The default value is `awx_session_id` which you can see later in the `Set-Cookie` headers.

NOTE:
You can change the session expiration time by specifying it in the `SESSION_COOKIE_AGE` parameter.

Basic authentication

Basic authentication is stateless. You must send the base64-encoded username and password along with each request through the Authorization header. You can use this method for API calls from curl requests, Python scripts, or individual requests to the API.

OAuth 2 token authentication through platform gateway is the recommended method for accessing the API.

The following is an example of basic authentication with curl:

```
# the --user flag adds this Authorization header for us

curl -X GET --user 'user:password' https://<gateway server name>/api/gateway/v1/
tokens/ -k -L
```

Related information
The 'Basic' HTTP Authentication Scheme

Disable basic authentication

You can disable basic authentication for security purposes.

Procedure

1. From the navigation panel, select **Settings > Platform gateway**.
2. Click **Edit platform gateway settings**.
3. Disable the option **Gateway basic auth enabled**.
4. Click **Save platform gateway settings**.

OAuth 2 token authentication

OAuth (Open Authorization) is an open standard for token-based authentication and authorization. OAuth 2 authentication is commonly used when interacting with the platform gateway API programmatically.

You provide an OAuth 2 token with each API request through the Authorization header. Unlike Basic authentication, OAuth 2 tokens have a configurable timeout and are scopable. Tokens have a configurable expiration time and can be revoked for one user or for the entire platform gateway system by an administrator if needed. You can do this with the *revoke_oauth2_tokens* management command, or by using the API as explained in Revoke an access token.

The different methods for obtaining OAuth 2 access tokens in automation controller include the following:

- Personal access tokens (PAT)
- Application token: Password grant type
- Application token: Implicit grant type
- Application token: Authorization Code grant type

You can create an OAuth 2 token in the API or in the **Access Management > OAuth Applications** tab of the platform gateway UI.

For the purpose of this example, use the PAT method for creating a token in the API. After you create it, you can set the scope.

NOTE:
You can configure the expiration time of the token system-wide..

Token authentication is the recommended method for any programmatic use of the platform gateway API, such as Python scripts or tools such as curl.

Curl example

Create a token through the platform gateway tokens endpoint:

```
curl -u user:password -k -X POST https://<gateway server name>/api/gateway/v1/
tokens/
```

This call returns JSON data with the following:

You can use the value of the `token` property to perform a `GET` request for a resource, such as Hosts:

```
curl -k -X GET \
  -H “Content-Type: application/json”
  -H “Authorization: Bearer <oauth2-token-value>” \
  https://<platform-host>/api/controller/v2/hosts/
```

You can also run a job by making a `POST` to the job template that you want to start:

```
curl -k -X POST \
  -H "Authorization: Bearer <oauth2-token-value>" \
  -H "Content-Type: application/json" \
  --data '{"limit" : "ansible"}' \
  https://<platform-host>/api/controller/v2/job_templates/14/launch/
```

Related information
Configuring access to external applications with token-based authentication
revoke_oauth2_tokens
Revoke an access token
Associate tokens with applications
Configure access to external applications with tokens

Enable external users to create OAuth 2 tokens

By default, external users such as those created by single sign-on are not able to generate OAuth tokens for security purposes.

Procedure

1. From the navigation panel, select **Settings > Platform gateway**.
2. Select **Edit platform gateway settings**.
3. Enable the option to **Allow external users to create OAuth2 tokens**.

Single sign-on authentication

Single sign-on (SSO) authentication methods are different from other methods because the authentication of the user happens external to platform gateway, such as Google SSO, Microsoft Azure SSO, SAML, or GitHub.

You can configure SSO authentication through platform gateway to integrate with a central identity provider in your organization. Once you have configured an SSO method, an option for that SSO is available on the login screen. If you select that option, the platform redirects you to the identity provider, for example GitHub, where you present your credentials. If the identity provider verifies you successfully, platform gateway creates a user linked to your GitHub user (if this is your first time logging in with this SSO method) and logs you in.

Related information

Configuring an authentication type

Renew and change SSL/TLS certificates

If your current SSL/TLS certificates have expired or will expire soon, you can either renew or replace the SSL/TLS certificates used by Ansible Automation Platform.

You must renew the SSL/TLS certificates if you need to regenerate them with new information such as new hosts.

You must replace the SSL/TLS certificates if you want to use certificates signed by an internal certificate authority.

Container-based installations

You can change the TLS certificates and keys for your container-based Ansible Automation Platform installation. This process involves a preparation step, either providing new custom certificates or deleting or moving the old certificates, followed by running the installation program.

Change TLS certificates and keys using the installation program

The following procedure describes how to update the TLS certificates and keys by using the installation program.

Procedure

1. To prepare the certificates and keys, choose one of the following methods:
    - To provide custom certificates - For each service that requires updated TLS certificates, copy the new certificates and keys to a path relative to the Ansible Automation Platform installer. Then update the inventory file variables with the absolute paths to the new files.

```
 # Platform gateway
gateway_tls_cert=<path_to_tls_certificate>
gateway_tls_key=<path_to_tls_key>
gateway_pg_tls_cert=<path_to_tls_certificate>
gateway_pg_tls_key=<path_to_tls_key>
gateway_redis_tls_cert=<path_to_tls_certificate>
gateway_redis_tls_key=<path_to_tls_key>

# Automation controller
controller_tls_cert=<path_to_tls_certificate>
controller_tls_key=<path_to_tls_key>
controller_pg_tls_cert=<path_to_tls_certificate>
controller_pg_tls_key=<path_to_tls_key>

# Automation hub
hub_tls_cert=<path_to_tls_certificate>
hub_tls_key=<path_to_tls_key>
hub_pg_tls_cert=<path_to_tls_certificate>
hub_pg_tls_key=<path_to_tls_key>

# Event-Driven Ansible
eda_tls_cert=<path_to_tls_certificate>
eda_tls_key=<path_to_tls_key>
eda_pg_tls_cert=<path_to_tls_certificate>
eda_pg_tls_key=<path_to_tls_key>
eda_redis_tls_cert=<path_to_tls_certificate>
eda_redis_tls_key=<path_to_tls_key>

# PostgreSQL
postgresql_tls_cert=<path_to_tls_certificate>
postgresql_tls_key=<path_to_tls_key>

# Receptor
receptor_tls_cert=<path_to_tls_certificate>
receptor_tls_key=<path_to_tls_key>
```

- To generate new certificates - If you want the installation program to generate a new certificate for a service, delete or move the existing certificates and keys.

Certificate and key file paths per service

Service	Certificate file path	Key file path
Automation controller	`~/aap/controller/etc/tower.cert`	`~/aap/controller/etc/tower.key`
Event-Driven Ansible	`~/aap/eda/etc/eda.cert`	`~/aap/eda/etc/eda.key`
Platform gateway	`~/aap/gateway/etc/gateway.cert`	`~/aap/gateway/etc/gateway.key`
Automation hub	`~/aap/hub/etc/pulp.cert`	`~/aap/hub/etc/pulp.key`
PostgreSQL	`~/aap/postgresql/server.crt`	`~/aap/postgresql/server.key`
Receptor	`~/aap/receptor/etc/receptor.crt`	`~/aap/receptor/etc/receptor.key`
Redis	`~/aap/redis/server.crt`	`~/aap/redis/server.key`

2. After preparing your certificates, run the `install` playbook from your installation directory:

```
 ansible-playbook -i <inventory_file_name>
ansible.containerized_installer.install
```

Result

Verify that the new TLS certificates are in use by checking that the services are running and accessible. To do this, check a specific endpoint by using `curl`:

```
$ curl -vk https://<hostname_or_ip>:<port_number>/api/v2/
```

The output of this command gives details about the TLS handshake. Look for the following output to confirm the correct certificate is being used:

```
*  SSL certificate verify OK
```

Operator-based installations

You can change the TLS certificates and keys for your operator-based Ansible Automation Platform installation.

Change the SSL/TLS certificate and key on automation controller on OpenShift Container Platform

The following procedure describes how to change the SSL/TLS certificate and key for automation controller running on OpenShift Container Platform.

Procedure

1. Copy the signed SSL/TLS certificate and key to a secure location.
2. Create a TLS secret within OpenShift:

```
oc create secret tls controller-certs-$(date +%F) --cert=/path/to/ssl.crt --
key=/path/to/ssl.key -n <namespace>
```

3. Edit the Ansible Automation Platform custom resource to add `route_tls_secret` under the `controller` section:

```
oc edit ansibleautomationplatform <aap-instance-name> -n <namespace>
```

4. Add or update the `route_tls_secret` parameter under `spec.controller`:

```
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
  namespace: ansible-automation-platform
spec:
  controller:
    route_tls_secret: controller-certs-2024-03-24
```

NOTE:
The name of the TLS secret is arbitrary. In this example, it is timestamped with the date that the secret is created, to differentiate it from other TLS secrets applied to the automation controller instance.

The operator automatically applies this configuration to the automation controllerinstance managed by this Ansible Automation Platform deployment.

5. Wait a few minutes for the changes to be applied.
6. Verify that new SSL/TLS certificate and key have been installed:

```
true | openssl s_client -showcerts -connect ${CONTROLLER_FQDN}:443
```

Change the SSL/TLS certificate and key for automation hub on OpenShift Container Platform

The following procedure describes how to change the SSL/TLS certificate and key for your operator-based Ansible Automation Platform installation of automation hub running on OpenShift Container Platform.

Procedure

1. Copy the signed SSL/TLS certificate and key to a secure location.
2. Create a TLS secret within OpenShift:

```
oc create secret tls hub-certs-$(date +%F) --cert=/path/to/ssl.crt --key=/
path/to/ssl.key -n <namespace>
```

3. Edit the Ansible Automation Platform custom resource to add `route_tls_secret` under the `hub` section:

```
oc edit ansibleautomationplatform <aap-instance-name> -n <namespace>
```

4. Add or update the `route_tls_secret` parameter under `spec.hub`:

```
apiVersion: aap.ansible.com/v1alpha1
kind: AnsibleAutomationPlatform
metadata:
  name: myaap
  namespace: ansible-automation-platform
spec:
  hub:
    route_tls_secret: hub-certs-2024-03-24
```

NOTE:
The name of the TLS secret is arbitrary. In this example, it is timestamped with the date that the secret is created, to differentiate it from other TLS secrets applied to the automation hub instance.

The operator automatically applies this configuration to the automation hub instance managed by this Ansible Automation Platform deployment.

5. Wait a few minutes for the changes to be applied.
6. Verify that new SSL/TLS certificate and key have been installed:

```
true | openssl s_client -showcerts -connect ${HUB_FQDN}:443
```

RPM-based installations

To renew or change SSL/TLS certificates for RPM-based installations, you can edit the inventory file and run the installation program. The installation program verifies that all Ansible Automation Platform components are working.

Alternatively, you can change the SSL/TLS certificates manually. This is quicker, but there is no automatic verification.

Red Hat recommends that you use the installation program to make changes to your Ansible Automation Platform deployment.

Renewing the self-signed SSL/TLS certificates

The following steps regenerate new SSL/TLS certificates for all Ansible Automation Platform components.

Procedure

1. Add `aap_service_regen_cert=true` to the inventory file in the `[all:vars]` section:

```
[all:vars]

aap_service_regen_cert=true
```

2. Run the installation program.

Result

- Validate the CA file and certificate file on Event-Driven Ansible controller:

```
openssl verify -CAfile ansible-automation-platform-managed-ca-cert.crt /etc/
ansible-automation-platform/eda/server.cert

openssl s_client -connect <EDA_FQDN>:443
```

- Validate the CA file and certificate file on platform gateway:

```
openssl verify -CAfile ansible-automation-platform-managed-ca-cert.crt /etc/
ansible-automation-platform/gateway/gateway.cert

openssl s_client -connect <GATEWAY_FQDN>:443
```

- Validate the CA file and certificate file on automation hub:

```
openssl verify -CAfile ansible-automation-platform-managed-ca-cert.crt /etc/
pulp/certs/pulp_webserver.crt

openssl s_client -connect <HUB_FQDN>:443
```

- Validate the CA file and certificate file on automation controller:

```
openssl verify -CAfile ansible-automation-platform-managed-ca-cert.crt /etc/
tower/tower.cert

openssl s_client -connect <CONTROLLER_FQDN>:443
```

Change SSL/TLS certificates and keys using the installation program

The following procedure describes how to change the SSL/TLS certificate and key in the inventory file.

Before you begin

- The certificates must be in PEM format.
- If there is an intermediate certificate authority, you must append it to the server certificate.
- Use the correct order for the certificates: The server certificate comes first, followed by the intermediate certificate authority.

For further information, see the ssl certificate section of the NGINX documentation.

Procedure

1. Copy the new SSL/TLS certificates and keys to a path relative to the Ansible Automation Platform installer.
2. Add the absolute paths of the SSL/TLS certificates and keys to the inventory file. Refer to Inventory file variables for guidance on setting these variables.
    - Event-Driven Ansible controller: `automationedacontroller_ssl_cert`, `automationedacontroller_ssl_key`, `custom_ca_cert`
    - Platform gateway: `automationgateway_ssl_cert`, `automationgateway_ssl_key`, `custom_ca_cert`
    - Automation hub: `automationhub_ssl_cert`, `automationhub_ssl_key`, `custom_ca_cert`

- Automation controller: `web_server_ssl_cert`, `web_server_ssl_key`, `custom_ca_cert`

> **NOTE:**
> The `custom_ca_cert` must be the root certificate authority that signed the intermediate certificate authority. This file is installed in `/etc/pki/ca-trust/source/anchors`.

3. Run the installation program.

Change SSL/TLS certificates and keys manually

The following procedure describes how to change SSL/TLS certificates and keys manually for all Ansible Automation Platform components.

Procedure

1. Backup the current SSL/TLS certificate:

```
cp <CERT_PATH> <CERT_PATH>-$(date +%F)
```

2. Backup the current key files:

```
cp <KEY_PATH> <KEY_PATH>-$(date +%F)
```

3. Copy the new SSL/TLS certificate to the certificate path.
4. Copy the new key to the key path.
5. Restore the SELinux context:

```
restorecon -v <CERT_PATH> <KEY_PATH>
```

6. Set appropriate permissions for the certificate and key files:

```
chown <OWNER>:<GROUP> <CERT_PATH> <KEY_PATH>
chmod 0600 <CERT_PATH> <KEY_PATH>
```

7. Test the NGINX configuration:

```
nginx -t
```

8. Reload NGINX:

```
systemctl reload nginx.service
```

9. Verify that new SSL/TLS certificate and key have been installed:

```
true | openssl s_client -showcerts -connect <COMPONENT_FQDN>:443
```

SSL/TLS certificate and key file paths per service

Service	Certificate file path	Key file path	Owner:Group
Automation controller	`/etc/tower/tower.cert`	`/etc/tower/tower.key`	`root:awx`
Automation hub	`/etc/pulp/certs/pulp_webserver.crt`	`/etc/pulp/certs/pulp_webserver.key`	`root:pulp`
Event-Driven Ansible controller	`/etc/ansible-automation-platform/eda/server.cert`	`/etc/ansible-automation-platform/eda/server.key`	`root:eda`
Platform gateway	`/etc/ansible-automation-platform/gateway/gateway.cert`	`/etc/ansible-automation-platform/gateway/gateway.key`	`root:gateway`

Configure a CA file

Use this example to customize the default definition file to include a CA certificate to the `additional-build-files` section, move the file to the appropriate directory and, run the

command to update the dynamic configuration of CA certificates to allow the system to trust this certificate.

Prerequisites

- A custom CA certificate, for example `rootCA.crt`.

NOTE:
Customizing the CA certificate using `prepend_base` means that the resulting CA configuration is displayed in all other build stages and the final image, because all other build stages inherit from the base image.

```
additional_build_files:

  # copy the CA public key into the build context, we will copy and use it in the
base image later

  - src: files/rootCA.crt

    dest: configs


additional_build_steps:

  prepend_base:

    # copy a custom CA cert into the base image and recompute the trust database

    # because this is in "base", all stages will inherit (including the final EE)

    - COPY _build/configs/rootCA.crt /usr/share/pki/ca-trust-source/anchors

    - RUN update-ca-trust


options:

  package_manager_path: /usr/bin/microdnf  # downstream images use non-standard
package manager


[galaxy]

server_list = automation_hub
```

Harden the platform security posture

Harden the security posture of your Ansible Automation Platform deployment on Red Hat Enterprise Linux. Applying these guidelines during the planning, architecture, and installation phases helps ensure your automation components remain protected.

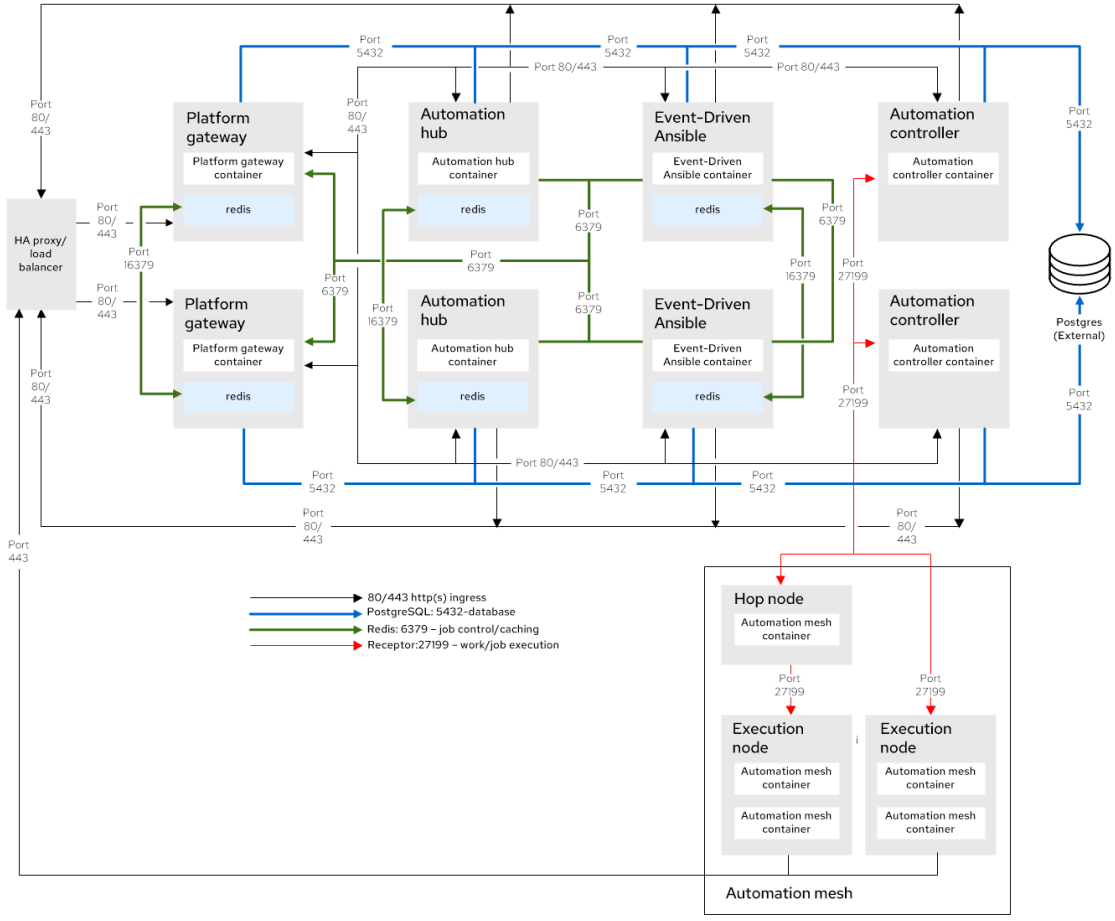
Plan your topology and networking configuration

Install Ansible Automation Platform using a tested deployment model. Choose between an enterprise reference architecture for high performance and scalability or a growth architecture for resource-constrained environments.

It is possible to install Ansible Automation Platform on different infrastructure reference architectures and with different environment configurations. Red Hat does not fully test architectures outside of published reference architectures. Red Hat recommends using a tested reference architecture for all new deployments and provides commercially reasonable support for deployments that meet minimum requirements.

The following diagram is a tested container enterprise architecture:

Figure: Reference architecture overview



DNS, NTP, and service planning

When installing Ansible Automation Platform, DNS and NTP configurations are crucial for a successful deployment and proper operation.

DNS

Define a valid Fully Qualified Domain Name (FQDN) for all infrastructure nodes in your installation inventory file. Resolving these to a routable IP address in DNS passes installer checks and helps ensure a successful setup.

DNS and load balancing

When using a load balancer with Ansible Automation Platform, an additional FQDN is required. For example, `aap.example.com` could be used for the load balancer to direct traffic to each platform gateway component defined in the installation inventory.

Manage platform credentials

Red Hat Ansible Automation Platform uses credentials to authenticate requests to jobs against machines, synchronize with inventory sources, and import project content from a version control system.

Ansible Automation Platform manages three sets of secrets:

- User passwords for **local Ansible Automation Platform users**.
- Secrets for Ansible Automation Platform **operational use** (database password, message bus password, and so on).
- Secrets for **automation use** (SSH keys, cloud credentials, external password vault credentials, and so on).

Implementing a privileged access or credential management solution to protect credentials from compromise is a highly recommended practice. Organizations should audit the use of, and provide additional programmatic control over, access and privilege escalation.

You can further secure automation credentials by ensuring they are unique and stored only in Ansible Automation Platform or in a supported external secrets management system. Services such as OpenSSH can be configured to allow credentials on connections only from specific addresses. Use different credentials for automation from those used by system administrators to log in to a server. Although direct access should be limited where possible, it can be used for disaster recovery or other ad hoc management purposes, allowing for easier auditing.

Different automation jobs might need to access a system at different levels. For example, you can have low-level system automation that applies patches and performs security baseline checking, while a higher-level piece of automation deploys applications. By using different keys or credentials for each piece of automation, the effect of any one key vulnerability is minimized. This also allows for easy baseline auditing.

External credential vault considerations

Secrets management is an essential component of maintaining a secure automation platform. We recommend the following secrets management practice:

Use an external system to manage secrets. In cases where credentials need to be updated, an external system can retrieve updated credentials with less complexity than an internal system. External systems for managing secrets include CyberArk, HashiCorp Vault, {Azure} Key Management, and others.

Related information
Configure an external secret management system for automation

Understand how Ansible Automation Platform manages secrets

Ansible Automation Platform uses several secrets (passwords, keys, and so on) operationally. These secrets are stored unencrypted on the various Ansible Automation Platform servers, as each component service must read them at startup.

All files are protected by UNIX permissions, and restricted to the root user or the appropriate service account user. These files should be routinely monitored to ensure there has been no unauthorized access or modification.

RPM-based installation secrets

The following table provides the location of these secrets for RPM-based installs of Ansible Automation Platform.

Ansible Automation Platform operational secrets

Automation controller secrets	
File path	**Description**
`/etc/tower/SECRET_KEY`	A secret key used for encrypting automation secrets in the database. If the `SECRET_KEY` changes or is unknown, no encrypted fields in the database will be accessible.

`/etc/tower/tower.cert`
 	
`/etc/tower/tower.key`
	SSL/TLS certificate and key for the automation controller web service.
	
	A self-signed `cert/key` is installed by default; you can provide a locally appropriate certificate and key.
	
	For more information, see *Installing with user-provided PKI certificates* in See "Installation settings to secure your platform" on page 1 208.

`/etc/tower/conf.d/postgres.py`	Contains the password used by automation controller to connect to the database, unless TLS authentication is used for the database connection.
`/etc/tower/conf.d/channels.py`	Contains the secret used by automation controller for WebSocket broadcasts.
`/etc/tower/conf.d/gateway.py`	Contains the key used by automation controller to sync state with the platform gateway.
Platform gateway secrets	
File path	**Description**
`/etc/ansible-automation-platform/gateway/SECRET_KEY`	A secret key used for encrypting automation secrets in the database. If the `SECRET_KEY changes` or is unknown, the platform gateway cannot access the encrypted secrets in the database.
`/etc/ansible-automation-platform/gateway/gateway.cert`	SSL/TLS certificate for the platform gateway web service.

	A self-signed certificate is installed by default, although a user-provided certificate and key pair can be used. For more information, see *Installing with user-provided PKI certificates* in See "Installation settings to secure your platform" on page 1 208.
`/etc/ansible-automation-platform/gateway/gateway.key`	SSL/TLS key for the platform gateway web service. automation controller config A self-signed certificate is installed by default, although a user-provided certificate and key pair can be used. For more information, see *Installing with user-provided PKI certificates* in See "Installation settings to secure your platform" on page 1 208.
`/etc/ansible-automation-platform/gateway/cache.cert`	SSL/TLS certificate used for mutual TLS (mTLS) authentication with the Redis cache used by the platform gateway.
`/etc/ansible-automation-platform/gateway/cache.key`	SSL/TLS key used for mutual TLS (mTLS) authentication with the Redis cache used by the platform gateway.
`/etc/ansible-automation-platform/gateway/settings.py`	Contains the password used by the platform gateway to connect to the database, unless TLS authentication is used for the database connection. Also contains the password used to connect to the Redis cache used by the platform gateway.

Automation hub secrets

File path	Description
`/etc/pulp/settings.py`	Contains the password used by automation hub to connect to the database, unless TLS authentication is used for the database connection. Contains the Django secret key used by the automation hub web service.
`/etc/pulp/certs/token_public_key.pem`	OpenSSL public key in PEM format for the automation hub EE token authentication. It is generated by default from the `token_private_key.pem` file.

`/etc/pulp/certs/token_private_key.pem`	OpenSSL private key in PEM format for the automation hub EE token authentication. It is generated by default, although a user can provide their own private key with the `pulp_token_auth_key` installation inventory variable.
`/etc/pulp/certs/pulp_webserver.crt`	SSL/TLS certificate for the automation hub web service. A self-signed certificate is installed by default, although a user-provided certificate and key pair can be used. For more information, see *Installing with user-provided PKI certificates* in See "Installation settings to secure your platform" on page 1 208.
`/etc/pulp/certs/pulp_webserver.key`	SSL/TLS key for the automation hub web service. A self-signed certificate is installed by default, although a user-provided certificate and key pair can be used. For more information, see *Installing with user-provided PKI certificates* in See "Installation settings to secure your platform" on page 1 208.
`/etc/pulp/certs/database_fields.symmetric.key`	A key used for encrypting sensitive fields in the automation hub database table. If the key changes or is unknown, automation hub cannot access the encrypted fields in the database.
Event-Driven Ansible secrets	
File path	**Description**
`/etc/ansible-automation-platform/eda/SECRET_KEY`	A secret key used for encrypting fields in the Event-Driven Ansible controller database table. If the SECRET_KEY changes or is unknown, the Event-Driven Ansible controller cannot access the encrypted fields in the database.

`/etc/ansible-automation-platform/eda/settings.yaml`	Contains the password used by the Event-Driven Ansible gateway to connect to the database, unless TLS authentication is used for the database connection. Contains the password used to connect to the Redis cache used by the Event-Driven Ansible controller. Contains the key used by the Event-Driven Ansible controller to sync state with the platform gateway.
`/etc/ansible-automation-platform/eda/server.cert`	SSL/TLS certificate for the Event-Driven Ansible controller web service. A self-signed certificate is installed by default, although a user-provided certificate and key pair can be used. For more information, see see *Installing with user-provided PKI certificates* in See "Installation settings to secure your platform" on page 1 208.
`/etc/ansible-automation-platform/eda/server.key`	SSL/TLS key for the Event-Driven Ansible controller web service. A self-signed certificate is installed by default, although a user-provided certificate and key pair can be used. For more information, see *Installing with user-provided PKI certificates* in See "Installation settings to secure your platform" on page 1 208.
`/etc/ansible-automation-platform/eda/cache.cert`	SSL/TLS certificate used for mutual TLS (mTLS) authentication with the Redis cache used by the Event-Driven Ansible controller
`/etc/ansible-automation-platform/eda/cache.key`	SSL/TLS key used for mutual TLS (mTLS) authentication with the Redis cache used by the Event-Driven Ansible controller
`/etc/ansible-automation-platform/eda/websocket.cert`	SSL/TLS certificate for the Event-Driven Ansible controller WebSocket endpoint.

	A self-signed certificate is installed by default, although a user-provided certificate and key pair can be used. For more information, see *Installing with user-provided PKI certificates* in See "Installation settings to secure your platform" on page 1208.
`/etc/ansible-automation-platform/eda/websocket.key`	SSL/TLS key for the Event-Driven Ansible controller WebSocket endpoint. A self-signed certificate is installed by default, although a user-provided certificate and key pair can be used. For more information, see *Installing with user-provided PKI certificates* in See "Installation settings to secure your platform" on page 1208.
Redis secrets	
File path	**Description**
`/etc/ansible-automation-platform/ca/ansible-automation-platform-managed-ca-cert.crt`	SSL/TLS certificate for the internal self-signed certificate authority used by the installation program to generate the default self-signed certificates for each component service.
`/etc/ansible-automation-platform/ca/ansible-automation-platform-managed-ca-cert.key`	SSL/TLS key for the internal self-signed certificate authority used by the installation program to generate the default self-signed certificates for each component service.

NOTE:
Some of these file locations reflect the previous product name of automation controller, formerly named Ansible Tower.

Container-based installation secrets

The secrets listed for RPM-based installations are also used in container-based installations, but they are stored in a different manner. Container-based installations of Red Hat Ansible Automation Platform use Podman secrets to store operational secrets. These secrets can be listed using the `podman secret list` command.

By default, Podman stores data in the home directory of the user who installed and runs the containerized Red Hat Ansible Automation Platform services. Podman secrets are stored in the file `$HOME/.local/share/containers/storage/secrets/filedriver/secretsdata.json` as base64-encoded strings, so while they are not in plain text the values are only obfuscated.

The data stored in a Podman secret can be shown using the command `podman secret inspect --showsecret <secret>`.

This file should be routinely monitored to ensure there has been no unauthorized access or modification.

Related information
Installation settings to secure your platform

Automation use secrets

Ansible Automation Platform stores a variety of secrets in the database that are either used for automation or are a result of automation. Automation use secrets include:

- All secret fields of all credential types (passwords, secret keys, authentication tokens, secret cloud credentials).
- Secret tokens and passwords for external services defined in automation controller settings.
- "password" type survey field entries.

You can grant users and teams the ability to use these credentials without actually exposing the credential to the user. This means that if a user moves to a different team or leaves the organization, you do not have to re-key all of your systems.

Ansible Automation Platform uses SSH (or the Windows equivalent) to connect to remote hosts. To pass the key from automation controller to SSH, the key must be decrypted before it can be written to a named pipe. Automation controller then uses that pipe to send the key to SSH (so that it is never written to disk). If passwords are used, automation controller handles those by responding directly to the password prompt and decrypting the password before writing it to the prompt.

As an administrator with superuser access, you can define a custom credential type in a standard format by using a YAML/JSON-like definition. This allows the assignment of new credential types to jobs and inventory updates. This, in turn, lets you to define a custom credential type that works in ways similar to existing credential types. For example, you can create a custom credential type that injects an API token for a third-party web service into an environment variable. Your playbook or custom inventory script can then consume this.

To encrypt secret fields, Ansible Automation Platform uses the *Advanced Encryption Standard* (AES) in *Cipher Block Chaining* (CBC) mode with a 256-bit key for encryption, *Public-Key cryptography Standard* (PKCS7) padding, and *Hash-Based Message Authentication Code* (HMAC) using SHA256 for authentication. The encryption and decryption processes derive the AES-256 bit encryption key from the `SECRET_KEY`, the field name of the model field, and the

database-assigned auto-incremented record ID. Thus, if any attribute used in the key generation process changes, Ansible Automation Platform fails to correctly decrypt the secret. Ansible Automation Platform is designed such that the `SECRET_KEY` is never readable in playbooks Ansible Automation Platform launches. This means that these secrets are never readable by Ansible Automation Platform users, and no secret field values are ever made available through the Ansible Automation Platform REST API. If a secret value is used in a playbook, you must use `no_log` on the task so that it is not accidentally logged.

Protect sensitive data with no_log

If you save Ansible output to a log, you expose any secret data in your Ansible output, such as passwords and usernames. To keep sensitive values out of your logs, mark tasks that expose them with the `no_log: True` attribute.

However, the `no_log` attribute does not affect debugging output, so be careful not to debug playbooks in a production environment.

Best practices for securing user accounts

When planning user authentication for Ansible Automation Platform, consider both infrastructure-level and application-level authentication requirements.

There are two types of user accounts to consider in an Ansible Automation Platform environment:

- Infrastructure accounts: user accounts on the RHEL servers that run the Ansible Automation Platform services.
- Application accounts: user accounts for the Ansible Automation Platform web UI and API.

Infrastructure server account planning

For user accounts on the RHEL servers that run Ansible Automation Platform services, follow your organizational policies to determine if individual user accounts should be local or should use an external authentication source.

Only users who have a valid need to perform maintenance tasks on the Ansible Automation Platform components themselves should have access to the underlying RHEL servers, as the servers store configuration files that contain sensitive information, such as encryption keys and service passwords. Because these individuals must have privileged access to support Ansible Automation Platform services, minimizing access to the underlying RHEL servers is critical. Do not grant sudo access to the root account or local Ansible Automation Platform service accounts (awx, pulp, postgres) to untrusted users.

NOTE:
Some local service accounts are created and managed by the RPM-based installation program. These particular accounts on the underlying RHEL hosts cannot come from an external authentication source.

Ansible Automation Platform account planning

Ansible Automation Platform user accounts for accessing the user interface or API can either be local (stored in the Ansible Automation Platform database) or mapped to an external authentication source, such as a *Lightweight Directory Access Protocol* (LDAP) server.

Where possible, map all primary user accounts to an external authentication source. Using external account sources eliminates a source of error when working with permissions in this context and minimizes the amount of time devoted to maintaining a full set of users exclusively within Ansible Automation Platform. This includes accounts assigned to individual persons and for non-person entities, such as service accounts used for external application integration. Reserve any local accounts, such as the default "admin" account, for emergency access or "break glass" scenarios where the external authentication mechanism isn't available.

- LDAP
- SAML
- TACACS+
- Radius
- Azure Active Directory
- Google OAuth
- Generic OIDC
- Keycloak
- GitHub
- GitHub Organization
- GitHub team
- GitHub enterprise
- GitHub enterprise organization
- GitHub enterprise team

Select an authentication mechanism compliant with your organization's policies, and review Access management and authentication for prerequisite details. The authentication mechanism used

must ensure that the authentication-related traffic between Ansible Automation Platform and the authentication back-end is encrypted when the traffic occurs on a public or insecure network (for example, LDAPS or LDAP over TLS, HTTPS for OAuth2 and SAML providers, and so on.).

In the Ansible Automation Platform UI, any "system administrator" account can edit, change, and update any inventory or automation definition. Restrict these account privileges to the minimum set of users possible for low-level automation controller configuration and disaster recovery.

NOTE:
Ansible Automation Platform 2.6 introduces a new central authentication mechanism used by all of the platform components:

- Automation controller
- Private automation hub
- Event-Driven Ansible controller

Before 2.6, each of these components had their own authentication configuration.

Best practices for setting up secure logging

Visibility and analytics are important pillars of Enterprise Security and Zero Trust architectures. Logging is key to capturing actions and auditing.

You can manage logging and auditing by using the built-in audit support described in the Auditing the suystem section of the Security hardening for Red Hat Enterprise Linux guide. Ansible Automation Platform's built-in logging and activity stream logs all change within Red Hat Ansible Automation Platform and automation logs for auditing purposes.

Ansible Automation Platform and the underlying Red Hat Enterprise Linux systems should be configured to collect logging and auditing centrally, rather than reviewing it on the local system. Configure Ansible Automation Platform to use external logging to compile log records from multiple components within the Ansible Automation Platform server. The events occurring must be time-correlated to conduct accurate forensic analysis.

Another critical capability of logging is the ability to use cryptography to protect the integrity of log tools. Log data includes all information (for example, log records, log settings, and log reports) needed to successfully log information system activity. It is common for attackers to replace the log tools or inject code into the existing tools to hide or erase system activity from the logs. To address this risk, log tools must be cryptographically signed so that you can identify when the log tools have been modified, manipulated, or replaced. For example, one way to validate that the log tool(s) have not been modified, manipulated or replaced is to use a checksum hash against the tool file(s). This ensures the integrity of the tool(s) has not been compromised.

Related information

Auditing the system
Send log files to third-party aggregation services

Configure centralized logging

Configure centralized logging to collect all Ansible Automation Platform logs in a single location. Consolidating this data makes it easier to troubleshoot issues, detect tampering, and helps ensure the overall security and stability of your environment.

There are several additional benefits including:

- The data is sent in JSON format over a HTTP connection using minimal service-specific tweaks engineered in a custom handler or through an imported library. The types of data that are most useful to the controller are job fact data, job events/job runs, activity stream data, and log messages.
- Deeper insights into the automation process by analyzing logs from different parts of the infrastructure, including playbook execution details, task outcomes, and system events.
- Identifying performance bottlenecks and optimizing the Ansible playbooks by analyzing execution times and resource usage from the logs.
- Centralized logging helps meet compliance mandates by providing a single source of truth for auditing purposes.
- Third Party integration with a centralized log management platform like Splunk, Logstash, ElasticSearch, or Loggly to collect and analyze logs.

The logging aggregator service works with the following monitoring and data analysis systems:

- Splunk
- Loggly
- Sumologic
- Elastic stack (formerly ELK stack)

Set up logging

To set up logging to any of the aggregator types for centralized logging follow these steps:

- From the navigation panel, select **Settings › Automation Execution › Logging**.
- On the **Logging settings** page, click **Edit**.
- You can configure the following options:

- **Logging Aggregator**: Enter the hostname or IP address that you want to send logs to.
- **Logging Aggregator Port**: Specify the port for the aggregator if it requires one.

> **NOTE:**
> When the connection type is HTTPS, you can enter the hostname as a URL with a port number, after which, you are not required to enter the port again. However, TCP and UDP connections are determined by the hostname and port number combination, rather than URL. Therefore, in the case of a TCP or UDP connection, supply the port in the specified field. If a URL is entered in the **Logging Aggregator** field instead, its hostname portion is extracted as the hostname.

- **Logging Aggregator Type**: Click to select the aggregator service from the list:
- **Logging Aggregator Username**: Enter the username of the logging aggregator if required.
- **Logging Aggregator Password/Token**: Enter the password of the logging aggregator if required.
- **Loggers to Send Data to the Log Aggregator Form**: All four types of data are pre-populated by default. Click the tooltip ⓘ icon next to the field for additional information on each data type. Delete the data types you do not want.
- **Cluster wide unique identifier**: Use this to uniquely identify instances.
- **Logging Aggregator Protocol**: Click to select a connection type (protocol) to communicate with the log aggregator. Subsequent options vary depending on the selected protocol.
- **TCP Connection Timeout**: Specify the connection timeout in seconds. This option is only applicable to HTTPS and TCP log aggregator protocols.
- **Logging Aggregator Level Threshold**: Select the level of severity you want the log handler to report.
- **Maximum number of messages that can be stored in the log action queue**:Defines how large the `rsyslog` action queue can grow in number of messages stored. This can have an impact on memory use. When the queue reaches 75% of this number, the queue starts writing to disk (`queue.highWatermark` in `rsyslog`). When it reaches 90%, `NOTICE` , `INFO` , and `DEBUG` messages start to be discarded (`queue.discardMark` with `queue.discardSeverity=5`).
- **Maximum disk persistence for rsyslogd action queuing (in GB)**: The amount of data to store (in gigabytes) if an `rsyslog` action takes time to process an incoming message (defaults to 1). Equivalent to the `rsyslogd queue.maxdiskspace` setting on the action (e.g. `omhttp`). It stores files in the directory specified by `LOG_AGGREGATOR_MAX_DISK_USAGE_PATH` .

- **File system location for rsyslogd disk persistence**: Location to persist logs that should be retried after an outage of the external log aggregator (defaults to `/var/lib/awx`). Equivalent to the `rsyslogd queue.spoolDirectory` setting.: Configure a specific error message. When the API encounters an issue with a request, it typically returns an HTTP error code in the 400 range along with an error. When this happens, an error message is generated in the log that follows the following pattern:
- **Log Format For API 4XX Errors:** When the API encounters an issue with a request, it typically returns an HTTP error code in the 400 range along with an error. When this happens, an error message is generated in the log that follows the following pattern: `' status {status_code} received by user {user_name} attempting to access {url_path} from {remote_addr} '`

- You can set the following options:
  - **Log System Tracking Facts Individually**: Click the tooltip ⓘ icon for additional information, such as whether or not you want to turn it on, or leave it off by default.
- Review your entries for your chosen logging aggregation.
  - **Enable External Logging**: Select this checkbox if you want to send logs to an external log aggregator.
  - **Enable/disable HTTPS certificate verification**: Certificate verification is enabled by default for the HTTPS log protocol. Select this checkbox if you want the log handler to verify the HTTPS certificate sent by the external log aggregator before establishing a connection.
  - **Enable rsyslogd debugging**: Select this checkbox to enable high verbosity debugging for `rsyslogd`. Useful for debugging connection issues for external log aggregation.
- Click **Save** or **Cancel** to abandon the changes.

Related information
API 4XX Error Configuration

Configure LDAP logging

Enable debug logging for LDAP in the platform gateway settings to capture detailed authentication messages. Reviewing these logs ensures that you can effectively troubleshoot and resolve LDAP connection issues.

Procedure

1. Edit the gateway settings file:
   a. On Ansible Automation Platform 2.6 Containerized, the file is `~/aap/gateway/etc/settings.py` (as the user running the platform gateway container).

b. On Ansible Automation Platform 2.6 RPM-based, the file is `/etc/ansible-automation-platform/gateway/settings.py`.

```
  (...)

  CACHES['fallback']['LOCATION'] = '/var/cache/ansible-automation-
platform/gateway'


  LOGGING['loggers']['aap']['level'] = 'INFO'

  LOGGING['loggers']['ansible_base']['level'] = 'INFO'

  LOGGING['loggers']['django_auth_ldap']['level'] = 'DEBUG'      ######
   add this line


  (...)
```

2. Restart the platform gateway service or container:

   a. On Ansible Automation Platform 2.6 Containerized, restart the platform gateway service so that it restarts the platform gateway container:

> **NOTE:**
> Ensure that you run `systemctl with the `--user`` flag as follows:
>
> + `$ systemctl --user restart automation-gateway`

   b. On Ansible Automation Platform 2.6 RPM-based, use the `automation-gateway-service` command:

```
# automation-gateway-service restart
```

Implement security control

Some of the following examples of meeting compliance requirements come from the US DoD *Security Technical Implementation Guide*, but go back to integrity and security best practices.

Automation controller must use external log providers that can collect user activity logs in independent, protected repositories to prevent modification or repudiation. Automation controller must be configured to use external logging to compile log records from multiple components within the server. The events occurring must be time-correlated in order to conduct accurate forensic analysis. In addition, the correlation must meet certain tolerance criteria.

The following steps implement the security control:

Procedure

1. Log in to automation controller as an administrator.
2. From the navigation panel, select **Settings › Automation Execution › Logging**.
3. On the **Logging settings** page, click **Edit**.
4. Set the following fields:
    - Set **Logging Aggregator** to `Not configured`. This is the default.
    - Set **Enable External Logging** to `On`.
    - Set **Logging Aggregator Level Threshold** to DEBUG.
    - Set **TCP Connection Timeout** to 5 (the default) or to the organizational timeout.
    - Set **Enable/disable HTTPS certificate verification** to `On`.
5. Click **Save**.

    Automation controller must allocate log record storage capacity and shut down by default upon log failure (unless availability is an overriding concern). It is critical that when a system is at risk of failing to process logs, it detects and takes action to mitigate the failure. Log processing failures include software/hardware errors, failures in the log capturing mechanisms, and log storage capacity being reached or exceeded. During a failure, the application server must be configured to shut down unless the application server is part of a high availability system. When availability is an overriding concern, other approved actions in response to a log failure are as follows:
6. If the failure was caused by the lack of log record storage capacity, the application must continue generating log records if possible (automatically restarting the log service if necessary), overwriting the oldest log records in a first-in-first-out manner.
7. If log records are sent to a centralized collection server and communication with this server is lost or the server fails, the application must queue log records locally until communication is restored or until the log records are retrieved manually. Upon restoration of the connection to the centralized collection server, action must be taken to synchronize the local log data with the collection server.

    The following steps implement the security control:

    a. Open a web browser and navigate to the logging API, `/api/v2/settings/logging/`

        Ensure that you are authenticated as an automation controller administrator.

    b. In the **Content** section, modify the following values:

        - `LOG_AGGREGATOR_ACTION_MAX_DISK_USAGE_GB` = organization-defined requirement for log buffering.
        - `LOG_AGGREGATOR_MAX_DISK_USAGE_PATH` = `/var/lib/awx`

    c. Click **PUT**.

Implement security control for each host

Restrict access to automation controller log files using explicitly defined privileges. Protecting log confidentiality prevents attackers from gathering sensitive system details and helps ensure your environment is safe from privilege escalation or lateral movement.

To implement the security control, use the following procedure:

Procedure

1. As a system administrator for each automation controller host, set the permissions and owner of the automation controller NGINX log directory:
   - `chmod 770 /var/log/nginx`
   - `chown nginx:root /var/log/nginx`
2. Set the permissions and owner of the automation controller log directory:
   - `chmod 770 /var/log/tower`
   - `chown awx:awx /var/log/tower`
3. Set the permissions and owner of the automation controller supervisor log directory:
   - `chmod 770 /var/log/supervisor/`
   - `chown root:root /var/log/supervisor/`
     Automation controller must be configured to fail over to another system in case of log subsystem failure. Automation controller hosts must be capable of failing over to another automation controller host which can handle application and logging functions upon detection of an application log processing failure. This enables continual operation of the application and logging functions while minimizing the loss of operation for the users and loss of log data.
   - If automation controller is not in an HA configuration, the administrator must reinstall automation controller.

Implement security control for system administrators

Configure your automation controller web server to log detailed user session records. Capturing this data supports troubleshooting, debugging, and forensic analysis, and helps ensure you retain essential auditing tools for event investigations.

Use the following procedure to implement the security control as a System Administrator for each automation controller host:

Procedure

1. From the navigation panel, select **Settings › Automation Execution › System**. The System Settings page is displayed.
2. Click **Edit**.
3. Set the following:
    - **Enable Activity Stream** = On
    - **Enable Activity Stream for Inventory Sync** = On
    - **Organization Admins Can Manage Users and Teams** = On
    - **All Users Visible to Organization Admins** = On
4. Click **Save**.

Apply the NIST Cybersecurity Framework

Ansible Automation Platform should be used to fulfill security policy requirements by applying the NIST Cybersecurity Framework for common use cases, such as:

- Requiring HTTPS for web servers on Red Hat Enterprise Linux.
- Requiring TLS encryption for internal communication between web servers and database servers on Red Hat Enterprise Linux.
- Generating reports showing that the policy is properly deployed.
- Monitoring for drift that violates the policy.
- Automating correction of any policy violation.

This can be done through 5 steps of the cybersecurity framework:

IDENTIFY

Define the requirements to be implemented according to the security policy.

PROTECT

Implement and apply the requirements as an Ansible Playbook.

DETECT

Monitor for drift and generate an audit report.

RESPOND

Explore actions that could be taken when an incident is detected.

RECOVER

Use Ansible to restore the systems to the known good configuration.

Secure your Red Hat Enterprise Linux hosts

The security of Ansible Automation Platform relies in part on the configuration of the underlying Red Hat Enterprise Linux servers.

For this reason, the underlying Red Hat Enterprise Linux hosts for each Ansible Automation Platform component must be installed and configured in accordance with the Security hardening for Red Hat Enterprise Linux 8 or Security hardening for Red Hat Enterprise Linux 9 (depending on which operating system is used). In addition, install and configure any security profile requirements (*Center for Internet Security* (CIS), STIG, *Health Insurance Portability and Accountability Act* (HIPAA), and so on) used by your organization. This document recommends Red Hat Enterprise Linux 9 for all new deployments. When using the container-based installation method, Red Hat Enterprise Linux 9 is required.

Related information
Security hardening for Red Hat Enterprise Linux 8
Security hardening for Red Hat Enterprise Linux 9

Ansible Automation Platform and additional software

Dedicate your Red Hat Enterprise Linux servers exclusively to Ansible Automation Platform components. Avoiding the installation of additional server capabilities helps protect system security and performance while maintaining a supported configuration.

Similarly, when Ansible Automation Platform is deployed on a Red Hat Enterprise Linux host, it installs software such as the nginx web server, the Pulp software repository, and the PostgreSQL database server (unless a user-provided external database is used). This software should not be modified or used in a more generic fashion (for example, do not use nginx to serve additional website content or PostgreSQL to host additional databases) as this is an unsupported configuration and might affect the security and performance of Ansible Automation Platform. The configuration of this software is managed by the Ansible Automation Platform installation program, and any manual changes might be undone when performing upgrades.

Installation settings to secure your platform

Installation decisions directly impact the security posture of Ansible Automation Platform. The process involves setting several variables critical to infrastructure hardening. Before installing, review the installation guidance to ensure your configuration meets security standards.

Install from a dedicated installation host

The Ansible Automation Platform installation program can be run from one of the infrastructure servers, such as an automation controller, or from an external system that has SSH access to the Ansible Automation Platform infrastructure servers.

The Ansible Automation Platform installation program is also used not just for installation, but for subsequent day-two operations, such as backup and restore, and upgrades. Perform installation and day-two operations from a dedicated external server, hereafter referred to as the installation host. Doing so eliminates the need to log in to one of the infrastructure servers to run these functions. The installation host must only be used for management of Ansible Automation Platform and must not run any other services or software.

The installation host must be a Red Hat Enterprise Linux server that has been installed and configured in accordance with Security hardening for Red Hat Enterprise Linux and any security profile requirements relevant to your organization (CIS, STIG, and so on). Obtain the Ansible Automation Platform installer and create the installation program inventory file. This inventory file is used for upgrades, adding infrastructure components, and day-two operations by the installation program, so preserve the file after installation for future operational use.

Access to the installation host must be restricted only to those personnel who are responsible for managing the Ansible Automation Platform infrastructure. Over time, it will contain sensitive information, such as the installation inventory (which contains the initial login credentials for Ansible Automation Platform), copies of user-provided PKI keys and certificates, backup files, and so on. The installation host must also be used for logging in to the Ansible Automation Platform infrastructure servers through SSH when necessary for infrastructure management and maintenance.

Related information
Security hardening

Security-relevant variables in the installation inventory

Customize the installation inventory file to define your Ansible Automation Platform architecture and change the initial configuration of its components.

The following table lists several security-relevant variables and their recommended values for an RPM-based deployment.

Security-relevant inventory variables

RPM deployment variable	Recommended Value	Details
`postgres_use_ssl`	true	The installation program configures the installation program-managed Postgres database to accept SSL/TLS-based connections when this variable is set. The default for this variable is false which means SSL/TLS is not used for PostgreSQL connections. When set to true, the platform connects to PostgreSQL by using SSL/TLS.

`pg_sslmode`
`automation_gateway_pg_sslmode`
`automationhub_pg_sslmode`
`automationcontroller_pg_sslmode`
	verify-full	These variables control mutual TLS (mTLS) authentication to the database. By default, when each service connects to the database, it tries an encrypted connection, but it is not enforced.
		
		Setting this variable to `verify-full` enforces an mTLS negotiation between the service and the database. The `postgres_use_ssl` variable must also be set to `true`

RPM deployment variable	Recommended Value	Details
		for this pg_sslmode to be effective. **NOTE**: If a third-party database is used instead of the installation program-managed database, the third-party database must be set up independently to accept mTLS connections.

`nginx_disable_hsts`
`automation_gateway_disable_hsts`
`automationhub_disable_hsts`
`automationcontroller_disable_hsts`
	false	If set to `true`, these variables disable HTTPS *strict transport Security* (HSTS) connections to each of the component web services.
		
		The default is `false`. If these variables are absent from the installation program inventory it is effectively equivalent to defining the variables as `false`.

The following table lists several security-relevant variables and their recommended values for a container-based deployment.

Security-relevant containerized inventory variables

Container deployment variable	Recommended Value	Details
`postgresql_disable_tls`	false	If set to `true`, this variable disables TLS connections to the installation program-managed PostgreSQL database. The default is `false` If this variable is absent from the installation program inventory, it is effectively equivalent to defining the variable as `false`.

Container deployment variable	Recommended Value	Details

`controller_pg_sslmode`
`gateway_pg_sslmode`
`hub_pg_sslmode`
`eda_pg_sslmode`
	verify-full	These variables control mutual TLS (mTLS) authentication to the database.
		
		By default, when each service connects to the database, it tries an encrypted connection, but it is not enforced. Setting this variable to `verify-full` enforces an mTLS negotiation between the service and the database.
		
		NOTE:
		If a third-party database is used instead of the installation program-managed database, the third-party database must be set up independently to accept mTLS connections.

`controller_nginx_disable_https`
`gateway_nginx_disable_https`
`hub_nginx_disable_https`
`da_nginx_disable_https`
	`false`	If set to `true`, these variables disable HTTPS connections to each of the component web services.
		
		The default is `false`.
		
		If these variables are absent from the installation program inventory, it is effectively equivalent to defining the variables as `false`.

`controller_nginx_disable_hsts`
`gateway_nginx_disable_hsts`
`hub_nginx_disable_hsts`
`eda_nginx_disable_hsts`
	`false`	If set to 'true', these variables disable HTTPS Strict Transport Security (HSTS) connections to each of the component web services.

Container deployment variable	Recommended Value	Details
		The default is `false`. If these variables are absent from the installation program inventory it is effectively equivalent to defining the variables as `false`.

In an enterprise architecture where a load balancer is used in front of multiple platform gateways, SSL/TLS client connections can be terminated at the load balancer or passed through to the individual AAP servers. If SSL/TLS is being terminated at the load balancer, this section recommends that the traffic gets re-encrypted from the load balancer to the individual Ansible Automation Platform servers. This ensures that end-to-end encryption is in use. In this scenario, the `*_disable_https` variables listed are set to the default value of `false`.

Install with user-provided PKI certificates

Replace the default self-signed certificates with custom PKI certificates for your Ansible Automation Platform components. Using your existing PKI infrastructure helps ensure trusted and secure communication across the platform.

Procedure

1. Copy the certificate files and their relevant key files to the installation program directory, along with the CA certificate used to verify the certificates.
2. Use the following inventory variables to configure the infrastructure components with the new certificates.

PKI certificate inventory variables

RPM Variable	Containerized Variable	Details
`custom_ca_cert`	`custom_ca_cert`	The path to the custom CA certificate file. If set, this will install a custom CA certificate to the system truststore.
`web_server_ssl_cert`	`controller_tls_cert`	The file name of the automation controller PKI certificate located in

		the installation program directory.
`web_server_ssl_key`	`controller_tls_key`	The file name of the automation controller PKI key located in the installation program directory.
`automationhub_ssl_cert`	`hub_tls_cert`	The file name of the private automation hub PKI certificate located in the installation program directory.
`automationhub_ssl_key`	`hub_tls_key`	The file name of the private automation hub PKI key located in the installation program directory.
`postgres_ssl_cert`	`postgresql_tls_cert`	The file name of the database server PKI certificate located in the installation program directory. This variable is only needed for the installation program managed database server, not if a third-party database is used.
`postgres_ssl_key`	`postgresql_tls_key`	The file name of the database server PKI key located in the installation program directory. This variable is only needed for the installation program-managed database server, not if a third-party database is used.
`automationedacontroller_ssl_cert`	`eda_tls_cert`	The file name of the Event-Driven Ansible controller PKI certificate located in the installation program directory.

`automationedacontroller_ssl_key`	`eda_tls_key`	The file name of the Event-Driven Ansible controller PKI key located in the installation program directory.
-	`gateway_tls_cert`	The filename of the platform gateway PKI certificate located in the installation program directory.
-	`gateway_tls_key`	The file name of the platform gateway PKI key located in the installation program directory.

When multiple platform gateways are deployed with a load balancer, `gateway_tls_cert` and `gateway_tls_key` are shared by each platform gateway. To prevent hostname mismatches, the certificate's *Common Name* (CN) must match the DNS FQDN used by the load balancer. If your organizational policies require unique certificates for each service, each certificate requires a *Subject Alt Name* (SAN) that matches the DNS FQDN used for the load-balanced service. To install unique certificates and keys on each platform gateway, the certificate and key variables in the installation inventory file must be defined as host variables instead of in the `[all:vars]` section. For example:

```
 [automationgateway]

gateway0.example.com gateway_tls_cert=/path/to/cert0 gateway_tls_key=/path/to/
key0

gateway1.example.com gateway_tls_cert=/path/to/cert1 gateway_tls_key=/path/to/
key1



[automationcontroller]

controller0.example.com web_server_ssl_cert=/path/to/cert0
web_server_ssl_key=/path/to/key0

controller1.example.com web_server_ssl_cert=/path/to/cert1
web_server_ssl_key=/path/to/key1

controller2.example.com web_server_ssl_cert=/path/to/cert2
web_server_ssl_key=/path/to/key2



[automationhub]

hub0.example.com automationhub_ssl_cert=/path/to/cert0 automationhub_ssl_key=/
path/to/key0

hub1.example.com automationhub_ssl_cert=/path/to/cert1 automationhub_ssl_key=/
path/to/key1

hub2.example.com automationhub_ssl_cert=/path/to/cert2 automationhub_ssl_key=/
path/to/key2
```

Secure sensitive variables with ansible vault

By securing sensitive values in the installation inventory file with Ansible Vault, both RPM-based and containerized Ansible Automation Platform installations benefit from improved security, password hygiene, and maintainability.

Procedure

1. Navigate to the install directory by using the following command:

   `cd /path/to/ansible-automation-platform-setup-bundle-2.5-<version>`

2. Create a vault file by using the following command:

   `ansible-vault create vault.yml`

3. When prompted, enter a vault password This password is required to access or modify the vault and is required for day-two operations such as backups and reconfigurations.

IMPORTANT:
Passwords with special characters must be in double quotes.

4. Store the vault password securely, in accordance with your organizations security policy, for example, using a password manager or vault service.
5. Add your sensitive variables to the vault and ensure they are not also defined in the inventory file.

   To edit your vault file use:

   `ansible-vault edit <file>`

Use an external vault file with an RPM-based Ansible Automation Platform deployment

When installing Ansible Automation Platform using RPM packages, you can use an external Ansible vault file to securely provide sensitive variables, such as passwords, during the installation process.

For RPM-based installations, you can provide the Ansible vault at runtime when executing the setup script.

Add the following sensitive variables to the vault file:

```
admin_password: <secure_password>
pg_password: <secure_password>
automationgateway_admin_password: <secure_password>
automationgateway_pg_password: <secure_password>
automationhub_admin_password: <secure_password>
automationhub_pg_password: <secure_password>
automationedacontroller_admin_password: <secure_password>
automationedacontroller_pg_password: <secure_password>

*In the case of a connected installation:

registry_password: <secure_cdn_password>
```

To use the vault during installation, use the following procedure:

Procedure

1. Ensure the vault file, for example, `vault.yml`, contains all required sensitive variables.
2. Run the installation using the following command:

   ```
   ./setup.sh -e @vault.yml -ask-vault-pass
   ```

   Using this procedure ensures that the installation program reads encrypted variables from the vault and prompts for the vault password.

Use an external vault file with a containerized installation

For containerized installations of Ansible Automation Platform, use the provided automation execution playbook with the external vault file.

Add the following sensitive variables to the vault file:

```
postgresql_admin_password:  <secure_password>
gateway_admin_password:  <secure_password>
gateway_pg_password:  <secure_password>
controller_admin_password:  <secure_password>
controller_pg_password:  <secure_password>
hub_admin_password:  <secure_password>c
hub_pg_password:  <secure_password>
eda_admin_password:  <secure_password>
eda_pg_password: <secure_password>

*In the case of a connected installation:

registry_password: <secure_cdn_password>
```

To use the new Ansible vault with the installation program, use the following procedure:

Procedure

1. Ensure your vault file, for example, `vault.yml`, contains all required sensitive variables.
2. Run the container installer using the following command:

`ansible-playbook ansible.containerized_installer.install -e @vault.yml -ask-become-pass`.

Ensure that the vault file is located in the working directory, or provide the full path. Do not duplicate the encrypted variables in the `plaintext` inventory file.

Ensure compliance with host-level security controls

You can use Ansible Automation Platform to manage systems where security controls have been applied to managed RHEL nodes to meet the requirements of a compliance profile such as CIS, PCI/DSS, the DISA STIG, or similar.

In environments where these controls are required, discuss waiving the controls with your security auditor.

Fapolicyd

Set the `fapolicyd` daemon to permissive mode before installing Ansible Automation Platform. This prevents the pre-flight checks from stopping your installation and avoids subsequent operational failures caused by enforcing policies.

Procedure

1. Edit the file `/etc/fapolicyd/fapolicyd.conf` and set "permissive = 1".
2. Restart the service with the command

   `sudo systemctl restart fapolicyd.service`

NOTE:
If this security control is also applied to the installation host, the default `fapolicyd` configuration causes the Ansible Automation Platform installation program to fail. In this case, the recommendation is to set `fapolicyd` to permissive mode on the installation host.

File systems mounted with "noexec"

Remove the `noexec` mount option from the `/tmp`, `/var`, and `/var/tmp` file systems so the Ansible Automation Platform RPM installer can execute binaries. This prevents preflight check failures and helps ensure a successful installation.

To install Ansible Automation Platform, you must re-mount these file systems with the `noexec` option removed. When installation is complete, proceed with the following steps:

Procedure

1. Reapply the `noexec` option to the `/tmp` and `/var/tmp` file systems.
2. Change the automation controller job execution path from `/tmp` to an alternate directory that does not have the `noexec` option enabled.
3. To make this change, log in to the automation controller UI as an administrator, navigate to Settings and select **Jobs settings**.
4. Change the "Job execution path" setting to the alternate directory.

   During normal operations, the file system which contains the `/var/lib/awx` subdirectory (typically `/var`) must not be mounted with the `noexec` option, or the automation controller cannot run automation jobs in execution environments.
5. In environments where STIG controls are routinely audited, discuss waiving the STIG controls related to file system `noexec` with your security auditor.

User namespaces

To support Ansible Automation Platform execution environments, you must allow Linux containers. If a compliance profile (like DISA STIG) has set `user.max_user_namespaces` to "0," you must disable this control.

To check the `user.max_user_namespaces` kernel setting, complete the following steps on each Ansible Automation Platform component in the installation inventory.

Procedure

1. Log in to your automation controller at the command line.
2. Run the command `sudo sysctl user.max_user_namespaces`.
3. If the output indicates that the value is zero, look at the contents of the file `/etc/sysctl.conf` and all files under `/etc/sysctl.d/`, edit the file containing the `user.max_user_namespaces` setting, and set the value to "65535".
4. To apply this new value, run the command `sudo sysctl -p <file>`, where `<file>` is the file just modified.

5. Re-run the command `sudo sysctl user.max_user_namespaces` and verify that the value is now set to "65535".

Interactive session timeout

Temporarily increase the interactive session timeout during lengthy operations like installations, backups, and restores. This prevents compliance policies from automatically logging you out and helps ensure these critical processes complete successfully.

There are multiple ways in which this control can be enforced, including shell timeout variables, setting the idle session timeout for `systemd-logind`, or setting SSH connection timeouts, and different compliance profiles can use one or more of these methods. The one that most often interrupts the installation and day two operations is the idle session timeout for `systemd-logind`, which was introduced in the DISA STIG version V2R1 (Red Hat Enterprise Linux 8) and V2R2 (Red Hat Enterprise Linux 9). To increase the idle session timeout for `systemd-logind`, as the root user:

- Edit the file `/etc/systemd/logind.conf`.
- If the `StopIdleSessionSec` setting is set to zero, no change is needed.
- If the `StopIdleSessionSec` setting is non-zero, this indicates that the session will be terminated after that number of seconds.
  Change `StopIdleSessionSec=7200` to increase the timeout, then run `systemctl restart systemd-logind` to apply the change.
- Log out of the interactive session entirely and log back in to ensure the new setting applies to the current login session.

NOTE:
This change only needs to be made on the installation host, or if an installation host is not used, the host where the Ansible Automation Platform installation program is run.

Sudo and NOPASSWD

A compliance profile might require that all users with sudo privileges must provide a password (the `NOPASSWD` directive must not be used in a sudoers file). The installation program runs many tasks as a privileged user, and by default expects to be able to elevate privileges without a password.

To provide a password to the installation program for elevating privileges, append the following options when launching the RPM installer script:

`./setup.sh <setup options> --ask-become-pass`.

For the container-based installation program:

```
ansible-playbook ansible.containerized_installer.install --ask-become-pass
```

When the installation program is run, you are prompted for the user's password to elevate privileges.

NOTE:
Using the `--ask-become-pass` option also applies when running the installation program for day-two operations such as backup and restore.

Recommended security practices for access controls

Granting access to certain parts of the system exposes security vulnerabilities. Apply the following practices to help secure access:

- Minimize access to system administrative accounts. There is a difference between the user interface (web interface) and access to the operating system that the automation controller is running on. A system administrator or super user can access, edit, and disrupt any system application. Anyone with root access to automation controller has the potential ability to decrypt those credentials, and so minimizing access to system administrative accounts is crucial for maintaining a secure system.
- Minimize local system access. Automation controller should not require local user access except for administrative purposes. Non-administrator users should not have access to the automation controller system.
- Enforce separation of duties. Different components of automation might need to access a system at different levels. Use different keys or credentials for each component so that the effect of any one key or credential vulnerability is minimized.
- Restrict automation controller to the minimum set of users possible for low-level automation controller configuration and disaster recovery only. In an automation controller context, any automation controller 'system administrator' or 'superuser' account can edit, change, and update any inventory or automation definition in automation controller.

Use a configuration as code paradigm

Red Hat collections provide automation content to manage Red Hat Ansible Automation Platform infrastructure and configuration as code (CaC). This enables platform automation through CaC. Review the security implications of this approach.

The following Ansible content collections are available for managing Ansible Automation Platform components using an infrastructure as code methodology, all of which are found on the Ansible Automation Hub:

Ansible content collections

Validated Collection	Collection Purpose
`infra.aap_utilities`	Ansible content for automating day 1 and day 2 operations of Ansible Automation Platform, including installation, backup and restore, certificate management, and more.
`infra.aap_configuration`	A collection of roles to manage the creation of Ansible Automation Platform components, including users and groups (RBAC), projects, job templates and workflows, credentials, and more. This collection includes functionality from the older `infra.controller_configuration`, `infra.ah_configuration` and `infra.eda_configuration` and should be used in their place with Ansible Automation Platform 2.6.
`infra.ee_utilities`	A collection of roles for creating and managing execution environment images, or migrating from the older Tower virtualenvs to execution environments.

Many organizations use CI/CD platforms to configure pipelines or other methods to manage this type of infrastructure. However, using Ansible Automation Platform natively, a webhook can be configured to link a Git-based repository natively. In this way, Ansible can respond to Git events to trigger Job Templates directly. This removes the need for external CI components from this overall process and thus reduces the attack surface.

These practices enable version control of all infrastructure and configuration. Apply Git best practices to ensure proper code quality inspection before being synchronized into Ansible Automation Platform. Relevant Git best practices include the following:

- Creating pull requests.
- Ensuring that inspection tools are in place.
- Ensuring that no plain text secrets are committed.
- Ensuring that pre-commit hooks and any other policies are followed.

CaC also encourages using external vault systems which removes the need to store any sensitive data in the repository, or deal with having to individually vault files as needed. This is particularly important when storing Ansible Automation Platform configuration in a source code repository, as automation controller credentials and Event-Driven Ansible credentials must be provided to the collection variables in plain text which should not be committed to a source repository. For more information on using external vault systems, see External credential vault considerations.

Configure centralized logging

Configure centralized logging to collect all Ansible Automation Platform logs in a single location. Consolidating this data makes it easier to troubleshoot issues, detect tampering, and helps ensure the overall security and stability of your environment.

There are several additional benefits including:

- The data is sent in JSON format over a HTTP connection using minimal service-specific tweaks engineered in a custom handler or through an imported library. The types of data that are most useful to the controller are job fact data, job events/job runs, activity stream data, and log messages.
- Deeper insights into the automation process by analyzing logs from different parts of the infrastructure, including playbook execution details, task outcomes, and system events.
- Identifying performance bottlenecks and optimizing the Ansible playbooks by analyzing execution times and resource usage from the logs.
- Centralized logging helps meet compliance mandates by providing a single source of truth for auditing purposes.
- Third Party integration with a centralized log management platform like Splunk, Logstash, ElasticSearch, or Loggly to collect and analyze logs.

The logging aggregator service works with the following monitoring and data analysis systems:

- Splunk
- Loggly
- Sumologic
- Elastic stack (formerly ELK stack)

RBAC security considerations for day two operations

Day 2 Operations include Cluster Health and Scaling Checks, including Host, Project, and environment level Sustainment. You must continually analyze configuration and security drift.

RBAC considerations

Delegate access to resources and centralize credential management using Role-Based Access Controls (RBAC) in platform gateway. This helps ensure users can securely utilize secrets without exposure, increasing security and streamlining platform management.

RBAC is the practice of granting roles to users or teams. RBAC is easiest to think of in terms of Roles which define precisely who or what can see, change, or delete an “object” for which a specific capability is being set.

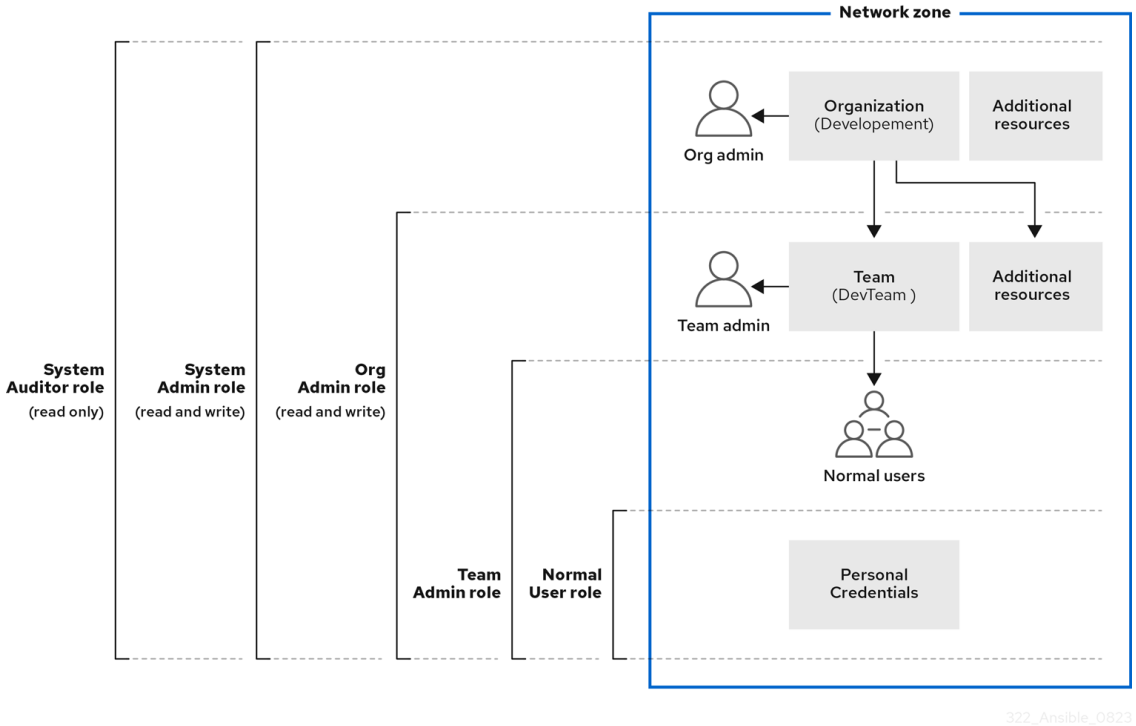
There are a few main concepts that you should become familiar with regarding Ansible Automation Platform’s RBAC design–roles, resources, and users. Users can be members of a role, which gives them certain access to any resources associated with that role, or any resources associated with “descendant” roles.

A role is essentially a collection of capabilities. Users are granted access to these capabilities and automation controller’s resources through the roles to which they are assigned or through roles inherited through the role hierarchy.

Roles associate a group of capabilities with a group of users. All capabilities are derived from membership within a role. Users receive capabilities only through the roles to which they are assigned or through roles they inherit through the role hierarchy. All members of a role have all capabilities granted to that role. Within an organization, roles are relatively stable, while users and capabilities are both numerous and may change rapidly. Users can have many roles.

The following is an example of an organization with roles and resource permissions:

Figure: RBAC role scopes within automation controller



User access is based on managing permissions to system objects (users, groups, namespaces) rather than by assigning permissions individually to specific users. You can assign permissions to the groups you create. You can then assign users to these groups. This means that each user in a group has the permissions assigned to that group.

Teams created in automation hub can range from system administrators responsible for governing internal collections, configuring user access, and repository management to groups with access to organize and upload internally developed content to automation hub.

View-only access can be enabled for further lockdown of the private automation hub. By enabling view-only access, you can grant access for users to view collections or namespaces on your private automation hub without the need for them to log in. View-only access allows you to share content with unauthorized users while restricting their ability to only view or download source code, without permissions to edit anything on your private automation hub. Enable view-only access for your private automation hub by editing the inventory file found on your Red Hat Ansible Automation Platform installer.

Related information
Managing access with role-based access control

Disaster recovery and operational continuity

Regularly back up Red Hat Ansible Automation Platform to ensure effective disaster recovery.

An important aspect of backups is that they contain a copy of the database as well as the secret key used to decrypt credentials stored in the database, so the backup files should be stored in a secure encrypted location. This means that access to endpoint credentials are protected properly. Access to backups should be limited only to Ansible Automation Platform administrators who have root shell access to automation controller and the dedicated installation host.

The two main reasons an Ansible Automation Platform administrator needs to back up their Ansible Automation Platform environment are:

- To save a copy of the data from your Ansible Automation Platform environment, so you can restore it if needed.
- To use the backup to restore the environment into a different set of servers if you're creating a new Ansible Automation Platform cluster or preparing for an upgrade.

In all cases, the recommended and safest process is to always use the same versions of PostgreSQL and Ansible Automation Platform to back up and restore the environment.

Using some redundancy on the system is highly recommended. If the secrets system is down, the automation controller cannot fetch the information and can fail in a way that would be recoverable once the service is restored. If you believe the SECRET_KEY automation controller generated for you has been compromised and has to be regenerated, you can run a tool from the installation program that behaves much like the automation controller backup and restore tool.

To generate a new secret key, perform the following steps:

Procedure

1. Backup your Ansible Automation Platform database before you do anything else! See the Back up and restore section of the Configuring automation execution guide,
2. Using the inventory from your install (same inventory with which you run backups/restores), run `setup.sh -k`.

Result

A backup copy of the previous key is saved in `/etc/tower/`.

Integrate with HashiCorp to secure sensitive data

You can integrate HashiCorp Vault with Ansible Automation Platform to manage and retrieve sensitive data.

Configure Ansible Automation Platform to communicate with HashiCorp vault

In enterprise environments, managing secrets externally is vital. A recommended HashiCorp Vault method uses AppRoles. To use these secrets, configure a new Ansible Automation Platform credential using the HashiCorp Vault Secret Lookup type.

Enter relevant information such as an identifiable credential name, organization, and the URL of the vault server, for example, https://vault.domain.com:8200.

Populate the necessary fields with your information such as Token, AppRole role_id, and AppRole secret_id, then select v2 for the API version.

To test the credential to test for functionally and operation, use the following procedure:

Procedure

1. Before clicking on **Create Credential**, click **Test**.
2. In the pop-up box, enter the **Path to Secret** and the **Key Name**.

> **NOTE:**
> The **Path to Secret** will be prefixed by `kv` if storing a key-value pair, for example, `kv/key_name`.

3. Click **Run**.
4. When the test is successful, click **Create Credential**.
5. When complete, Ansible Automation Platform is properly configured to use HashiCorp Vault as an external secret source.

Use HashiCorp Vault credentials within Ansible Automation Platform

To use HashiCorp vault credentials within Ansible Automation Platform, create a new credential with the type **Machine Credential**. Enter relevant information such as an identifiable credential name and an organization.

To configure the use of HashiCorp Vault credentials, use the following procedure:

Procedure

1. To configure the **Username**, click the 🔑 icon.
2. Select the HashiCorp Vault credentials that were created in step 1.
3. Populate **Path to Secret** and the **Key Name**.
4. Optionally, click **Test**. Otherwise, click **Finish**.

Configure the machine credential's SSH private key

Link your machine credential's SSH private key to HashiCorp Vault. Retrieving this key from an external secret management system helps ensure that sensitive authentication details are securely injected into your automation workflows.

Procedure

1. To configure the **Username**, click the 🔑 icon.
2. Select the HashiCorp Vault credentials that you created.

3. Populate the **Path to Secret** and the **Key Name**.
4. Select the name of the private key as the **Key Name**.
5. Optionally, click **Test**. Otherwise, click **Finish**.

Improve the security of nodes managed by Ansible Automation Platform

Ansible Automation Platform is an agentless technology that relies on making a remote connection to the devices it manages, called managed nodes, to run automation tasks.

NOTE: Managed node configuration can vary significantly based on factors such as operating system, compliance profiles, configuration, and organizational policies.

Any recommendations on managed node configuration presented here must be thoroughly tested and reviewed before implementation to ensure that they meet organizational policies and requirements.

Red Hat Enterprise Linux managed node configuration

The following section provides guidance for *Red Hat Enterprise Linux* (RHEL) managed nodes, but the concepts may be applicable to other Linux distributions as well.

Examples are provided for manual configuration of RHEL managed nodes. These steps can also be automated with Ansible Automation Platform, or they can be added to a *Standard Operating Environment* (SOE) or "golden image" created with a tool such as the Red Hat Lightspeed image builder.

Create a dedicated service account with access limits

Ansible Automation Platform can be configured to use various users or accounts for connecting to managed nodes.

Create a single, dedicated service account for this purpose. This service account must be a local account on each managed node to ensure automation jobs continue to run, even if an external authentication mechanism experiences an outage. This recommendation applies unless organizational policy mandates centrally managed service accounts. The service account must be clearly named to indicate its purpose, for instance, `ansible` or `aapsvc`.

"Ansible" is used here as the assumed name for a local service account in the examples.

The local service account is configured as follows:

- It is granted sufficient privileges to run any automation job required.
- It is limited to SSH key authentication only. No password authentication is allowed.
- Access is only granted to connections made from the Ansible Automation Platform {ControllerNames}s and execution nodes.

> **NOTE:**
> To execute tasks in an Ansible playbook or job template as a user other than the service account, use the `become` and `become_user` keywords. Connecting to the managed node as a different user is not necessary.

- The `useradd` command can be used to create a local service account. For example:

```
sudo useradd ansible \
    --system --create-home \
    --comment "Ansible Automation Platform service account"
```

Configure sudo privileges for service account

The service account requires sufficient privileges to run any current or future automation job on the managed node. This section describes the use of `sudo`, though other privilege escalation methods are available.

Since Ansible Automation Platform defaults to using the `ansible.builtin.sudo` become plugin on Linux-based managed nodes, the service account must be permitted to run any command on the RHEL managed node using the sudo command.

To configure this, use the following procedure:

Procedure

1. Create the file `/etc/sudoers.d/ansible`, and include the following content:

```
# Rules for the ansible service account

ansible ALL=(ALL) NOPASSWD: ALL
```

2. Set restrictive permissions on the file:

   `sudo chmod 0440 /etc/sudoers.d/ansible`

3. Verify the file uses the proper syntax:

   `sudo visudo -cf /etc/sudoers.d/ansible`

Require SSH key authentication for service account

The service account must not be permitted to use password authentication with SSH connections to the managed node.

Given that password authentication is prohibited, at least one SSH public key must be appended to the service account's authorized_keys file (typically located at `/home/ansible/.ssh/authorized_keys`).

These public keys must correspond with the private keys used to establish Machine credentials in Ansible Automation Platform, as these credentials facilitate connections to the RHEL managed nodes.

This section advocates for a single service account for connections to managed RHEL nodes, but this does not mean using a single SSH key across all nodes is required. In larger organizations, individual teams managing RHEL servers can generate their own machine credentials within Ansible Automation Platform. Teams can then add the corresponding keys to the authorized keys file on their specific RHEL servers. This approach ensures consistent access to managed nodes organization-wide by using a common service account, while enabling each team to independently manage the credentials for their assigned nodes.

Use the following procedure to configure the SSH daemon:

Procedure

1. Create the file `/etc/sshd/sshd_config.d/60-ansible.conf`, and include the following content:

```
# sshd config applied to the ansible service account
Match User ansible
    PasswordAuthentication no
Match all
```

2. Verify that the file uses the proper syntax:

   `sudo sshd -t`

3. Restart the SSH daemon

   `sudo systemctl restart sshd.service`

Enable and configure pam_access controls for service accounts

To restrict remote login access to connections originating from the Ansible Automation Platform automation controller and execution nodes and ensure the service account is exclusively used by Ansible Automation Platform, use the following procedure:

Procedure

1. Use the FQDNs of your automation controller and execution nodes to add the following content to the `/etc/security/access.conf` file, using the FQDNs of your controller nodes and execution nodes:

```
# allow the ansible service account to log in from local sources and
# the hybrid controller or execution nodes, and deny all other sources
+:ansible:LOCAL controller1.example.com controller2.example.com
en1.example.com
-:ansible:ALL
```

2. Enable the `with-pamaccess` feature in the current authselect profile. All authselect profiles included by default with RHEL have this feature.

   `sudo authselect enable-feature with-pamaccess`

Automate nodes that comply with security profiles

Edit specific security controls on your compliance-hardened RHEL nodes so Ansible Automation Platform can manage them properly. This helps ensure smooth automation in environments governed by strict profiles like CIS, PCI/DSS, or DISA STIG.

Fapolicyd on managed RHEL nodes

Ansible jobs on RHEL nodes often fail if the `fapolicyd` service is enabled. This occurs because the service prevents the execution of Python code copied to the node during task processing.

To prevent this issue from occurring, use one of the following methods:

- Option 1: Set the fapolicyd service to permissive mode
- Option 2: Create custom fapolicyd rules

Option 1: Set the fapolicyd service to permissive mode

The fapolicyd service can be set to "permissive" mode, meaning that it only logs fapolicyd rule violations, rather than enforcing them.

To configure permissive mode for fapolicyd, use the following procedure:

Procedure

1. Edit the file `/etc/fapolicyd/fapolicyd.conf`, and set "permissive = 1".
2. Restart the `fapolicy` service by running `systemctl restart fapolicyd.service`.
3. In environments where this configuration might not meet a required compliance profile or local policy, discuss waiving the relevant security control with your security auditor.

Option 2: Create custom fapolicyd rules

Where the `fapolicyd` service must enforce its rules, consider crafting a custom set of rules to permit Ansible Automation Platform to run its Python code.

The following example procedure treats the "ansible" service account as a trusted entity and enables it to run content in the local Ansible temporary directory (by default, `$HOME/.ansible/tmp`).

Procedure

1. Create the file `/etc/fapolicy/rules.d/50-ansible.rules` with the following content:

   ```
   allow perm=any uid=ansible trust=1 : dir=/home/ansible/.ansible/tmp/
   ```

2. Restart the fapolicyd service:

   ```
   sudo systemctl restart fapolicyd.service
   ```

   This example rule might require modification to work with any other `fapolicyd` rules that exist on the managed RHEL nodes, and must be thoroughly tested and approved by your security auditor before being put into production.

Security best practices

You can deploy automation controller to automate typical environments securely. However, managing certain operating system environments, automation, and automation platforms, can require additional best practices to ensure security.

Related information
Enhancing security of Red Hat Enterprise Linux 8 systems
Enhancing security of Red Hat Enterprise Linux 9 systems

Understand the architecture of Ansible Automation Platform and automation controller

Ansible Automation Platform and automation controller comprise a general-purpose, declarative automation platform. When an Ansible Playbook is launched (by automation controller, the playbook, inventory, and credentials provided to Ansible are considered to be the source of truth.

If you want policies around external verification of specific playbook content, job definition, or inventory contents, you must complete these processes before the automation is launched, either by the automation controller web UI, or the automation controller API.

The use of source control, branching, and mandatory code review is best practice for Ansible automation. There are tools that can help create process flow around using source control in this manner.

At a higher level, tools exist that enable creation of approvals and policy-based actions around arbitrary workflows, including automation. These tools can then use Ansible through the automation controller's API to perform automation.

You must use a secure default administrator password at the time of automation controller installation.

Automation controller exposes services on certain well-known ports, such as port 80 for HTTP traffic and port 443 for HTTPS traffic. Do not expose automation controller on the open internet, which reduces the threat surface of your installation.

Granting access

Granting access to certain parts of the system exposes security risks. Apply the following practices to help secure access:

Related information
Minimize administrative accounts
Minimize local system access
Remove access to credentials from users
Enforce separation of duties

Minimize administrative accounts

Minimizing the access to system administrative accounts is crucial for maintaining a secure system. A system administrator or root user can access, edit, and disrupt any system application.

Limit the number of people or accounts with root access, where possible. Do not give out *sudo* to *root* or *awx* (the automation controller user) to untrusted users. Note that when restricting administrative access through mechanisms such as *sudo*, restricting to a certain set of commands can still give a wide range of access. Any command that enables execution of a shell or arbitrary shell prompt commands, or any command that can change files on the system, is equal to full root access.

With automation controller, any automation controller "system administrator" or "superuser" account can edit, change, and update an inventory or automation definition in automation

controller. Restrict this to the minimum set of users possible for low-level automation controller configuration and disaster recovery only.

Minimize local system access

When you use automation controller with best practices, it does not require local user access except for administrative purposes. Non-administrator users do not have access to the automation controller system.

Remove user access to credentials

You can remove user access to credentials stored in an automation controller controller to enhance security.

If an automation controller credential is only stored in the controller, you can further secure it. You can configure services such as OpenSSH to only allow credentials on connections from specific addresses. Credentials used by automation can be different from credentials used by system administrators for disaster-recovery or other ad hoc management, allowing for easier auditing.

Enforce separation of duties

Separation of duties is a security principle that prevents fraud and errors by dividing responsibilities among multiple individuals or systems. You can enforce separation of duties by using different credentials for different levels of access or different types of automation tasks.

Different pieces of automation might require access to a system at different levels. For example, you can have low-level system automation that applies patches and performs security baseline checking, while a higher-level piece of automation deploys applications. By using different keys or credentials for each piece of automation, the effect of any one key vulnerability is minimized, while also enabling baseline auditing.

Available resources

Several resources exist in automation controller and elsewhere to ensure a secure platform.

Consider using the following functionalities:

- Existing security functionality

- External account stores
- Django password policies

Existing security functionality

Consider the following security best practices when implementing or deploying any new system to protect your organization's assets:

- Do not disable SELinux or automation controller's existing multitenant containment.
- Use automation controller's role-based access control (RBAC) to delegate the minimum level of privileges required to run automation. For more information, see Managing access with role based access control.
- Use teams in automation controller to assign permissions to groups of users rather than to users individually.

External account stores

Maintaining a full set of users in automation controller can be a time-consuming task in a large organization. Automation controller supports connecting to external account sources by LDAP, SAML 2.0, and certain OAuth providers. Using this eliminates a source of error when working with permissions.

Django password policies

Learn how to set password policies to validate user passwords upon creation, ensuring compliance with organizational security standards.

Add the following code block example in the `custom.py` file located at `/etc/tower/conf.d` of your automation controller instance:

SECURE

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
        'OPTIONS': {
            'min_length': 9,
        }
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

Ensure that you restart your automation controller instance for the change to take effect. For more information, see Start, stop, and restart automation controller.

Related information
Password validation

Manage firewall policies and rules with security automation

As a security operator, you can use Ansible security automation to manage multiple firewall policies or create and delete firewall rules to block or unblock a source IP address from accessing a destination IP address.

About firewall policy management

An organization's network firewall is the first line of defense against an attack and a vital component for maintaining a secure environment.

As a security operator, you construct and manage secure networks to ensure that your firewall only allows inbound and outbound network traffic defined by your organization's firewall policies. A firewall policy consists of security rules that protect the network against harmful incoming and outgoing traffic.

Managing multiple firewall rules across various products and vendors can be both challenging and time consuming for security teams. Manual workflow processes that involve complex tasks can result in errors and ultimately cause delays in investigating an application's suspicious behavior or stopping an ongoing attack on a server. When every solution in a security portfolio is automated through the same language, both security analysts and operators can perform a series of actions across various products in a fraction of the time. This automated process maximizes the overall efficiency of the security team.

Ansible security automation interacts with a wide variety of security technologies from a range of vendors. Ansible enables security teams to manage different products, interfaces, and workflows in a unified way to produce a successful deployment. For example, your security team can automate tasks such as blocking and unblocking IP and URLs on supported technologies such as enterprise firewalls.

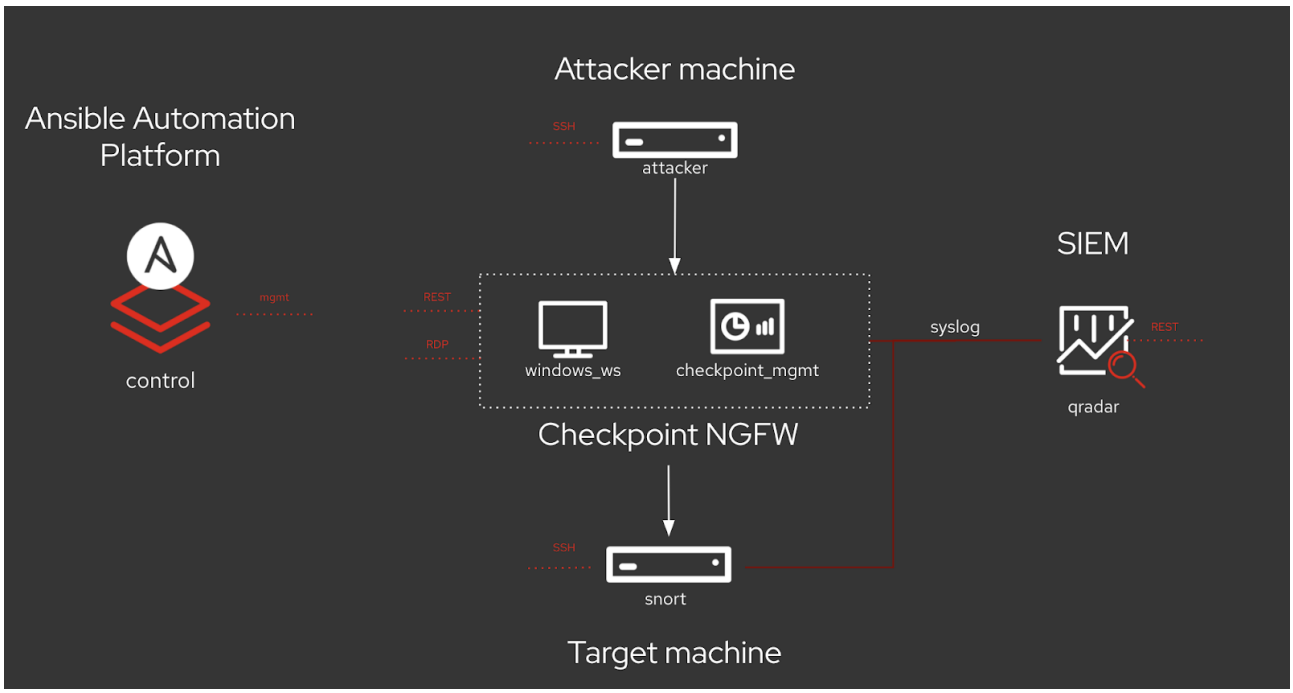
Automate firewall rules

Ansible security automation enables you to automate various firewall policies that require a series of actions across various products.

You can use an Ansible role, such as the `acl_manager` role to manage your *Access Control Lists* (ACLs) for many firewall devices such as blocking or unblocking an IP or URL. Roles let you automatically load related vars, files, tasks, handlers, and other Ansible artifacts based on a known file structure. After you group your content in roles, you can easily reuse them and share them with other users.

The following lab environment is a simplified example of a real-world enterprise security architecture, which can be more complex and include additional vendor-specific tools. This is a typical incident response scenario where you receive an intrusion alert and immediately execute a playbook with the acl_manger role that blocks the attacker's IP address.

Your entire team can use Ansible security automation to address investigations, threat hunting, and incident response all on one platform. Red Hat Ansible Automation Platform provides you with certified content collections that are easy to consume and reuse within your security team.



Related information
[acl_manager]
[Red Hat Ansible Automation Platform]

Create a new firewall rule

Use the `acl_manager` role to create a new firewall rule for blocking a source IP address from accessing a destination IP address.

Before you begin

- You have installed the latest version of ansible-core.
- You have access to the Check Point Management server to enforce the new policies

Procedure

1. Install the acl_manager role using the ansible-galaxy command.

   ```
   $ ansible-galaxy install ansible_security.acl_manager
   ```

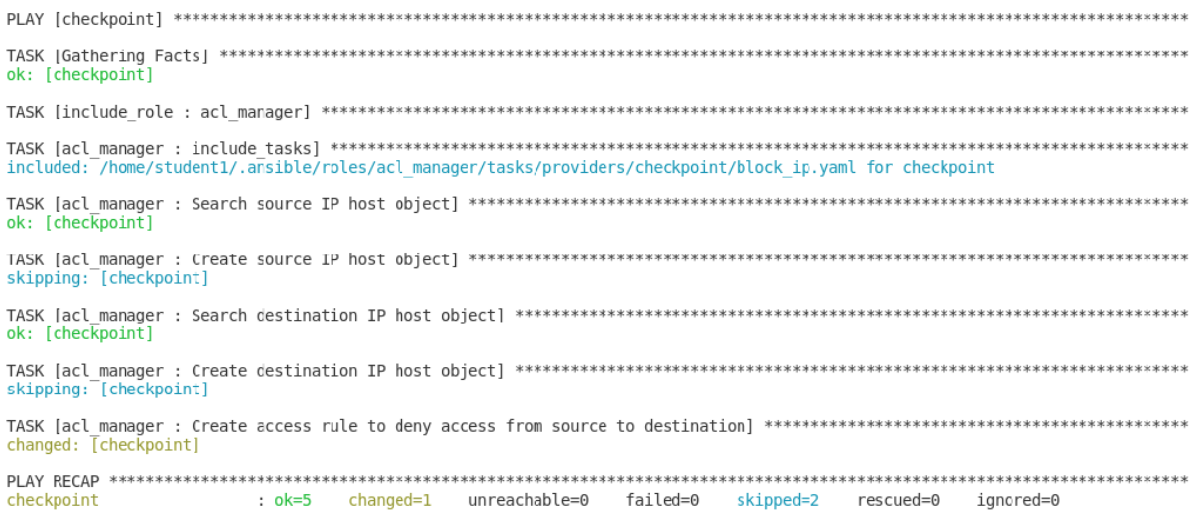
2. Create a new playbook and set the following parameter. For example, source object, destination object, access rule between the two objects and the actual firewall you are managing, such as Check Point:

```
- name: block IP address
  hosts: checkpoint
  connection: httpapi


  tasks:
    - include_role:
        name: acl_manager
        tasks_from: block_ip
      vars:
        source_ip: 172.17.13.98
        destination_ip: 192.168.0.10
        ansible_network_os: checkpoint
```

3. Run the playbook:

   `$ ansible-navigator run --ee false <playbook.yml>`.



Result

You have created a new firewall rule that blocks a source IP address from accessing a destination IP address. Access the MGMT server and verify that the new security policy has been created.

Delete a firewall rule

Use the acl_manager role to delete a security rule.

Before you begin

- You have installed Ansible 2.9 or later
- You have access to the firewall MGMT servers to enforce the new policies

Procedure

1. Install the acl_manager role using the ansible-galaxy command:

   `$ ansible-galaxy install ansible_security.acl_manager`

2. Using the CLI, create a new playbook with the acl_manger role and set the parameters, for example, source object, destination object, access rule between the two objects:

```
- name: delete block list entry
  hosts: checkpoint
  connection: httpapi


    - include_role:
        name: acl_manager
        Tasks_from: unblock_ip
      vars:
        source_ip: 192.168.0.10
        destination_ip: 192.168.0.11
        ansible_network_os: checkpoint
```

3. Run the playbook:

   `$ ansible-navigator run --ee false <playbook.yml>`:

```
PLAY [checkpoint] ***************************************************************************************************

TASK [Gathering Facts] **********************************************************************************************
ok: [checkpoint]

TASK [include_role : acl_manager] ***********************************************************************************

TASK [acl_manager : include_tasks] **********************************************************************************
included: /home/student1/.ansible/roles/acl_manager/tasks/providers/checkpoint/block_ip.yaml for checkpoint

TASK [acl_manager : Search source IP host object] *******************************************************************
ok: [checkpoint]

TASK [acl_manager : Create source IP host object] *******************************************************************
skipping: [checkpoint]

TASK [acl_manager : Search destination IP host object] **************************************************************
ok: [checkpoint]

TASK [acl_manager : Create destination IP host object] **************************************************************
skipping: [checkpoint]

TASK [acl_manager : Create access rule to deny access from source to destination] ***********************************
changed: [checkpoint]

TASK [include_role : acl_manager] ***********************************************************************************

TASK [acl_manager : include_tasks] **********************************************************************************
included: /home/student1/.ansible/roles/acl_manager/tasks/providers/checkpoint/unblock_ip.yaml for checkpoint

TASK [acl_manager : Delete access rule that deny access from source to destination] *********************************
changed: [checkpoint]

PLAY RECAP **********************************************************************************************************
checkpoint                 : ok=7    changed=2    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

4. You have deleted the firewall rule. Access the MGMT server and verify that the new security policy has been removed.

Related information
Installing roles from Galaxy

Automate network intrusion detection and prevention systems

You can use Ansible Automation Platform to automate your *Intrusion Detection and Prevention System* (IDPS). In this section, we use Snort as the IDPS. Use automation hub to consume content collections, such as tasks, roles, and modules to create automated workflows.

Requirements and prerequisites

Before you begin automating your IDPS with Ansible Automation Platform, ensure that you have the proper installations and configurations necessary to successfully manage your IDPS.

- You have installed Ansible-core 2.15 or later.
- SSH connection and keys are configured.
- IDPS software (Snort) is installed and configured.
- You have access to the IDPS server (Snort) to enforce new policies.

Verify your IDPS installation

Use the following procedure to verify that Snort has been configured successfully:

Procedure

1. Call snort using `sudo` and ask for the version:

```
   $ sudo snort --version


    ,,_     -*> Snort! <*-
   o"  )~   Version 2.9.13 GRE (Build 15013)
   ""    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
         Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights
reserved.
         Copyright (C) 1998-2013 Sourcefire, Inc., et al.
         Using libpcap version 1.5.3
         Using PCRE version: 8.32 2012-11-30
         Using ZLIB version: 1.2.7
```

2. Verify that the service is actively running using the following command:

   `sudo systemctl`:

```
 $ sudo systemctl status snort
● snort.service - Snort service
   Loaded: loaded (/etc/systemd/system/snort.service; enabled; vendor preset:
disabled)
   Active: active (running) since Mon 2019-08-26 17:06:10 UTC; 1s ago
  Main PID: 17217 (snort)
   CGroup: /system.slice/snort.service
           └─17217 /usr/sbin/snort -u root -g root -c /etc/snort/snort.conf -i
eth0 -p -R 1 --pid-path=/var/run/snort --no-interface-pidfile --nolock-pidfile
[...]
```

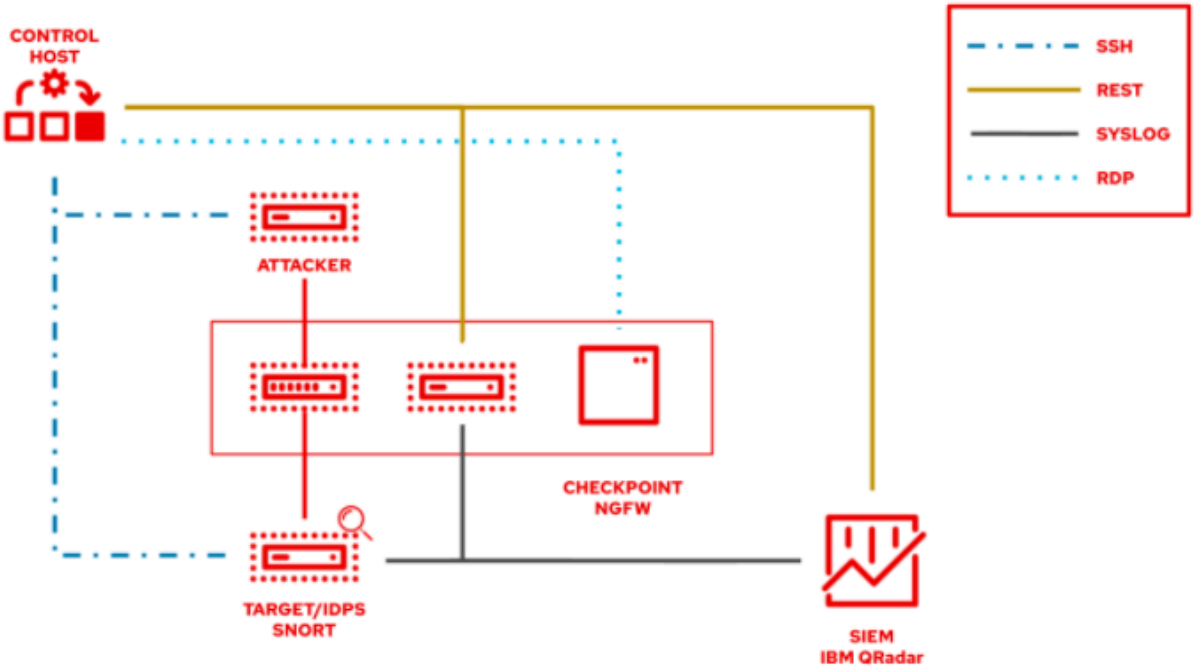
3. If the Snort service is not actively running, restart it with `systemctl restart snort` and recheck the status.
4. When you confirm that the service is actively running, exit the Snort server by simultaneously pressing `CTRL` and `D`, or by typing `exit` on the command line. All further interaction will be done through Ansible Automation Platform from the Ansible control host.

Automate your IDPS rules with Ansible Automation Platform

To automate your IDPS, use the `ids_rule` role to create and change Snort rules. Snort uses rule-based language that analyzes your network traffic and compares it against the given rule set.

The following lab environment demonstrates what an Ansible security automation integration would look like. A machine called “Attacker” simulates a potential attack pattern on the target machine on which the IDPS is running.

Keep in mind that a real world setup will feature other vendors and technologies.



Create a new IDPS rule

Use the `ids_rule` role to manage your rules and signatures for IDPS.

Before you begin

- You need `root` privileges to make any changes on the Snort server.

For example, you can set a new rule that looks for a certain pattern aligning with a previous attack on your firewall.

NOTE:
Currently, the `ids_rule` role only supports Snort IDPS.

Procedure

1. Install the `ids_rule` role using the ansible-galaxy command:

   ```
   $ ansible-galaxy install ansible_security.ids_rule
   ```

2. Create a new playbook file titled `add_snort_rule.yml`. Set the following parameters:

   ```
   - name: Add Snort rule
     hosts: snort
   ```

3. Add the `become` flag to ensure that Ansible handles privilege escalation.

   ```
   - name: Add Snort rule
     hosts: snort
     become: true
   ```

4. Specify the name of your IDPS provider by adding the following variables:

   ```
   - name: Add Snort rule
     hosts: snort
     become: true

     vars:
       ids_provider: snort
   ```

5. Add the following tasks and task-specific variables, for example, rules, Snort rules file, and the state of the rule - present or absent, to the playbook:

```
- name: Add Snort rule

  hosts: snort

  become: true


  vars:

    ids_provider: snort


  tasks:

    -  name: Add snort password attack rule

       include_role:

         name: "ansible_security.ids_rule"

       vars:

         ids_rule: 'alert tcp any any -> any any (msg:"Attempted /etc/passwd
Attack"; uricontent:"/etc/passwd"; classtype:attempted-user; sid:99000004;
priority:1; rev:1;)'

         ids_rules_file: '/etc/snort/rules/local.rules'

         ids_rule_state: present
```

Tasks are components that make changes on the target machine. Since you are using a role that defines these tasks, the `include_role` is the only entry you need.

The `ids_rules_file` variable specifies a defined location for the `local.rules` file, while the `ids_rule_state` variable indicates that the rule should be created if it does not already exist.

6. Run the playbook by executing the following command:

   `$ ansible-navigator run add_snort_rule.ym --mode stdout`

   When the playbook runs, all of your tasks will be executed in addition to your newly created rules. Your playbook output confirms your PLAY, TASK, RUNNING HANDLER, and PLAY RECAP.

Result

To verify that your IDPS rules were successfully created, SSH to the Snort server and view the content of the `/etc/snort/rules/local.rules` file.

Security automation use cases

Red Hat Ansible Automation Platform helps organizations automate manual security tasks. You can automate security event response, remediation, routine operations, compliance, and infrastructure hardening.

Red Hat Ansible Automation Platform as part of a Security Operations Center

Automating functions of your *Security Operations Center* (SOC) can help you streamline security operations, response, and remediation activities at scale to reduce the risk and cost of breaches.

Red Hat Ansible Automation Platform can connect your security teams, tools, and processes for more successful automation adoption and use. Learn how automation can help you safeguard your business and respond to growing security threats faster.

Simplify your security operations center provides an overview of the benefits to automating SOC operations, including such use cases as:

- Investigation enrichment
- Threat hunting
- Incident response

Related information
Simplify your security operations center

Automate software patching

Software patching is a fundamental activity of security and IT operations teams everywhere. Keeping patches up to date is critical to remediating software vulnerabilities and meeting compliance requirements, but patching systems manually at scale can be time-consuming and error-prone.

Organizations should put thought into patch management strategies that meet their security, compliance, and business objectives, to prioritize the types of patches to apply (known exploits, critical or important vulnerabilities, optimizations, routine updates, new features, and so on) against the IT assets available across the enterprise. Once policies and priorities have been defined and a patching plan is established, the manual tasks involved in patch management can be automated using Red Hat Ansible Automation Platform to improve patch deployment speed and accuracy, reduce human error, and limit downtime.

Benefits of patch automation

Patch automation reduces manual effort, accelerates patch deployment across all systems, and improves consistency by eliminating human errors in complex updates. In more detail it provides:

- Reduces error-prone manual effort.

- Decreases time to deploy patches at scale.
- Ensures consistency of patches across similar systems. Manual patching of similar systems can result in human error (forgetting one or more, patching using different versions) that impacts consistency.
- Enables orchestration of complex patching scenarios where an update might require taking a system snapshot before applying a patch, or might require additional configuration changes when the patch is applied.

Patching examples

You can modify and thoroughly test the following playbooks that serve as patching examples, to fit the target environment before production use.

These examples use the `ansible.builtin.dnf` module for managing packages on RHEL and other operating systems that use the `dnf` package manager. Modules for patching other Linux operating systems, Microsoft Windows, and many network devices are also available.

Keep everything up to date

For non-production systems like labs, you can automate weekly patching using a job template. This example playbook updates all RPMs to the latest versions on a regular cadence to keep your servers current.

```
- name: Install all available RPM updates
  hosts: target_hosts
  become: true

  tasks:
    - name: Install latest RPMs
      ansible.builtin.dnf:
        name: '*'
        state: latest
```

Install security updates only

For organizations with a policy requiring that all RPMs including security errata be kept up to date, the following playbook might be used in a regularly scheduled job template.

```
- name: Install all security-related RPM updates
  hosts: target_hosts
  become: true

  tasks:
    - name: Install latest RPMs with security errata
      ansible.builtin.dnf:
        name: '*'
        security: true
        state: latest
```

Specify package versions

For production systems, a well-established configuration management practice is to deploy only known, tested combinations of software to ensure that systems are configured correctly and perform as expected.

This includes deploying only known versions of operating system software and patches to ensure that system updates do not introduce problems with production applications.

NOTE:
The following example playbook installs a specific version of the `httpd` RPM and its dependencies when the target host uses the RHEL 9 operating system. This playbook does not take action if the specified versions are already in place or if a different version of RHEL is installed.

```
- name: Install specific RPM versions
  hosts: target_hosts
  gather_facts: true
  become: true

  vars:
    httpd_packages_rhel9:
      - httpd-2.4.53-11.el9_2.5
      - httpd-core-2.4.53-11.el9_2.5
      - httpd-filesystem-2.4.53-11.el9_2.5
      - httpd-tools-2.4.53-11.el9_2.5
      - mod_http2-1.15.19-4.el9_2.4
      - mod_lua-2.4.53-11.el9_2.5

  tasks:
    - name: Install httpd and dependencies
      ansible.builtin.dnf:
        name: '{{ httpd_packages_rhel9 }}'
        state: present
        allow_downgrade: true
    when:
      - ansible_distribution == "RedHat"
      - ansible_distribution_major_version == "9"
```

NOTE:
By setting `allow_downgrade: true`, if a newer version of any defined package is installed on the system, it is downgraded to the specified version instead.

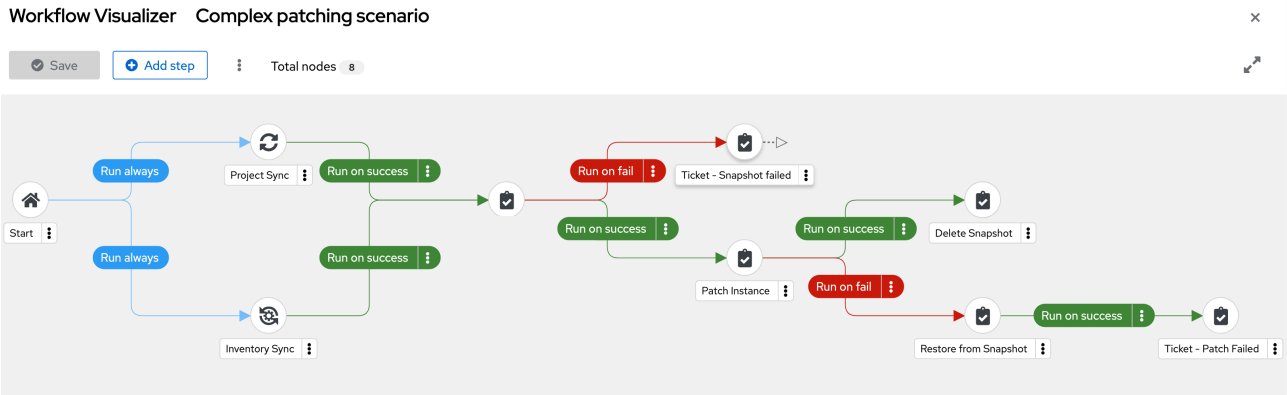
Complex patching scenarios

In Ansible Automation Platform, multiple automation jobs can be chained together into workflows, which can be used to coordinate multiple steps in a complex patching scenario.

The following example complex patching scenario demonstrates taking virtual machine snapshots, patching the virtual machines, and creating tickets when an error is encountered in the workflow.

1. Run a project sync to ensure the latest playbooks are available. In parallel, run an inventory sync to make sure the latest list of target hosts is available.
2. Take a snapshot of each target host.
    a. If the snapshot task fails, submit a ticket with the relevant information.
3. Patch each of the target hosts.
    a. If the patching task fails, restore the snapshot and submit a ticket with the relevant information.
4. Delete each snapshot where the patching task was successful.

The following workflow visualization shows how the components of the example complex patching scenario are executed:



Related information
Workflows in automation controller

Configure automatic security reactions with Event-Driven Ansible

Integrate different security technologies using Event-Driven Ansible to resolve complex configuration challenges. This helps ensure that different products, interfaces, and team workflows align seamlessly across your organization.

Use of Event-Driven Ansible for security

Event-Driven Ansible is a powerful automation framework that enables organizations to respond to real-time events dynamically. It listens for triggers from various sources, evaluates conditions, and executes automated responses using Ansible Playbooks.

In the context of security operations, Event-Driven Ansible enables rapid incident response, threat mitigation, and system hardening by automating reactions to security-related events. Event-driven automation is the process of responding automatically to changing conditions in an IT environment, enabling faster issue resolution and reducing routine, repetitive tasks. Event-Driven Ansible connects sources of events with corresponding actions using rules. Its decision-making capabilities receive an "event" from a monitoring tool and trigger the required action. Ansible Rulebooks define the source of the event and, using "if-this-then-that" instructions, explains the action to take when the event is encountered. Ansible Rulebooks map event conditions to an action, like running a playbook or directly executing a module. Through Ansible, this event-driven automation process is applied to security-related events for event-driven security. An extensive set of monitoring tools is required to promptly identify and address any security risk. When these tools identify an issue or concern, an event-driven automation solution delivers log sources back to a Security Information and Event Management (SIEM) system for human intervention, triage, or resolution. Example automated event-driven threat responses include shutting ports, IPs, or devices. If your event source is watching network routers and discovers that a router is not responding, it recognizes this as an event. Event-Driven Ansible receives this event and matches the event to the condition defined by the rule in the Rulebook, which in this case would be "if an event indicating 'no response' is encountered, then reset the router". Event-Driven Ansible triggers the instructions in the Rulebook and the router is reset, restoring it to normal function. This can happen at any time without human intervention.

Event-Driven Ansible can automate the following common security use cases:

- Enterprise firewalls
- *Intrusion Detection and Prevention Systems* (IDPS)
- *Security Information and Event Management* (SIEM) systems
- *Privileged Access Management* (PAM) tools
- *Endpoint Protection Platform* (EPP)
- Threat detection and response
- Automated incident response
- *Zero Trust Network Access* (ZTNA)
- Compliance and hardening
- Phishing mitigation

The following is an example workflow scenario using Event-Driven Ansible for detection of and response to unauthorized SSH access:

1. **Event Source**: A security monitoring tool detects multiple failed SSH login attempts.
2. **Trigger**: The event is sent to Event-Driven Ansible.
3. **Event-Driven Ansible rulebook evaluation**: If the failed login count exceeds a threshold, execute an Ansible Playbook.
    - Automated response actions:
    - Block the source IP in the firewall.

- Send a notification to security teams.
- Collect logs for forensic analysis.

Case study with F5 example

Configure Event-Driven Ansible rulebooks to automatically mitigate threats detected by monitoring tools like Elasticsearch or Kibana. Triggering F5 security solutions immediately stops potential attacks and helps ensure secure application delivery.

This agentless automation system uses existing transport mechanisms, such as APIs and webhooks, for easier interoperability. F5 content collections for Event-Driven Ansible are developed by F5 and certified by Red Hat to ensure reliable automation and support. Together, F5 and Red Hat help organizations to reduce risk, achieve a faster mean time to resolution, and ultimately free up limited resources to focus on high-value tasks.

Security operations use cases

The following security operations use cases benefit from automation with F5 and Event-Driven Ansible:

Enriched security investigations

Cyberattacks are perpetual threats to organizations, and security tools generate more alerts than understaffed security teams can investigate. Organizations can generate significant savings by enabling their security teams to identify and remediate issues more efficiently through automation. A common first step for security automation is to expedite the investigation phase of potential security incidents by following pre-defined investigation playbooks. When a new security event triggers an Ansible rulebook, automated workflows gather and correlate data from multiple F5 solutions to significantly decrease the amount of time that the security analyst must spend on the investigation, which results in a faster mean time to identify and contain an incident.

Improved threat hunting

Enterprises manage a large number of endpoint devices. This attack surface exposes an organization to multiple threat vectors. Many security teams lack the resources to invest in proactive threat hunting, but by using automation to monitor and correlate threat data and produce actionable insights, security teams can prevent security issues more effectively and quickly detect threat exposure.

Faster response to security incidents

In the context of automated cyberattacks, immediate threat response is vital. Security teams can use automation that executes pre-built, verified workflows for instant response to contain or prevent security incidents, which reduces attacker dwell time and damage. Rules determine which workflows to trigger based on specific events. When the event is detected, automation takes effect to remediate the issue, prevent attacker access, quarantine endpoints, or update security policies to prevent future occurrences. For example, if a malicious user is detected trying to access an application, event monitoring can trigger an Ansible Rulebook that instructs F5 Advanced WAF to block the malicious user while continuing to allow application access by legitimate users.

Case study with F5 example

Configure Event-Driven Ansible rulebooks to automatically mitigate threats detected by monitoring tools like Elasticsearch or Kibana. Triggering F5 security solutions immediately stops potential attacks and helps ensure secure application delivery.

This agentless automation system uses existing transport mechanisms, such as APIs and webhooks, for easier interoperability. F5 content collections for Event-Driven Ansible are developed by F5 and certified by Red Hat to ensure reliable automation and support. Together, F5 and Red Hat help organizations to reduce risk, achieve a faster mean time to resolution, and ultimately free up limited resources to focus on high-value tasks.

Drive responses from logging events

Ansible validated content provides pre-tested, trusted Roles and playbooks for secure, consistent infrastructure management. Use this content out-of-the-box to reduce the time needed for custom development, such as automating Event-Driven Ansible's response to log events.

Use case: AWS CloudTrail

Configure Event-Driven Ansible rulebooks to automatically monitor and secure your AWS CloudTrail logs. Triggering actions to re-enable encryption or restore deleted trails helps ensure your sensitive data remains protected and compliant.

AWS CloudTrail is a service that logs all the API calls made in your AWS account, including API calls made by other AWS services. By default, CloudTrail logs are stored in an S3 bucket in an unencrypted form. To verify that your CloudTrail logs are secure, enable encryption for CloudTrail logs using AWS KMS. Enable encryption for CloudTrail logs by creating a KMS key that is used to

encrypt the S3 bucket where your CloudTrail logs are stored. Then configure CloudTrail to use this key to encrypt the logs.

With encryption enabled, all CloudTrail logs are automatically encrypted when they are written to the S3 bucket. The logs can only be decrypted using the KMS key that you specified. This establishes that your logs are secure and can only be accessed by authorized users and services.

Encrypting AWS CloudTrail logs is important for several reasons:

- Protection of sensitive information: CloudTrail logs contain a wealth of information about the AWS account, including API calls, user identities, and resource information. Encrypting CloudTrail logs helps protect this sensitive information from unauthorized access or tampering.
- Compliance requirements: Many compliance standards, such as HIPAA and PCI DSS, require log encryption to protect sensitive information. Encrypting CloudTrail logs enables compliance with these standards.
- Prevent tampering: CloudTrail's log encryption helps prevent logs from being tampered with. This helps maintain log integrity and an accurate record of all API calls made to your AWS account.
- Secure data: CloudTrail log's encryption provides an additional layer of security for data. In the event that your S3 bucket is compromised, the encrypted logs cannot be accessed without the encryption key.

The Event-Driven Ansible rulebook is comprised of the following components to assist in actions on the log files:

- **Sources**: define which event source will be used
- **Rules**: define which conditionals will be matched from the event source
- **Actions**: trigger events when conditions are met

In the following example, the rulebook implements a ruleset with three rules as follows:

Rule #1: Enable trail encryption

This rule handles the case when trail encryption is disabled. It is triggered when an UpdateTrail operation is performed on the trail and the parameters contained in the UpdateTrail request match these conditions:

`event.CloudTrailEvent.requestParameters.kmsKeyId==""` AND
`event.CloudTrailEvent.requestParameters.name==vars.cloudtrail_name.`

The action that is taken to mitigate this drift will run the

'playbooks/eda/aws_restore_cloudtrail_encryption.yml playbook` This playbook runs the Ansible validated role `cloud.aws_ops.enable_cloudtrail_encryption_with_kms` that re-enables the trail's encryption, restoring the system to its status quo.

Rule#2: Re-create the trail

This rule handles the case when the trail is deleted.

When the following conditions are met:

`event.CloudTrailEvent.eventName=="DeleteTrail"` AND
`event.CloudTrailEvent.requestParameters.name==vars.cloudtrail_name`

The action is running the `playbooks/eda/aws_restore_cloudtrail.yml` playbook. This playbook runs the Ansible validated content `cloud.aws_ops.awsconfig_multiregion_cloudtrail` role first, which re-creates the trail and then the `cloud.aws_ops.enable_cloudtrail_encryption_with_kms` role, to enable the encryption on the newly created trail.

Rule#3: Cancels the deletion of the KMS key and re-enables it

This rule responds to the case of a KMS key being deleted or disabled. This results in the condition

`event.CloudTrailEvent.eventName=="ScheduleKeyDeletion"` OR
`event.CloudTrailEvent.eventName=="DisableKey"`

When someone attempts to delete a KMS key intentionally or accidentally, a `ScheduleKeyDeletion` event is displayed in AWS CloudTrail. The KMS key is not deleted immediately, because deleting a KMS key is destructive and potentially dangerous. AWS KMS requires setting a 7–30 day waiting period. This situation is handled promptly by running playbooks/eda/aws_restore_kms_key.yml playbook, which cancels the deletion of the KMS key. Similarly, when the KMS key is disabled, the playbook reactivates it to restore the original state of the system. The playbook sets the KMS key ARN and uses it to determine whether to cancel the KMS key deletion, to re-enable the KMS key, or both.

Ansible validated content for cloud.aws_ops and Event-Driven Ansible create many opportunities for automated issue resolution and observation of cloud computing environments, helping you to easily automate, mitigate security issues, and maximize your mastery of cloud environments. For more information on using rulebooks, see Validated content for Event-Driven ansible for AWS.

Red Hat product documentation legal notices

Copyright © Red Hat

Except as otherwise noted below, the text of and illustrations in this documentation are licensed by Red Hat under the Creative Commons Attribution–Share Alike 3.0 Unported license. If you distribute this document or an adaptation of it, you must provide the URL for the original version. Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

XFS is a trademark or registered trademark of Hewlett Packard Enterprise Development LP or its subsidiaries in the United States and other countries.

The OpenStack® Word Mark and OpenStack logo are trademarks or registered trademarks of the Linux Foundation, used under license.

All other trademarks are the property of their respective owners.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. link:https://fsf.org/. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If

such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions. "This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code. The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component,

or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions. All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law. No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies. You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions. You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so. A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms. You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms. "Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination. You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies. You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients. Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents. A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom. If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License. Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License. The Free Software Foundation may publish revised and/ or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty. THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES

SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16. If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see link:https://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read link:https://www.gnu.org/licenses/why-not-lgpl.html.

Apache license

Version 2.0, January 2004

http://www.apache.org/licenses/

Terms and Conditions for use, reproduction, and distribution

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, **"control"** means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or **"Your"**) shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, **"submitted"** means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the

Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as **"Not a Contribution."**

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a **"NOTICE"** text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications,

or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Copyright 2026. All rights reserved.

www.redhat.com