Red Hat

Develop

Ansible Automation Platform 2.6

May 12, 2026

Contents

1 Develop

Get started automating with playbooks

An Ansible Playbook is a blueprint for automation tasks, which are actions executed with limited manual effort across an inventory of solutions. Playbooks tell Ansible what to do on which devices.

A playbook automatically executes the same action across a specified inventory type, such as a set of routers, replacing manual repetitive work.

Playbooks are regularly used to automate IT infrastructure, such as operating systems, Kubernetes platforms, networks, security systems, and code repositories such as GitHub.

You can use playbooks to program applications, services, server nodes, and other devices, without the effort of creating everything from scratch. Playbooks, and the conditions, variables, and tasks within them, can be saved, shared, or reused indefinitely. This makes it easier for you to codify operational knowledge and ensure that the same actions are performed consistently.

How do Ansible Playbooks work

Ansible Playbooks are lists of tasks that run automatically for your specified inventory or groups of hosts. One or more Ansible tasks can be combined to make a play, that is, an ordered grouping of tasks mapped to specific hosts.

Tasks are executed in the order in which they are written.

A playbook can include one or more plays.

A playbook is composed of one or more `plays` in an ordered list.

The terms `playbook` and `play` are sports analogies.

Each play executes part of the overall goal of the playbook, running one or more tasks.

Each task calls an Ansible module.

Playbook

A list of plays that define the order in which Ansible performs operations, from top to bottom, to achieve an overall goal.

Play

An ordered list of tasks that maps to managed nodes in an inventory.

Task

A reference to a single module that defines the operations that Ansible performs.

Roles

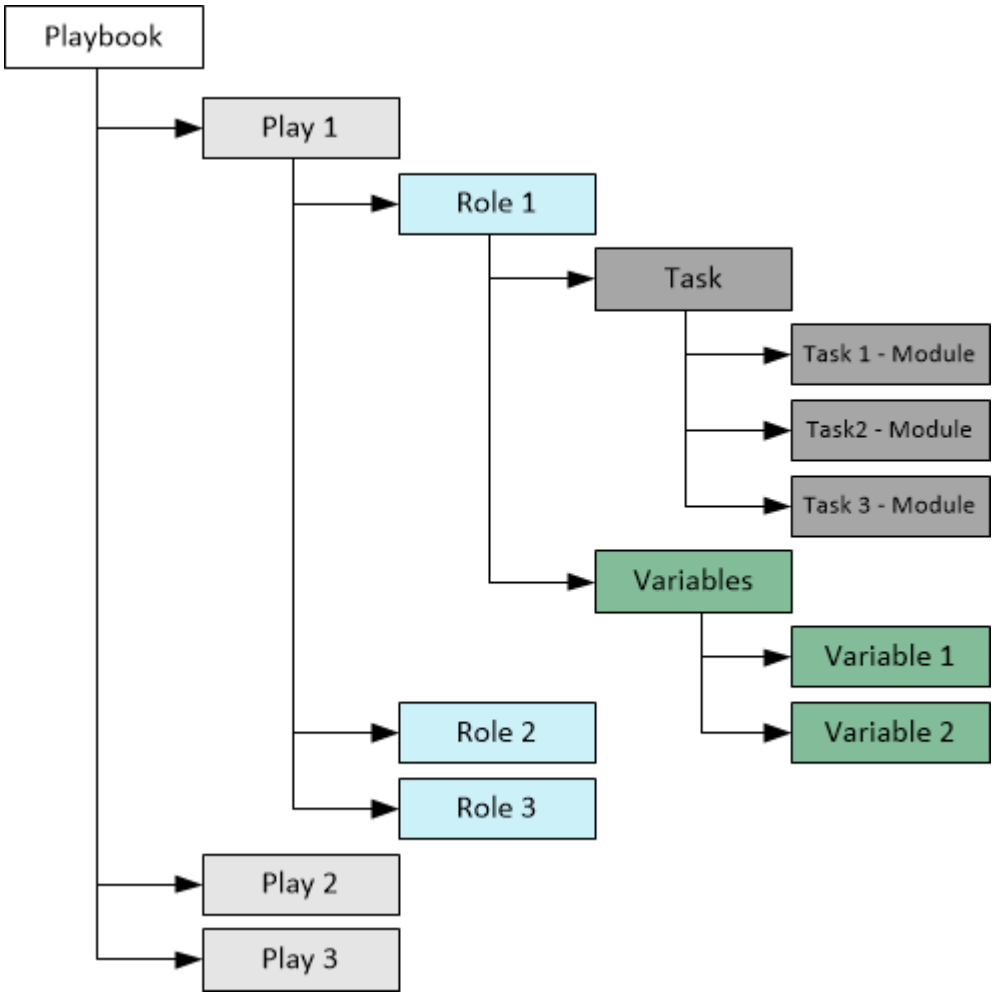
Roles are a way to make code in playbooks reusable by putting the functionality into "libraries" that can then be used in any playbook as needed.

Module

A unit of code or binary that Ansible runs on managed nodes.

Ansible modules are grouped in collections with a *Fully Qualified Collection Name* (FQCN) for each module. Tasks are executed by modules, each of which performs a specific task in a playbook. A module contains metadata that determines when and where a task is executed, and which user executes it. There are thousands of Ansible modules that perform all kinds of IT tasks, such as:

- Cloud management
- User management
- Networking
- Security
- Configuration management
- Communication



How do I use Ansible Playbooks

Ansible Playbooks are files containing a series of instructions that define the required state of a system or application. They are written in YAML, a human-readable data serialization format, and used to automate IT tasks such as configuration management, application deployment, and orchestration.

You can use YAML to create playbooks without having to learn a complicated coding language.

There are two ways of using Ansible Playbooks:

- From the *command line interface* (CLI)
- Using Red Hat Ansible Automation Platform's push-button deployments.

From the CLI

After installing the open source Ansible project or Red Hat Ansible Automation Platform by using

```
$ sudo dnf install ansible
```

in the Red Hat Enterprise Linux CLI, you can use the `ansible-playbook` command to run Ansible Playbooks.

From within the platform

The Red Hat Ansible Automation Platform user interface offers push-button Ansible Playbook deployments that can be used as part of larger jobs or job templates. These deployments come with additional safeguards that are particularly helpful to users who are newer to IT automation, or those without as much experience working in the CLI.

Start automating with Ansible

Get started with Ansible by creating an automation project, building an inventory, and creating a `Hello World` playbook.

Before you begin

- The Ansible package must be installed.
- A text editor must be available to create and edit files.

- Create a project folder on your filesystem.

```
mkdir ansible_quickstart
cd ansible_quickstart
```

Using a single directory structure makes it easier to add to source control, and reuse and share automation content.

Define which hosts to manage in an inventory file

Inventories organize managed nodes in centralized files that provide Ansible with system information and network locations. Using an inventory file, Ansible can manage a large number of hosts with a single command.

Before you begin

- To complete the following steps, you need the IP address or fully qualified domain name (FQDN) of at least one host system. For demonstration purposes, the host could be running locally in a container or a virtual machine.
- You must also ensure that your public SSH key is added to the `authorized_keys` file on each host. Use the following procedure to build an inventory.

Procedure

1. Create a file named `inventory.ini` in the `ansible_quickstart` directory that you created.
2. Add a new `[myhosts]` group to the inventory.ini file and specify the IP address or fully qualified domain name (FQDN) of each host system.

```
[myhosts]
192.0.2.50
192.0.2.51
192.0.2.52
```

3. Verify your inventory, using:

   `ansible-inventory -i inventory.ini --list`

4. Ping the `myhosts` group in your inventory, using:

   'ansible myhosts -m ping -i inventory.ini`

Pass the `-u` option with the Ansible command if the username is different on the control node and the managed node(s).

```
192.0.2.50 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.0.2.51 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.0.2.52 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

You have successfully built an inventory.

Inventories in INI or YAML format

You can create inventories by using either INI files or in YAML. In most cases, such as the preceding example, INI files are straightforward and easy to read for a small number of managed nodes. Creating an inventory in YAML format becomes a sensible option as the number of managed nodes increases.

The following is the same as `inventory.ini` that declares unique names for managed nodes and uses the `ansible_host` field:

```
myhosts:
  hosts:
    my_host_01:
      ansible_host: 192.0.2.50
    my_host_02:
      ansible_host: 192.0.2.51
    my_host_03:
      ansible_host: 192.0.2.52
```

Tips for building inventories

When building inventories for Ansible automation, consider the following best practices to ensure efficient and effective management of your hosts.

- Ensure that group names are meaningful and unique.
- Group names are also case sensitive.
- Do not use spaces, hyphens, or preceding numbers (use floor_19, not 19th_floor) in group names.
- Group hosts in your inventory logically according to their What, Where, and When:
    - What: Group hosts according to the topology, for example: db, web, leaf, spine.
    - Where: Group hosts by geographic location, for example: data center, region, floor, building.
    - When: Group hosts by stage, for example: development, test, staging, production.

Use metagroups

Organize your inventory by using metagroups to group multiple groups together.

Create a metagroup that organizes multiple groups in your inventory with the following syntax:

```
metagroupname:
  children:
```

The following inventory illustrates a basic structure for a data center. This example inventory has a network metagroup that includes all network devices and a data center metagroup that includes the network group and all webservers.

```
leafs:
  hosts:
    leaf01:
      ansible_host: 192.0.2.100
    leaf02:
      ansible_host: 192.0.2.110

spines:
  hosts:
    spine01:
      ansible_host: 192.0.2.120
    spine02:
      ansible_host: 192.0.2.130

network:
  children:
    leafs:
    spines:

webservers:
  hosts:
    webserver01:
      ansible_host: 192.0.2.140
    webserver02:
      ansible_host: 192.0.2.150

datacenter:
  children:
    network:
    webservers:
```

Create inventory file variables to set values for managed nodes

Variables set values for managed nodes, such as the IP address, FQDN, operating system, and SSH user, so you do not need to pass them when running Ansible commands.

Variables can apply to specific hosts.

```
webservers:
  hosts:
    webserver01:
      ansible_host: 192.0.2.140
      http_port: 80
    webserver02:
      ansible_host: 192.0.2.150
      http_port: 443
```

Variables can also apply to all hosts in a group.

```
webservers:
  hosts:
    webserver01:
      ansible_host: 192.0.2.140
      http_port: 80
    webserver02:
      ansible_host: 192.0.2.150
      http_port: 443
  vars:
    ansible_user: my_server_user
```

For more information about inventories and Ansible inventory variables, see About the Installer Inventory file and Inventory file variables.

Create a simple playbook to connect to managed hosts

Learn how to create a playbook that pings your hosts and prints a “Hello world” message.

Procedure

1. Create a file named `playbook.yaml` in your `ansible_quickstart` directory, with the following content:

```
- name: My first play
  hosts: myhosts
  tasks:
   - name: Ping my hosts
     ansible.builtin.ping:


   - name: Print message
     ansible.builtin.debug:
      msg: Hello world
```

2. Run your playbook, using the following command:

   `ansible-playbook -i inventory.ini playbook.yaml`

3. Ansible returns the following output:

```
 PLAY [My first play]
*********************************************************************


TASK [Gathering Facts]
*********************************************************************

ok: [192.0.2.50]

ok: [192.0.2.51]

ok: [192.0.2.52]


TASK [Ping my hosts]
*********************************************************************

ok: [192.0.2.50]

ok: [192.0.2.51]

ok: [192.0.2.52]


TASK [Print message]
*********************************************************************

ok: [192.0.2.50] => {

    "msg": "Hello world"

}

ok: [192.0.2.51] => {

    "msg": "Hello world"

}

ok: [192.0.2.52] => {

    "msg": "Hello world"

}


PLAY RECAP
*********************************************************************
********

192.0.2.50: ok=3    changed=0    unreachable=0    failed=0    skipped=0
 rescued=0    ignored=0

192.0.2.51: ok=3    changed=0    unreachable=0    failed=0    skipped=0
 rescued=0    ignored=0

192.0.2.52: ok=3    changed=0    unreachable=0    failed=0    skipped=0
 rescued=0    ignored=0
```

In this output you can see:

- The names that you give the play and each task. Always use descriptive names that make it easy to verify and troubleshoot playbooks.

- The Gather Facts task runs implicitly. By default Ansible gathers information about your inventory that it can use in the playbook.
- The status of each task. Each task has a status of `ok` which means it ran successfully.
- The play recap that summarizes results of all tasks in the playbook per host. In this example, there are three tasks so `ok=3` indicates that each task ran successfully.

Gather and display network device info with a playbook

Learn how to create and run a simple Ansible Playbook that connects to a network device, gathers facts, and displays them.

To confirm your credentials, you can connect to a network device manually and retrieve its configuration. Replace the sample user and device name with your real credentials.

For example, for a VyOS router:

```
ssh my_vyos_user@vyos.example.net
show config
exit
```

Run a network Ansible command

Instead of manually connecting and running a command on the network device, you can retrieve its configuration with a single Ansible command.

```
ansible all -i vyos.example.net, -c ansible.netcommon.network_cli -u \
my_vyos_user -k -m vyos.vyos.vyos_facts -e \
ansible_network_os=vyos.vyos.vyos
```

The flags in this command set seven values:

- the host group(s) to which the command should apply (in this case, `all`)
- the inventory (`-i`, the device or devices to target - without the trailing comma `-i` points to an inventory file)
- the connection method (`-c`, the method for connecting and executing ansible)

- the user (`-u` , the username for the SSH connection)
- the SSH connection method (- `k` , prompt for the password)
- the module (`-m` , the Ansible module to run, using the fully qualified collection name (FQCN))
- an extra variable (`-e` , in this case, setting the network operating system value)

NOTE:
If you use `ssh-agent` with SSH keys, Ansible loads them automatically. You can omit the `-k` flag.

If you are running Ansible in a virtual environment, you must also add the variable `ansible_python_interpreter=/path/to/venv/bin/python` .

Run a network Ansible Playbook

If you want to run a particular command every day, you can save it in a playbook and run it with `ansible-playbook` instead of ansible.

Before you begin

Download `first_playbook.yml` from here.

The playbook looks like this:

```
---


- name: Network Getting Started First Playbook
  connection: ansible.netcommon.network_cli
  gather_facts: false                                    ①
  hosts: all
  tasks:


    - name: Get config for VyOS devices
      vyos.vyos.vyos_facts:
        gather_subset: all


    - name: Display the config
      debug:
        msg: "The hostname is {{ ansible_net_hostname }} and the OS is
{{ ansible_net_version }}"
```

Label	Description
`gather_facts`	Ansible's native fact gathering (`ansible.builtin.setup`) is disabled here because the playbook relies on the facts provided by a platform-specific module (`vyos.vyos.vyos_facts`) in this networking collection.

The playbook sets three of the seven values from the command line above:

- the group (`hosts: all`)
- the connection method (`connection: ansible.netcommon.network_cli`) and
- the module (in each task).

With those values set in the playbook, you can omit them on the command line. The playbook also adds a second task to show the configuration output.

When facts are gathered from a system, either through a collection-specific fact module such as `vyos.vyos.vyos_facts` or `ansible.builtin.setup`, the gathered data is held in memory for use by future tasks instead of being written to the console.

When a module runs in a playbook, the output is held in memory for use by future tasks instead of written to the console. With most other modules you must explicitly register a variable to store and reuse the output of a module or task.

The following debug task lets you see the results in your shell.

The playbook can store many of the parameters you provided with flags at the command line, leaving less to type at the command line. You need two files for this, a playbook and an inventory file.

Procedure

1. Run the playbook with the following command.

   `ansible-playbook -i vyos.example.net, -u ansible -k -e ansible_network_os=vyos.vyos.vyos first_playbook.yml`

   The playbook contains one play with two tasks, and generates output similar to the following:

```
 $ ansible-playbook -i vyos.example.net, -u ansible -k -e
ansible_network_os=vyos.vyos.vyos first_playbook.yml


PLAY [Network Getting Started First Playbook]

**********************************************************************************
*****************************************

TASK [Get config for VyOS devices]

**********************************************************************************
*****************************************

ok: [vyos.example.net]


TASK [Display the config]

**********************************************************************************
*****************************************

ok: [vyos.example.net] => {

    "msg": "The hostname is vyos and the OS is VyOS 1.1.8"

}
```

2. Now that you can retrieve the device configuration, you can try updating it with Ansible.
3. Download `first_playbook_ext.yml` from here, which is an extended version of the first playbook:

   The playbook looks like this:

```
---


- name: Network Getting Started First Playbook Extended
  connection: ansible.netcommon.network_cli
  gather_facts: false
  hosts: all
  tasks:


    - name: Get config for VyOS devices
      vyos.vyos.vyos_facts:
        gather_subset: all


    - name: Display the config
      debug:
        msg: "The hostname is {{ ansible_net_hostname }} and the OS is
{{ ansible_net_version }}"


    - name: Update the hostname
      vyos.vyos.vyos_config:
        backup: yes
        lines:
          - set system host-name vyos-changed


    - name: Get changed config for VyOS devices
      vyos.vyos.vyos_facts:
        gather_subset: all


    - name: Display the changed config
      debug:
        msg: "The new hostname is {{ ansible_net_hostname }} and the OS is
{{ ansible_net_version }}"
```

4. The extended first playbook has five tasks in a single play.
5. Run the playbook with the following command.

```
$ ansible-playbook -i vyos.example.net, -u ansible -k -e
ansible_network_os=vyos.vyos.vyos first_playbook_ext.yml
```

6. The output shows you the change Ansible made to the configuration:

```
 $ ansible-playbook -i vyos.example.net, -u ansible -k -e
ansible_network_os=vyos.vyos.vyos first_playbook_ext.yml


PLAY [Network Getting Started First Playbook Extended]

*********************************************************************************
**************************************************

TASK [Get config for VyOS devices]

*********************************************************************************
**************************************************

ok: [vyos.example.net]


TASK [Display the config]

*********************************************************************************
**************************************************

ok: [vyos.example.net] => {

    "msg": "The hostname is vyos and the OS is VyOS 1.1.8"

}


TASK [Update the hostname]

*********************************************************************************
**************************************************

changed: [vyos.example.net]


TASK [Get changed config for VyOS devices]

*********************************************************************************
**************************************************

ok: [vyos.example.net]


TASK [Display the changed config]

*********************************************************************************
**************************************************

ok: [vyos.example.net] => {

    "msg": "The new hostname is vyos-changed and the OS is VyOS 1.1.8"

}


PLAY RECAP

*********************************************************************************
**************************************************

vyos.example.net           : ok=5    changed=1    unreachable=0    failed=0
```

Gather facts from network devices

The `gather_facts` keyword supports gathering network device facts in standardized key/value pairs. You can feed these network facts into further tasks to manage the network device.

You can also use the `gather_network_resources` parameter with the network `*_facts` modules (such as `arista.eos.eos_facts`) to return a subset of the device configuration, as the following example shows:

```
- hosts: arista
  gather_facts: True
  gather_subset: interfaces
  module_defaults:
    arista.eos.eos_facts:
      gather_network_resources: interfaces
```

The playbook returns the following interface facts:

```
"network_resources": {
      "interfaces": [
          {
              "description": "test-interface",
              "enabled": true,
              "mtu": "512",
              "name": "Ethernet1"
          },
          {
              "enabled": true,
              "mtu": "3000",
              "name": "Ethernet2"
          },
          {
              "enabled": true,
              "name": "Ethernet3"
          },
          {
              "enabled": true,
              "name": "Ethernet4"
          },
          {
              "enabled": true,
              "name": "Ethernet5"
          },
          {
              "enabled": true,
              "name": "Ethernet6"
          },
      ]
  }
```

NOTE:
`gather_network_resources` renders configuration data as facts for all supported resources (`interfaces/bgp/ospf/etc`), whereas `gather_subset` is primarily used to fetch operational data.

Example: automate software updates

Ansible can communicate with many different device classifications, from cloud-based REST APIs, to Linux and Windows systems, networking hardware, and much more. The following is a sample of two Ansible modules automatically updating two types of servers.

Playbook execution

A playbook runs in order from top to bottom. Within each play, tasks also run in order from top to bottom. Playbooks with multiple 'plays' can orchestrate multi-machine deployments, running one play on your webservers, then another play on your database servers, and so on.

At a minimum, each play defines two things:

- the managed nodes to target, using a pattern
- at least one task to run

NOTE:
Use the fully-qualified collection name in your playbooks to ensure the correct module is selected, because multiple collections can contain modules with the same name (for example, `user`).

In this example, the first play targets the web servers; the second play targets the database servers.

```
---
- name: Update web servers
  hosts: webservers
  become: true

  tasks:
    - name: Ensure apache is at the latest version
      ansible.builtin.yum:
        name: httpd
        state: latest
    - name: Write the apache config file
      ansible.builtin.template:
        src: /srv/httpd.j2
        dest: /etc/httpd.conf
        mode: "0644"

- name: Update db servers
  hosts: databases
  become: true

  tasks:
    - name: Ensure postgresql is at the latest version
      ansible.builtin.yum:
        name: postgresql
        state: latest
    - name: Ensure that postgresql is started
      ansible.builtin.service:
        name: postgresql
        state: started
```

The playbook contains two plays:

- The first checks if the web server software is up to date and runs the update if necessary.
- The second checks if database server software is up to date and runs the update if necessary.

Your playbook can include more than just a hosts line and tasks.

For example, this example playbook sets a remote_user for each play. This is the user account for the SSH connection. You can add other playbook keywords at the playbook, play, or task level to influence how Ansible behaves. Playbook keywords can control the connection plugin, whether to use privilege escalation, how to handle errors, and more.

To support a variety of environments, Ansible enables you to set many of these parameters as command-line flags, in your Ansible configuration, or in your inventory. Learning the precedence rules for these sources of data can help you as you expand your Ansible ecosystem

Create, test, and deploy automation content with ansible-dev-tools

Ansible development tools (`ansible-dev-tools`) is a suite of tools provided with Ansible Automation Platform to help automation creators to create, test, and deploy playbook projects, execution environments, and collections.

The Ansible VS Code extension by Red Hat integrates most of the Ansible development tools: you can use these tools from the VS Code user interface.

Use Ansible development tools during local development of playbooks, local testing, and in a CI pipeline (linting and testing).

Use Ansible development tools to create a playbook project that contains playbooks and roles that you can reuse within the project. It also describes how to test the playbooks and deploy the project on your Ansible Automation Platform instance so that you can use the playbooks in automation jobs.

Ansible development tools components and workflow

You can operate some Ansible development tools from the VS Code UI when you have installed the Ansible extension, and the remainder from the command line. VS Code is a free open-source code editor available on Linux, Mac, and Windows.

Ansible VS Code extension

This is not packaged with the Ansible Automation Platform RPM package, but it is an integral part of the automation creation workflow. From the VS Code UI, you can use the Ansible development tools for the following tasks:

- Scaffold directories for a playbook project or a collection.
- Write playbooks with the help of syntax highlighting and auto-completion.
- Debug your playbooks with a linter.

- Execute playbooks with Ansible Core using `ansible-playbook`.
- Execute playbooks in an execution environment with `ansible-navigator`.

From the VS Code extension, you can also connect to Red Hat Ansible Lightspeed with IBM watsonx Code Assistant.

Command-line Ansible development tools

You can perform the following tasks with Ansible development tools from the command line, including the terminal in VS Code:

- Create an execution environment.
- Test your playbooks, roles, modules, plugins and collections.

Development workflow

The development workflow for automation content includes three stages: Create, Test, and Deploy. Follow these stages to successfully manage and publish your automation.

Workflow

The workflow for developing automation content involves three key stages: Create, Test, and Deploy.

Create

In the create stage, you create a new playbook project locally, using VS Code. The following is a typical workflow:

1. Install and run the Ansible extension in VS Code.
2. Scaffold a playbook project from VS Code.
3. Add playbook files to your project and edit them in VS Code.

Test

1. Debug your playbook with the help of `ansible-lint`.
2. Select or create an automation execution environment so that your local environment replicates the environment on Ansible Automation Platform.

3. Run your playbooks from VS Code, using `ansible-playbook` or using `ansible-navigator` with an execution environment.
4. Test your playbooks by running them on an execution environment that replicates your production environment.

Deploy

1. Push your playbooks project to a source control repository.
2. Set up credentials on Ansible Automation Platform to pull from your source control repository and create a project for your playbook repository.
3. If you have created an execution environment, push it to private automation hub.
4. Create a job template on Ansible Automation Platform that runs a playbook from your project, and specify the execution environment that you want to use.

Auto-generate the structure and files for your automation project

Use the Ansible Automation Platform VS Code extension to scaffold a new Ansible playbook project. This process creates the necessary directory structure and configuration files, preparing your environment for playbook development.

Scaffold a playbook project

The following steps describe the process for scaffolding a new playbook project with the Ansible VS Code extension.

Before you begin

- You have installed Ansible development tools.
- You have installed and opened the Ansible VS Code extension.
- You have identified a directory where you want to save the project.

Procedure

1. Open VS Code.
2. Click the Ansible icon in the VS Code activity bar to open the Ansible extension.

3. Select **Get started** in the **Ansible content creator** section.

   The **Ansible content creator** tab opens.

4. In the **Create** section, click **Ansible playbook project**.

   The **Create Ansible project** tab opens.

5. In the form in the **Create Ansible project** tab, enter the following:

   - **Destination directory**: Enter the path to the directory where you want to scaffold your new playbook project.

     NOTE:
     If you enter an existing directory name, the scaffolding process overwrites the contents of that directory. The scaffold process only allows you to use an existing directory if you enable the `Force` option.

     - If you are using the containerized version of Ansible Dev tools, the destination directory path is relative to the container, not a path in your local system. To discover the current directory name in the container, run the `pwd` command in a terminal in VS Code. If the current directory in the container is `workspaces`, enter `workspaces/<destination_directory_name>`.
     - If you are using a locally installed version of Ansible Dev tools, enter the full path to the directory, for example `/user/<username>/projects/<destination_directory_name>`.

   - **SCM organization and SCM project**: Enter a name for the directory and subdirectory where you can store roles that you create for your playbooks.

6. Enter a name for the directory where you want to scaffold your new playbook project.

Result

After the project directory has been created, the following message appears in the **Logs** pane of the **Create Ansible Project** tab. In this example, the destination directory name is `destination_directory_name`.

```
------------------ ansible-creator logs ------------------

    Note: ansible project created at /Users/username/test_project
```

The following directories and files are created in your project directory:

```
$ tree -a -L 5 .
├── .devcontainer
│   ├── devcontainer.json
│   ├── docker
│   │   └── devcontainer.json
│   └── podman
│       └── devcontainer.json
├── .gitignore
├── README.md
├── ansible-navigator.yml
├── ansible.cfg
├── collections
│   ├── ansible_collections
│   │   └── scm_organization_name
│   │       └── scm_project_name
│   └── requirements.yml
├── devfile.yaml
├── inventory
│   ├── group_vars
│   │   ├── all.yml
│   │   └── web_servers.yml
│   ├── host_vars
│   │   ├── server1.yml
│   │   ├── server2.yml
│   │   ├── server3.yml
│   │   ├── switch1.yml
│   │   └── switch2.yml
│   └── hosts.yml
├── linux_playbook.yml
├── network_playbook.yml
└── site.yml
```

Write your first automation task using the VS Code extension

Learn how to write, inspect, debug, and run Ansible playbooks directly within VS Code using Ansible development tools.

Set up the Ansible configuration file

When you scaffolded your playbook project, an Ansible configuration file, `ansible.cfg`, was added to the root directory of your project.

- If you have configured a default Ansible configuration file in `/etc/ansible/ansible.cfg`, copy any settings that you want to reuse in your project from your default Ansible configuration file to the `ansible.cfg` file in your project's root directory.

  To learn more about the Ansible configuration file, see Reviewing your Ansible configuration with automation content navigator.

Write your first playbook

Create your first Ansible playbook within VS Code using the Ansible extension. The tools available help ensure that your syntax is correct and ready to run.

Before you begin

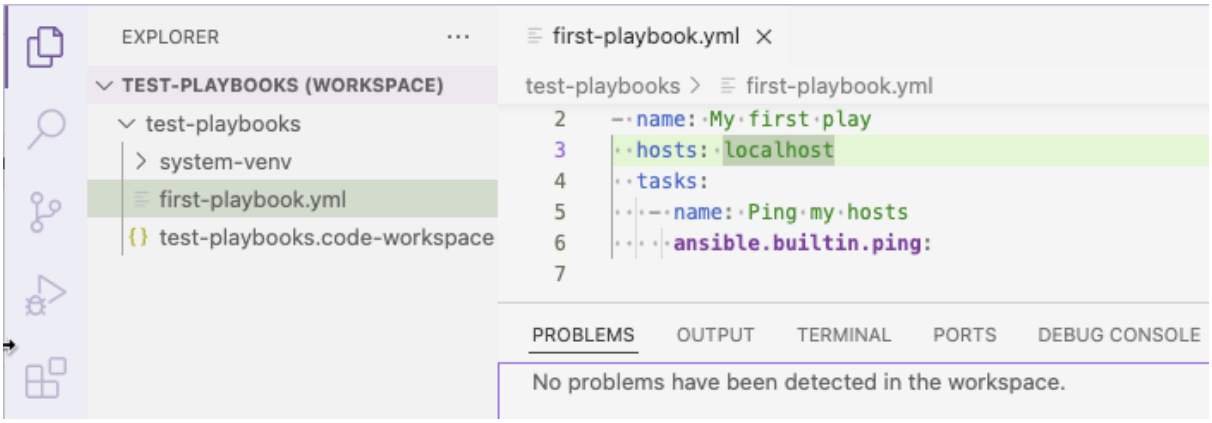
- You have installed and opened the Ansible VS Code extension.
- You have opened a terminal in VS Code.
- You have installed `ansible-devtools`.

Procedure

1. Create a new .yml file in VS Code for your playbook, for example `example_playbook.yml`. Put it in the same directory level as the example `site.yml` file.
2. Add the following example code into the playbook file and save the file. The playbook consists of a single play that executes a `ping` to your local machine.

```
---
- name: My first play
  hosts: localhost
  tasks:
    - name: Ping my hosts
      ansible.builtin.ping:
```

`Ansible-lint` runs in the background and displays errors in the **Problems** tab of the terminal. There are no errors in this playbook:



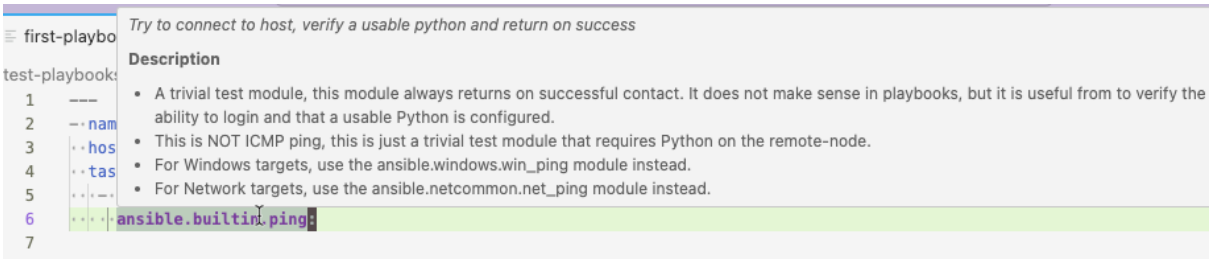
3. If you want to add new content to the playbook, use the following rules:
    - Every playbook file must finish with a blank line.
    - Trailing spaces at the end of lines are not allowed.
    - Every playbook and every play require an identifier (name).
4. Save your playbook file.

Inspect your playbook

The Ansible VS Code extension provides inline help, syntax highlighting, and assists you with indentation in `.yml` files.

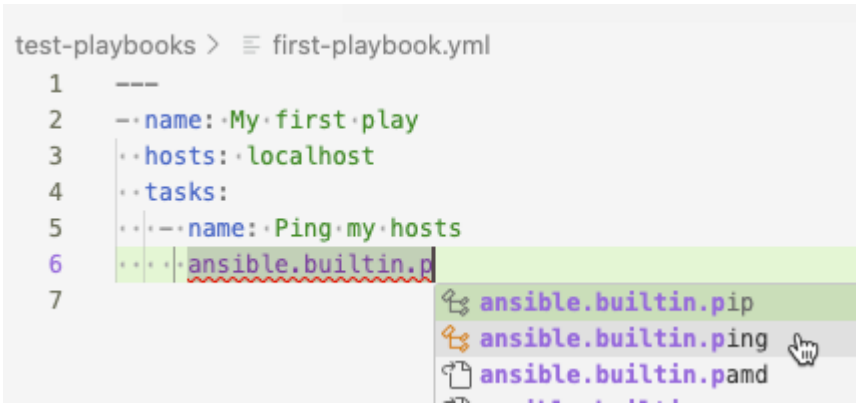
Procedure

1. Open a playbook in VS Code.
2. Hover your mouse over a keyword or a module name: the Ansible extension provides documentation:



3. If you begin to type the name of a module, for example `ansible.builtin.ping`, the extension provides a list of suggestions.

   Select one of the suggestions to autocomplete the line.



Debug your playbook

Learn how to use VS Code to identify and understand error messages in playbooks.

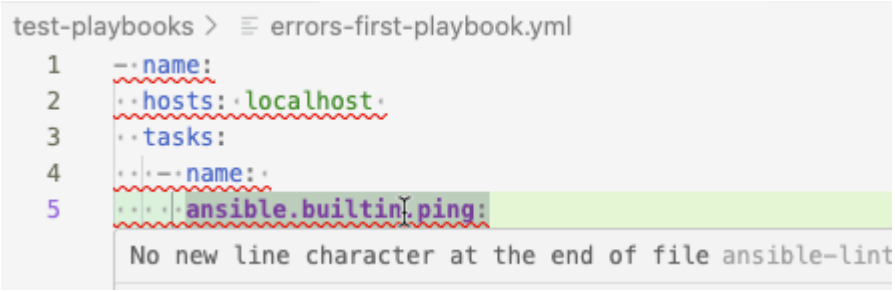
Procedure

1. The following playbook contains multiple errors. Copy and paste it into a new file in VS Code.

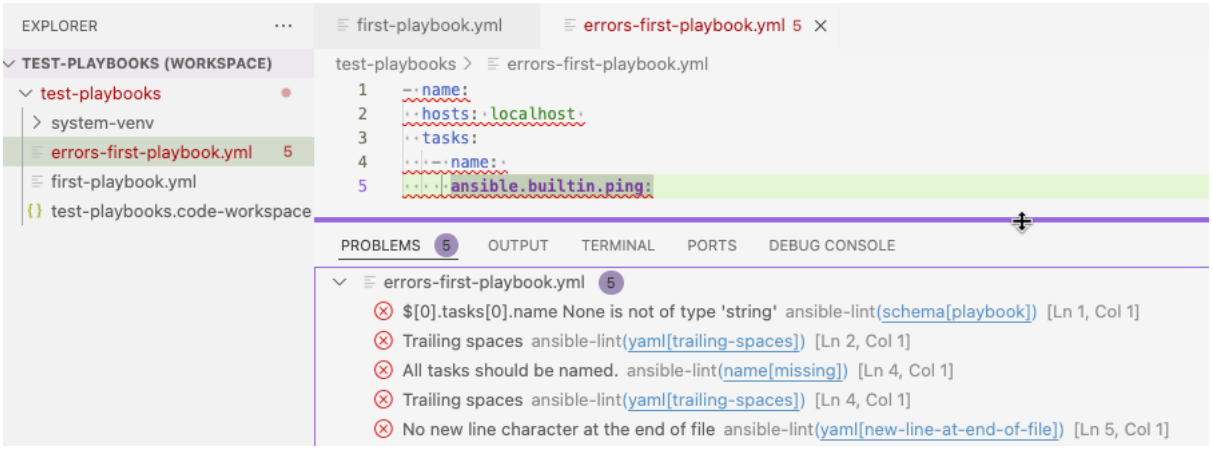
```
- name:
  hosts: localhost
  tasks:
   - name:
     ansible.builtin.ping:
```

   The errors are indicated with a wavy underline in VS Code.

2. Hover your mouse over an error to view the details:



3. Playbook files that contain errors are indicated with a number in the **Explorer** pane.
4. Select the **Problems** tab of the VS Code terminal to view a list of the errors.



`$[0].tasks[0].name None is not of type 'string'` indicates that the playbook does not have a label.

Run your playbook automation to test behavior

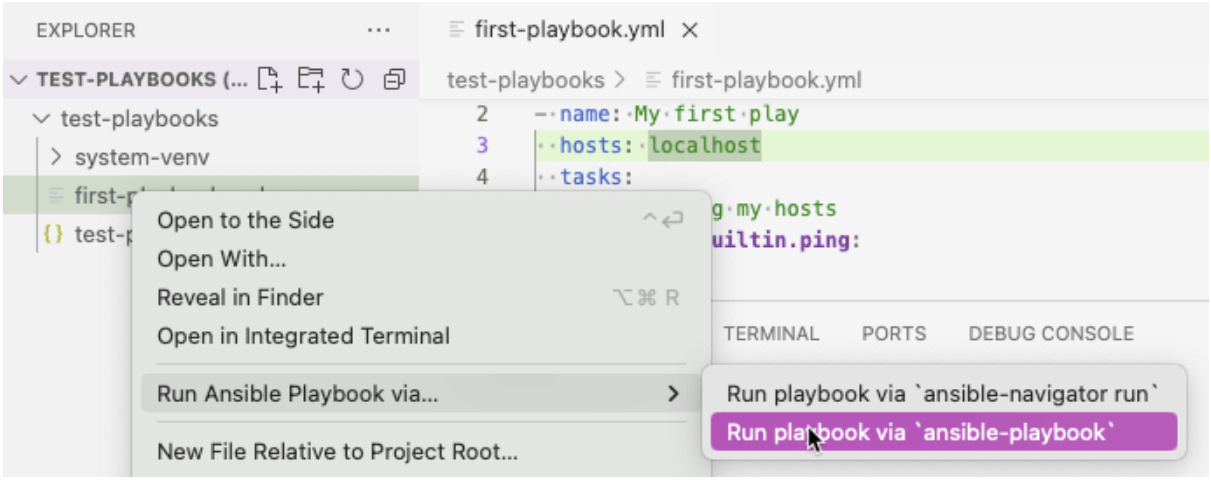
Run your playbook using two methods provided by the Ansible VS Code extension.

- The Ansible VS Code extension provides two options to run your playbook:
  - `ansible-playbook` runs the playbook on your local machine using Ansible Core.
  - `ansible-navigator` runs the playbook in an execution environment in the same manner that Ansible Automation Platform runs an automation job. You specify the base image for the execution environment in the Ansible extension settings.

Run your playbook with `ansible-playbook`

You can run your Ansible playbook locally by using the `ansible-playbook` command directly within the VS Code extension.

- To run a playbook, right-click the playbook name in the **Explorer** pane, then select **Run Ansible Playbook via > Run playbook via ansible-playbook**.



The output is displayed in the **Terminal** tab of the VS Code terminal. The `ok=2` and `failed=0` messages indicate that the playbook ran successfully.

Run your playbook with `ansible-navigator`

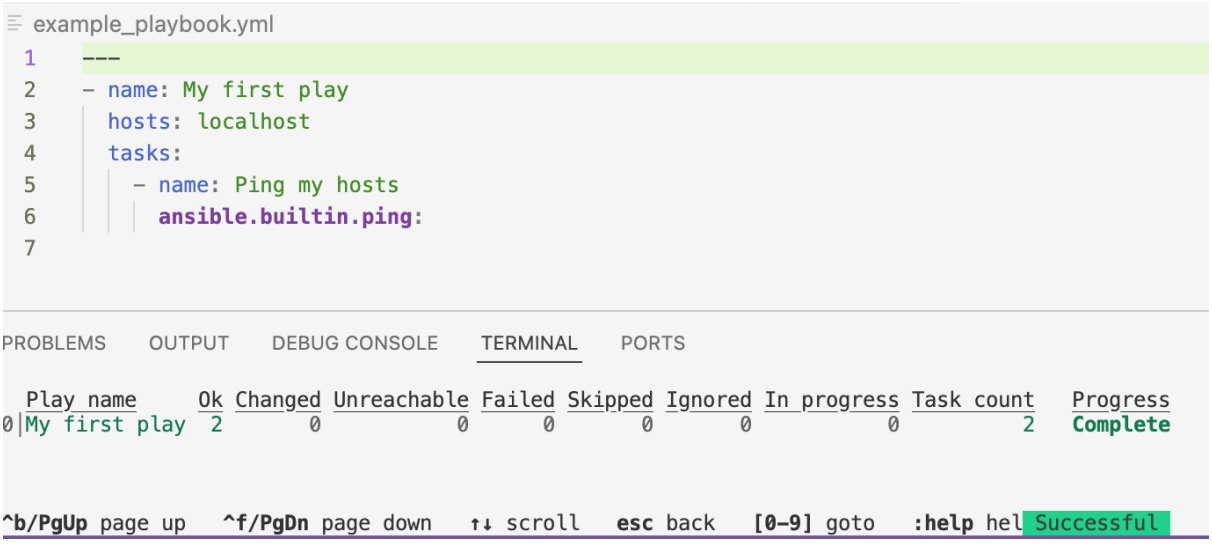
You can run an Ansible playbook through `ansible-navigator` by right-clicking the playbook name in the Explorer pane. This procedure explains how to view the playbook's output and navigate the results for each play and task within the terminal.

Before you begin

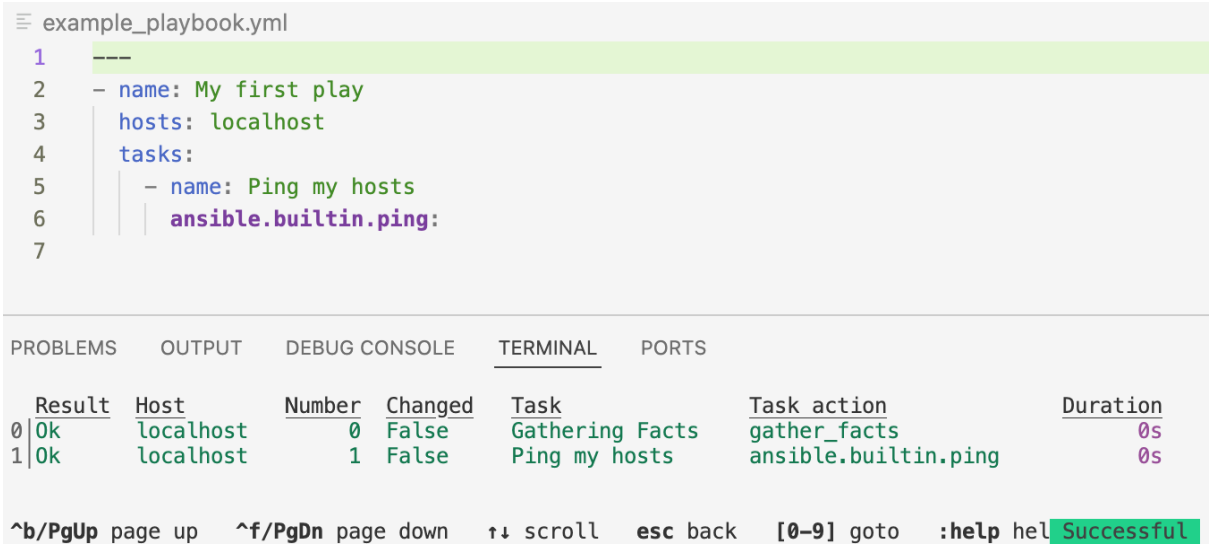
- You enabled the use of an execution environment in the Ansible extension settings.
- You entered the path or URL for the execution environment image in the Ansible extension settings.

Procedure

1. To run a playbook, right-click the playbook name in the Explorer pane, then select **Run Ansible Playbook via > Run playbook via ansible-navigator run**.
2. View the output in the **Terminal** tab of the VS Code terminal. The **Successful** status indicates that the playbook ran successfully.



3. Enter the number next to a play to step into the play results. The example playbook only contains one play. Enter `0` to view the status of the tasks executed in the play.



Type the number next to a task to review the task results.

Related information

Executing a playbook from automation content navigator

Publish and run your playbook automation

The following procedures describe how to deploy your new playbooks in your instance of Ansible Automation Platform so that you can use them to run automation jobs.

Save your project in SCM

Save your playbook project as a repository in your source control management system, for example GitHub.

Procedure

1. Initialize your project directory as a git repository.
2. Push your project up to a source control system such as GitHub.

Run your playbook in Ansible Automation Platform

To run your playbook in Ansible Automation Platform, you must create a project in automation controller for the repository where you stored your playbook project. You can then create a job template for each playbook from the project.

Procedure

1. In a browser, log in to automation controller.
2. Configure a Source Control credential type for your source control system if necessary. See the Creating new credentials section of *Using automation execution* for more details.
3. In automation controller, create a project for the GitHub repository where you stored your playbook project. Refer to the Projects chapter of *Using automation execution*.
4. Create a job template that uses a playbook from the project that you created. Refer to the Job Templates chapter of *Using automation execution*.
5. Run your playbook from automation controller by launching the job template. Refer to the Launching a job template section of *Using automation execution*.

Package and distribute automation content with collections

Collections are a distribution format for Ansible content that can include playbooks, roles, modules, and plugins. Red Hat provides Ansible Content Collections on Ansible automation hub that contain both Red Hat Ansible Certified Content and Ansible validated content.

If you have installed private automation hub, you can create collections for your organization and push them to private automation hub so that you can use them in job templates in Ansible Automation Platform. You can use collections to package and distribute plug-ins. These plug-ins are written in Python.

You can also create collections to package and distribute Ansible roles, which are expressed in YAML. You can also include playbooks and custom plug-ins that are required for these roles in the collection. Typically, collections of roles are distributed for use within your organization.

Understand collections for distributing roles

An Ansible role is a self-contained unit of Ansible automation content that groups related tasks and associated variables, files, handlers, and other assets in a defined directory structure.

You can run Ansible roles in one or more plays, and reuse them across playbooks. Invoking roles instead of tasks simplifies playbooks. You can migrate existing standalone roles into collections, and push them to private automation hub to share them with other users in your organization. Distributing roles in this way is a typical way to use collections.

With Ansible collections, you can store and distribute multiple roles in a single unit of reusable automation. Inside a collection, you can share custom plug-ins across all roles in the collection instead of duplicating them in each role.

You must move roles into collections if you want to use them in Ansible Automation Platform.

You can add existing standalone roles to a collection, or add new roles to it. Push the collection to source control and configure credentials for the repository in Ansible Automation Platform.

Plan your collection

Organize smaller bundles of curated automation into separate collections for specific functions, rather than creating one big general collection for all of your roles.

For example, you could store roles that manage the networking for an internal system called `myapp` in a `company_namespace.myapp_network` collection, and store roles that manage and deploy networking in AWS in a collection called `company_namespace.aws_net`.

Prerequisites

You must install and configure the essential software and accounts listed in this section before you proceed.

- VS Code and the Ansible extension
- Microsoft Dev Containers extension in VS Code
- Ansible development tools
- A containerization platform, for example Podman, Podman Desktop, Docker, or Docker Desktop
- A Red Hat account and log in access to the Red Hat container registry at registry.redhat.io. For information about logging in to registry.redhat.io, see Authenticating with the Red Hat container registry.

Related information
Authenticating with the Red Hat container registry

Generate the collection structure for roles

Scaffold a collection using the Ansible extension in VS Code. This process creates the necessary directory structure for packaging and distributing your roles and plug-ins.

Procedure

1. Open VS Code.
2. Navigate to the directory where you want to create your roles collection.
3. Click the Ansible icon in the VS Code activity bar to open the Ansible extension.
4. Select **Get started** in the **Ansible content creator** section.

   The **Ansible content creator** tab opens.
5. In the **Create** section, click **Ansible collection project**.

   The **Create new Ansible project** tab opens.
6. In the form in the **Create Ansible project** tab, enter the following:

   - **Namespace**: Enter a name for your namespace, for example `company_namespace`.
   - **Collection**: Enter a name for your collection, for example, `myapp_network`.
   - **Init path**: Enter the path to the directory where you want to scaffold your new collection.

If you enter an existing directory name, the scaffolding process overwrites the contents of that directory. The scaffold process only allows you to use an existing directory if you enable the Force option.

- If you are using the containerized version of Ansible development tools, the destination directory path is relative to the container, not a path in your local system. To discover the current directory name in the container, run the pwd command in a terminal in VS Code. If the current directory in the container is `workspaces`, enter `workspaces/<current_project>/collections`.
- If you are using a locally installed version of Ansible Dev tools, enter the full path to the directory, for example `/user/<username>/path/to/<collection_directory>`.

7. Click **Create**.

Result

The following message appears in the **Logs** pane of the **Create Ansible collection** tab.

```
---------------------- ansible-creator logs ----------------------


    Note: collection company_namespace.myapp_network created at /path/to/
collections/directory
```

The following directories and files are created in your `collections/` directory:

```
├── .devcontainer
├── .github
├── .gitignore
├── .isort.cfg
├── .pre-commit-config.yaml
├── .prettierignore
├── .vscode
├── CHANGELOG.rst
├── CODE_OF_CONDUCT.md
├── CONTRIBUTING
├── LICENSE
├── MAINTAINERS
├── README.md
├── changelogs
├── devfile.yaml
├── docs
├── extensions
├── galaxy.yml
├── meta
├── plugins
├── pyproject.toml
├── requirements.txt
├── roles
├── test-requirements.txt
├── tests
└── tox-ansible.ini
```

Migrate existing roles to your collection

Migrate existing standalone roles into the `roles/` directory of your new collection. You must rename roles to remove hyphens and update playbooks to use the fully qualified collection name.

```
my_role
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

An Ansible role has a defined directory structure with seven main standard directories. Each role must must include at least one of these directories. You can omit any directories the role does not use. Each directory contains a `main.yml` file.

Procedure

1. If necessary, rename the directory that contains your role to reflect its content, for example, `acl_config` or `tacacs`.

   Roles in collections cannot have hyphens in their names. Use the underscore character (`_`) instead.

2. Copy the roles directories from your standalone role into the `roles/` directory in your collection.

   For example, in a collection called `myapp_network`, add your roles to the `myapp_network/roles/` directory.

3. Copy any plug-ins from your standalone roles into the `plugins directory/` for your new collection. The collection directory structure resembles the following.

```
company_namespace
└── myapp_network
    ├── ...
    ├── galaxy.yml
    ├── docs
    ├── extensions
    ├── meta
    ├── plugins
    ├── roles
    │   ├── acl_config
    │   │   ├── README.md
    │   │   ├── defaults
    │   │   ├── files
    │   │   ├── handlers
    │   │   ├── meta
    │   │   ├── tasks
    │   │   ├── templates
    │   │   ├── tests
    │   │   └── vars
    │   └── tacacs
    │       ├── README.md
    │       ├── default
    │       ├── files
    │       ├── handlers
    │       ├── meta
    │       ├── tasks
    │       ├── templates
    │       ├── tests
    │       └── vars
    │   ├── run
    ├── ...
    ├── tests
    └── vars
```

The `run` role is a default role directory that is created when you scaffold the collection.

4. Update your playbooks to use the fully qualified collection name (FQDN) for your new roles in your collection.

> **NOTE:**
> Not every standalone role will seamlessly integrate into your collection without modification of the code. For example, if a third-party standalone role from Galaxy that contains a plug-in uses the `module_utils/` directory, then the plug-in itself has import statements.

Create a new role in your collection

Create a new Ansible role within your collection by copying the default role directory.

Procedure

1. To create a new role, copy the default `run` role directory that was scaffolded when you created the collection.
2. Define the tasks that you want your role to perform in the `tasks/main.yml` file. If you are creating a role to reuse tasks in an existing playbook, copy the content in the tasks block of your playbook YAML file. Remove the whitespace to the left of the tasks. Use `ansible-lint` in VS Code to check your YAML code.
3. If your role depends on another role, add the dependency in the `meta/main.yml` file.

Add documentation for your roles collection

It is important to provide documentation for your roles and roles collection, so that other users understand what your roles do and how to use them.

Procedure

1. To add documentation for a role, navigate to the role directory.
2. Open the `README.md` file in an editor. This file was added in the role directory when you scaffolded your collection directory.
3. Provide the following information in the `README.md` files for every role in your collection:
    - Role description: A brief summary of what the role does
    - Requirements: List the collections, libraries, and required installations
    - Dependencies
    - Role variables: Provide the following information about the variables your role uses.
        - Description
        - Defaults
        - Example values

    - Required variables

- Example playbook: Show an example of a playbook that uses your role. Use comments in the playbook to help users understand where to set variables. The `README.md` file in `controller_configuration.ad_hoc_command_cancel` is an example of a role with standard documentation.

4. To add documentation for your roles collection, navigate to the collection directory.
5. In the `README.md` file for your collection, provide the following information:

    - Collection description: Describe what the collection does.
    - Requirements: List required collections.
    - List the roles as a component of the collection.
    - Using the collection: Describe how to run the components of the collection.
    - Add a troubleshooting section.
    - Versioning: Describe the release cycle of your collection.

Publish your collection in private automation hub

Publish your collection by packaging it into a tarball and uploading it to a namespace in private automation hub. This makes the collection available for internal use in Ansible Automation Platform projects.

Before you begin

- Package your collection into a tarball.
- Format your collection file name as follows `<my_namespace-my_collection-x.y.z.tar.gz>` .For example, `company_namespace-myapp_network-2.0.0.tar.gz`

Procedure

1. Create a namespace for your collection in private automation hub. See Creating a namespace .
2. Optional: Add information to your namespace. See Adding additional information and resources to a namespace in *Managing automation content* .
3. Upload your roles collections tarballs to your namespace. See Uploading collections to your namespaces in *Managing automation content* .
4. Approve your collection for internal publication. See Uploading collections to your namespaces in *Managing automation content*.

Use your collection in projects in Red Hat Ansible Automation Platform

To use your collection in automation controller projects, add the collection to a custom execution environment. Then tag the new image and push it to private automation hub.

The following procedure describes the workflow to add a collection to an execution environment. Refer to Customizing an existing automation executions environment image for the commands to execute these steps.

Procedure

1. You can pull an execution environment base image from automation hub, or you can add your collection to your own custom execution environment.
2. Add the collections that you want to include in the execution environment.
3. Build the new execution environment.
4. Verify that the collections are in the execution environment.
5. Tag the image and push it to private automation hub.
6. Pull your new image into your automation controller instance.
7. The playbooks that use the roles in your collection must use the fully qualified domain name (FQDN) for the roles.

Emulate a platform environment locally with automation content navigator

As a content creator, you can use automation content navigator to develop Ansible playbooks, collections, and roles compatible with the Red Hat Ansible Automation Platform. It provides seamless and predictable results across all supported environments.

- Local development machines
- Automation execution environments

Automation content navigator also produces an artifact file you can use to help you develop your playbooks and troubleshoot problem areas.

NOTE:
Automation content navigator is a component of Ansible development tools. To use automation content navigator, you must install Ansible development tools.

What is automation content navigator?

Automation content navigator is a command line, content-creator-focused tool with a text-based user interface. You can use automation content navigator to:

- Launch and watch jobs and playbooks.
- Share stored, completed playbook and job run artifacts in JSON format.
- Browse and introspect automation execution environments.
- Browse your file-based inventory.
- Render Ansible module documentation and extract examples you can use in your playbooks.
- View a detailed command output on the user interface.

Automation content navigator modes

automation content navigator provides two operating modes to support your development needs: a command line interface for quick tasks and an interactive text-based interface for complex evaluation and troubleshooting.

stdout mode

Accepts most of the existing Ansible commands and extensions at the command line.

text-based user interface mode

Provides an interactive, text-based interface to the Ansible commands. Use this mode to evaluate content, run playbooks, and troubleshoot playbooks after they run using artifact files.

stdout mode

Use the `-m stdout` subcommand with automation content navigator to use the familiar Ansible commands, such as `ansible-playbook` within automation execution environments or on your local development environment. You can use commands you are familiar with for quick tasks.

Automation content navigator also provides extensive help in this mode:

`--help`

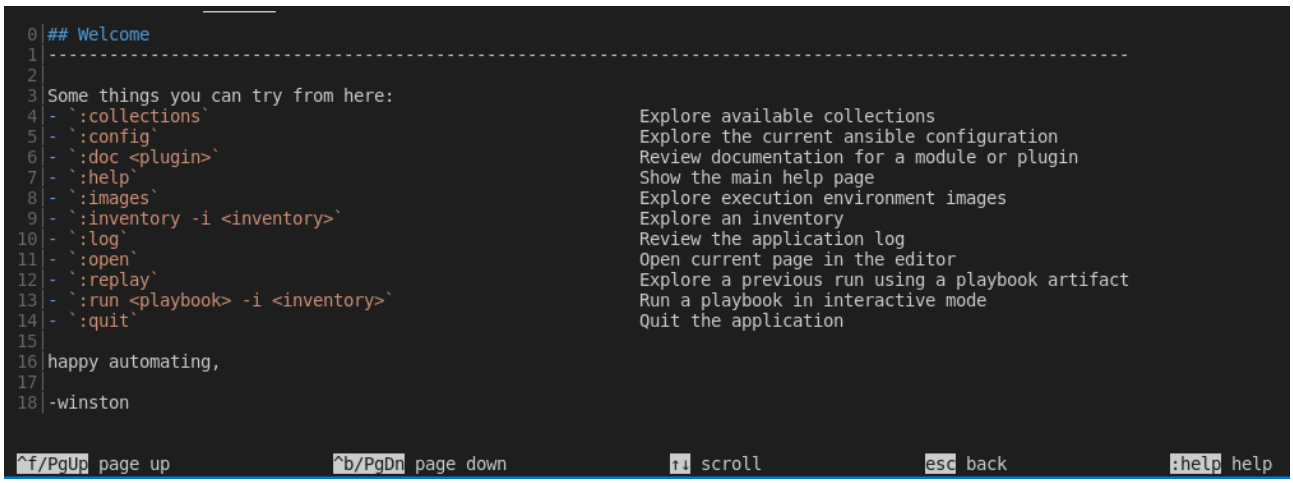
Accessible from `ansible-navigator` command or from any subcommand, such as `ansible-navigator config --help`.

subcommand help

Accessible from the subcommand, for example `ansible-navigator config --help-config`. This help displays the details of all the parameters supported from the related Ansible command.

Text-based user interface mode

The text-based user interface mode provides enhanced interaction with automation execution environments, collections, playbooks, and inventory. This mode is compatible with integrated development environments (IDE), such as Visual Studio Code.



This mode includes a number of helpful user interface options:

colon commands

You can access all the automation content navigator commands with a colon, such as `:run` or `:collectio`

navigating the text-based interface

The screen shows how to page up or down, scroll, escape to a prior screen or access `:help`.

output by line number

You can access any line number in the displayed output by preceding it with a colon, for example `:12`.

color-coded output

With colors enabled, automation content navigator displays items, such as deprecated modules, in red.

pagination and scrolling

You can page up or down, scroll, or escape by using the options displayed at the bottom of each automation content navigator screen.

You cannot switch between modes after automation content navigator is running.

This document uses the text-based user interface mode for most procedures.

Automation content navigator modes

automation content navigator provides two operating modes to support your development needs: a command line interface for quick tasks and an interactive text-based interface for complex evaluation and troubleshooting.

stdout mode

Accepts most of the existing Ansible commands and extensions at the command line.

text-based user interface mode

Provides an interactive, text-based interface to the Ansible commands. Use this mode to evaluate content, run playbooks, and troubleshoot playbooks after they run using artifact files.

stdout mode

Use the `-m stdout` subcommand with automation content navigator to use the familiar Ansible commands, such as `ansible-playbook` within automation execution environments or on your local development environment. You can use commands you are familiar with for quick tasks.

Automation content navigator also provides extensive help in this mode:

`--help`

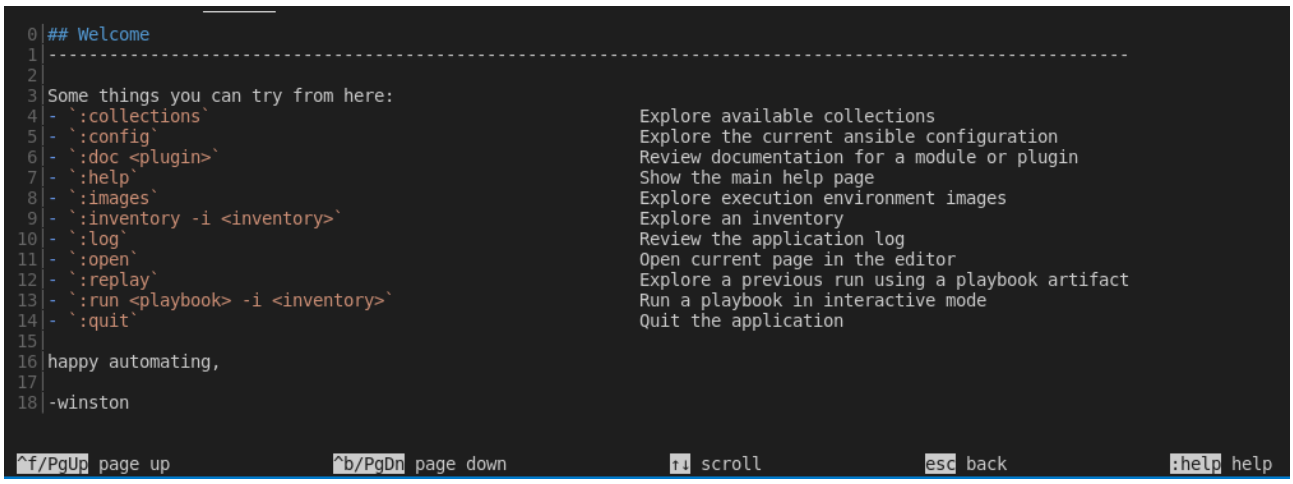
Accessible from `ansible-navigator` command or from any subcommand, such as `ansible-navigator config --help`.

subcommand help

Accessible from the subcommand, for example `ansible-navigator config --help-config`. This help displays the details of all the parameters supported from the related Ansible command.

Text-based user interface mode

The text-based user interface mode provides enhanced interaction with automation execution environments, collections, playbooks, and inventory. This mode is compatible with integrated development environments (IDE), such as Visual Studio Code.



This mode includes a number of helpful user interface options:

colon commands

You can access all the automation content navigator commands with a colon, such as `:run` or `:collectio`

navigating the text-based interface

The screen shows how to page up or down, scroll, escape to a prior screen or access `:help`.

output by line number

You can access any line number in the displayed output by preceding it with a colon, for example `:12`.

color-coded output

With colors enabled, automation content navigator displays items, such as deprecated modules, in red.

pagination and scrolling

You can page up or down, scroll, or escape by using the options displayed at the bottom of each automation content navigator screen.

You cannot switch between modes after automation content navigator is running.

This document uses the text-based user interface mode for most procedures.

Automation content navigator modes

automation content navigator provides two operating modes to support your development needs: a command line interface for quick tasks and an interactive text-based interface for complex evaluation and troubleshooting.

stdout mode

Accepts most of the existing Ansible commands and extensions at the command line.

text-based user interface mode

Provides an interactive, text-based interface to the Ansible commands. Use this mode to evaluate content, run playbooks, and troubleshoot playbooks after they run using artifact files.

stdout mode

Use the `-m stdout` subcommand with automation content navigator to use the familiar Ansible commands, such as `ansible-playbook` within automation execution environments or on your local development environment. You can use commands you are familiar with for quick tasks.

Automation content navigator also provides extensive help in this mode:

`--help`

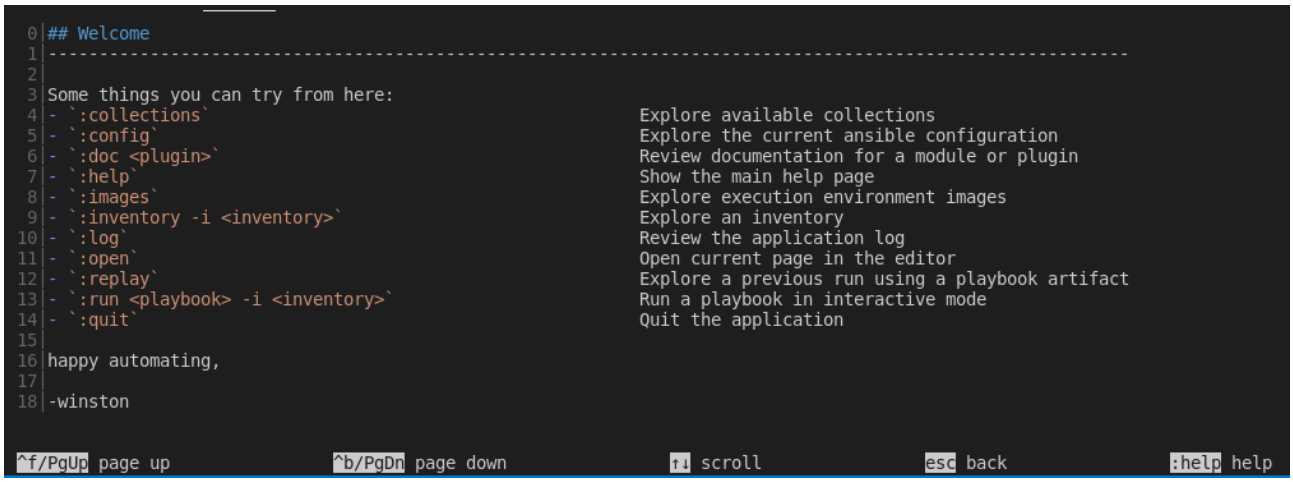
Accessible from `ansible-navigator` command or from any subcommand, such as `ansible-navigator config --help`.

subcommand help

Accessible from the subcommand, for example `ansible-navigator config --help-config`. This help displays the details of all the parameters supported from the related Ansible command.

Text-based user interface mode

The text-based user interface mode provides enhanced interaction with automation execution environments, collections, playbooks, and inventory. This mode is compatible with integrated development environments (IDE), such as Visual Studio Code.



This mode includes a number of helpful user interface options:

colon commands

You can access all the automation content navigator commands with a colon, such as `:run` or `:collectio`

navigating the text-based interface

The screen shows how to page up or down, scroll, escape to a prior screen or access `:help`.

output by line number

You can access any line number in the displayed output by preceding it with a colon, for example `:12`.

color-coded output

With colors enabled, automation content navigator displays items, such as deprecated modules, in red.

pagination and scrolling

You can page up or down, scroll, or escape by using the options displayed at the bottom of each automation content navigator screen.

You cannot switch between modes after automation content navigator is running.

This document uses the text-based user interface mode for most procedures.

Automation content navigator commands

The automation content navigator commands run familiar Ansible CLI commands in `-m stdout` mode. You can use all the subcommands and options from the related Ansible CLI command. Use `ansible-navigator --help` for details.

Automation content navigator commands

Command	Description	CLI example
collections	Explore available collections	`ansible-navigator collections --help`
config	Explore the current Ansible configuration	`ansible-navigator config --help`
doc	Review documentation for a module or plugin	`ansible-navigator doc --help`
images	Explore execution environment images	`ansible-navigator images --help`
inventory	Explore an inventory	`ansible-navigator inventory --help`
replay	Explore a previous run using a playbook artifact	`ansible-navigator replay --help`
run	Run a playbook	`ansible-navigator run --help`
welcome	Start at the welcome page	`ansible-navigator welcome --help`

Relationship between Ansible and automation content navigator commands

The automation content navigator commands run familiar Ansible CLI commands in `-m stdout` mode. You can use all the subcommands and options available in the related Ansible CLI command. Use `ansible-navigator --help` for details.

Comparison of automation content navigator and Ansible CLI commands

automation content navigator command	Ansible CLI command
`ansible-navigator collections`	`ansible-galaxy collection`
`ansible-navigator config`	`ansible-config`
`ansible-navigator doc`	`ansible-doc`
`ansible-navigator inventory`	`ansible-inventory`
`ansible-navigator run`	`ansible-playbook`

Run a playbook to display execution environment contents

As a content developer, you can review your automation execution environment with automation content navigator and display the packages and collections included in the automation execution environments. Automation content navigator runs a playbook to extract and display the results.

Review automation execution environments from automation content navigator

Use the automation content navigator text-based interface to review your automation execution environments. This allows you to quickly inspect packages, versions, and collections installed in the environment.

Before you begin

- Automation execution environments

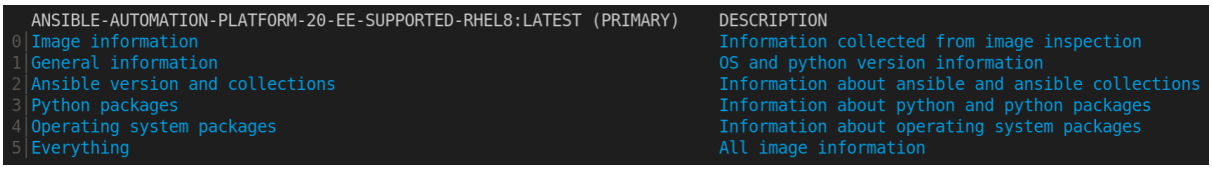
Procedure

1. Review the automation execution environments included in your automation content navigator configuration.

```
$ ansible-navigator images
```

	NAME	TAG	EXECUTION ENVIRONMENT	CREATED	SIZE
0	ansible-automation-platform-20-ee-minimal-rhel8	latest	True	4 weeks ago	411 MB
1	ansible-automation-platform-20-ee-supported-rhel8 (primary)	latest	True	45 hours ago	923 MB
2	ansible-runner	devel	True	4 weeks ago	652 MB
3	ccutil	amazing	False	16 months ago	1.67 GB

2. Type the number of the automation execution environment you want to delve into for more details.



	ANSIBLE-AUTOMATION-PLATFORM-20-EE-SUPPORTED-RHEL8:LATEST (PRIMARY)	DESCRIPTION
0	Image information	Information collected from image inspection
1	General information	OS and python version information
2	Ansible version and collections	Information about ansible and ansible collections
3	Python packages	Information about python and python packages
4	Operating system packages	Information about operating system packages
5	Everything	All image information

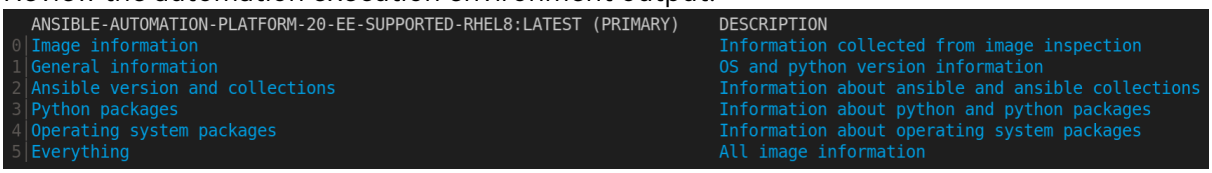
You can review the packages and versions of each installed automation execution environment and the Ansible version any included collections.

3. Optional: pass in the automation execution environment that you want to use. This becomes the primary and is the automation execution environment that automation content navigator uses.

```
$ ansible-navigator images --eei registry.example.com/example-enterprise-
ee:latest
```

Result

- Review the automation execution environment output.



	ANSIBLE-AUTOMATION-PLATFORM-20-EE-SUPPORTED-RHEL8:LATEST (PRIMARY)	DESCRIPTION
0	Image information	Information collected from image inspection
1	General information	OS and python version information
2	Ansible version and collections	Information about ansible and ansible collections
3	Python packages	Information about python and python packages
4	Operating system packages	Information about operating system packages
5	Everything	All image information

Browse collections in a text-based format

As a content creator, you can browse your Ansible collections with automation content navigator and interactively delve into each collection developed locally or within Automation execution environments.

Automation content navigator collections display

Understand the information displayed by Automation content navigator about your collections, such as shadowing, type, and path. This overview helps you verify the collection source and search priority.

Automation content navigator displays information about your collections with the following details for each collection:

SHADOWED

Indicates that an additional copy of the collection is higher in the search order, and playbooks prefer that collection.

TYPE

Shows if the collection is contained within an automation execution environment or volume mounted on onto the automation execution environment as a `bind_mount`.

PATH

Reflects the collections location within the automation execution environment or local file system based on the collection TYPE field.

	NAME	VERSION	SHADOWED	TYPE	PATH
0	amazon.aws	1.5.0	False	contained	/usr/share/ansible/collections/ansible_collections/amazon/aws/
1	ansible.posix	1.2.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/posix/
2	ansible.utils	2.2.0	False	bind_mount	/home/samccann/aap/collections/ansible_collections/ansible/utils/
3	ansible.windows	1.6.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/windows/
4	awx.awx	19.2.2	False	bind_mount	/home/samccann/aap/collections/ansible_collections/awx/awx/
5	awx.awx	19.2.0	True	contained	/usr/share/ansible/collections/ansible_collections/awx/awx/
6	azure.azcollection	1.7.0	False	contained	/usr/share/ansible/collections/ansible_collections/azure/azcollection/

Browse collections from automation content navigator

You can browse Ansible collections with the automation content navigator text-based user interface in interactive mode and delve into each collection. automation content navigator shows collections within the current project directory and those available in the automation execution environments

Before you begin

- A locally accessible collection or installed automation execution environments.

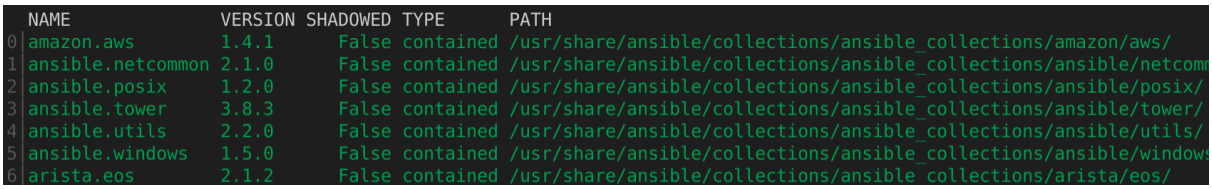
Procedure

1. Start automation content navigator

```
$ ansible-navigator
```

2. Browse the collection. Alternately, you can type `ansible-navigator collections` to directly browse the collections.

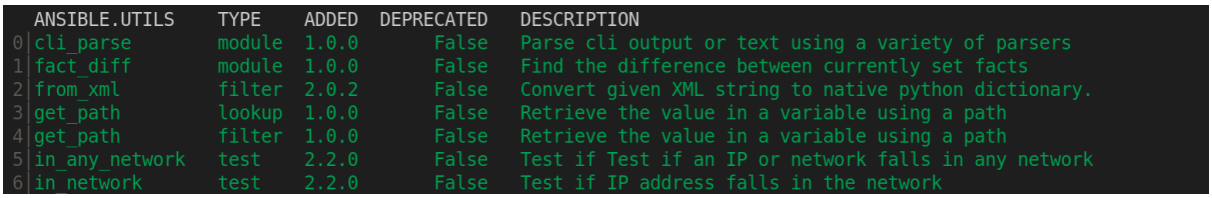
```
$ :collections
```



	NAME	VERSION	SHADOWED	TYPE	PATH
0	amazon.aws	1.4.1	False	contained	/usr/share/ansible/collections/ansible_collections/amazon/aws/
1	ansible.netcommon	2.1.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/netcomm
2	ansible.posix	1.2.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/posix/
3	ansible.tower	3.8.3	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/tower/
4	ansible.utils	2.2.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/utils/
5	ansible.windows	1.5.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/windows
6	arista.eos	2.1.2	False	contained	/usr/share/ansible/collections/ansible_collections/arista/eos/

3. Type the number of the collection you want to explore.

```
:4
```



	ANSIBLE.UTILS	TYPE	ADDED	DEPRECATED	DESCRIPTION
0	cli_parse	module	1.0.0	False	Parse cli output or text using a variety of parsers
1	fact_diff	module	1.0.0	False	Find the difference between currently set facts
2	from_xml	filter	2.0.2	False	Convert given XML string to native python dictionary.
3	get_path	lookup	1.0.0	False	Retrieve the value in a variable using a path
4	get_path	filter	1.0.0	False	Retrieve the value in a variable using a path
5	in_any_network	test	2.2.0	False	Test if Test if an IP or network falls in any network
6	in_network	test	2.2.0	False	Test if IP address falls in the network

4. Type the number corresponding to the module you want to delve into.

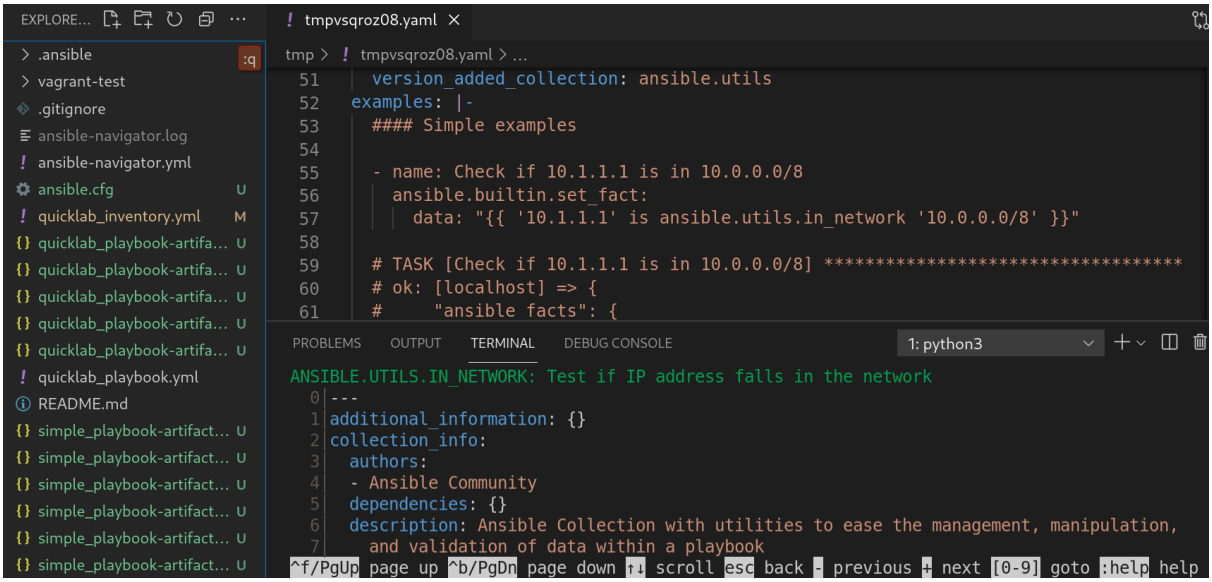
```
ANSIBLE.UTILS.IP_ADDRESS: Test if something in an IP address
 0|---
 1|additional_information: {}
 2|collection_info:
 3|  authors:
 4|  - Ansible Community
 5|  dependencies: {}
 6|  description: Ansible Collection with utilities to ease the management,
manipulation,
 7|    and validation of data within a playbook
 8|  documentation: null
 9|  homepage: null
10|  issues: null
11|  license: []
12|  license_file: LICENSE
13|  name: ansible.utils
14|  namespace: ansible
15|  path:/usr/share/ansible/collections/ansible_collections/ansible/utils/
16|  readme: README.md
<... output truncated...>
```

5. Optional: jump to the documentation examples for this module.

```
:{{ examples }}


0|
1|
2|#### Simple examples
3|
4|- name: Check if 10.1.1.1 is a valid IP address
5|  ansible.builtin.set_fact:
6|    data: "{{ '10.1.1.1' is ansible.utils.ip_address }}"
7|
8|# TASK [Check if 10.1.1.1 is a valid IP address] *********************
9|# ok: [localhost] => {
10|#     "ansible_facts": {
11|#         "data": true
12|#     },
13|#     "changed": false
14|# }
15|
```

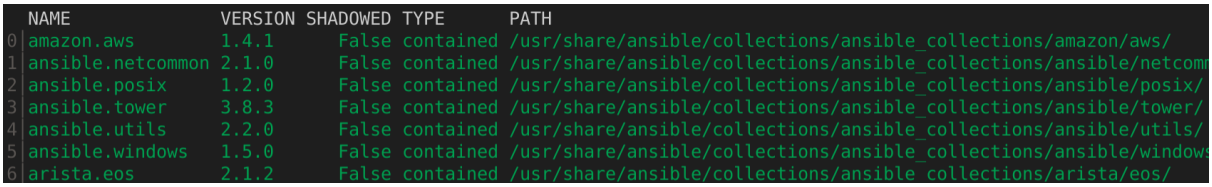
6. Optional: open the example in your editor to copy it into a playbook.

```
:open
```



Result

- Browse the collection list.



	NAME	VERSION	SHADOWED	TYPE	PATH
0	amazon.aws	1.4.1	False	contained	/usr/share/ansible/collections/ansible_collections/amazon/aws/
1	ansible.netcommon	2.1.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/netcomm
2	ansible.posix	1.2.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/posix/
3	ansible.tower	3.8.3	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/tower/
4	ansible.utils	2.2.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/utils/
5	ansible.windows	1.5.0	False	contained	/usr/share/ansible/collections/ansible_collections/ansible/windows
6	arista.eos	2.1.2	False	contained	/usr/share/ansible/collections/ansible_collections/arista/eos/

Review collection and plugin documentation

You can review Ansible documentation for collections and plugins with the automation content navigator text-based user interface in interactive mode. automation content navigator shows collections within the current project directory and those available in the automation execution environments

Before you begin

- A locally accessible collection or installed automation execution environments.

Procedure

1. Start automation content navigator

```
$ ansible-navigator
```

2. Review the module you are interested in. Alternately, you can type `ansible-navigator doc` to access the documentation.

```
:doc ansible.utils.ip_address
```

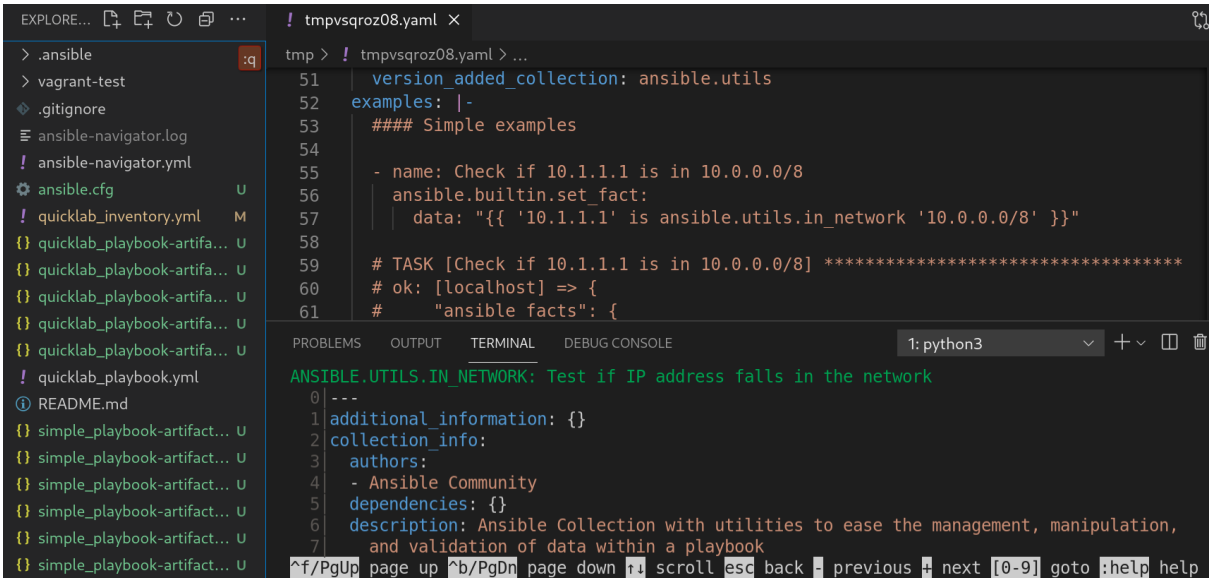
```
ANSIBLE.UTILS.IP_ADDRESS: Test if something in an IP address

 0|---

 1|additional_information: {}

 2|collection_info:

 3|  authors:

 4|  - Ansible Community

 5|  dependencies: {}

 6|  description: Ansible Collection with utilities to ease the management,
manipulation,

 7|    and validation of data within a playbook

 8|  documentation: null

 9|  homepage: null

10|  issues: null

11|  license: []

12|  license_file: LICENSE

13|  name: ansible.utils

14|  namespace: ansible

15|  path:/usr/share/ansible/collections/ansible_collections/ansible/utils/

16|  readme: README.md

<... output truncated...>
```

3. Jump to the documentation examples for this module.

```
:{{ examples }}


0|
1|
2|#### Simple examples
3|
4|- name: Check if 10.1.1.1 is a valid IP address
5|  ansible.builtin.set_fact:
6|    data: "{{ '10.1.1.1' is ansible.utils.ip_address }}"
7|
8|# TASK [Check if 10.1.1.1 is a valid IP address] *********************
9|# ok: [localhost] => {
10|#     "ansible_facts": {
11|#         "data": true
12|#     },
13|#     "changed": false
14|# }
15|
```

4. Optional: open the example in your editor to copy it into a playbook.

```
:open
```



See Automation content navigator general settings for details on how to set up your editor.

View groups and hosts in your inventory

As a content creator, you can review your Ansible inventory with automation content navigator and interactively delve into the groups and hosts.

Review inventory from automation content navigator

You can review Ansible inventories with the automation content navigator text-based user interface in interactive mode and delve into groups and hosts for more details.

Before you begin

- A valid inventory file or an inventory plugin.

Procedure

1. Start automation content navigator.

```
$ ansible-navigator
```

Optional: type `ansible-navigator inventory -i simple_inventory.yml` from the command line to view the inventory.

2. Review the inventory.

```
  :inventory -i simple_inventory.yml


   TITLE            DESCRIPTION
0|Browse groups    Explore each inventory group and group members members
1|Browse hosts     Explore the inventory with a list of all hosts
```

3. Type `0` to brows the groups.

```
   NAME              TAXONOMY                    TYPE
0|general            all                         group
1|nodes              all                         group
2|ungrouped          all                         group
```

The `TAXONOMY` field details the hierarchy of groups the selected group or node belongs to.

4. Type the number corresponding to the group you want to delve into.

```
   NAME              TAXONOMY                    TYPE
0|node-0             all▸nodes                   host
1|node-1             all▸nodes                   host
2|node-2             all▸nodes                   host
```

5. Type the number corresponding to the host you want to delve into, or type `:<number>` for numbers greater than 9.

```
 [node-1]
0|---
1|ansible_host: node-1.example.com
2|inventory_hostname: node-1
```

Result

- Review the inventory output.

```
  TITLE            DESCRIPTION
0|Browse groups   Explore each inventory group and group members members
1|Browse hosts    Explore the inventory with a list of all hosts
```

Run playbooks locally with automation content navigator

Use automation content navigator to run playbooks and interactively delve into play and task results. Compare execution inside and outside of execution environments to identify and troubleshoot problems.

Run a playbook from automation content navigator

You can run Ansible playbooks with the automation content navigator text-based user interface to follow the execution of the tasks and delve into the results of each task.

Before you begin

- A playbook.
- A valid inventory file if not using `localhost` or an inventory plugin.

Procedure

1. Start automation content navigator

```
$ ansible-navigator
```

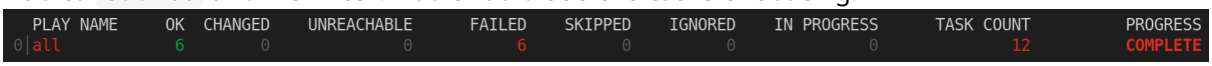
2. Run the playbook.

```
$ :run
```

3. Optional: type `ansible-navigator run simple-playbook.yml -i inventory.yml` to run the playbook.
4. Verify or add the inventory and any other command line parameters.

```
INVENTORY OR PLAYBOOK NOT FOUND, PLEASE CONFIRM THE FOLLOWING
──────────────────────────────────────────────────────────────────────────────

   Path to playbook: /home/ansible-navigator_demo/simple_playbook.yml

   Inventory source: /home/ansible-navigator-demo/inventory.yml

  Additional command line parameters: Please provide a value (optional)
──────────────────────────────────────────────────────────────────────────────

                                                          Submit Cancel
```

5. Tab to `Submit` and hit Enter. You should see the tasks executing.



6. Type the number next to a play to step into the play results, or type `:<number>` for numbers above 9.

```
  RESULT   HOST       NUMBER   CHANGED   TASK                          TASK ACTION                          DURATION
3 OK       node-0          3     False   Gathering Facts               gather_facts                               1s
4 OK       node-1          4     False   Gathering Facts               gather_facts                               1s
5 OK       node-2          5     False   Gathering Facts               gather_facts                               1s
6 FAILED   main-0          6     False   Gather the package facts      ansible.builtin.package_facts              1s
7 FAILED   infra-0         7     False   Gather the package facts      ansible.builtin.package_facts              1s
8 FAILED   lb-0            8     False   Gather the package facts      ansible.builtin.package_facts              1s
9 FAILED   node-0          9     False   Gather the package facts      ansible.builtin.package_facts              1s
10 FAILED  node-1         10     False   Gather the package facts      ansible.builtin.package_facts              1s
11 FAILED  node-2         11     False   Gather the package facts      ansible.builtin.package_facts              0s
```

Notice failed tasks show up in red if you have colors enabled for automation content navigator.

7. Type the number next to a task to review the task results, or type `:<number>` for numbers above 9.

```
PLAY [all:6] ***************************************************************************************************************
TASK [Gather the package facts] ********************************************************************************************
FAILED: [main-0] Could not detect a supported package manager from the following list: ['apt', 'apk', 'rpm', 'portage', 'pkg'
 0 ---
 1 duration: 1.339719
 2 end: '2021-06-10T18:52:32.968770'
 3 event_loop: null
 4 host: main-0
 5 ignore_errors: null
 6 play: all
```

8. Optional: type `:doc` bring up the documentation for the module or plugin used in the task to aid in troubleshooting.

```
ANSIBLE.BUILTIN.PACKAGE_FACTS (MODULE)
  0|---
  1|doc:
  2|  author:
  3|  - Matthew Jones (@matburt)
  4|  - Brian Coca (@bcoca)
  5|  - Adam Miller (@maxamillion)
  6|  collection: ansible.builtin
  7|  description:
  8|  - Return information about installed packages as facts.
<... output omitted ...>
 11|  module: package_facts
 12|  notes:
 13|  - Supports C(check_mode).
 14|  options:
 15|    manager:
 16|      choices:
 17|      - auto
 18|      - rpm
 19|      - apt
 20|      - portage
 21|      - pkg
 22|      - pacman


<... output truncated ...>
```

Related information
Getting started with playbooks

Review playbook results

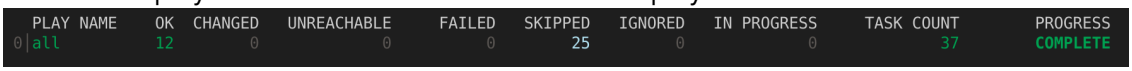
Automation content navigator saves playbook runs as JSON artifact files. You can use these files to share, review, or troubleshoot results without requiring access to the original playbook or inventory.

Before you begin

- A automation content navigator artifact JSON file from a playbook run.
- Start automation content navigator with the artifact file.

```
$ ansible-navigator replay simple_playbook_artifact.json
```

a. Review the playbook results that match when the playbook ran.



Result

You can now type the number next to the plays and tasks to step into each to review the results, as you would after executing the playbook.

Related information
Getting started with playbooks

View Ansible configuration file contents in text-based format

As a content creator, you can review your Ansible configuration with automation content navigator and interactively delve into settings.

Review your Ansible configuration from automation content navigator

Use the automation content navigator interactive interface to review Ansible settings. The tool pulls data from your configuration file or returns defaults if no configuration file is present.

Before you begin

- You have authenticated to the Red Hat registry if you need to access additional automation execution environments. See Red Hat Container Registry Authentication for details.

Procedure

1. Start automation content navigator

```
$ ansible-navigator
```

Optional: type `ansible-navigator config` from the command line to access the Ansible configuration settings.

2. Review the Ansible configuration.

```
:config
```

Some values reflect settings from within the automation execution environments needed for the automation execution environments to function. These display as non-default settings you cannot set in your Ansible configuration file.

3. Type the number corresponding to the setting you want to delve into, or type `:<number>` for numbers greater than 9.

```
ANSIBLE_COW_ACCEPTLIST (current: ['bud-frogs', 'bunny', 'cheese']) (default:
0|---
1|current:
2|- bud-frogs
3|- bunny
4|- cheese
5|default:
6|- bud-frogs
7|- bunny
8|- cheese
9|- daemon
```

The output shows the current `setting` as well as the `default`. Note the `source` in this example is `env` since the setting comes from the automation execution environments.

Result

- Review the configuration output.

Troubleshoot collections, execution environments, and playbooks

Troubleshoot Ansible collections, playbooks, and execution environments interactively with automation content navigator. Compare results inside or outside environments to resolve issues fast.

Review playbook results

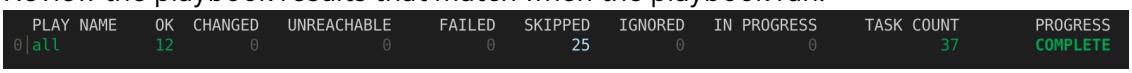
Automation content navigator saves playbook runs as JSON artifact files. You can use these files to share, review, or troubleshoot results without requiring access to the original playbook or inventory.

Before you begin

- A automation content navigator artifact JSON file from a playbook run.
- Start automation content navigator with the artifact file.

```
$ ansible-navigator replay simple_playbook_artifact.json
```

a. Review the playbook results that match when the playbook ran.



	PLAY NAME	OK	CHANGED	UNREACHABLE	FAILED	SKIPPED	IGNORED	IN PROGRESS	TASK COUNT	PROGRESS
0	all	12	0	0	0	25	0	0	37	COMPLETE

Result

You can now type the number next to the plays and tasks to step into each to review the results, as you would after executing the playbook.

Related information
Getting started with playbooks

Frequently asked questions about automation content navigator

Review frequently asked questions about automation content navigator configuration and behavior. This information helps you troubleshoot common problems and optimize your environment setup.

Where should the `ansible.cfg` file go when using an automation execution environment?

The easiest place to have the `ansible.cfg` file is in the project directory next to the playbook. The playbook directory is automatically mounted in the automation execution environment and automation content navigator finds the `ansible.cfg` file there. If the `ansible.cfg` file is in another directory, set the `ANSIBLE_CONFIG` variable, and specify the directory as a custom volume mount. (See automation content navigator settings for `execution-environment-volume-mounts`)

Where should the `ansible.cfg` file go when not using an automation execution environment?

Ansible looks for the `ansible.cfg` in the typical locations when not using an automation execution environment. See Reviewing your Ansible configuration with automation content navigator.

Where should Ansible collections be placed when using an automation execution environment?

The easiest place to have Ansible collections is in the project directory, in a playbook adjacent collections directory (for example, `ansible-galaxy collections install ansible.utils -p ./collections`). The playbook directory is automatically mounted in the automation execution environment and automation content navigator finds the collections there. Another option is to build the collections into an automation execution environment using Ansible Builder. This helps content creators author playbooks that are production ready, since automation controller supports playbook adjacent collection directories. If the collections are in another directory, set the `ANSIBLE_COLLECTIONS_PATHS` variable and configure a custom volume mount for the directory. (See Automation content navigator general settings for `execution-environment-volume-mounts`).

Where should Ansible collections be placed when not using an automation execution environment?

When not using an automation execution environment, Ansible looks in the default locations for collections. See Creating a collection for distributing roles for more information.

Why does the playbook hang when `vars_prompt` or `pause/prompt` is used?

By default, automation content navigator runs the playbook in the same manner that automation controller runs the playbook. This helps content creators author playbooks that are production ready. If you cannot avoid the use of `vars_prompt` or `pause\prompt`, disabling `playbook-artifact` creation causes automation content navigator to run the playbook in a manner that is compatible with `ansible-playbook` and allows for user interaction.

Why does automation content navigator change the terminal colors or look terrible?

Automation content navigator queries the terminal for its OSC4 compatibility. OSC4, 10, 11, 104, 110, 111 indicate the terminal supports color changing and reverting. It is possible that the

terminal is misrepresenting its ability. You can disable OSC4 detection by setting `--osc4 false`. (See Automation content navigator general settings for how to handle this with an environment variable or in the settings file).

How can I change the colors used by automation content navigator?

Use `--osc4 false` to force automation content navigator to use the terminal defined colors. (See Automation content navigator general settings for how to handle this with an environment variable or in the settings file).

What is with all these `site-artifact-2021-06-02T16:02:33.911259+00:00.json` files in the playbook directory?

Automation content navigator creates a playbook artifact for every playbook run. These can be helpful for reviewing the outcome of automation after it is complete, sharing and troubleshooting with a colleague, or keeping for compliance or change-control purposes. The playbook artifact file has the detailed information about every play and task, and the `stdout` from the playbook run. You can review playbook artifacts with `ansible-navigator replay <filename>` or `:replay <filename>` while in an automation content navigator session. You can review all playbook artifacts with both `--mode stdout` and `--mode interactive`, depending on the required view. You can disable playbook artifacts writing and the default file naming convention. (See Automation content navigator general settings for how to handle this with an environment variable or in the settings file).

Why does `vi` open when I use `:open`?

Automation content navigator opens anything showing in the terminal in the default editor. The default is set to either `vi +{line_number} {filename}` or the current value of the `EDITOR` environment variable. Related to this is the `editor-console` setting which indicates if the editor is console or terminal based. Here are examples of alternate settings that might be useful:

```
# emacs
ansible-navigator:
  editor:
    command: emacs -nw +{line_number} {filename}
    console: true
```

```
# vscode
ansible-navigator:
  editor:
    command: code -g {filename}:{line_number}
    console: false
```

```
#pycharm
ansible-navigator:
  editor:
    command: charm --line {line_number} {filename}
    console: false
```

What is the order in which configuration settings are applied?

The automation content navigator configuration system pulls in settings from various sources and applies them hierarchically in the following order (where the last applied changes are the most prevalent):

1. Default internal values
2. Values from a settings file
3. Values from environment variables
4. Flags and arguments specified on the command line
5. While issuing `:` commands within the text-based user interface

Something did not work, how can I troubleshoot it?

Automation content navigator has reasonable logging messages. You can enable `debug` logging with `--log-level debug`. If you think you might have found a bug, log an issue and include the details from the log file.

Automation content navigator command reference

Configure automation content navigator settings to customize the tool for your specific development environment needs. This helps you optimize your workflow for efficiency.

Create an automation content navigator settings file

Alter default automation content navigator settings using the command line, environment variables, or a dedicated settings file. Using a settings file helps ensure consistent configuration across sessions.

Automation content navigator checks for a settings file in the following order and uses the first match:

- `ANSIBLE_NAVIGATOR_CONFIG` - The settings file path environment variable if set.
- `./ansible-navigator.<ext>` - The settings file within the current project directory, with no dot in the file name.
- `\~/.ansible-navigator.<ext>` - Your home directory, with a dot in the file name.

Consider the following when you create an automation content navigator settings file:

- The settings file can be in `JSON` or `YAML` format.
- For settings in `JSON` format, the extension must be `.json`.
- For settings in `YAML` format, the extension must be `.yml` or `.yaml`.
- The project and home directories can only contain one settings file each.
- If automation content navigator finds more than one settings file in either directory, it results in an error.

You can copy the example settings file below into one of those paths to start your `ansible-navigator` settings file.

```
---
ansible-navigator:
#   ansible:
#     config: /tmp/ansible.cfg
#     cmdline: "--forks 15"
#     inventories:
#     - /tmp/test_inventory.yml
#     playbook: /tmp/test_playbook.yml
#   ansible-runner:
#     artifact-dir: /tmp/test1
#     rotate-artifacts-count: 10
#     timeout: 300
#   app: run
#   collection-doc-cache-path: /tmp/cache.db
#   color:
#     enable: False
#     osc4: False
#   editor:
#     command: vim_from_setting
#     console: False
```

```
#   documentation:
#     plugin:
#       name: shell
#       type: become
#   execution-environment:
#     container-engine: podman
#     enabled: False
#     environment-variables:
#       pass:
#         - ONE
#         - TWO
#         - THREE
#       set:
#         KEY1: VALUE1
#         KEY2: VALUE2
#         KEY3: VALUE3
#     image: test_image:latest
#     pull-policy: never
#     volume-mounts:
#     - src: "/test1"
#       dest: "/test1"
#       label: "Z"
#   help-config: True
#   help-doc: True
#   help-inventory: True
#   help-playbook: False
#   inventory-columns:
#     - ansible_network_os
#     - ansible_network_cli_ssh_type
#     - ansible_connection
  logging:
#     append: False
    level: critical
#     file: /tmp/log.txt
#   mode: stdout
#   playbook-artifact:
```

```
    #     enable: True
    #     replay: /tmp/test_artifact.json
    #     save-as: /tmp/test_artifact.json
```

Automation content navigator general settings

The following table describes each general parameter and setting options for automation content navigator.

Automation content navigator general parameters settings

Parameter	Description	Setting options

ansible-runner-artifact-dir	The directory path to store artifacts generated by ansible-runner.	**Default:** No default value set
		CLI: `--rad` or `--ansible-runner-artifact-dir`
		ENV: `ANSIBLE_NAVIGATOR_ANSIBLE_RUNNER_ARTIFACT_DIR`
		Settings file:
		`ansible-navigator:`
		`ansible-runner:`
		`artifact-dir:`

ansible-runner-rotate-artifacts-count	Keep ansible-runner artifact directories, for last n runs. If set to 0, artifact directories are not deleted.	**Default:** No default value set **CLI:** `--rac` or `--ansible-runner-rotate-artifacts-count` **ENV:** `ANSIBLE_NAVIGATOR_ANSIB`

Parameter	Description	Setting options

		`LE_RUNNER_ROTATE_ARTIFACTS_COUNT`
		
		Settings file:
		
		`ansible-navigator:`
		`  ansible-runner:`
		`  rotate-artifacts-count:`

ansible-runner-timeout	The timeout value after which `ansible-runner` force stops the execution.	**Default:** No default value set
		
		CLI: `--rt` or `--ansible-runner-timeout`
		
		ENV: `ANSIBLE_NAVIGATOR_ANSIBLE_RUNNER_TIMEOUT`
		
		Settings file:
		
		`ansible-navigator:`
		`  ansible-runner:`
		`  timeout:`

app	Entry point for automation content navigator.	**Choices**: `collections`, `config`, `doc`, `images`, `inventory`, `replay`, `run` or `welcome` **Default**: `welcome` **CLI example**: `ansible-navigator collections` **ENV**: `ANSIBLE_NAVIGATOR_APP` **Settings file:**

Parameter	Description	Setting options
		`ansible-navigator:` `  app:`

cmdline	Extra parameters passed to the corresponding command.	**Default**: No default value
		CLI: positional
		ENV: `ANSIBLE_NAVIGATOR_CMDLINE`
		Settings file:
		`ansible-navigator:`
		`  ansible:`
		`  cmdline:`

collection-doc-cache-path	The path to the collection doc cache.	**Default**: `$HOME/.cache/ansible-navigator/collection_doc_cache.db`
		CLI: `--cdcp` or `--collection-doc-cache-path`
		ENV: `ANSIBLE_NAVIGATOR_COLLECTION_DOC_CACHE_PATH`
		Settings file:
		`ansible-navigator:`
		`  collection-doc-cache-path:`

container-engine	Specify the container engine (`auto` = `podman` then `docker`).	**Choices:** `auto` , `podman` or `docker` **Default:** `auto`

Parameter	Description	Setting options

		CLI: `--ce` or `--container-engine`
		
		ENV: `ANSIBLE_NAVIGATOR_CONTAINER_ENGINE`
		
		Settings file:
		
		`ansible-navigator:`
		`  execution-environment:`
		`  container-engine:`

display-color	Enable the use of color in the display.	**Choices:** `True` or `False`
		
		Default: `True`
		
		CLI: `--dc` or `--display-color`
		
		ENV: `NO_COLOR`
		
		Settings file:
		
		`ansible-navigator:`
		`  color:`
		`  enable:`

editor-command	Specify the editor used by automation content navigator	Default:* vi +{line_number} {filename} **CLI:** `--ecmd` or `--editor-command` **ENV:** `ANSIBLE_NAVIGATOR_EDITOR_COMMAND` **Settings file:**

Parameter	Description	Setting options

		`ansible-navigator:`
		`  editor:`
		`  command:`

editor-console	Specify if the editor is console based.	**Choices:** `True` or `False`
		Default: `True`
		CLI: `--econ` or `--editor-console`
		ENV: `ANSIBLE_NAVIGATOR_EDITOR_CONSOLE`
		Settings file:
		`ansible-navigator:`
		`  editor:`
		`  console:`

execution-environment	Enable or disable the use of an automation execution environment.	**Choices:** `True` or `False`
		Default: `True`
		CLI: `--ee` or `--execution-environment`
		ENV:* `ANSIBLE_NAVIGATOR_EXECUTION_ENVIRONMENT`
		Settings file:
		`ansible-navigator:`
		`  execution-environment:`
		`  enabled:`

Parameter	Description	Setting options

execution-environment-image	Specify the name of the automation execution environment image.	**Default:** `quay.io/ansible/ansible-runner:devel`
		
		CLI: `--eei` or `--execution-environment-image`
		
		ENV: `ANSIBLE_NAVIGATOR_EXECUTION_ENVIRONMENT_IMAGE`
		
		Settings file:
		
		`ansible-navigator:`
		`  execution-environment:`
		`  image:`

execution-environment-volume-mounts	Specify volume to be bind mounted within an automation execution environment (`--eev /home/user/test:/home/user/test:Z`)	**Default:** No default value set
		
		CLI: `--eev` or `--execution-environment-volume-mounts`
		
		ENV: `ANSIBLE_NAVIGATOR_EXECUTION_ENVIRONMENT_VOLUME_MOUNTS`
		
		Settings file:
		
		`ansible-navigator:`
		`  execution-environment:`
		`  volume-mounts:`

log-append	Specify if log messages should be appended to an existing log file, otherwise a new log file is created per session.	**Choices:** `True` or `False` **Default:** True

Parameter	Description	Setting options

		CLI: `--la` or `--log-append`
		
		ENV: `ANSIBLE_NAVIGATOR_LOG_APPEND`
		
		Settings file:
		
		`ansible-navigator:`
		`  logging:`
		`  append:`

log-file	Specify the full path for the automation content navigator log file.	**Default:** `$PWD/ansible-navigator.log`
		
		CLI: `--lf` or `--log-file`
		
		ENV: `ANSIBLE_NAVIGATOR_LOG_FILE`
		
		Settings file:
		
		`ansible-navigator:`
		`  logging:`
		`  file:`

log-level	Specify the automation content navigator log level.	**Choices:** `debug`, `info`, `warning`, `error` or `critical` **Default:** `warning` **CLI:** `--ll` or `--log-level` **ENV:** `ANSIBLE_NAVIGATOR_LOG_LEVEL` **Settings file:**

Parameter	Description	Setting options

		`ansible-navigator:`
		`  logging:`
		`  level:`

mode	Specify the user-interface mode.	**Choices:** `stdout` or `interactive`
		Default: `interactive`
		CLI: `-m` or `--mode`
		ENV: `ANSIBLE_NAVIGATOR_MODE`
		Settings file:
		`ansible-navigator:`
		`  mode:`

osc4	Enable or disable terminal color changing support with OSC 4.	**Choices:** `True` or `False`
		Default: `True`
		CLI: `--osc4` or `--osc4`
		ENV: `ANSIBLE_NAVIGATOR_OSC4`
		Settings file:
		`ansible-navigator:`
		`  color:`
		`  osc4:`

pass-environment-variable	Specify an exiting environment variable to be passed through to and set within the automation execution environment (`--penv MY_VAR`)	**Default:** No default value set **CLI:** `--penv` or `--pass-environment-variable`

Parameter	Description	Setting options

		ENV: `ANSIBLE_NAVIGATOR_PASS_ENVIRONMENT_VARIABLES`
		
		Settings file:
		
		`ansible-navigator:`
		`  execution-environment:`
		`  environment-variables:`
		`  pass:`

pull-policy	Specify the image pull policy.
	
	`always` - Always pull the image
	
	`missing` - Pull if not locally available
	
	`never` - Never pull the image
	
	`tag` - If the image tag is `latest` always pull the image, otherwise pull if not locally available
		Choices: `always`, `missing`, `never`, or `tag`
		
		Default: `tag`
		
		CLI: `--pp` or `--pull-policy`
		
		ENV: `ANSIBLE_NAVIGATOR_PULL_POLICY`
		
		Settings file:
		
		`ansible-navigator:`
		`  execution-environment:`
		`  pull-policy:`

set-environment-variable	Specify an environment variable and a value to be set within the automation execution environment `(--senv MY_VAR=42)`	**Default:** No default value set **CLI:** `--senv` or `--set-environment-variable` **ENV:** `ANSIBLE_NAVIGATOR_SET_ENVIRONMENT_VARIABLES` **Settings file:**

Parameter	Description	Setting options

		`ansible-navigator:`
		`  execution-environment:`
		`  environment-variables:`
		`  set:`

Automation content navigator `config` subcommand settings

The following table describes each parameter and setting options for the automation content navigator `config` subcommand.

Automation content navigator `config` subcommand parameters settings

Parameter	Description	Setting options

config	Specify the path to the Ansible configuration file.	**Default:** No default value set
		CLI: `-c` or `--config`
		ENV: `ANSIBLE_CONFIG`
		Settings file:
		`ansible-navigator:`
		`  ansible:`
		`  config:`
		`  path:`

help-config	Help options for the `ansible-config` command in `stdout` mode.	**Choices:*** `True` or `False` **Default:** `False`

Parameter	Description	Setting options

		CLI: `--hc` or `--help-config`
		ENV: `ANSIBLE_NAVIGATOR_HELP_CONFIG`
		Settings file:
		`ansible-navigator:`
		`  help-config:`

automation content navigator `doc` subcommand settings

The following table describes each parameter and setting options for the automation content navigator `doc` subcommand.

automation content navigator `doc` subcommand parameters settings

Parameter	Description	Setting options

help-doc	Help options for the `ansible-doc` command in `stdout` mode.	**Choices:** `True` or `False`
		Default: `False`
		CLI: `--hd` or `--help-doc`
		ENV: `ANSIBLE_NAVIGATOR_HELP_DOC`
		Settings file:
		`ansible-navigator:`
		`  help-doc:`

Parameter	Description	Setting options

plugin-name	Specify the plugin name.	**Default:** No default value set
		
		CLI: positional
		
		ENV: `ANSIBLE_NAVIGATOR_PLUGIN_NAME`
		
		Settings file:
		
		`ansible-navigator:`
		`  documentation:`
		`  plugin:`
		`  name:`

plugin-type	Specify the plugin type.	**Choices:** `become`, `cache`, `callback`, `cliconf`, `connection`, `httpapi`, `inventory`, `lookup`, `module`, `netconf`, `shell`, `strategy`, or `vars`
		
		Default: `module`
		
		CLI: `-t` or `----type`
		
		ENV: `ANSIBLE_NAVIGATOR_PLUGIN_TYPE`
		
		Settings file:
		
		`ansible-navigator:`
		`  documentation:`
		`  plugin:`
		`  type:`

Automation content navigator `inventory` subcommand settings

The following table describes each parameter and setting options for the automation content navigator `inventory` subcommand.

Automation content navigator `inventory` subcommand parameters settings

Parameter	Description	Setting options

help-inventory	Help options for the `ansible-inventory` command in `stdout` mode.	**Choices:** `True` or `False`
		Default: `False`
		CLI: `--hi` or `--help-inventory`
		ENV: `ANSIBLE_NAVIGATOR_INVENTORY_DOC`
		Settings file:
		`ansible-navigator:`
		`  help-inventory:`

inventory	Specify an inventory file path or comma separated host list.	**Default:** no default value set
		CLI: `--i` or `--inventory`
		ENV: `ANSIBLE_NAVIGATOR_INVENTORIES`
		Settings file:
		`ansible-navigator:`
		`  inventories:`

Parameter	Description	Setting options

inventory-column	Specify a host attribute to show in the inventory view.	**Default:** No default value set
		
		CLI: `--ic` or `--inventory-column`
		
		ENV:* `ANSIBLE_NAVIGATOR_INVENTORY_COLUMNS` **Settings file:**
		
		`ansible-navigator:`
		`  inventory-columns:`

Automation content navigator `replay` subcommand settings

The following table describes each parameter and setting options for the automation content navigator `replay` subcommand.

Automation content navigator `replay` subcommand parameters settings

Parameter	Description	Setting options
playbook-artifact-replay	Specify the path for the playbook artifact to replay.	**Default:** No default value set **CLI:** positional **ENV:** `ANSIBLE_NAVIGATOR_PLAYBOOK_ARTIFACT_REPLAY` **Settings file:**

Parameter	Description	Setting options

		`ansible-navigator:`
		`  playbook-artifact:`
		`  replay:`

Automation content navigator `run` subcommand settings

The following table describes each parameter and setting options for the automation content navigator `run` subcommand.

Automation content navigator `run` subcommand parameters settings

Parameter	Description	Setting options

playbook-artifact-replay	Specify the path for the playbook artifact to replay.	**Default:** No default value set
		CLI: positional
		ENV: `ANSIBLE_NAVIGATOR_PLAYBOOK_ARTIFACT_REPLAY`
		Settings file:
		`ansible-navigator:`
		`  playbook-artifact:`
		`  replay:`

help-playbook	Help options for the `ansible-playbook` command in `stdout` mode.	**Choices:** `True` or `False` **Default:** `False` **CLI:** `--hp` or `--help-playbook`

Parameter	Description	Setting options

		ENV: `ANSIBLE_NAVIGATOR_HELP_PLAYBOOK`
		
		Settings file:
		
		`ansible-navigator:`
		`  help-playbook:`

inventory	Specify an inventory file path or comma separated host list.	**Default:** no default value set
		
		CLI: `--i` or `--inventory`
		
		ENV: `ANSIBLE_NAVIGATOR_INVENTORIES`
		
		Settings file:
		
		`ansible-navigator:`
		`  inventories:`

inventory-column	Specify a host attribute to show in the inventory view.	**Default:** No default value set
		
		CLI: `--ic` or `--inventory-column`
		
		ENV:* `ANSIBLE_NAVIGATOR_INVENTORY_COLUMNS` **Settings file:**
		
		`ansible-navigator:`
		`  inventory-columns:`

playbook	Specify the playbook name.	**Default:** No default value set **CLI:** positional

Parameter	Description	Setting options

		ENV: `ANSIBLE_NAVIGATOR_PLAYBOOK`
		
		Settings file:*
		
		`ansible-navigator:`
		`  ansible:`
		`  playbook:`

playbook-artifact-enable	Enable or disable the creation of artifacts for completed playbooks. Note: not compatible with `--mode stdout` when playbooks require user input.	**Choices:** `True` or `False`
		
		Default: `True`
		
		CLI: `--pae` or `--playbook-artifact-enable` **ENV:** `ANSIBLE_NAVIGATOR_PLAYBOOK_ARTIFACT_ENABLE`
		Settings file:
		
		`ansible-navigator:`
		`  playbook-artifact:`
		`  enable:`

playbook-artifact-save-as	Specify the name for artifacts created from completed playbooks.	**Default:** `{playbook_dir}/{playbook_name}-artifact-{ts_utc}.json` **CLI:** `--pas` or `--playbook-artifact-save-as` **ENV:** `ANSIBLE_NAVIGATOR_PLAYBOOK_ARTIFACT_SAVE_AS` **Settings file:**

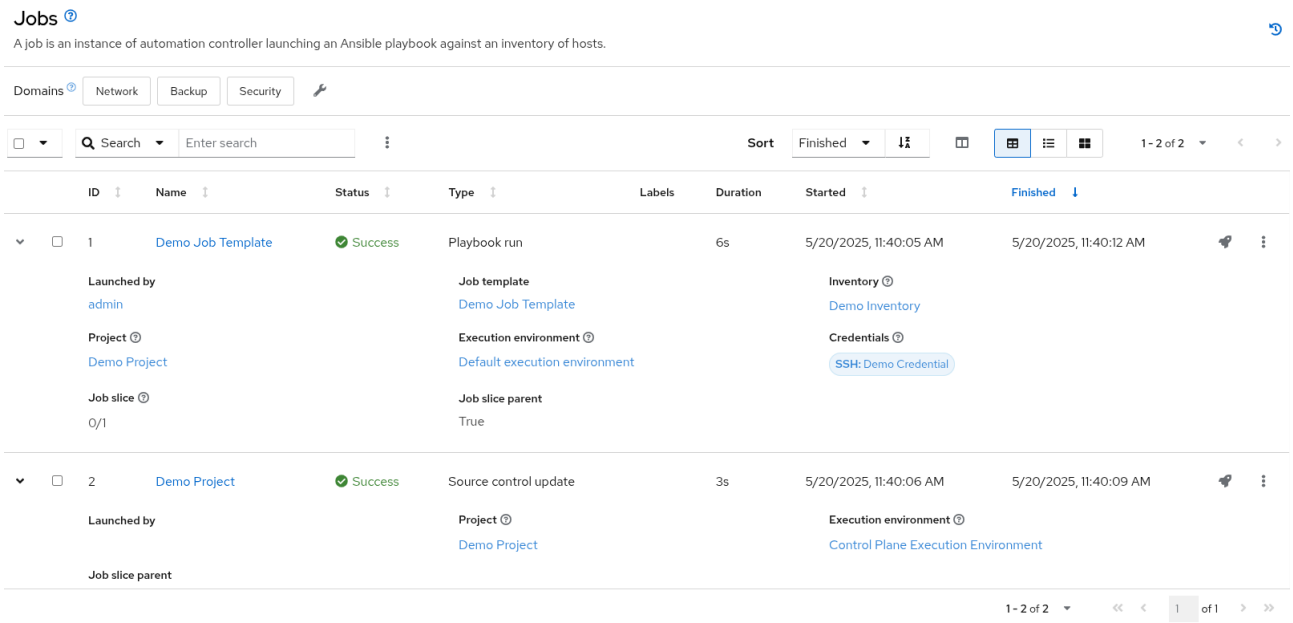
Parameter	Description	Setting options

		`ansible-navigator:`
		`  playbook-artifact:`
		`  save-as:`

Use jobs to run playbooks against an inventory of hosts

A job is an instance of automation controller launching an Ansible Playbook against an inventory of hosts.

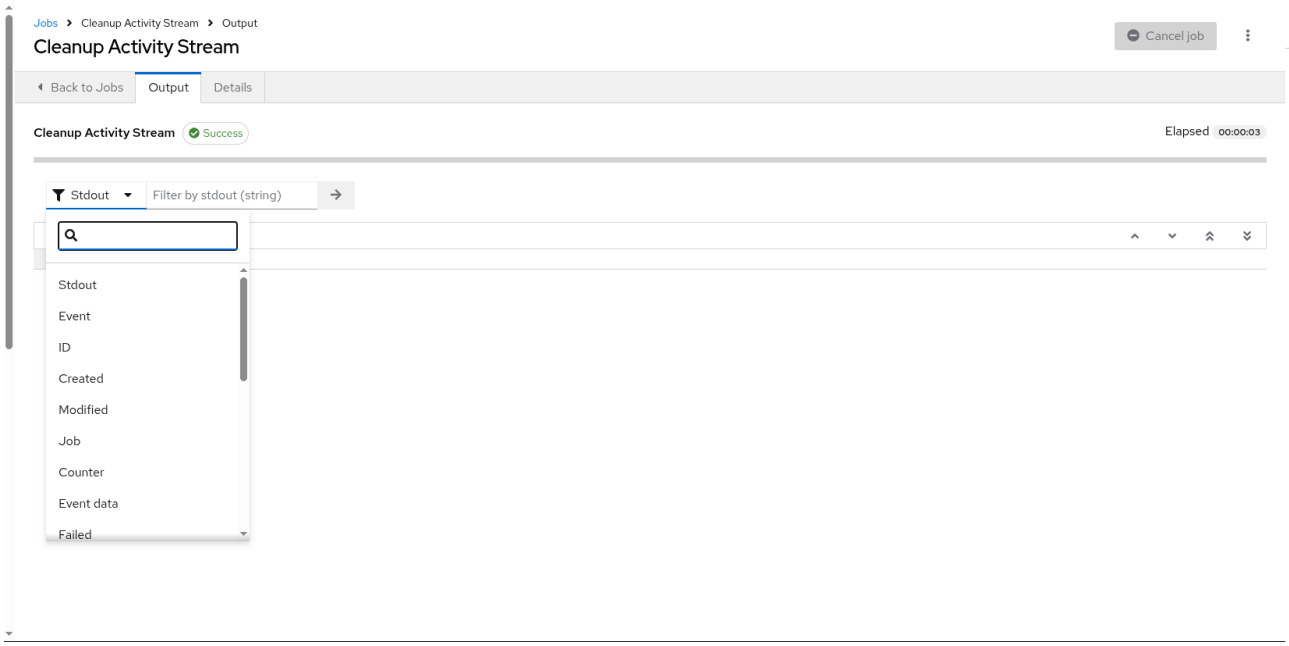
The **Jobs** list view displays a list of jobs and their statuses, shown as completed successfully, failed, or as an active (running) job. The default view is collapsed (Compact) with the job name, status, job type, start, and finish times. You can click the arrow > icon to expand and see more information. You can sort this list by various criteria, and perform a search to filter the jobs of interest.



From this screen you can complete the following tasks:

- In the **Domains** taskbar you can specify a domain to make relevant resources easily accessible. Click the icon to edit the existing labels or **Add Domain** to set up your own.
- View details and standard output of a particular job
- Relaunch jobs
- Cancel or delete selected jobs

The relaunch operation only applies to relaunches of playbook runs and does not apply to project or inventory updates, system jobs, and workflow jobs. When a job relaunches, the **Output** view is displayed. Selecting any type of job also takes you to the **Output** view for that job, where you can filter jobs by various criteria:



- The **Event** option in the **Search output** list enables you to filter by the events of interest, such as errors, host failures, host retries, and items skipped. You can include as many events in the filter as necessary.

Sync inventory data with external sources

Inventory synchronization jobs update automation controller data by pulling the latest information from configured sources. This ensures your inventory reflects the current state of the managed infrastructure.

Inventory sync jobs can be scheduled to run at regular intervals or triggered manually by users. These jobs gather data such as host details, group memberships, and variables from various sources such as cloud providers, dynamic inventory scripts, or static files.

When an inventory synchronization is executed, the results display in the **Output** tab.

If used, the Ansible CLI displays the same information. This can be useful for debugging. The `ANSIBLE_DISPLAY_ARGS_TO_STDOUT` parameter is set to `False` for all playbook runs. This parameter matches Ansible's default behavior and does not display task arguments in task headers in the Job **Details** interface to avoid leaking certain sensitive module parameters to `stdout`. To restore the earlier behavior, set `ANSIBLE_DISPLAY_ARGS_TO_STDOUT` to `True` through the `AWX_TASK_ENV` configuration setting.

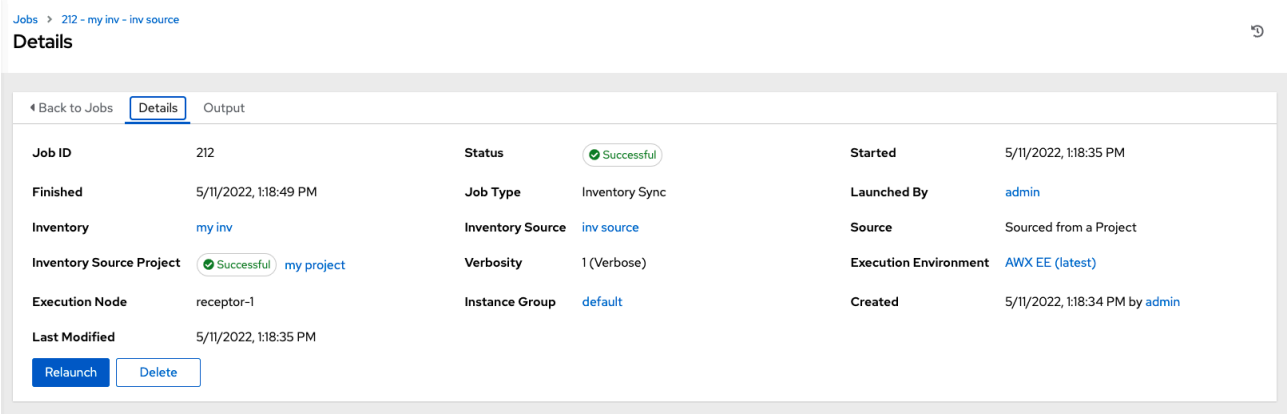
You can **Relaunch job**, **Cancel job**, download the job output, or delete the job.

NOTE:
You can perform an inventory update while a related job is running. In cases where you have a large project (around 10 GB), disk space on `/tmp` can be an issue.

Related information
Create dynamic groups with constructed inventories

Inventory sync details

Access the **Details** tab to view details about the job execution:



You can view the following details for an executed job:

- **Status**: It can be any of the following:
  - **Pending**: The inventory sync has been created, but not queued or started yet. Any job, not just inventory source syncs, stays in pending until it is ready to be run by the system. Reasons for inventory source syncs not being ready include:
    - Dependencies that are currently running (all dependencies must be completed before the next step can run).
    - Insufficient capacity to run in the locations it is configured for.
  - **Waiting**: The inventory sync is in the queue waiting to be executed.
  - **Running**: The inventory sync is currently in progress.
  - **Successful**: The inventory sync job succeeded.
  - **Failed**: The inventory sync job failed.
- **Inventory**: The name of the associated inventory group.
- **Source**: The type of cloud inventory.
- **Inventory Source Project**: The project used as the source of this inventory sync job.
- **Execution Environment**: The execution environment used.

- **Execution node**: The node used to run the job.
- **Instance Group**: The name of the instance group used with this job (automation controller is the default instance group).

Selecting these items enables you to view the corresponding job templates, projects, and other objects.

Sync inventory data with a source control management system

When you synchronize an inventory source that is configured to use a source control management (SCM) system, such as Git, automation controller creates and runs an SCM inventory job. This job pulls the latest inventory data from the SCM repository and updates the inventory in automation controller.

SCM inventory jobs function similarly to standard inventory source update jobs, but they specifically handle the interaction with the SCM system. These jobs ensure that the inventory data in automation controller remains up-to-date with the latest changes made in the SCM repository.

When an inventory sourced from an SCM, for example git, is executed, the results are displayed in the **Output** tab. If used, the Ansible CLI displays the same information. This can be useful for debugging.

Use the navigation menu to **Relaunch job**, **Cancel job**, download the job output, or delete the job.

SCM inventory details

To view details about the job execution and its associated project, select the **Details** tab.

You can view the following details for an executed job:

- **Status**: It can be any of the following:
  - **Pending**: The SCM job has been created, but not queued or started yet. Any job, not just SCM jobs, stay in pending until it is ready to be run by the system. Reasons for SCM jobs not being ready include dependencies that are currently running (all dependencies must be completed before the next step can run), or there is not enough capacity to run in the locations it is configured to.
  - **Waiting**: The SCM job is in the queue waiting to be executed.
  - **Running**: The SCM job is currently in progress.
  - **Successful**: The last SCM job succeeded.
  - **Failed**: The last SCM job failed.

- **Type**: SCM jobs display Source Control Update.
- **Project**: The name of the project.
- **Status**: Indicates whether the associated project was successfully updated.
- **Revision**: Indicates the revision number of the sourced project that was used in this job.
- **Execution environment**: Specifies the execution environment used to run this job.
- **Execution node**: Indicates the node on which the job ran.
- **Instance group**: Indicates the instance group on which the job ran, if specified.
- **Job tags**: Tags show the various job operations executed.

Select these items to view the corresponding job templates, projects, and other objects.

View output for your playbook job runs

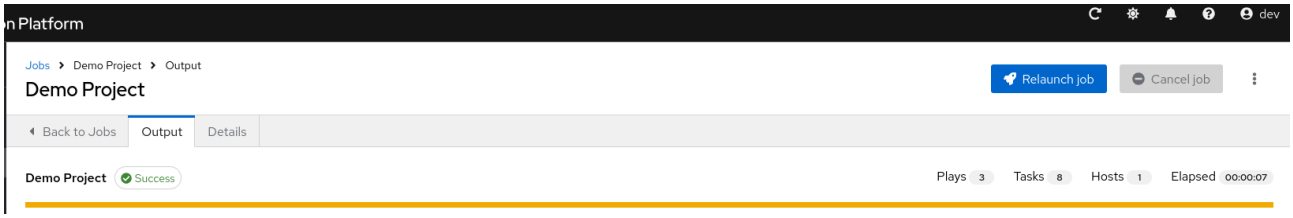
You can run playbook jobs to run Ansible playbooks on one or more managed nodes directly from the automation controller interface without creating a job template.

Use playbook run jobs to perform tasks that are more complex than those that can be accomplished with remote command execution. Any task that you can describe as an Ansible Playbook can be run on a host or group of hosts in your inventory. You can manage your systems quickly and easily. Because of an RBAC engine and detailed audit logging, you know which user has completed a specific task.

When a playbook is run, the results display in the **Output** tab. If used, the Ansible CLI displays the same information. This can be useful for debugging.

The events summary displays the following events that are run as part of this playbook:

- The number of times this playbook has run is shown in the **Plays** field
- The number of tasks associated with this playbook is shown in the **Tasks** field
- The number of hosts associated with this playbook is shown in the **Hosts** field
- The amount of time it took to complete the playbook run is shown in the **Elapsed** field



You can **Relaunch job**, **Cancel job**, download the job output, or delete the job.

Hover over a section of the host status bar in the **Output** view and the number of hosts associated with that status displays.

The output for a playbook job is also available after launching a job from the **Jobs** tab of its **Jobs Templates** page. View its host details by clicking the line item tasks in the output.

Search

Use **Search** to look up specific events, hostnames, and their statuses. To filter only certain hosts with a particular status, specify one of the following valid statuses:

ok

Indicates that a task completed successfully but no change was executed on the host.

changed

The playbook task executed. Since Ansible tasks should be written to be idempotent, tasks can exit successfully without executing anything on the host. In these cases, the task returns **ok**, but not **changed**.

failed

The task failed. Further playbook execution stopped for this host.

unreachable

The host is unreachable from the network or has another unrecoverable error associated with it.

skipped

The playbook task skipped because no change was necessary for the host to reach the target state.

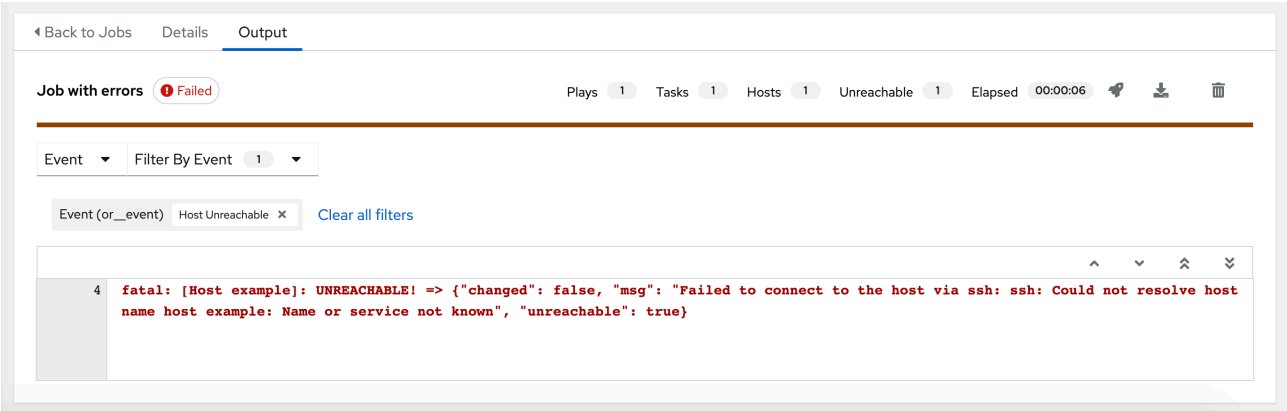
rescued

This shows the tasks that failed and then executes a rescue section.

ignored

This shows the tasks that failed and have `ignore_errors: yes configured`.

The following example shows a search with only unreachable hosts:



For more information about using the search, see the Search section.

The standard output view displays the events that occur on a particular job.

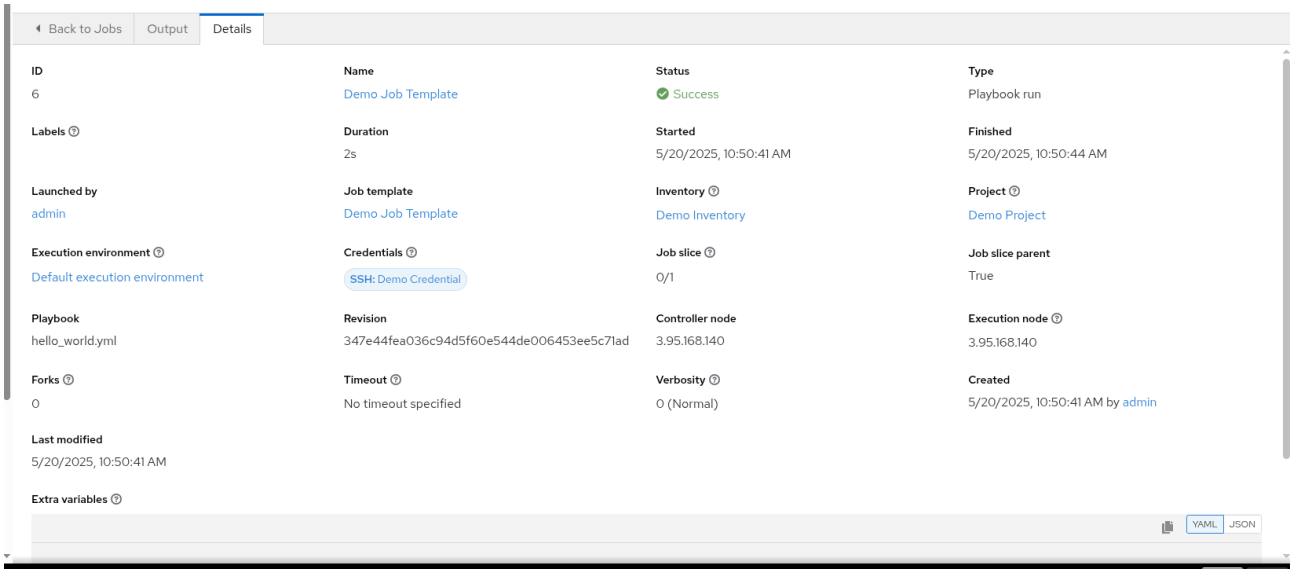
Click a line of an event from the **Stdout** pane and a **Host Events** window displays in a separate window. This window shows the host that was affected by that particular event.

NOTE:
Upgrading to the latest versions of Ansible Automation Platform involves progressively migrating all historical playbook output and events. This migration process is gradual, and happens automatically in the background after installation is complete. Installations with very large amounts of historical job output (tens or hundreds of GB of output) can have missing job output until migration is complete. The most recent data shows up at the top of the output, followed by older events.

Playbook run details

Learn how to view the details of a playbook run in Automation controller.

Access the **Details** tab to view details about the job execution:



You can view the following details for an executed job:

- **Status**: It can be any of the following:
  - **Pending**: The playbook run has been created, but not queued or started yet. Any job, not just playbook runs, stay in pending until it is ready to be run by the system. Reasons for playbook runs not being ready include dependencies that are currently running (all dependencies must be completed before the next step can run), or there is not enough capacity to run in the locations it is configured to.
  - **Waiting**: The playbook run is in the queue waiting to be executed.
  - **Running**: The playbook run is currently in progress.
  - **Successful**: The last playbook run succeeded.

    - **Failed**: The last playbook run failed.
- **Job template**: The name of the job template from which this job launched.
- **Inventory**: The inventory selected to run this job against.
- **Project**: The name of the project associated with the launched job.
- **Project Status**: The status of the project associated with the launched job.
- **Playbook**: The playbook used to launch this job.
- **Execution environment**: The name of the execution environment used in this job.
- **Credentials**: The credentials used in this job.
- **Extra variables**: Any extra variables passed when creating the job template are displayed here.

Select one of these items to view the corresponding job templates, projects, and other objects.

Playbook access and information sharing

Automation controller's use of automation execution environments and Linux containers prevents playbooks from reading files outside of their project directory.

By default, the only data exposed to the ansible-playbook process inside the container is the current project being used.

You can customize this in the Job Settings and expose additional directories from the host into the container.

Isolation functionality and variables

Automation controller uses container technology to isolate jobs from each other. By default, only the current project is exposed to the container running a job template.

If you need to expose additional directories, you must customize your playbook runs. To configure job isolation, you can set variables.

By default, automation controller uses the system's `tmp` directory (`/tmp` by default) as its staging area. You can change this in the **Job Execution Path** field of the **Jobs settings** page, or in the REST API at `/api/v2/settings/jobs` :

```
AWX_ISOLATION_BASE_PATH = "/opt/tmp"
```

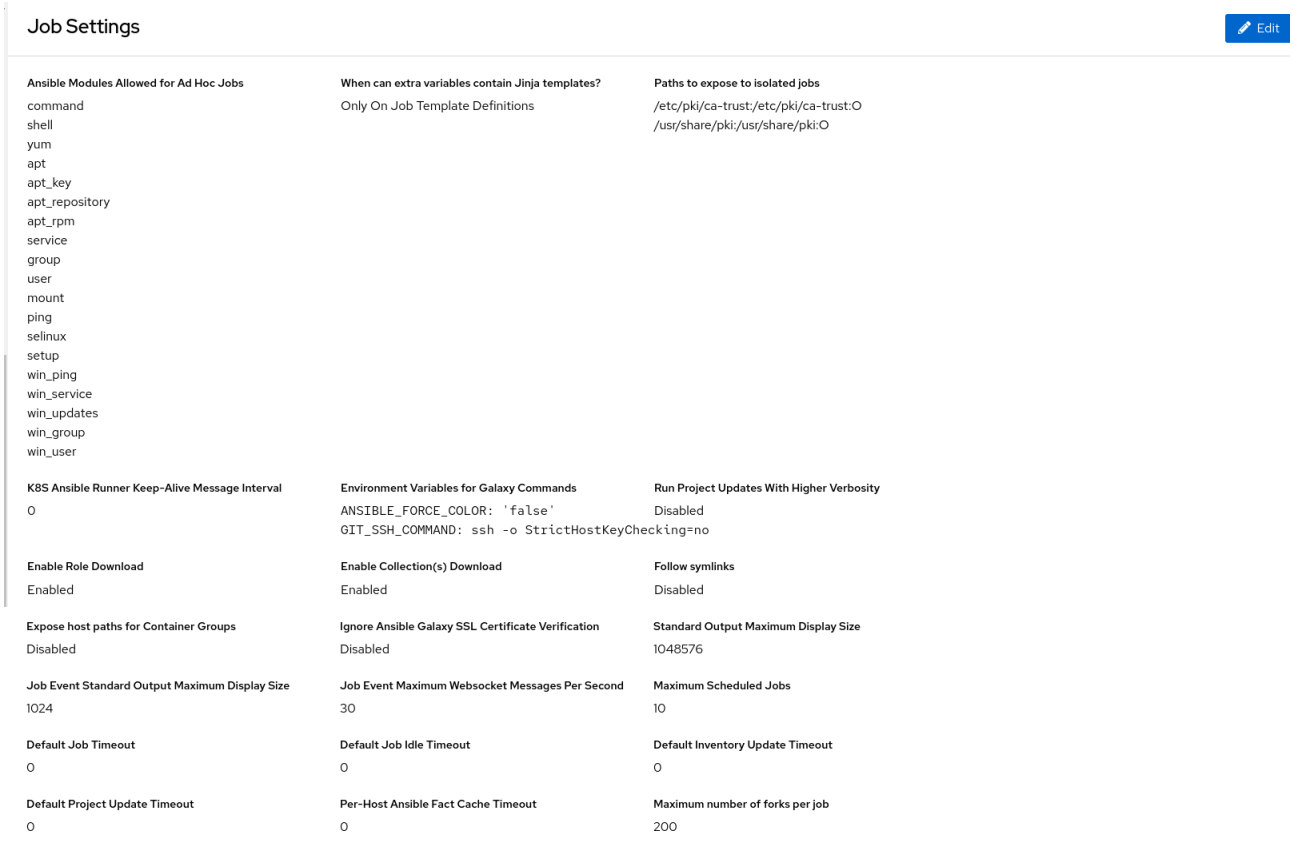
If there are any additional directories that should specifically be exposed from the host to the container that playbooks run in, you can specify those in the **Paths to expose to isolated jobs** field of the **Jobs Settings** page, or in the REST API at `/api/v2/settings/jobs`:

```
AWX_ISOLATION_SHOW_PATHS = ['/list/of/', '/paths']
```

NOTE:

- If a path to a specific file is entered, then the entire directory containing that file will be mounted inside the execution environment.
- If your playbooks need to use keys or settings defined in `AWX_ISOLATION_SHOW_PATHS`, then add this file to `/var/lib/awx/.ssh`.

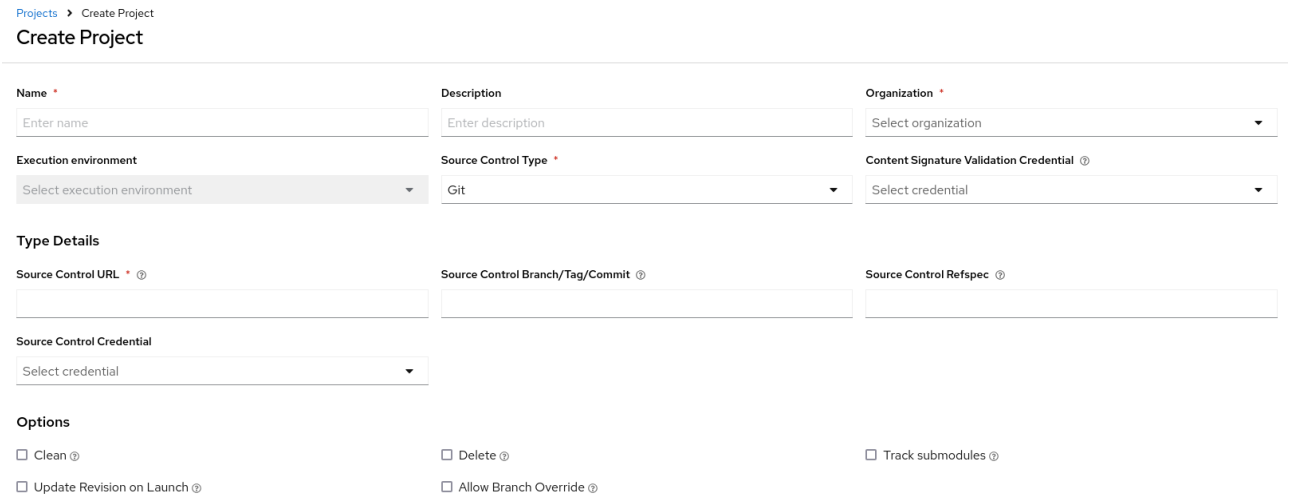
The fields described here can be found on the **Jobs settings** page:



Advanced configuration for jobs tied to source control management systems

In automation controller, you can configure projects to allow job templates to override the branch, tag, or reference used for source control.

Projects specify the branch, tag, or reference to use from source control in the `scm_branch` field. These are represented by the values specified in the **Type Details** fields:



When creating or editing a job you have the option to **Allow branch override**. When this option is checked, project administrators can delegate branch selection to the job templates that use that project, requiring only project `use_role`.

Source tree copy behavior

When automation controller runs a job, it creates a private copy of the project's source tree for that job run.

Every job run has its own private data directory. This directory contains a copy of the project source tree for the given `scm_branch` that the job is running. Jobs are free to make changes to the project folder and make use of those changes while it is still running. This folder is temporary and is removed at the end of the job run.

If you check the **Clean** option, modified files are removed in automation controller's local copy of the repository. This is done through use of the force parameter in its corresponding Ansible modules pertaining to git or sub-version.

Project revision behavior

Automation controller integrates with SCM systems like Git to manage project revisions. This ensures the correct version of your project files is tracked and used consistently during job execution.

During a project update, the revision of the default branch (specified in the **Source control branch** field of the project) is stored when updated. If providing a non-default **Source control branch** (not a commit hash or tag) in a job, the newest revision is pulled from the source control remote immediately before the job starts. This revision is shown in the **Source control revision** field of the job and its project update.

As a result, offline job runs are impossible for non-default branches. To ensure that a job is running a static version from source control, use tags or commit hashes. Project updates do not save all branches, only the project default branch.

The **Source control branch** field is not validated, so the project must update to assure it is valid. If this field is provided or prompted for, the **Playbook** field of job templates is not validated, and you have to launch the job template to verify presence of the expected playbook.

Git refspec

The **Source control refspec** field specifies the extra references the update should download from the remote. A refspec maps references from the remote repository to references in the local repository. If you leave this field blank, only the default references are fetched.

Examples include the following:

- `refs/:refs/remotes/origin/`: This fetches all references, including remotes of the remote
- `refs/pull/:refs/remotes/origin/pull/` (GitHub-specific): This fetches all refs for all pull requests
- `refs/pull/62/head:refs/remotes/origin/pull/62/head`: This fetches the ref for one GitHub pull request

For large projects, consider performance impact when using the first or second examples.

The **Source control refspec** parameter affects the availability of the project branch, and can enable access to references not otherwise available. Use the earlier examples to supply a pull request from the **Source control branch**, which is not possible without the **Source control refspec** field.

The Ansible git module fetches `refs/heads/` by default. This means that you can use a project's branches, tags and commit hashes, as the **Source control branch** if **Source control refspec** is blank. The value specified in the **Source control refspec** field affects which **Source control**

branch fields can be used as overrides. Project updates (of any type) perform an extra `git fetch` command to pull that refspec from the remote.

Example You can set up a project that enables branch override with the first or second refspec example. Use this in a job template that prompts for the **Source control branch**. A client can then start the job template when a new pull request is created, providing the branch `pull/N/head` and the job template can run against the provided GitHub pull request reference.

Standardize and streamline automation with automation job templates

You can create both Job templates and Workflow job templates from **Automation Execution > Templates**.

For Workflow job templates, see Workflow job templates.

A job template is a definition and set of parameters for running an Ansible job. Job templates are useful to run the same job many times. They also encourage the reuse of Ansible Playbook content and collaboration between teams.

Related information
Workflow job templates

Create repeatable, shareable job templates to standardize automation runs

A job template is a definition and set of parameters for running an Ansible job. Use job templates to launch jobs.

Procedure

1. From the navigation panel, select **Automation Execution > Templates**.
2. On the **Automation Templates** page, select **Create job template** from the **Create template** list.
3. Enter the appropriate details in the following fields:

NOTE:
If a field has the **Prompt on launch** checkbox selected, launching the job prompts you for the value for that field when launching.

Most prompted values override any values set in the job template.

Note the exceptions in the following table.

Field	Options	Prompt on Launch
Name	Enter a name for the job.	N/A
Description	Enter an arbitrary description as appropriate (optional).	N/A
Job type	Choose a job type: • Run: Start the playbook when launched, running Ansible tasks on the selected hosts. • Check: Perform a "dry run" of the playbook and report changes that would be made without actually making them. Tasks that do not support check mode are missed and do not report potential changes. For more information about job types see the Jobs in automation controller.	Yes
Inventory	Choose the inventory to use with this job template from the inventories available to the logged in user. A System Administrator must grant you or your team permissions to be able to use certain inventories in a job template.	Yes Inventory prompts show up as its own step in a later prompt window.
Project	Select the project to use with this job template	N/A

Field	Options	Prompt on Launch
	from the projects available to the user that is logged in.	
Source control branch	This field is only present if you chose a project that allows branch override. Specify the overriding branch to use in your job run. If left blank, the specified SCM branch (or commit hash or tag) from the project is used. For more information, see Job branch overriding.	Yes
Playbook	Choose the playbook to be launched with this job template from the available playbooks. This field automatically populates with the names of the playbooks found in the project base path for the selected project. Alternatively, you can enter the name of the playbook if it is not listed, such as the name of a file (such as `test.yml`) you want to use to run with that playbook. If you enter a filename that is not valid, the template displays an error, or causes the job to fail.	N/A
Execution Environment	Select the container image to be used to run this job. You must select a project before you can select an execution environment.	Yes Execution environment prompts show up as its own step in a later prompt window.

Field	Options	Prompt on Launch
Credentials	Select the icon to open a separate window. Choose the credential from the available options to use with this job template. Use the drop-down menu list to filter by credential type if the list is extensive. Some credential types are not listed because they do not apply to certain job templates.	• If selected, when launching a job template that has a default credential and supplying another credential replaces the default credential if it is the same type. The following is an example this message: `Job Template default credentials must be replaced with one of the same type. Please select a credential for the following types in order to proceed: Machine.` • You can add more credentials as you see fit. • Credential prompts show up as its own step in a later prompt window.
Labels	• Optionally supply labels that describe this job template, such as `dev` or `test`. • Use labels to group and filter job templates and completed jobs in the display. • Labels are created when they are added to the job	• If selected, even if a default value is supplied, you are prompted when launching to supply additional labels, if needed. • You cannot delete existing labels, selecting ✖ only removes the newly added labels, not

Field	Options	Prompt on Launch
	template. Labels are associated with a single Organization by using the Project that is provided in the job template. Members of the Organization can create labels on a job template if they have edit permissions (such as the admin role). • Once you save the job template, the labels appear in the **Job Templates** overview in the Expanded view. • Select ✖ beside a label to remove it. When a label is removed, it is no longer associated with that particular Job or Job Template, but it remains associated with any other jobs that reference it. • Jobs inherit labels from the Job Template at the time of launch. If you delete a label from a Job Template, it is also deleted from the Job.	existing default labels.
Forks	The number of parallel or simultaneous processes to use while executing	Yes

Field	Options	Prompt on Launch
	the playbook. A value of zero uses the Ansible default setting, which is five parallel processes unless overridden in `/etc/ansible/ansible.cfg`.	
Limit	A host pattern to further constrain the list of hosts managed or affected by the playbook. You can separate many patterns by colons (:). As with core Ansible: • a:b means "in group a or b" • a:b:&c means "in a or b but must be in c" • a:!b means "in a, and definitely not in b" If not selected, the job template executes against all nodes in the inventory or only the nodes predefined on the **Limit** field. When running as part of a workflow, the workflow job template limit is used instead.	
Verbosity	Control the level of output Ansible produces as the playbook executes. Choose the verbosity from Normal to various Verbose or Debug settings. This is only displayed in the **details** report view. Verbose logging includes the output of all	Yes

Field	Options	Prompt on Launch
	commands. Debug logging is exceedingly verbose and includes information about SSH operations that can be useful in certain support instances. Verbosity `5` causes automation controller to block heavily when jobs are running, which could delay reporting that the job has finished (even though it has) and can cause the browser tab to lock up.	
Job slicing	Specify the number of slices you want this job template to run. Each slice runs the same tasks against a part of the inventory. For more information about job slices, see Job Slicing.	Yes
Timeout	This enables you to specify the length of time (in seconds) that the job can run before it is canceled. Consider the following for setting the timeout value: • There is a global timeout defined in the settings which defaults to 0, indicating no timeout. • A negative timeout (<0) on a job template is a true "no timeout" on the job.	Yes

Field	Options	Prompt on Launch
	• A timeout of 0 on a job template defaults the job to the global timeout (which is no timeout by default). • A positive timeout sets the timeout for that job template.	
Show changes	Enables you to see the changes made by Ansible tasks.	Yes
Instance groups	Choose Instance and Container Groupsto associate with this job template. If the list is extensive, use the  icon to narrow the options. Job template instance groups contribute to the job scheduling criteria, see Job Runtime Behavior and Control where a job runs for rules. A System Administrator must grant you or your team permissions to be able to use an instance group in a job template. Use of a container group requires admin rights.	Yes If selected, you are providing the jobs preferred instance groups in order of preference. If the first group is out of capacity, later groups in the list are considered until one with capacity is available, at which point that is selected to run the job. • If you prompt for an instance group, what you enter replaces the normal instance group hierarchy and overrides all of the organizations' and inventories' instance groups. • The Instance Groups prompt shows up as its own step in a later prompt window.

Field	Options	Prompt on Launch
Job tags	Type and select the **Create** menu to specify which parts of the playbook should be executed.	Yes
Skip tags	Type and select the **Create** menu to specify certain tasks or parts of the playbook to skip.	Yes
Extra variables	• Pass extra command line variables to the playbook. This is the "-e" or "--extra-vars" command line parameter for ansible-playbook. • Give key or value pairs by using either YAML or JSON. These variables have a maximum value of precedence and overrides other variables specified elsewhere. The following is an example value: `git_branch: production release_version: 1.5`	Yes If you want to be able to specify `extra_vars` on a schedule, you must select **Prompt on launch** for Variables on the job template, or enable a survey on the job template. Those answered survey questions become `extra_vars`.

4. You can set the following options for launching this template, if necessary:

   - **Privilege escalation**: If checked, you enable this playbook to run as an administrator. This is the equal of passing the `--become` option to the `ansible-playbook` command.
   - **Provisioning callback**: If checked, you enable a host to call back to automation controller through the REST API and start a job from this job template. For more information, see Provisioning Callbacks.

- **Enable webhook**: If checked, you turn on the ability to interface with a predefined SCM system web service that is used to launch a job template. GitHub and GitLab are the supported SCM systems.
    - If you enable webhooks, other fields display, prompting for additional information:
    - **Webhook service**: Select which service to listen for webhooks from.
    - **Webhook URL**: Automatically populated with the URL for the webhook service to POST requests to.
    - **Webhook key**: Generated shared secret to be used by the webhook service to sign payloads sent to automation controller. You must configure this in the settings on the webhook service in order for automation controller to accept webhooks from this service.
    - **Webhook credential**: Optionally, give a GitHub or GitLab personal access token (PAT) as a credential to use to send status updates back to the webhook service.
      Before you can select it, the credential must exist.

      See Credential types to create one.
    - For additional information about setting up webhooks, see Working with Webhooks.
- **Concurrent jobs**: If checked, you are allowing jobs in the queue to run simultaneously if not dependent on one another. Check this box if you want to run job slices simultaneously. For more information, see Automation controller capacity determination and job impact.
- **Enable fact storage**: If checked, automation controller stores gathered facts for all hosts in an inventory related to the job running.
- **Prevent instance group fallback**: Check this option to allow only the instance groups listed in the **Instance Groups** field to run the job. If clear, all available instances in the execution pool are used based on the hierarchy described in Control where a job runs.

5. Click **Create job template**, when you have completed configuring the details of the job template.

   Creating the template does not exit the job template page but advances to the Job Template **Details** tab.

   After saving the template, you can click **Launch template** to start the job. You can also click **Edit** to add or change the attributes of the template, such as permissions, notifications, view completed jobs, and add a survey (if the job type is not a scan). You must first save the template before launching, otherwise, **Launch template** remains disabled.

Result

1. From the navigation panel, select **Automation Execution > Templates**.
2. Verify that the newly created template is displayed on the **Templates** page.

Automation templates

The **Automation Templates** page shows both **job templates** and **workflow job templates** that are currently available.

Automation Templates serve as a powerful blueprint for automating and orchestrating complex IT tasks.

Whether defined as a Job Template or Workflow Template, they standardize and streamline routine operations, enabling consistent execution across various environments.

By specifying playbooks, inventory, credentials, and other configuration details, an Automation Template eliminates manual intervention, reduces errors, and accelerates task completion.

It also provides flexibility by allowing the chaining of multiple tasks in a Workflow Template, supporting sophisticated automation use cases that can span across multiple systems and processes.

This ensures IT teams can reliably scale automation while maintaining high efficiency and control.

The default view is collapsed (Compact), showing the template name, template type, and the timestamp of the last job that ran using that template. You can click the icon next to each entry to expand and view more information. This list is sorted alphabetically by name, but you can sort by other criteria, or search by various fields and attributes of a template.

From this screen you can launch , edit , and duplicate a job template.

Select the template name to display more information about the template, including when it last ran.

This list is sorted alphabetically by name, but you can sort by other criteria, or search by various fields and attributes of a template.

NOTE:
Search functionality for Job templates is limited to alphanumeric characters only.

Workflow templates have the workflow visualizer icon as a shortcut for accessing the workflow editor.

NOTE:
You can use job templates to build a workflow template. Templates that show the **Workflow Visualizer** icon next to them are workflow templates. Clicking the icon allows you to build a workflow graphically. Many parameters in a job template enable you to select **Prompt on Launch** that you can change at the workflow level, and do not affect the values assigned at the job template level. For instructions, see the Workflow Visualizer section.

Add permissions to templates

You can add permissions to a template to allow specific users or teams to have access to it.

Use the following steps to add permissions for the team.

Procedure

1. From the navigation panel, select **Automation Execution > Templates**.
2. Select a template, and in the **Team Access** or **User Access** tab, click **Add roles**.
3. Select **Teams** or **Users** and click **Next**.
    - Select one or more users or teams from the list by clicking the check boxes next to the names to add them as members and click **Next**.
4. Choose the roles that you want users or teams to have. Ensure that you scroll down for a complete list of roles. Each resource has different options available.
5. Click **Finish** to apply the roles to the selected users or teams and to add them as members.

   The window to add users and teams closes to display the updated roles assigned for each user and team
6. To remove roles for a particular user, click the ✖ icon next to its resource.

   This launches a confirmation dialog, asking you to confirm the disassociation.

Set your domains of interest

With Domain filtering lets you customize content in Automation Execution's **Jobs** and **Templates** sections. Jobs and templates are linked to labels; selecting one filters out less-relevant items, providing easy access to resources tailored to your specific area of interest.

Procedure

1. From the navigation panel, select **Automation Execution > Jobs** or **Automation Execution > Templates**.
2. Beneath the page heading, next to **Domains**, is a list of topic-related labels. Select a label to filter jobs and job templates so that only content related to the labels is shown. You can choose more than one label.
3. To clear your selection, click the **X**.
4. To customize your domain options, select the 🔧 icon. In the modal that appears, select **Add Domain** to add new domains to filter with.

Next steps

You can add labels to your individual job templates to make the templates appear as resources when you filter with the related domain label. Go to **Automation Execution > Templates**, select your job template, and click **Edit template**. On the editing screen, enter the label you want to use in the **Labels** field and click **Save job template**.

Schedule job templates

You can schedule job templates in automation controller to run at specific times or intervals.

Access the schedules for a particular job template from the **Schedules** tab.

- To schedule a job template, select the **Schedules** tab from the job template, and select the appropriate method:
    - If schedules are already set up, review, edit, enable or disable your schedule preferences.
    - If schedules have not been set up, see Schedules for more information.
- If you select **Prompt on Launch** for the **Credentials field**, and you create or edit scheduling information for your job template, a **Prompt** option displays on the Schedules form.

  You cannot remove the default machine credential in the **Prompt** dialog without replacing it with another machine credential before you can save it.

> **NOTE:**
> To set `extra_vars` on schedules, you must select **Prompt on Launch** for **Variables** on the job template, or configure and enable a survey on the job template.
>
> The answered survey questions then become `extra_vars`.

Set extra variables in job templates

Along with any extra variables set in the job template and survey, automation controller automatically adds the following variables to the job environment.

> **NOTE:**
> - `awx_*` variables are defined by the system and cannot be overridden.
> - Variables about the job context, such as `awx_job_template_name` are not affected if they are set in `extra_vars`.

- `awx_job_id`: The job ID for this job run.
- `awx_job_launch_type`: The description to indicate how the job was started:
  - **manual**: The job was started manually by a user.
  - **relaunch**: The job was started via relaunch.
  - **callback**: The job was started via host callback.
  - **scheduled**: The job was started from a schedule.
  - **dependency**: The job was started as a dependency of another job.
  - **workflow**: The job was started from a workflow job.
  - **sync**: The job was started from a project sync.
  - **scm**: The job was created as an Inventory SCM sync.
- `awx_job_template_id`: The job template ID that this job run uses.
- `awx_job_template_name`: The job template name that this job uses.
- `awx_execution_node`: The Execution Node name that launched this job.
- `awx_project_revision`: The revision identifier for the source tree that this particular job uses (it is also the same as the job's field scm_revision).
- `awx_project_scm_branch`: The configured default project SCM branch for the project the job template uses.
- `awx_job_scm_branch`: If the SCM Branch is overwritten by the job, the value is shown here.
- `awx_user_email`: The user email of the controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_user_first_name`: The user's first name of the automation controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_user_id`: The user ID of the automation controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_user_last_name`: The last name of the automation controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_user_name`: The user name of the automation controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_schedule_id`: If applicable, the ID of the schedule that launched this job.
- `awx_schedule_name`: If applicable, the name of the schedule that launched this job.
- `awx_workflow_job_id`: If applicable, the ID of the workflow job that launched this job.
- `awx_workflow_job_name`: If applicable, the name of the workflow job that launched this job. Note this is also the same as the workflow job template.
- `awx_inventory_id`: If applicable, the ID of the inventory this job uses.

- `awx_inventory_name`: If applicable, the name of the inventory this job uses.

For compatibility, all variables are also given an "awx" prefix, for example, `awx_job_id`.

Create a survey

You can create a survey for a job template to prompt users for input when they launch a job based on that template. Surveys can include many questions of various types, such as text input, multiple choice, and passwords.

The responses provided by users are stored in Ansible variables that can be used within the playbook associated with the job template.

Procedure

1. From the navigation panel, select **Automation Execution > Templates**.
2. Select the job template you want to create a survey for.
3. From the **Survey** tab, click **Create survey question**.
4. A survey can consist of any number of questions. For each question, enter the following information:
    - **Question**: The question to ask the user.
    - Optional: **Description**: A description of what is being asked of the user.
    - **Answer variable name**: The Ansible variable name to store the user's response in. This is the variable to be used by the playbook. Variable names cannot contain spaces.
    - **Answer type**: Choose from the following question types:
        - **Text**: A single line of text. You can set the minimum and maximum length (in characters) for this answer.
        - **Textarea**: A multi-line text field. You can set the minimum and maximum length (in characters) for this answer.
        - **Password**: Responses are treated as sensitive information, similar to the way an actual password is treated. You can set the minimum and maximum length (in characters) for this answer.
        - **Multiple Choice (single select)**: A list of options, of which only one can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** field.
        - **Multiple Choice (multiple select)**: A list of options, any number of which can be selected at a time. Enter the options, one per line, in the **Multiple Choice Options** field.
        - **Integer**: An integer number. You can set the minimum and maximum length (in characters) for this answer.

    - **Float**: A decimal number. You can set the minimum and maximum length (in characters) for this answer.
- **Required**: Whether or not an answer to this question is required from the user.
- **Minimum length** and **Maximum length**: Specify if a certain length in the answer is required.
- **Default answer**: The default answer to the question. This value is pre-filled in the interface and is used if the answer is not provided by the user.

5. Once you have entered the question information, click **Create question** to add the question.

   The survey question displays in the **Survey** list. For any question, you can click to edit it.

   Check the box next to each question and click **Delete** to delete the question, or use the toggle option in the menu bar to enable or disable the survey prompts.

   If you have more than one survey question, click **Edit Order** to rearrange the order of the questions by clicking and dragging on the grid icon.
6. To add more questions, click **Add**.

Optional survey questions

The **Required** setting on a survey question determines whether the answer is optional or not for the user interacting with it.

Optional survey variables can also be passed to the playbook in `extra_vars`.

- If a non-text variable (input type) is marked as optional, and is not filled in, no survey `extra_var` is passed to the playbook.
- If a text input or text area input is marked as optional, is not filled in, and has a minimum `length > 0`, no survey `extra_var` is passed to the playbook.
- If a text input or text area input is marked as optional, is not filled in, and has a minimum `length === 0`, that survey `extra_var` is passed to the playbook, with the value set to an empty string ("").

Extra variables

You can pass extra variables to a automation controller job template in several ways, including through surveys and the API.

When you pass survey variables, they are passed as extra variables (`extra_vars`) within automation controller. However, passing extra variables to a job template (as you would do with a survey) can override other variables being passed from the inventory and project.

By default, `extra_vars` are marked as `!unsafe` unless you specify them on the Job Template's Extra Variables section. These are trusted, because they can only be added by users with enough privileges to add or edit a Job Template. For example, nested variables do not expand when entered as a prompt, as the Jinja brackets are treated as a string.

NOTE:
`extra_vars` passed to the job launch API are only honored if one of the following is true:

- They correspond to variables in an enabled survey.
- `ask_variables_on_launch` is set to **True**.

Example You have a defined variable for an inventory for `debug = true` . It is possible that this variable, `debug = true` , can be overridden in a job template survey.

To ensure the variables that you pass are not overridden, ensure they are included by redefining them in the survey. You can define extra variables at the inventory, group, and host levels.

If you are specifying the `ALLOW_JINJA_IN_EXTRA_VARS` parameter, which controls whether Jinja templating is allowed in extra variables for job templates in automation controller, you can configure the parameter in the UI Jobs Settings.

- Data: Setting ALLOW_JINJA_IN_EXTRA_VARS = template only works for saved job template extra variables. Prompted variables and survey variables are excluded from the 'template'.
- Data: This parameter has three values: Only on Template Definitions (default); Never (recommended); Always (strongly discouraged)

Setting The job template extra variables dictionary is merged with the survey variables.
The following are some simplified examples of `extra_vars` in YAML and JSON formats:

- The configuration in YAML format:

```
launch_to_orbit: true

satellites:

  - sputnik

  - explorer

  - satcom
```

- The configuration in JSON format:

```
{
  "launch_to_orbit": true,
  "satellites": ["sputnik", "explorer", "satcom"]
}
```

The following table notes the behavior (hierarchy) of variable precedence in automation controller as it compares to variable precedence in Ansible.

Automation controller Variable Precedence Hierarchy (last listed wins)

Ansible	automation controller
role defaults	role defaults
dynamic inventory variables	dynamic inventory variables
inventory variables	automation controller inventory variables
inventory `group_vars`	automation controller group variables
inventory `host_vars`	automation controller host variables
playbook `group_vars`	playbook `group_vars`
playbook `host_vars`	playbook `host_vars`
host facts	host facts
registered variables	registered variables
set facts	set facts
play variables	play variables
play `vars_prompt`	(not supported)
play `vars_files`	play `vars_files`
role and include variables	role and include variables
block variables	block variables
task variables	task variables
extra variables	Job Template extra variables
	Job Template Survey (defaults)
	Job Launch extra variables

Launch a job template

You can configure a job template to store all the parameters that you would normally pass to the Ansible Playbook on the command line. In addition to the playbooks, the template passes the inventory, credentials, extra variables, and all options and settings that you can specify on the command line.

Easier deployments drive consistency, by running your playbooks the same way each time, and allowing you to delegate responsibilities.

- Launch a job template by using one of these methods:
    - From the navigation panel, select **Automation Execution > Templates** and click **Launch template** on the job template card.
    - In the job template **Details** tab of the job template you want to launch, click **Launch template**.

Additional information requested when launching a job template

A job can require additional information to run. The following data can be requested at launch:

- Credentials that were setup
- The option **Prompt on Launch** is selected for any parameter
- Passwords or passphrases that have been set to **Ask**
- A survey, if one has been configured for the job templates
- Extra variables, if requested by the job template

NOTE:
If a job has user-provided values, then those are respected upon relaunch. If the user did not specify a value, then the job uses the default value from the job template. Jobs are not relaunched as-is. They are relaunched with the user prompts re-applied to the job template.

If you give values on one tab, return to a previous tab, continuing to the next tab results in having to re-provide values on the rest of the tabs. Ensure that you complete the tabs in the order that the prompts appear.

When launching, automation controller automatically redirects the web browser to the **Job Status** page for this job under the **Jobs** tab.

You can re-launch the most recent job from the list view to re-run on all hosts or just failed hosts in the specified inventory. For more information, see the Jobs in automation controller section.

When slice jobs are running, job lists display the workflow and job slices, and a link to view their details individually.

NOTE:
You can launch jobs in bulk by using the newly added endpoint in the API, `/api/v2/bulk/job_launch`. This endpoint accepts JSON and you can specify a list of unified job templates (such as job templates and project updates) to launch. The user must have the appropriate permission to launch all the jobs. If all jobs are not launched an error is returned indicating why the operation was not able to complete. Use the `OPTIONS` request to return relevant schema. For more information, see See "Use the REST API to browse, query, filter, and authenticate" on page 1 264

Set extra variables in job templates

Along with any extra variables set in the job template and survey, automation controller automatically adds the following variables to the job environment.

NOTE:

- `awx_*` variables are defined by the system and cannot be overridden.
- Variables about the job context, such as `awx_job_template_name` are not affected if they are set in `extra_vars`.

- `awx_job_id`: The job ID for this job run.
- `awx_job_launch_type`: The description to indicate how the job was started:
  - **manual**: The job was started manually by a user.
  - **relaunch**: The job was started via relaunch.
  - **callback**: The job was started via host callback.
  - **scheduled**: The job was started from a schedule.
  - **dependency**: The job was started as a dependency of another job.
  - **workflow**: The job was started from a workflow job.
  - **sync**: The job was started from a project sync.
  - **scm**: The job was created as an Inventory SCM sync.
- `awx_job_template_id`: The job template ID that this job run uses.

- `awx_job_template_name` : The job template name that this job uses.
- `awx_execution_node` : The Execution Node name that launched this job.
- `awx_project_revision` : The revision identifier for the source tree that this particular job uses (it is also the same as the job's field scm_revision).
- `awx_project_scm_branch` : The configured default project SCM branch for the project the job template uses.
- `awx_job_scm_branch` : If the SCM Branch is overwritten by the job, the value is shown here.
- `awx_user_email` : The user email of the controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_user_first_name` : The user's first name of the automation controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_user_id` : The user ID of the automation controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_user_last_name` : The last name of the automation controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_user_name` : The user name of the automation controller user that started this job. This is not available for callback or scheduled jobs.
- `awx_schedule_id` : If applicable, the ID of the schedule that launched this job.
- `awx_schedule_name` : If applicable, the name of the schedule that launched this job.
- `awx_workflow_job_id` : If applicable, the ID of the workflow job that launched this job.
- `awx_workflow_job_name` : If applicable, the name of the workflow job that launched this job. Note this is also the same as the workflow job template.
- `awx_inventory_id` : If applicable, the ID of the inventory this job uses.
- `awx_inventory_name` : If applicable, the name of the inventory this job uses.

For compatibility, all variables are also given an "awx" prefix, for example, `awx_job_id` .

Relaunch a job template

Relaunching a job template creates a new job based on a previous job.

Instead of manually relaunching a job, a relaunch is denoted by setting `launch_type` to `relaunch` . The relaunch behavior deviates from the launch behavior in that it does not inherit `extra_vars` .

Job relaunching does not go through the inherit logic. It uses the same `extra_vars` that were calculated for the job being relaunched.

Example

You launch a job template with no `extra_vars` which results in the creation of a job called **j1**. Then you edit the job template and add `extra_vars` (such as adding `"{ "hello": "world" }"`).

Relaunching **j1** results in the creation of **j2**, but because there is no inherit logic and **j1** has no extra_vars, **j2** does not have any `extra_vars`.

If you launch the job template with the `extra_vars` that you added after the creation of **j1**, the relaunch job created (**j3**) includes the extra_vars. Relaunching **j3** results in the creation of **j4**, which also includes `extra_vars`.

Delete a job template

You can delete job templates in automation controller when they are no longer needed.

Before deleting a job template, ensure that it is not used in a workflow job template.

Procedure

1. Delete a job template using the following method:
   - Click the ⋮ icon and select the Delete Template icon, or
   - Select the required job template, on the **Details** page click the ⋮ icon and select **Delete template**.

> **NOTE:**
> If deleting items that are used by other work items, a message opens listing the items that are affected by the deletion and prompts you to confirm the deletion. Some screens contain items that are invalid or previously deleted, and will fail to run.

Store host facts in cache for faster playbook execution

Automation controller can store and retrieve facts on a per-host basis through an Ansible Fact Cache plugin. This behavior is configurable on a per-job template basis.

Fact caching is turned off by default but can be enabled to serve fact requests for all hosts in an inventory related to the job running. This enables you to use job templates with `--limit` while still having access to the entire inventory of host facts. You can specify a global timeout setting that the plugin enforces per-host, (in seconds) from the navigation panel, select **Settings › Automation Execution › Job** and edit the **Per-Host Ansible Fact Cache Timeout** field.

After launching a job that uses fact cache (`use_fact_cache=True`), each host's `ansible_facts` are all stored by the controller in the job's inventory.

The Ansible Fact Cache plugin that includes automation controller is enabled on jobs with fact cache enabled (`use_fact_cache=True`).

When a job that has fact cache enabled (`use_fact_cache=True`) has run, automation controller restores all records for the hosts in the inventory. Any records with update times newer than the currently stored facts per-host are updated in the database.

New and changed facts are logged through automation controller's logging facility. Specifically, to the `system_tracking namespace` or logger. The logging payload includes the following fields:

- `host_name`
- `inventory_id`
- `ansible_facts`

`ansible facts` is a dictionary of all Ansible facts for `host_name` in the automation controller inventory, `inventory_id` .

NOTE:
If a hostname includes a forward slash (/), fact cache does not work for that host. If you have an inventory with 100 hosts and one host has a / in the name, the remaining 99 hosts still collect facts.

Benefits of fact caching

Fact caching saves you time over running fact gathering. If you have a playbook in a job that runs against a thousand hosts and forks, it can take 10 minutes to gather facts across all of those hosts.

If you run a job on a regular basis, the first run of it caches these facts and the next run pulls them from the database. This reduces the runtime of jobs against large inventories.

NOTE:
Do not change the ansible.cfg file to apply fact caching. Custom fact caching could conflict with the controller's fact caching feature. You must use the fact caching module that includes automation controller.

You can select to use cached facts in your job by checking the **Enable fact storage** option when you create or edit a job template.

The following is an example playbook that uses the Ansible `clear_facts` meta task.

```
- hosts: all
  gather_facts: false
  tasks:
    - name: Clear gathered facts from all currently targeted hosts
      meta: clear_facts
```

You can find the API endpoint for fact caching at:

http://<controller server name>/api/v2/hosts/x/ansible_facts

Associate cloud credentials with a job template

Automation controller can use Cloud Credentials to authenticate to cloud providers.

Cloud Credentials can be used when syncing a cloud inventory. They can also be associated with a job template and included in the runtime environment for use by a playbook. The following Cloud Credentials are supported:

- Openstack
- Amazon Web Services
- Google
- Azure
- VMware

Related information
OpenStack
Amazon Web Services
Google
Azure
VMware

OpenStack

Use this credential type to connect to OpenStack clouds. Automation controller uses the OpenStack SDK to interact with OpenStack clouds. When you create an OpenStack cloud credential, the controller prompts you for the following information:

- **Username**: The username to authenticate to the OpenStack cloud.

- **Password**: The password to authenticate to the OpenStack cloud.
- **Project name**: The project name (also called tenant name) to use when connecting to the OpenStack cloud.
- **Auth URL**: The authentication URL for the OpenStack cloud.
- **Cloud name**: The name of the cloud as defined in your OpenStack clouds.yaml file.
- **Region name** (optional): The region name to use when connecting to the OpenStack cloud.
- **Domain name** (optional): The domain name to use when connecting to the OpenStack cloud.
- **Project domain name** (optional): The project domain name to use when connecting to the OpenStack cloud.
- **Validate SSL certificate**: Select this option to validate the SSL/TLS certificate presented by the OpenStack cloud. Clear this option to disable SSL/TLS certificate validation.

The following sample playbook invokes the `nova_compute` Ansible OpenStack cloud module and requires credentials:

- `auth_url`
- `username`
- `password`
- `project name`

These fields are made available to the playbook through the environmental variable `OS_CLIENT_CONFIG_FILE`, which points to a YAML file written by the controller based on the contents of the cloud credential. The following sample playbooks load the YAML file into the Ansible variable space:

- OS_CLIENT_CONFIG_FILE example:

```
clouds:
  devstack:
    auth:
      auth_url: http://devstack.yoursite.com:5000/v2.0/
      username: admin
      password: your_password_here
      project_name: demo
```

- Playbook example:

```
- hosts: all
  gather_facts: false
  vars:
    config_file: "{{ lookup('env', 'OS_CLIENT_CONFIG_FILE') }}"
    nova_tenant_name: demo
    nova_image_name: "cirros-0.3.2-x86_64-uec"
    nova_instance_name: autobot
    nova_instance_state: 'present'
    nova_flavor_name: m1.nano



    nova_group:
      group_name: antarctica
      instance_name: deceptacon
      instance_count: 3
  tasks:
    - debug: msg="{{ config_file }}"
    - stat: path="{{ config_file }}"
      register: st
    - include_vars: "{{ config_file }}"
      when: st.stat.exists and st.stat.isreg



    - name: "Print out clouds variable"
      debug: msg="{{ clouds|default('No clouds found') }}"



    - name: "Setting nova instance state to: {{ nova_instance_state }}"
      local_action:
        module: nova_compute
        login_username: "{{ clouds.devstack.auth.username }}"
        login_password: "{{ clouds.devstack.auth.password }}"
```

Amazon Web Services

Amazon Web Services (AWS) cloud credentials are exposed as the following environment variables during playbook execution (in the job template, choose the cloud credential needed for your setup):

- `AWS_ACCESS_KEY_ID`
- `AWS-SECRET_ACCESS_KEY`

Each AWS module implicitly uses these credentials when run through the controller without having to set the `aws_access_key_id` or `aws_secret_access_key` module options.

Google

Use this credential type to authenticate to Google Cloud Platform services. Automation controller supports Google Cloud credentials for use with Ansible modules that manage Google Cloud resources.

Google cloud credentials are exposed as the following environment variables during playbook execution (in the job template, choose the cloud credential needed for your setup):

- `GCE_EMAIL`
- `GCE_PROJECT`
- `GCE_CREDENTIALS_FILE_PATH`

Each Google module implicitly uses these credentials when run through the controller without having to set the `service_account_email`, `project_id`, or `pem_file` module options.

Azure

Automation controller includes built-in support for managing Microsoft Azure cloud resources.

Azure cloud credentials exist as the following environment variables during playbook execution (in the job template, select the cloud credential needed for your setup):

- `AZURE_SUBSCRIPTION_ID`
- `AZURE_CERT_PATH`

Each Azure module uses these credentials when run using automation controller without having to set the `subscription_id` or `management_cert_path` module options.

VMware

Automation controller integrates with VMware vSphere to manage virtual machines (VMs) as part of its infrastructure. This integration allows users to automate the provisioning, management, and decommissioning of VMs within a VMware environment.

VMware cloud credentials are exposed as the following environment variables during playbook execution (in the job template, choose the cloud credential needed for your setup):

- `VMWARE_USER`
- `VMWARE_PASSWORD`
- `VMWARE_HOST`

The following sample playbook demonstrates the usage of these credentials:

```
- vsphere_guest:
    vcenter_hostname: "{{ lookup('env', 'VMWARE_HOST') }}"
    username: "{{ lookup('env', 'VMWARE_USER') }}"
    password: "{{ lookup('env', 'VMWARE_PASSWORD') }}"
    guest: newvm001
    from_template: yes
    template_src: linuxTemplate
    cluster: MainCluster
    resource_pool: "/Resources"
    vm_extra_config:
      folder: MyFolder
```

Increase capacity through cloud bursting by provisioning callbacks

Provisioning Callbacks are a feature of automation controller that enable a host to start a playbook run against itself, rather than waiting for a user to launch a job to manage the host from the automation controller console.

Provisioning Callbacks are only used to run playbooks on the calling host and are meant for cloud bursting. Cloud bursting is a cloud computing configuration that enables a private cloud to access public cloud resources by "bursting" into a public cloud when computing demand spikes.

Example

New instances with a need for client to server communication for configuration, such as transmitting an authorization key, not to run a job against another host. This provides for automatically configuring the following:

- A system after it has been provisioned by another system (such as AWS auto-scaling, or an operating system provisioning system such as Kickstart or preseed).
- Launching a job programmatically without invoking the automation controller API directly.

The job template launched only runs against the host requesting the provisioning.

This is often accessed with a firstboot type script or from `cron`.

Enable Provisioning Callbacks

Use the following procedure to enable provisioning callbacks for a job template.

- To enable callbacks, check the **Provisioning callback** option in the job template. This displays **Provisioning callback details** for the job template.

> **NOTE:**
> If you intend to use automation controller's provisioning callback feature with a dynamic inventory, set **Update on Launch** for the inventory group used in the job template.

Callbacks also require a host config key, to ensure that foreign hosts with the URL cannot request configuration. Give a custom value for the **Host config key**. The host key can be reused across many hosts to apply this job template against multiple hosts. If you want to control what hosts are able to request configuration, you can change the key can at any time.

Use REST manually to callback

You can use the REST API to trigger automation controller callbacks manually.

Procedure

1. Examine the callback URL in the UI, in the form: https://<CONTROLLER_SERVER_NAME>/api/v2/job_templates/7/callback/
    - The "7" in the sample URL is the job template ID in automation controller.
2. Ensure that the request from the host is a POST. The following is an example using `curl` (all on a single line):

```
curl -k -i -H 'Content-Type:application/json' -XPOST -d '{"host_config_key":
"redhat"}' \

                  https://<CONTROLLER_SERVER_NAME>/api/v2/job_templates/7/
callback/
```

3. Ensure that the requesting host is defined in your inventory for the callback to succeed.

Result

Successful requests result in an entry on the **Jobs** tab, where you can view the results and history. You can access the callback by using REST, but the suggested method of using the callback is to use one of the example scripts that includes automation controller:

- `/usr/share/awx/request_tower_configuration.sh` (Linux/UNIX)
- `/usr/share/awx/request_tower_configuration.ps1` (Windows)

Their usage is described in the source code of the file by passing the `-h` flag, as the following shows:

```
./request_tower_configuration.sh -h
Usage: ./request_tower_configuration.sh <options>



Request server configuration from Ansible Tower.



OPTIONS:
 -h      Show this message
 -s      Controller server (e.g. https://ac.example.com) (required)
 -k      Allow insecure SSL connections and transfers
 -c      Host config key (required)
 -t      Job template ID (required)
 -e      Extra variables
```

This script can retry commands and is therefore a more robust way to use callbacks than a simple `curl` request. The script retries once per minute for up to ten minutes.

NOTE:
This is an example script. Edit this script if you need more dynamic behavior when detecting failure scenarios, as any non-200 error code might not be a transient error requiring retry.

You can use callbacks with dynamic inventory in automation controller. For example, when pulling cloud inventory from one of the supported cloud providers. In these cases, along with setting **Update On Launch**, ensure that you configure an inventory cache timeout for the inventory source, to avoid hammering of your cloud's API endpoints. Since the `request_tower_configuration.sh` script polls once per minute for up to ten minutes, a suggested cache invalidation time for inventory (configured on the inventory source itself) would be one or two minutes.

Running the `request_tower_configuration.sh` script from a cron job is not recommended, however, a suggested cron interval is every 30 minutes. Repeated configuration can be handled by scheduling automation controller so that the primary use of callbacks by most users is to enable a base image that is bootstrapped into the latest configuration when coming online. Running at first boot is best practice. First boot scripts are init scripts that typically self-delete, so you set up an init script that calls a copy of the `request_tower_configuration.sh` script and make that into an auto scaling image.
If automation controller fails to locate the host either by name or IP address in one of your defined inventories, the request is denied. When running a job template in this way, ensure that the host initiating the playbook run against itself is in the inventory. If the host is missing from the inventory, the job template fails with a **No Hosts Matched** type error message.

If your host is not in the inventory and **Update on Launch** is checked for the inventory group, automation controller attempts to update cloud based inventory sources before running the callback.

Pass extra variables to Provisioning Callbacks

You can pass `extra_vars` in Provisioning Callbacks the same way you can in a regular job template. To pass `extra_vars`, the data sent must be part of the body of the POST as application or JSON, as the content type.

- Pass extra variables by using one of these methods:
    - Use the following JSON format as an example when adding your own `extra_vars` to be passed:

```
'{"extra_vars": {"variable1":"value1","variable2":"value2",...}}'
```

    - Pass extra variables to the job template call using `curl`:

```
root@localhost:~$ curl -f -H 'Content-Type: application/json' -XPOST \
-d '{"host_config_key": "redhat", "extra_vars": "{\"foo\": \"bar\"}"}' \
https://<CONTROLLER_SERVER_NAME>/api/v2/job_templates/7/callback
```

Related information
Launching Jobs with Curl

Distribute automation across a large number of hosts with job slicing

A sliced job refers to the concept of a distributed job. Use distributed jobs for running a job across a large number of hosts.

You can then run many ansible-playbooks, each on a subset of an inventory that you can be schedule in parallel across a cluster.

By default, Ansible runs jobs from a single control instance. For jobs that do not require cross-host orchestration, job slicing takes advantage of automation controller's ability to distribute work to many nodes in a cluster.

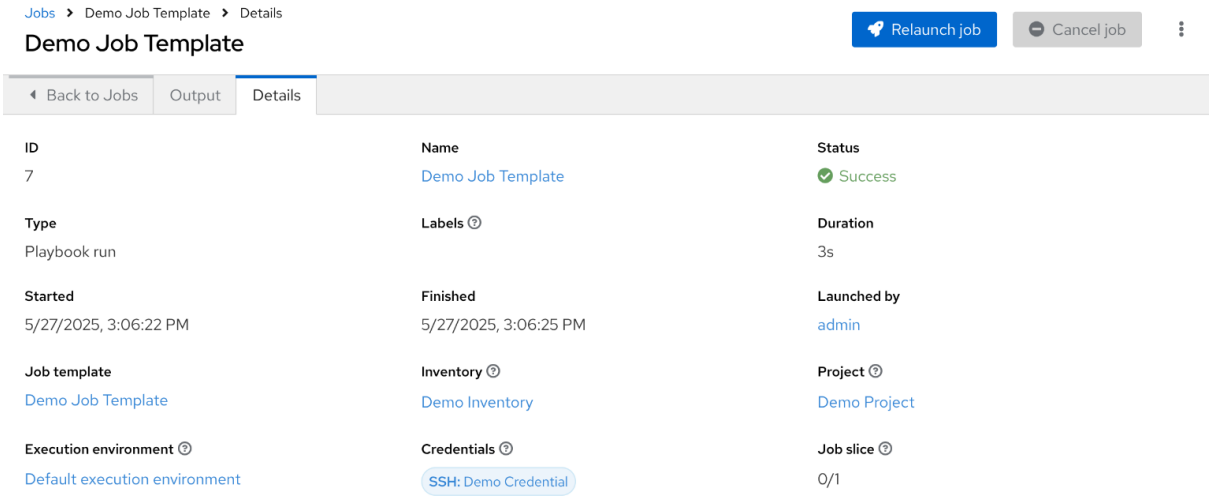
Job slicing works by adding a Job Template field `job_slice_count`, which specifies the number of jobs into which to slice the Ansible run. When this number is greater than `1`, automation controller generates a workflow from a job template instead of a job. The inventory is distributed evenly among the slice jobs. The workflow job is then started, and proceeds as though it were a normal workflow.

When launching a job, the API returns either a job resource (if `job_slice_count = 1`) or a workflow job resource. The corresponding User Interface (UI) redirects to the appropriate screen to display the status of the run.

Job slice considerations

When setting up job slices, consider the following:

- A sliced job creates a workflow job, which then creates jobs.
- A job slice consists of a job template, an inventory, and a slice count.
- When executed, a sliced job splits each inventory into several "slice size" chunks. It then queues jobs of ansible-playbook runs on each chunk of the appropriate inventory. The inventory fed into ansible-playbook is a shortened version of the original inventory that only has the hosts in that particular slice. The completed sliced job that displays on the **Jobs** list are labeled accordingly, with the number of sliced jobs that have run:



- These sliced jobs follow normal scheduling behavior (number of forks, queuing due to capacity, assignation to instance groups based on inventory mapping).

NOTE:
Job slicing is intended to scale job executions horizontally. Enabling job slicing on a job template divides an inventory to be acted upon in the number of slices configured at launch time and then starts a job for each slice.

Normally, the number of slices is equal to or less than the number of hybrid or execution nodes. A job template can also be limited to certain instances through its instance groups. You can set an extremely high number of job slices but it can cause performance degradation. The job scheduler is not designed to simultaneously schedule thousands of workflow nodes, which are what the sliced jobs become.

  - Sliced job templates with prompts or extra variables behave the same as standard job templates, applying all variables and limits to the entire set of slice jobs in the resulting workflow job. However, when passing a limit to a sliced job, if the limit causes slices to have no hosts assigned, those slices will fail, causing the overall job to fail.
  - A job slice job status of a distributed job is calculated in the same manner as workflow jobs. It fails if there are any unhandled failures in its sub-jobs.

- Any job that intends to orchestrate across hosts (rather than just applying changes to individual hosts) must not be configured as a slice job.
- If a sliced job fails, automation controller does not attempt to discover or account for the specific failed playbooks.

Job slice execution behavior

When jobs are sliced, they can run on any node. Insufficient capacity in the system can cause some to run at a different time. When slice jobs are running, job details display the workflow and job slices currently running, and a link to view their details individually.

By default, job templates are not normally configured to execute simultaneously (you must check `allow_simultaneous` in the API or **Concurrent jobs** in the UI). Slicing overrides this behavior and implies `allow_simultaneous` even if that setting is clear. See Job templates for information about how to specify this, and the number of job slices on your job template configuration.

The Job templates section provides additional detail on performing the following operations in the UI:

- Launch workflow jobs with a job template that has a slice number greater than one.
- Cancel the whole workflow or individual jobs after launching a slice job template.
- Relaunch the whole workflow or individual jobs after slice jobs finish running.
- View the details about the workflow and slice jobs after launching a job template.
- Search slice jobs specifically after you create them, according to the Searching job slices section.

Search job slices

You can search for job slices and their parent workflow jobs by using the search functionality in Automation controller.

To make it easier to find slice jobs, use the search functionality to apply a search filter to:

- Job lists to show only slice jobs
- Job lists to show only parent workflow jobs of job slices
- Job template lists to only show job templates that produce slice jobs
- Search for slice jobs by using one of the following methods:
    - To show only slice jobs in job lists, as with most cases, you can filter either on the type (jobs here) or `unified_jobs`:

      ```
      /api/v2/jobs/?job_slice_count__gt=1
      ```

    - To show only parent workflow jobs of job slices:

```
/api/v2/workflow_jobs/?job_template__isnull=false
```

- To show only job templates that produce slice jobs:

```
/api/v2/job_templates/?job_slice_count__gt=1
```

Orchestrate complex automation with workflow job templates

A workflow job template links together a sequence of disparate resources that tracks the full set of jobs that were part of the release process as a single unit.

These resources include the following:

- Job templates
- Workflow job templates
- Project syncs
- Inventory source syncs

You can create both Job templates and Workflow job templates from **Automation Execution > Templates**.

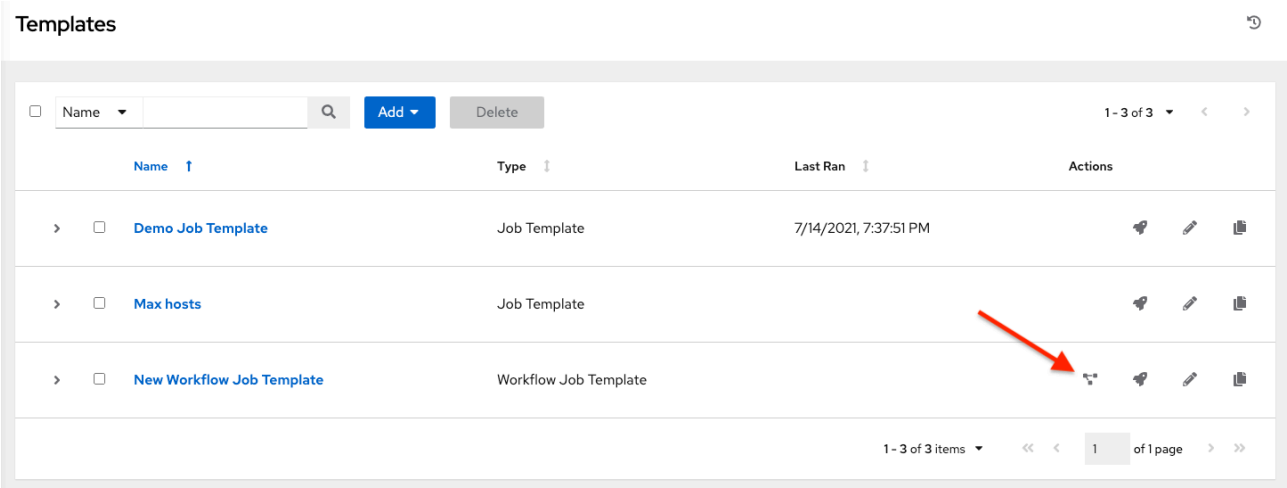
For Job templates, see Job templates.

The **Automation Templates** page shows the workflow and job templates that are currently available.

Select the template name to display more information about the template, including when it last ran. This list is sorted alphabetically by name, but you can sort by other criteria, or search by various fields and attributes of a template.

From this screen you can launch , edit , and duplicate a workflow job template.

Only workflow templates have the workflow visualizer icon as a shortcut for accessing the workflow editor.



NOTE:
Workflow templates can be used as building blocks for another workflow template. You can enable **Prompt on Launch** by setting up several settings in a workflow template, which you can edit at the workflow job template level. These do not affect the values assigned at the individual workflow template level. For further instructions, see the Workflow visualizer section.

Related information
Job templates
Workflow visualizer

Role-based access controls

Learn the specific roles required to create, edit, and run automation controller workflow job templates. Understanding these permissions help ensures that access is properly delegated among users and teams.

To edit and delete a workflow job template, you must have the administrator role. To create a workflow job template, you must be an organization administrator or a system administrator.

However, you can run a workflow job template that has job templates that you do not have permissions for. System administrators can create a blank workflow and then grant an `admin_role` to a low-level user, after which they can delegate more access and build the graph. You must have `execute` access to a job template to add it to a workflow job template.

You can also perform other tasks, such as making a duplicate copy or re-launching a workflow, depending on which permissions are granted to a user. You must have permissions to all the resources used in a workflow, such as job templates, before relaunching or making a copy.

For more information, see Managing access with role based access control.

For more information about performing the tasks described, see Workflow job templates.

Related information

Managing access with role based access control
Workflow job templates

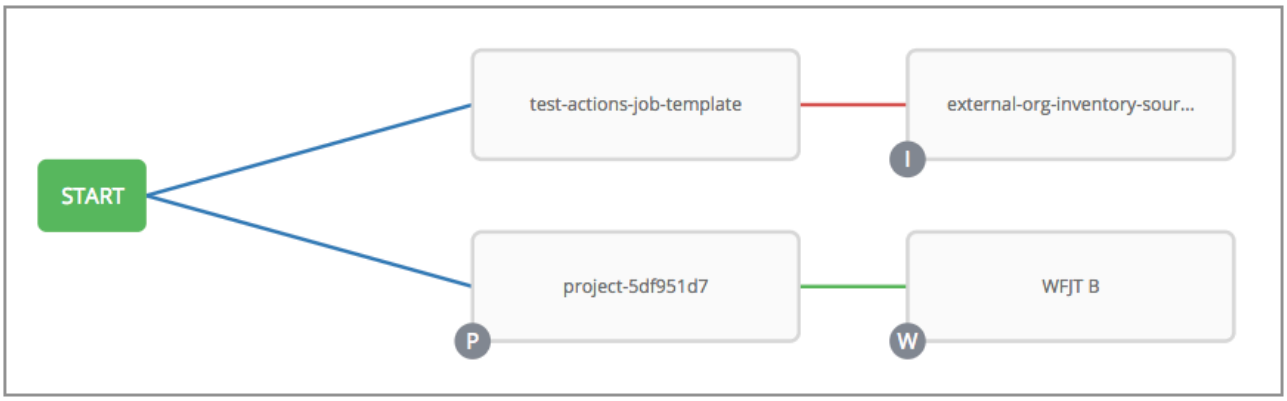
Understand how to configure workflows

Workflows enable you to configure a sequence of disparate job templates (or workflow templates) that might or might not share inventory, playbooks, or permissions.

Workflows have `admin` and `execute` permissions, similar to job templates. A workflow accomplishes the task of tracking the full set of jobs that were part of the release process as a single unit.

Job or workflow templates are linked together using a graph-like structure called nodes. These nodes can be jobs, project syncs, or inventory syncs. A template can be part of different workflows or used multiple times in the same workflow. A copy of the graph structure is saved to a workflow job when you launch the workflow.

The following example shows a workflow that has all three, and a workflow job template:



As the workflow runs, jobs are spawned from the node's linked template. Nodes linking to a job template which has prompt-driven fields (job_type, job_tags, skip_tags, limit) can contain those fields, and is not prompted on launch. Job templates that prompt for a credential or inventory, without defaults, are not available for inclusion in a workflow.

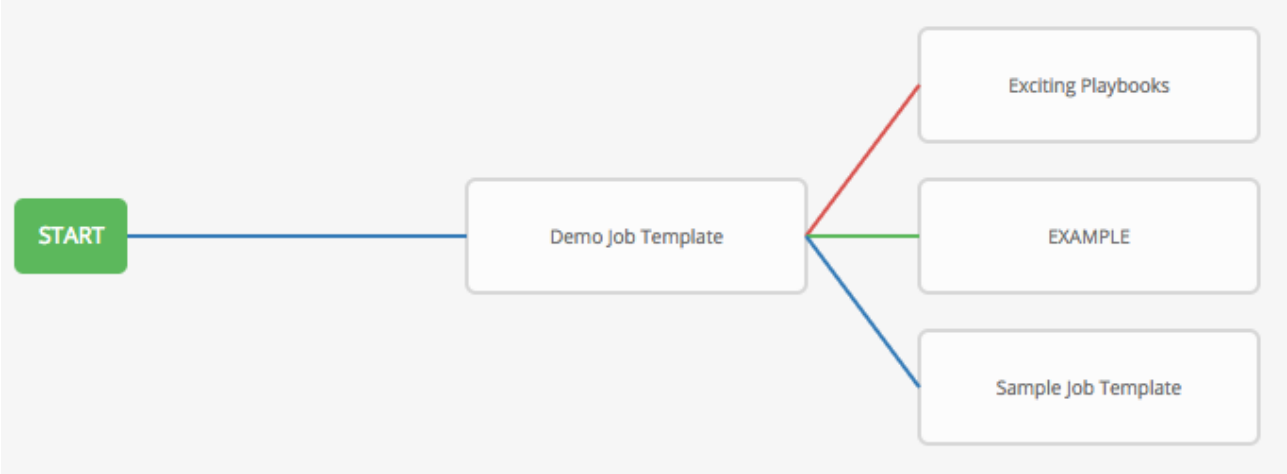
Workflow scenarios and considerations

Workflows in automation controller allow you to string together multiple job templates and other workflows into a single job run. This section describes several workflow scenarios and considerations to remember when building workflows.

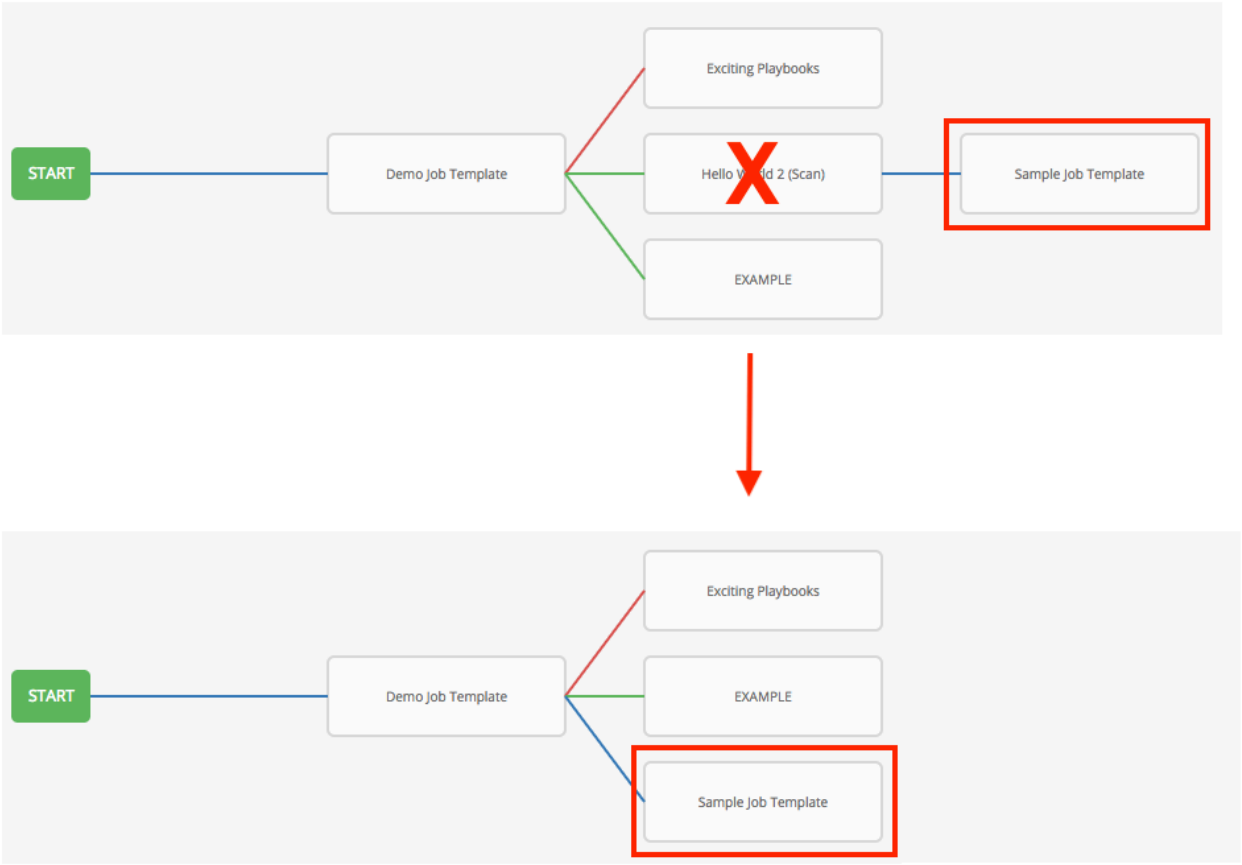
When building workflows, consider the following:

- A root node is set to **ALWAYS** by default and cannot be edited.

- A node can have multiple parents, and children can be linked to any of the states of success, failure, or always. If always, then the state is neither success nor failure. States apply at the node level, not at the workflow job template level. A workflow job is marked as successful unless it is canceled or encounters an error.



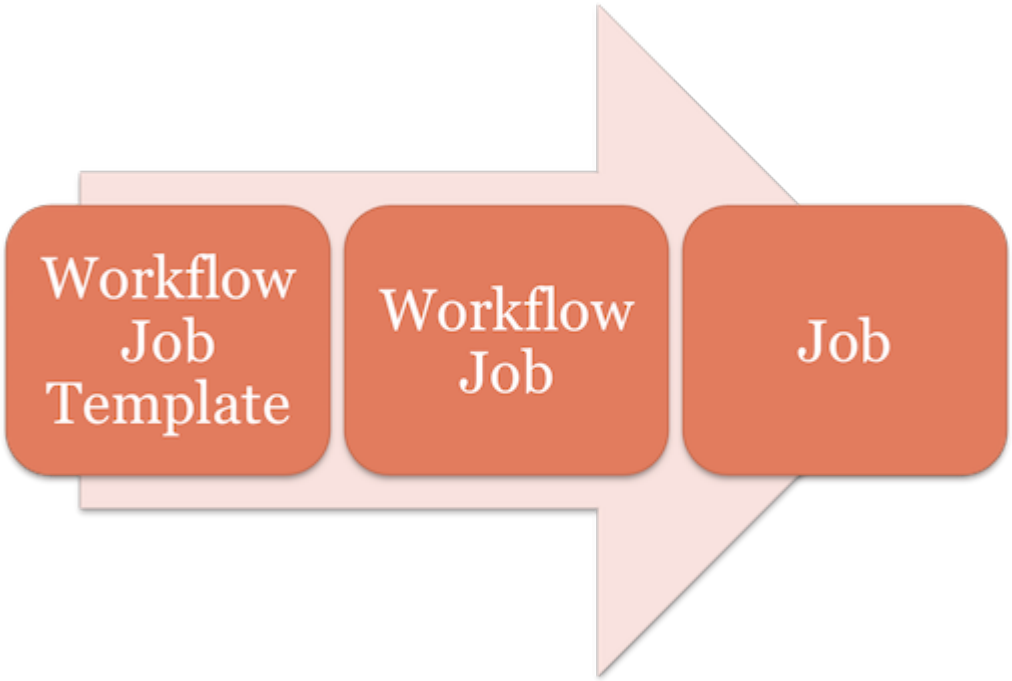
- If you remove a job or workflow template within the workflow, the nodes previously connected to those deleted, automatically get connected upstream and retain the edge type as in the following example:



- You can have a convergent workflow, where multiple jobs converge into one. In this scenario, any of the jobs or all of them must complete before the next one runs, as shown in the following example:

    - In this example, automation controller runs the first two job templates in parallel. When they both finish and succeed as specified, the third downstream (convergence node), triggers.

- Prompts for inventory and surveys apply to workflow nodes in workflow job templates.
- If you launch from the API, running a `get` command displays a list of warnings and highlights missing components. The following image illustrates a basic workflow for a workflow job template:



- It is possible to launch several workflows simultaneously, and set a schedule for when to launch them. You can set notifications on workflows, such as when a job completes, similar to that of job templates.

NOTE:
Job slicing is intended to scale job executions horizontally.

If you enable job slicing on a job template, it divides the inventory to be acted on in the number of slices configured at launch time. Then starts a job for each slice.

For more information see the Job slicing section.

- You can build a recursive workflow, but if automation controller detects an error, it stops at the time the nested workflow attempts to run.
- Artifacts gathered in jobs in the sub-workflow are passed to downstream nodes.
- An inventory can be set at the workflow level, or prompt for inventory on launch.
- When launched, all job templates in the workflow that have `ask_inventory_on_launch=true` use the workflow level inventory.
- Job templates that do not prompt for inventory ignore the workflow inventory and run against their own inventory.
- If a workflow prompts for inventory, schedules and other workflow nodes can provide the inventory.
- In a workflow convergence scenario, `set_stats` data is merged in an undefined way, therefore you must set unique keys.

Related information
Job slicing

Workflow extra variables

Workflows use surveys to define `extra_vars` for the playbooks within the workflow. These variables are combined with job template data to provide consistent configuration across all spawned jobs.

Workflows use the same behavior (hierarchy) of variable precedence as job templates with the exception of three additional variables. See the Automation controller Variable Precedence Hierarchy in the Extra variables section of Job templates. The three additional variables include:

- Workflow job template extra variables
- Workflow job template survey (defaults)
- Workflow job launch extra variables

Workflows included in a workflow follow the same variable precedence, they only inherit variables if they are specifically prompted for, or defined as part of a survey.

In addition to the workflow `extra_vars`, jobs and workflows run as part of a workflow can inherit variables in the artifacts dictionary of a parent job in the workflow (also combining with ancestors further upstream in its branch).

If you use the `set_stats` module in your playbook, you can produce results that can be consumed downstream by another job.

Example Notifying users as to the success or failure of an integration run. In this example, there are two playbooks that can be combined in a workflow to exercise artifact passing:

- invoke_set_stats.yml: first playbook in the workflow:

```
---
- hosts: localhost
  tasks:
    - name: "Artifact integration test results to the web"
      local_action: 'shell curl -F "file=@integration_results.txt" https://file.io'
      register: result


    - name: "Artifact URL of test results to Workflows"
      set_stats:
        data:
          integration_results_url:  "{{ (result.stdout|from_json).link }}"
```

- use_set_stats.yml: second playbook in the workflow:

```
---
- hosts: localhost
  tasks:
    - name: "Get test results from the web"
      uri:
        url: "{{ integration_results_url }}"
        return_content: true
      register: results



    - name: "Output test results"
      debug:
        msg: "{{ results.content }}"
```

The `set_stats` module processes this workflow as follows:

1. The contents of an integration result is uploaded to the web.
2. Through the `invoke_set_stats` playbook, `set_stats` is then invoked to artifact the URL of the uploaded `integration_results.txt` into the Ansible variable `integration_results_url`.
3. The second playbook in the workflow consumes the Ansible extra variable `integration_results_url`. It calls out to the web by using the URI module to get the contents of the file uploaded by the previous job template job. Then, it prints out the contents of the obtained file.

NOTE:
For artifacts to work, keep the default setting, `per_host = False` in the `set_stats` module.

Create a workflow job template

To create a new workflow job template, complete the following steps:

IMPORTANT:
If you set a limit to a workflow template, it is not passed down to the job template unless you check **Prompt on launch** for the limit. This can lead to playbook failures if the limit is mandatory for the playbook that you are running.

Procedure

1. From the navigation panel, select **Automation Execution > Templates**.
2. On the **Automation Templates** page, select **Create workflow job template** from the **Create template** list.
3. Enter the appropriate details in the following fields:

NOTE:
If a field has the **Prompt on launch** checkbox selected, either launching the workflow template, or using the workflow template within another workflow template, you are prompted for the value for that field. Most prompted values override any values set in the job template. Exceptions are noted in the following table.

Field	Options	Prompt on Launch
Name	Enter a name for the job.	N/A
Description	Enter an arbitrary description as appropriate (optional).	N/A
Organization	Choose the organization to use with this template from the organizations available to the logged in user.	N/A
Inventory	Optionally, select the inventory to use with this template from the inventories available to the logged in user.	Yes
Limit	A host pattern to further constrain the list of hosts managed or affected by the playbook. You can separate many patterns by colons (:). As with core Ansible: • a:b means "in group a or b" • a:b:&c means "in a or b but must be in c"	Yes If selected, even if a default value is supplied, you are prompted upon launch to select a limit.

Field	Options	Prompt on Launch
	• a:!b means "in a, and definitely not in b"	
Source control branch	Select a branch for the workflow. This branch is applied to all workflow job template nodes that prompt for a branch.	Yes
Labels	• Optionally, supply labels that describe this workflow job template, such as `dev` or `test`. Use labels to group and filter workflow job templates and completed jobs in the display. • Labels are created when they are added to the workflow template. Labels are associated to a single Organization using the Project that is provided in the workflow template. Members of the Organization can create labels on a workflow template if they have edit permissions (such as the admin role). • Once you save the job template, the labels appear in the workflow job	Yes If selected, even if a default value is supplied, you are prompted when launching to supply additional labels, if needed. - You cannot delete existing labels, selecting ✖ only removes the newly added labels, not existing default labels.

Field	Options	Prompt on Launch
	template **Details** view. • Labels are only applied to the workflow templates not the job template nodes that are used in the workflow. • Select ✖ beside a label to remove it. When a label is removed, it is no longer associated with that particular Job or Job Template, but it remains associated with any other jobs that reference it.	
Job tags	Type and select the **Create** drop-down to specify which parts of the playbook should run.	Yes
Skip tags	Type and select the **Create** drop-down to specify certain tasks or parts of the playbook to skip.	Yes
Extra variables	• Pass extra command line variables to the playbook. This is the "-e" or "--extra-vars" command line parameter for ansible-playbook that is documented in the Ansible documentation at Controlling how Ansible behaves: precedence rules. - Give	Yes If you want to be able to specify `extra_vars` on a schedule, you must select **Prompt on launch** for **Extra variables** on the workflow job template, or enable a survey on the job template. Those answered survey questions become `extra_vars`. For more

Field	Options	Prompt on Launch
	key or value pairs by using either YAML or JSON. These variables have a maximum value of precedence and overrides other variables specified elsewhere. The following is an example value: `git_branch: production release_version: 1.5`	information about extra variables, see Extra Variables.

4. Specify the following **Options** for launching this template, if necessary:

   - Check **Enable webhook** to turn on the ability to interface with a predefined SCM system web service that is used to launch a workflow job template. GitHub and GitLab are the supported SCM systems.
     - If you enable webhooks, other fields display, prompting for additional information:
       - **Webhook service**: Select which service to listen for webhooks from.
       - **Webhook URL**: Automatically populated with the URL for the webhook service to POST requests to.
       - **Webhook key**: Generated shared secret to be used by the webhook service to sign payloads sent to automation controller. You must configure this in the settings on the webhook service so that webhooks from this service are accepted in automation controller. For additional information about setting up webhooks, see Working with Webhooks.
   - Check **Enable concurrent jobs** to allow simultaneous runs of this workflow. For more information, see Automation controller capacity determination and job impact.

5. When you have completed configuring the workflow template, click **Create workflow job template**.

   Saving the template exits the workflow template page and the workflow visualizer opens where you can build a workflow. For more information, see the Workflow visualizer section. Otherwise, select one of these methods:

   - Close the workflow visualizer to return to the **Details** tab of the newly saved template. There you can complete the following tasks:
     - Review, edit, add permissions, notifications, schedules, and surveys
     - View completed jobs
     - Build a workflow template
   - Click **Launch template** to start the workflow.

NOTE:
Save the template before launching, or **Launch template** remains disabled. The **Notifications** tab is only present after you save the template.

Work with permissions

You can manage permissions for workflow templates to control which users and teams can view or modify them.

Click the **Team Access** or **User Access** tab to review, grand, edit, and remove associated permissions for users along with team members.

Click **Add roles** to create new permissions for this workflow template by following the prompts to assign them.

Work with notifications

From the navigation panel, select **Automation Execution > Administration > Notifiers**. You can review any notification integrations you have set up and their statuses, if they have run.

Use the toggles to enable or disable the notifications to use with your particular template. For more information, see Enable and disable notifications.

If no notifications have been set up, click **Add notifier** to create a new notification. For more information about configuring various notification types and extended messaging, see Notification types.

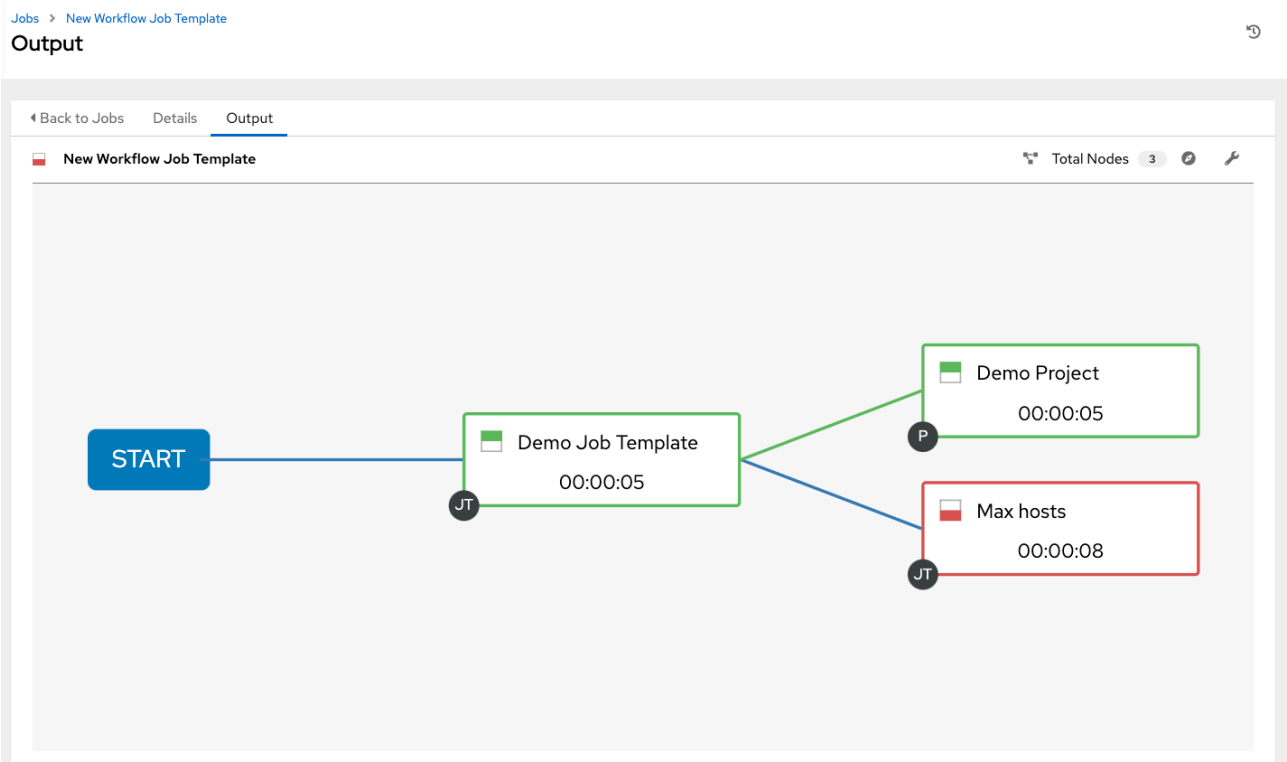
Related information
Enable and disable notifications
Notification types

View completed workflow jobs

You can view completed workflow jobs in automation controller to see the details of each job run.

The **Jobs** tab provides the list of job templates that have run. Click the expand icon next to each job to view the details of each job.

From this view, you can click the job ID, name of the workflow job and see its graphical representation. The following example shows the job details of a workflow job:



The nodes are marked with labels to help you identify them. For more information, see the legend in the Workflow visualizer section.

Related information
Workflow visualizer

Launch a workflow job template

You can launch a workflow job template to run a series of job templates and workflow nodes in a defined order.

- Launch a workflow job template by using one of these methods:
  - From the navigation panel, select **Automation Execution > Templates** and click the icon next to the job template.
  - Click **Launch template** in the **Details** tab of the workflow job template that you want to launch.
    Variables added for a workflow job template are automatically added in automation controller when launching, along with any extra variables set in the workflow job template and survey.

    Events related to approvals on workflows are displayed in the activity stream () with detailed information about the approval requests, if any.

Duplicate a workflow job template

Duplicate a workflow job template to create a copy of its structure. Note that associated schedules, notifications, and permissions are not copied and must be manually recreated for the new template.

Procedure

1. From the navigation panel, select **Automation Execution > Templates**.
2. Click the ⋮ icon associated with the template that you want to duplicate and select the Duplicate template icon.
    - The new template with the name of the template from which you duplicated and a timestamp displays in the list of templates.
3. Click to open the new template and click **Edit template**.
4. Replace the contents of the **Name** field with a new name, and give or change the entries in the other fields to complete this page.
5. Click **Save job template**.

> **NOTE:**
> If a resource has a related resource that you do not have the right level of permission to, you cannot duplicate the resource. For example, in the case where a project uses a credential that a current user only has Read access. However, for a workflow job template, if any of its nodes use an unauthorized job template, inventory, or credential, the workflow template can still be duplicated. But in the duplicated workflow job template, the corresponding fields in the workflow template node are absent.

Build a graphical workflow representation with workflow visualizer

The Workflow Visualizer provides a graphical way of linking together job templates, workflow templates, project syncs, and inventory syncs to build a workflow template.

Before you build a workflow template, see the Workflows in automation controller section for considerations associated with various scenarios on parent, child, and sibling nodes.

Related information
Workflows in automation controller

Build a workflow

You can build a workflow by adding nodes and defining their relationships in the workflow visualizer.

You can set up any combination of two or more of the following node types to build a workflow:

- Template (Job Template or Workflow Job Template)
- Project Sync
- Inventory Sync
- Approval

Procedure

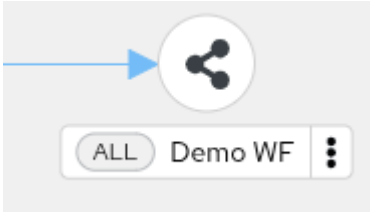
1. To launch the workflow visualizer, use one of these methods:
    - From the navigation panel, select **Automation Execution > Templates**.
        - Select a workflow template and click **View workflow visualizer**.
    - From the **Automation Templates** list view, click the icon next to a workflow job template.
2. Click **Add step** to display a list of nodes to add to your workflow.
3. From the **Node type** list, select the type of node that you want to add.
    - If you select an **Approval** node, see Approval nodes for more information. Selecting a node provides the available valid options associated with it.

> **NOTE:**
> If you select a job template that does not have a default inventory when populating a workflow graph, the inventory of the parent workflow is used. Though a credential is not required in a job template, you cannot select a job template for your workflow if it has a credential that requires a password, unless the credential is replaced by a prompted credential.

4. When you select a node type, the workflow begins to build, and you must specify the type of action to be taken for the selected node. This action is also referred to as edge type.
5. If the node is a root node, the edge type defaults to **Always** and is non-editable. For subsequent nodes, you can select one of the following scenarios (edge type) to apply to each:
    - **Always run**: Continue to run regardless of success or failure.
    - **Run on success**: After successful completion, run the next template.
    - **Run on fail**: After failure, run a different template.

6. Select the behavior of the node if it is a convergent node from the **Convergence** field:

    - **Any** is the default behavior, allowing any of the nodes to complete as specified, before triggering the next converging node. If the status of one parent meets one of those run conditions, an **any** child node will run. An **any** node requires all nodes to complete, but only one node must complete with the expected outcome.
    - Choose **All** to ensure that all nodes complete as specified, before converging and triggering the next node. The purpose of **all*** nodes is to make sure that every parent meets its expected outcome to run the child node. The workflow checks to make sure every parent behaves as expected to run the child node. Otherwise, it will not run the child node.
      If selected, the node is labeled as **ALL** in the graphical view:



> **NOTE:**
> If a node is a root node, or a node that does not have any nodes converging into it, setting the Convergence rule does not apply, as its behavior is dictated by the action that triggers it.

7. If a job template used in the workflow has **Prompt on launch** selected for any of its parameters, a **PROMPT** option is displayed, enabling you to change those values at the node level. Use the wizard to change the values in each of the tabs and click **Confirm** in the **Preview** tab.

   If a workflow template used in the workflow has **Prompt on launch** selected for the inventory option, use the wizard to supply the inventory at the prompt. If the parent workflow has its own inventory, it overrides any inventory that is supplied here.

NOTE:
For workflow job templates with required fields that prompt details, but do not have a default, you must give those values when creating a node before the **SELECT** option is enabled.

The following two cases disable the **SELECT** option until a value is provided by the **PROMPT** option:

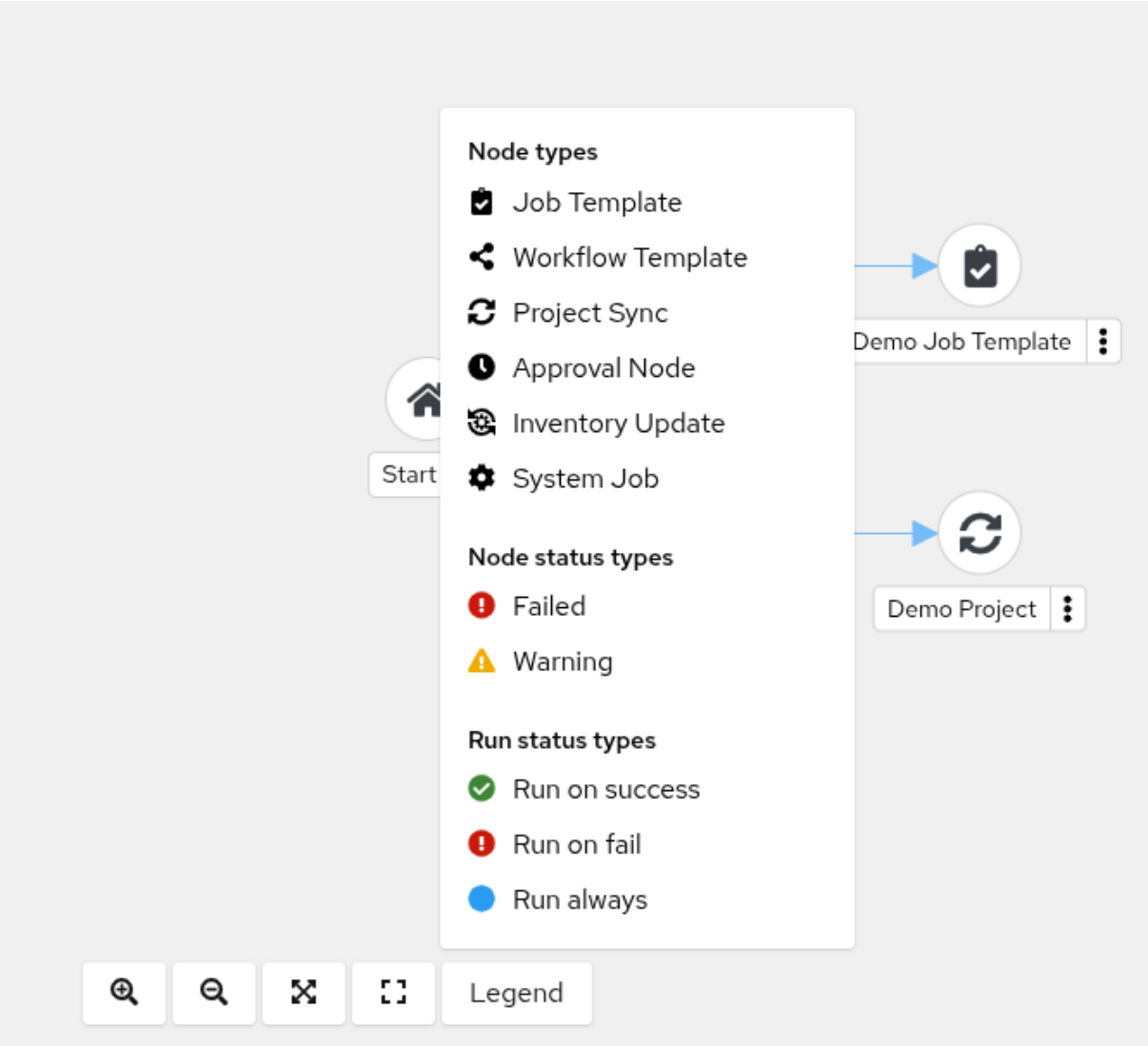
a. When you select the **Prompt on launch** checkbox in a workflow job template, but do not give a default.

b. When you create a survey question that is required but do not give a default answer.

However, this is not the case with credentials. Credentials that require a password on launch are not permitted when creating a workflow node, because everything required to launch the node must be provided when the node is created. If you are prompted for credentials in a workflow job template, it is not possible to select a credential that requires a password in automation controller.

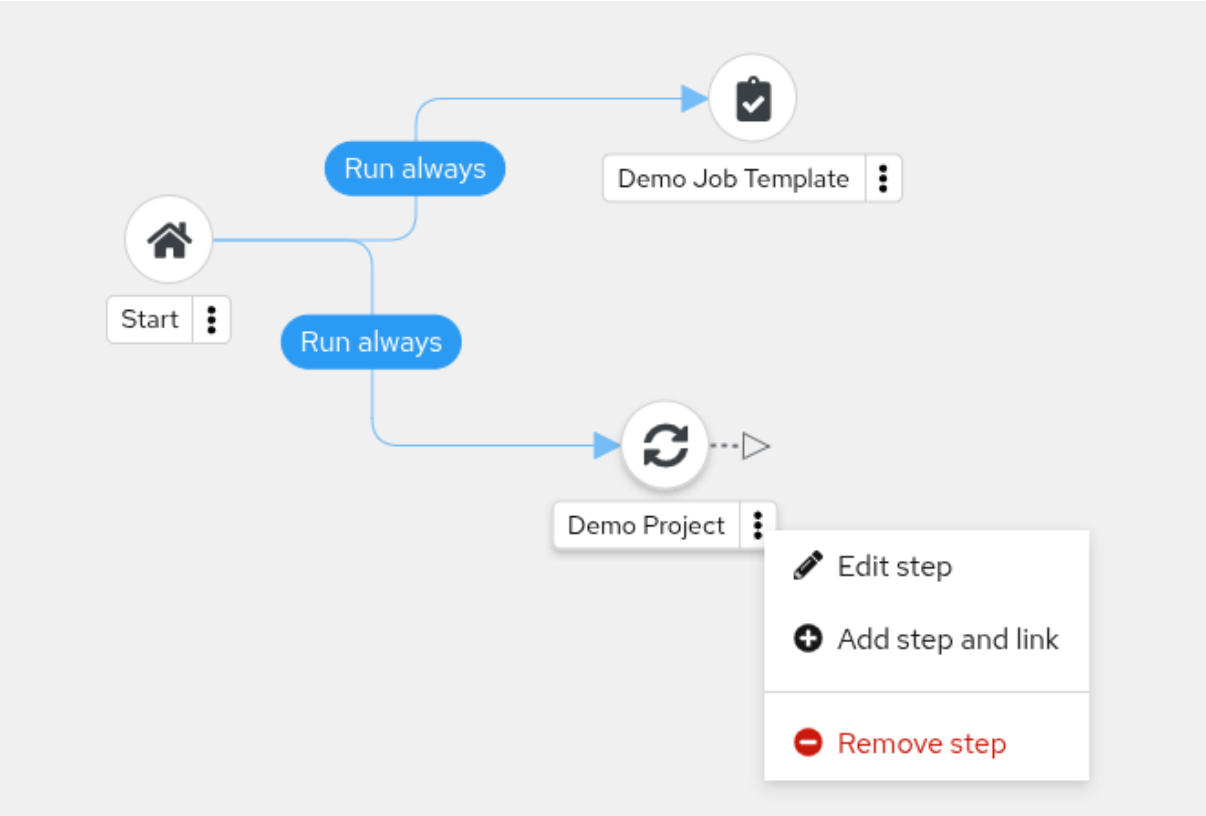
You must also click **SELECT** when the prompt wizard closes, to apply the changes at that node. Otherwise, any changes you make revert back to the values set in the job template.

When the node is created, it is labeled with its job type. A template that is associated with each workflow node runs based on the selected run scenario as it proceeds. Click **Legend** to display the legend for each run scenario and their job types.



8. Hover over a node to edit the node, add step and link, or delete the selected node:

NOTE:
If you hover over a step when adding a link and a red border is displayed, this means that you cannot connect those two steps together. This is a preventive measure to avoid users creating "circular dependencies", which can result in a workflow that ends up in an infinite loop and never finishes.



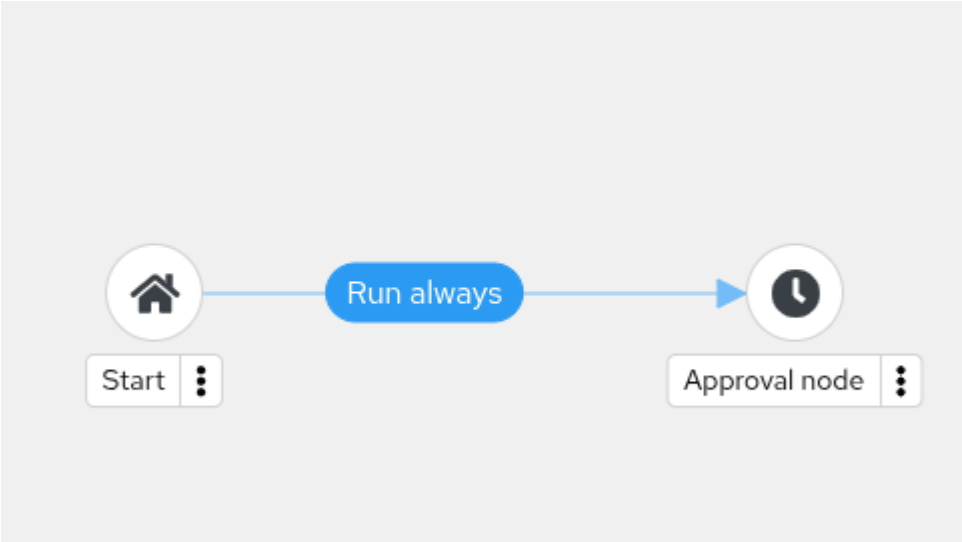
9. When you have added or edited a node, click **Finish** to save any modifications and render it on the graphical view. For possible ways to build your workflow, see Building nodes scenarios.
10. When you have built your workflow job template, click **Save** to save your entire workflow template and return to the new workflow job template details page.

IMPORTANT:
Clicking **Close** does not save your work, but instead, it closes the entire Workflow Visualizer so that you have to start again.

Configure nodes in workflow visualizer

Use an Approval node to manually pause a workflow between playbooks. This allows you to review and approve the next step within a set timeframe or advance the process immediately.

The default for the timeout is none, but you can specify the length of time before the request expires and is automatically denied. After you select and supply the information for the approval node, it displays on the graphical view with a pause icon beside it.



The approver is anyone who meets the following criteria:

- A user that can execute the workflow job template containing the approval nodes.
- A user who has organization administrator or above privileges (for the organization associated with that workflow job template).
- A user who has the **Approve** permission explicitly assigned to them within that specific workflow job template.

If pending approval nodes are not approved within the specified time limit (if an expiration was assigned) or they are denied, then they are marked as "timed out" or "failed", and move on to the next "on fail node" or "always node". If approved, the "on success" path is taken. If you try to `POST` in the API to a node that has already been approved, denied or timed out, an error message notifies you that this action is redundant, and no further steps are taken.

The following table shows the various levels of permissions allowed on approval workflows:

SCOPE	ROLE	CREATE WORKFLOW APPROVAL	GRANT APPROVAL	VIEW WORKFLOW APPROVAL	APPROVE/DENY	VIEW WORKFLOW APPROVAL IN ACTIVITY STREAM
Organization	Organization Admin	Yes	Yes	Yes	Yes	Yes
Organization Workflow Job Template	Workflow Admin	Yes	Yes (*)	Yes	Yes	Yes
	Workflow Executor	No	No	Yes	No	Yes
	Workflow Approver	No	No	Yes	Yes	Yes
	Read on Workflow	No	No	Yes (**)	No	Yes (***)
System	System Admin	Yes	Yes	Yes	Yes	Yes
	System Auditor	No	No	Yes	No	Yes
Random user in Organization		No	No	No	No	No
Random user outside Organization		No	No	No	No	No

* Exception: A User with WF Admin permission at the organization level would not be able to grant approval.

** Exception: A User with Read on WF permission at the organization level would not be able to view WF approvals.

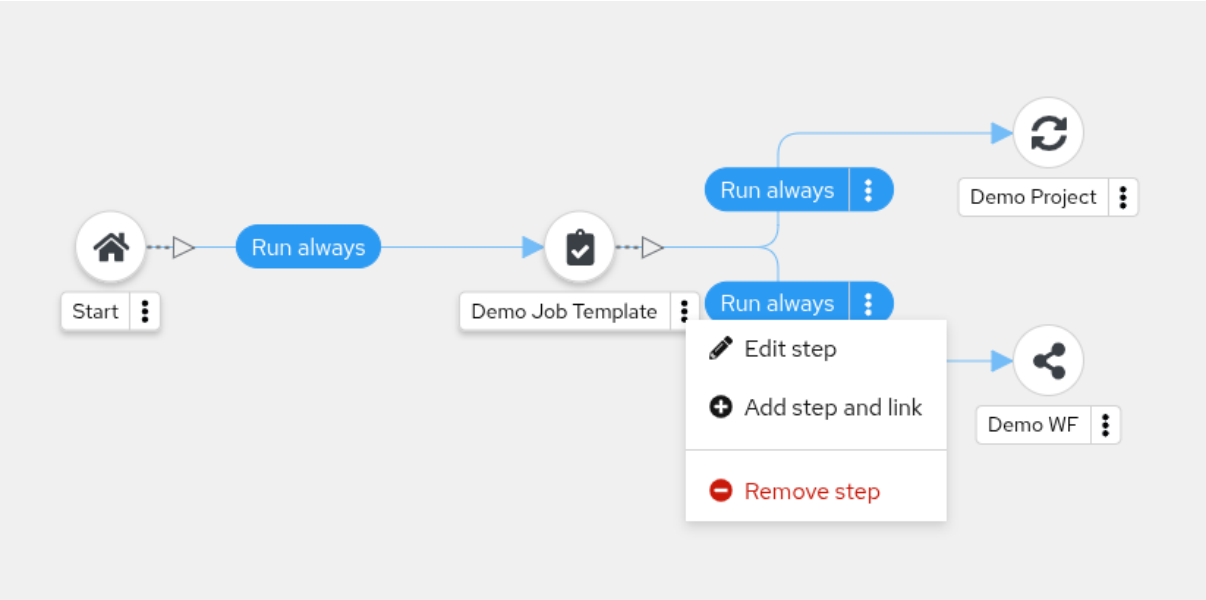
*** Exception: A User with Read on WF permission at the organization level would not be able to view approval jobs in the Activity Stream.

Build nodes scenarios

Learn how to manage nodes in the following scenarios.

Procedure

1. Click the (⋮) icon on the parent node and **Add step and link** to add a sibling node:



2. Click **Add step** or **Start** (⋮) and **Add step**, to add a root node to depict a split scenario.
3. At any node where you want to create a split scenario, hover over the node from which the split scenario begins and click the plus (⋮) icon on the parent node and **Add step and link**. This adds multiple nodes from the same parent node, creating sibling nodes.
4. Refer to the key by clicking **Legend** to identify the meaning of the symbols and colors associated with the graphical depiction.

> **NOTE:**
> If you remove a node that has a follow-on node attached to it in a workflow with a set of sibling nodes that has varying edge types, the attached node automatically joins the set of sibling nodes and retains its edge type:

Edit a node

Learn how to edit a node's details or link edge type (Run on success/fail/always) and remove links within a workflow, and how to adjust the workflow diagram's view.

- Edit a node by using one of these methods:
  - If you want to edit a node, click the icon of the node. The pane displays the current selections, click **Edit** to change these. Make your changes and click **Finish** to apply them to the graphical view.
  - To edit the edge type for an existing link, (**Run on success**, **Run on fail**, **Run always**), click (⋮) on the existing status.

- To remove a link, click (⋮) for the link and click **Remove link**. This option only appears in the pane if the target or child node has more than one parent. All nodes must be linked to at least one other node at all times so you must create a new link before removing an old one.

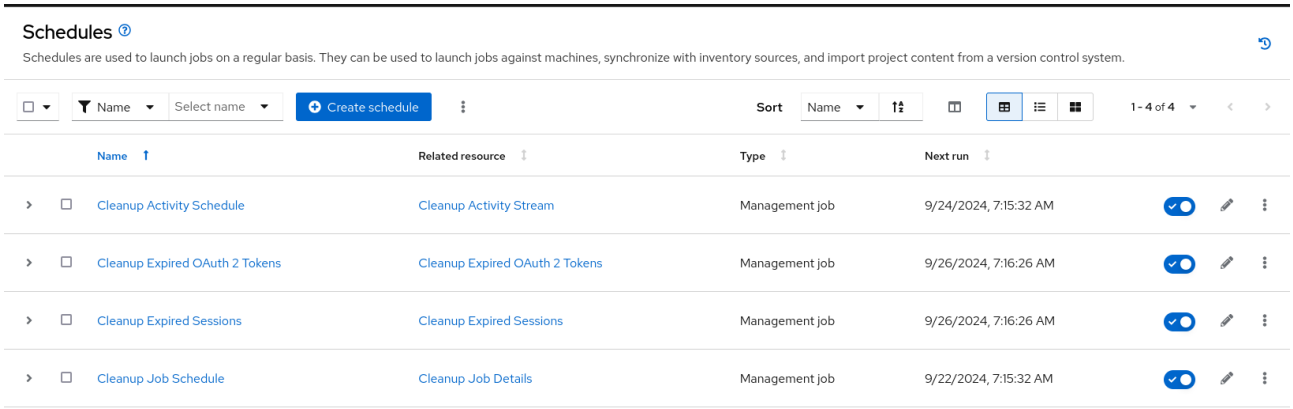
- Edit the view of the workflow diagram by using one of these methods:
  - Click the examine icon (🔍) to zoom in, the reduce icon (🔍) to zoom out, the expand icon (⤢) to fit to screen or the reset icon (⛶) to reposition the view.
  - Drag the workflow diagram to reposition it on the screen or use the scroll on your mouse to zoom.

Schedule recurring automation

From the navigation panel, click **Automation Execution > Schedules** to access your configured schedules. The schedules list can be sorted by any of the attributes from each column by using the directional arrows. You can also search by name, date, or the name of the month in which a schedule runs.

Use the **On** or **Off** toggle to stop an active schedule or activate a stopped schedule.

Click the Edit ✎ icon to edit a schedule.



If you are setting up a template, a project, or an inventory source, click the **Schedules** tab on the **Details** page for that resource, to configure schedules for these resources. When you create a schedule, it has the following parameters:

Name

Click the schedule name to open its details.

Related resource

Describes the function of the schedule.

Type

This identifies whether the schedule is associated with a source control update or a system-managed job schedule.

Next run

The next scheduled run of this task.

Add a new schedule

You can create a new schedule for a job template, workflow job template, inventory source, project sync, or management job template.

You can create schedules from a template, project, or inventory source, and directly on the main **Schedules** page.

To create a new schedule on the **Schedules** page:

Procedure

1. From the navigation panel, select **Automation Execution > Schedules**.
2. Click **Create schedule**. This opens the **Create schedule** window.
3. Select a **Resource type** onto which this schedule is applied.

   Select from:

   - **Job template**
     - For **Job template** select a **Job template** from the menu.
   - **Workflow job template**
     - For **Workflow job template** select a **Workflow job template** from the menu.
   - **Inventory source**
     - For **Inventory source** select an **Inventory** and an **Inventory source** from the appropriate menu.
   - **Project sync**
     - For **Project sync** select a **Project** from the menu.
   - **Management job template**
     - For **Management job template** select a **Workflow job template** from the menu.
4. For **Job template** and **Project sync** enter the appropriate details into the following fields:

   - **Schedule name**: Enter the name.
   - Optional: **Description**: Enter a description.
   - **Start date/time**: Enter the date and time to start the schedule.
   - **Time zone**: Select the time zone. The **Start date/time** that you enter must be in this time zone.

The **Schedule Details** display when you establish a schedule, enabling you to review the schedule settings and a list of the scheduled occurrences in the selected **Local Time Zone**.

IMPORTANT:
Jobs are scheduled in UTC. Repeating jobs that run at a specific time of day can move relative to a local time zone when Daylight Saving Time shifts occur. The system resolves the local time zone based time to UTC when the schedule is saved. To ensure your schedules are correctly created, set your schedules in UTC time.

5. Click **Next**. The **Define rules** page is displayed.

Add a new schedule from a resource page

You can create a new schedule from the **Schedules** tab of a resource page, such as a template, project, or inventory source.

To create a new schedule from a resource page:

Procedure

1. Click the **Schedules** tab of the resource that you are configuring. This can be a template, project, or inventory source.
2. Click **Create schedule**. This opens the **Create schedule** window.
3. Enter the appropriate details into the following fields:
   - **Schedule name**: Enter the name.
   - Optional: **Description**: Enter a description.
   - **Start date/time**: Enter the date and time to start the schedule.
   - **Time zone**: Select the time zone. The **Start date/time** that you enter must be in this time zone.
     The **Schedule Details** display when you establish a schedule, enabling you to review the schedule settings and a list of the scheduled occurrences in the selected **Local Time Zone**.

IMPORTANT:
Jobs are scheduled in UTC. Repeating jobs that run at a specific time of day can move relative to a local time zone when Daylight Saving Time shifts occur. The system resolves the local time zone based time to UTC when the schedule is saved. To ensure your schedules are correctly created, set your schedules in UTC time.

4. Click **Next**. The **Define rules** page is displayed.

Define rules for the schedule

After you create a schedule, you can define rules for when the schedule runs.

Procedure

1. Enter the following information:
   - **Frequency**: Enter how often the schedule runs.
   - **Interval**: Select the interval at which the rule will repeat.
   - **Week start**: Select the day of the week that you want the week to begin.
   - **Minutes of the hour**: Use this field to declare minute(s) of the hour that the schedule should run.
   - **Hours of day**: Use this field to declare the hours of day that the schedule should run.
   - **Days of the week**: Select the days of the week on which to run the schedule.
   - **Days of the month**: Select the months of the year on which to run the schedule
   - **Weeks of the year**: Use this field to declare numbered weeks of the year that the schedule should run.
   - **Months of the year**: Use this field to declare ordinal days number of the month that the schedule should run.
   - **Days of the year**: Use this field to declare ordinal number days of the year that the schedule should run.
   - **Occurrences**: Use this field to filter down indexed rules based on those declared using the form fields in the Rule section.
   - **Schedule ending type**: Use this field to select when the schedule is set to end. For more information, see the link to the `iCalendar RFC for bysetpos` field in the iCalendar documentation when you have set the rules for the schedule.
2. Click **Save rule**. The **Schedule Rules** summary page is displayed.
3. Click **Add rule** to add additional rules.
4. Click **Next**.

   The **Schedule Exceptions** page is displayed.

Set exceptions to the schedule

Define specific dates or times when automated jobs should skip execution by using schedule exceptions.

Procedure

1. On the **Schedule Exceptions** page, click **Create exception**.

   Use the same format as for the schedule rules to create a schedule exception.
2. Click **Next** to save and review both the schedule and the exception.

Logically group playbooks with projects

A project is a logical collection of Ansible playbooks, represented in automation controller.

You can manage playbooks and playbook directories different ways:

- By placing them manually under the Project Base Path on your automation controller server.
- By placing your playbooks into a source code management (SCM) system supported by the automation controller.

These include Git, Subversion, Mercurial and Red Hat Lightspeed.

For more information on creating a Red Hat Lightspeed project, see Setting up Red Hat Lightspeed Remediations.

NOTE:
The Project Base Path is `/var/lib/awx/projects`. However, this can be modified by the system administrator. It is configured in `/etc/tower/conf.d/custom.py`.

Use caution when editing this file, as incorrect settings can disable your installation.

The **Projects** page displays the list of the projects that are currently available.

A **Demo Project** is provided that you can work with initially.

For each project listed, you can get the latest SCM revision, edit the project, or duplicate the project attributes, using the icons next to each project.

Projects can be updated while a related job is running.

In cases where you have a large project (around 10 GB), disk space on `/tmp` may be an issue.

Status indicates the state of the project and may be one of the following (note that you can also filter your view by specific status types):

- **Pending** - The source control update has been created, but not queued or started yet. Any job (not just source control updates) stays in pending until it is ready to be run by the system. Possible reasons for it not being ready are:
  - It has dependencies that are currently running so it has to wait until they are done.
  - There is not enough capacity to run in the locations it is configured to.
- **Waiting** - The source control update is in the queue waiting to be executed.

- **Running** - The source control update is currently in progress.
- **Successful** - The last source control update for this project succeeded.
- **Failed** - The last source control update for this project failed.
- **Error** - The last source control update job failed to run at all.
- **Canceled** - The last source control update for the project was canceled.
- **Never updated** - The project is configured for source control, but has never been updated.
- **OK** - The project is not configured for source control, and is correctly in place.
- **Missing** - Projects are absent from the project base path of `/var/lib/awx/projects`. This is applicable for manual or source control managed projects.

NOTE:
Projects of credential type `Manual` cannot update or schedule source control-based actions without being reconfigured as an SCM type credential.

Related information
Setting up Red Hat Lightspeed Remediations

Add a new project

You can create a logical collection of playbooks, called projects in automation controller.

Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. On the **Projects** page, click **Create project** to launch the **Create Project** window.
3. Enter the appropriate details into the following required fields:
    - **Name** (required)
    - Optional: **Description**
    - **Organization** (required): A project must have at least one organization. Select one organization now to create the project. When the project is created you can add additional organizations.
    - Optional: **Execution environment**: Enter the name of the execution environment or search from a list of existing ones to run this project. For more information, see Define, create, and build execution environments.
    - **Source control type** (required): Select an SCM type associated with this project from the menu. Options in the following sections become available depending on the

type chosen. For more information, see Managing playbooks manually or Managing playbooks using source control.

- Optional: **Content signature validation credential**: Use this field to enable content verification. Specify the GPG key to use for validating content signature during project synchronization. If the content has been tampered with, the job will not run. For more information, see Project signing and verification.

4. Click **Create project**.

Related information
Managing playbooks manually
Managing playbooks using source control
SCM Types - Configuring playbooks to use Git and Subversion
SCM Type - Configuring playbooks to use Red Hat Lightspeed
SCM Type - Configuring playbooks to use a remote archive

Manage playbooks manually

Manage Ansible playbooks and directories directly on the automation controller filesystem. This approach helps ensure proper file ownership and permissions when you cannot use source control.

- Create one or more directories to store playbooks under the Project Base Path, for example, `/var/lib/awx/projects/`.
- Create or copy playbook files into the playbook directory.
- Ensure that the playbook directory and files are owned by the same UNIX user and group that the service runs as.
- Ensure that the permissions are appropriate for the playbook directories and files.
- If you have not added any Ansible Playbook directories to the base project path, an error message is displayed. Choose one of the following options to troubleshoot this error:
  - Create the appropriate playbook directories and check out playbooks from your SCM.
  - Copy playbooks into the appropriate playbook directories.

Configure playbooks to use source control management (SCM) systems

Choose one of the following options when managing playbooks by using source control:

- SCM Types - Configuring playbooks to use Git and Subversion
- SCM Type - Configuring playbooks to use Red Hat Lightspeed

- SCM Type - Configuring playbooks to use a remote archive

Configure playbooks to use Git and Subversion SCM types

Configure automation controller projects to synchronize Ansible playbooks directly from Git and Subversion. Integrating with Source Control Management supports collaboration and helps ensure you always deploy the latest automation code.

By following these steps, you can ensure your environment always uses the latest version of your playbooks directly from your chosen SCM.

Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. Click the project name you want to use.
3. In the project **Details** tab, click **Edit project**.
4. Select the appropriate option (Git or Subversion) from the **Source control type** menu.
5. Enter the appropriate details into the following fields:
    - **Source control URL** - See an example in the tooltip .
    - Optional: **Source control branch/tag/commit**: Enter the SCM branch, tags, commit hashes, arbitrary refs, or revision number (if applicable) from the source control (Git or Subversion) to checkout. Some commit hashes and references might not be available unless you also give a custom refspec in the next field. If left blank, the default is `HEAD` which is the last checked out Branch, Tag, or Commit for this project.
    - **Source control refspec** - This field is an option specific to git source control and only advanced users familiar and comfortable with git should specify which references to download from the remote repository. For more information, see Job branch overriding.
    - **Source control credential** - If authentication is required, select the appropriate source control credential.
6. Optional: **Options** - select the launch behavior, if applicable:
    - **Clean** - Removes any local modifications before performing an update.
    - **Delete** - Deletes the local repository in its entirety before performing an update. Depending on the size of the repository this can significantly increase the amount of time required to complete an update.
    - **Track submodules** - Tracks the latest commit. There is more information in the tooltip ⓘ.

- **Update revision on launch** - Updates the revision of the project to the current revision in the remote source control, and caching the roles directory from Ansible Galaxy support or Collections support. Automation controller ensures that the local revision matches and that the roles and collections are up-to-date with the last update. In addition, to avoid job overflows if jobs are spawned faster than the project can synchronize, selecting this enables you to configure a Cache Timeout to cache previous project synchronizations for a given number of seconds.
- **Allow branch override** - Enables a job template or an inventory source that uses this project to start with a specified SCM branch or revision other than that of the project. For more information, see Job branch overriding.

7. Click **Save project**.

Configure playbooks to use Red Hat Lightspeed

Configure your projects to retrieve Ansible playbooks directly from Red Hat Lightspeed. Integrate with Red Hat Lightspeed to manage and deploy remediation playbooks identified through its analysis of your Red Hat Enterprise Linux environment.

This integration streamlines the process of addressing identified vulnerabilities and optimizing system configurations, ensuring your automation aligns with best practices and security recommendations.

Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. Click the project name you want to use.
3. In the project **Details** tab, click **Edit project**.
4. Select **Red Hat Lightspeed** from the **Source Control Type** menu.
5. In the **Red Hat Lightspeed credential** field, select the appropriate credential for use with Red Hat Lightspeed, as Red Hat Lightspeed requires a credential for authentication.
6. Optional: In the **Options** field, select the launch behavior, if applicable:
   - **Clean** - Removes any local modifications before performing an update.
   - **Delete** - Deletes the local repository in its entirety before performing an update. Depending on the size of the repository this can significantly increase the amount of time required to complete an update.
   - **Update revision on launch** - Updates the revision of the project to the current revision in the remote source control, and caches the roles directory from Ansible Galaxy support or Collections support. Automation controller ensures that the local revision matches, and that the roles and collections are up-to-date. If jobs are spawned faster than the project can synchronize, selecting this enables you to

configure a Cache Timeout to cache previous project synchronizations for a certain number of seconds, to avoid job overflow.

7. Click **Save project**.

Configure playbooks to use a remote archive

Playbooks that use a remote archive enable projects to be based on a build process that produces a versioned artifact, or release, containing all the requirements for that project in a single archive.

Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. Click the project name you want to use.
3. In the project **Details** tab, click **Edit project**.
4. Select **Remote Archive** from the **Source control type** menu.
5. Enter the appropriate details into the following fields:
    - **Source control URL** - requires a URL to a remote archive, such as a *GitHub Release* or a build artifact stored in *Artifactory* and unpacks it into the project path for use.
    - **Source control credential** - If authentication is required, select the appropriate source control credential.
6. Optional: In the **Options** field, select the launch behavior, if applicable:
    - **Clean** - Removes any local modifications before performing an update.
    - **Delete** - Deletes the local repository in its entirety before performing an update. Depending on the size of the repository this can significantly increase the amount of time required to complete an update.
    - **Update revision on launch** - Not recommended. This option updates the revision of the project to the current revision in the remote source control, and caches the roles directory from Ansible Galaxy support or Collections support.
    - **Allow branch override** - Not recommended. This option enables a job template that uses this project to launch with a specified SCM branch or revision other than that of the project's.

> **NOTE:**
> Since this source control type is intended to support the concept of unchanging artifacts, it is advisable to disable Galaxy integration (for roles, at a minimum).

7. Click **Save project**.

Updating projects from source control

Regularly updating your projects ensures your Ansible Automation Platform environment is synchronized with the latest versions of playbooks, roles, and collections from your integrated SCM repositories.

Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. Click the sync icon next to the project that you want to update.

> **NOTE:**
> Immediately after adding a project setup to use source control, a sync starts that fetches the project details from the configured source control.

- Click the project's status under the **Status** column for further information about the update process.

Result

This brings you to the **Output** tab of the **Jobs** section.

Reuse prebuilt automation by referencing roles

At the end of a project update, automation controller searches for the `requirements.yml` file in the `roles` directory, located at `<project-top-level-directory>/roles/requirements.yml`.

If this file is found, the following command automatically runs:

```
ansible-galaxy role install -r roles/requirements.yml -p <project-specific cache
location>/requirements_roles -vvv
```

This file enables you to reference Ansible Galaxy roles or roles within other repositories which can be checked out in conjunction with your own project. The addition of Ansible Galaxy access eliminates the need to create git submodules to achieve this result. Given that SCM projects,

along with roles or collections, are pulled into and executed from a private job environment, a `<private job directory>` specific to the project within `/tmp` is created by default.

The cache directory is a subdirectory inside the global projects folder. You can copy the content from the cache location to `<job private directory>/requirements_roles`.

By default, automation controller has a system-wide setting that enables you to dynamically download roles from the `roles/requirements.yml` file for SCM projects. You can turn off this setting in the **Job Settings** screen from the navigation panel **Settings > Automation Execution > Job**, by unchecking the **Enable Role Download** box.

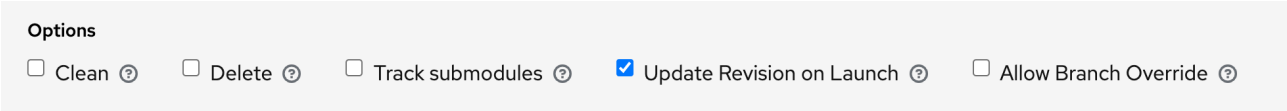
Whenever a project synchronization runs, automation controller determines if the project source and any roles from Galaxy or Collections are out of date with the project. Project updates download the roles inside the update.

If jobs need to pick up a change made to an upstream role, updating the project ensures that this happens. A change to the role means that a new commit was pushed to the *provision-role* source control.

To make this change take effect in a job, you do not have to push a new commit to the *playbooks* repository. You must update the project, which downloads roles to a local cache.

For instance, say you have two git repositories in source control. The first one is *playbooks* and the project in automation controller points to this URL. The second one is *provision-role* and it is referenced by the `roles/requirements.yml` file inside of the *playbooks* git repository.

Jobs download the most recent roles before every job run. Roles and collections are locally cached for performance reasons. You must select **Update revision on launch** in the project **Options** to ensure that the upstream role is re-downloaded before each job run:



The update happens much earlier in the process than the sync, so this identifies errors and details faster and in a more logical location.

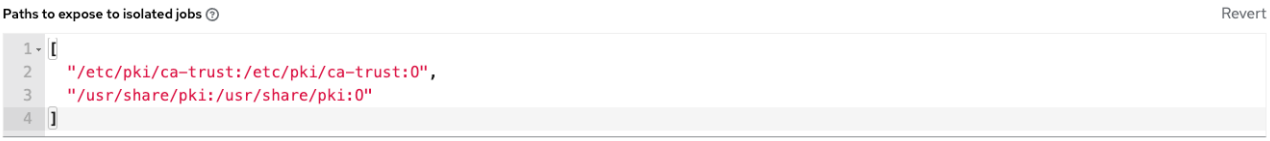
If there are any directories that must be specifically exposed, you can specify those in the **Job Settings** screen from the navigation panel **Settings > Automation Execution > Job**, in **Paths to expose to isolated Jobs**. You can also update the following entry in the settings file:

```
AWX_ISOLATION_SHOW_PATHS = ['/list/of/', '/paths']
```

NOTE:
If your playbooks need to use keys or settings defined in `AWX_ISOLATION_SHOW_PATHS`, you must add `AWX_ISOLATION_SHOW_PATHS` to `/var/lib/awx/.ssh`.

If you made changes in the settings file, be sure to restart services with the `automation-controller-service restart` command after your changes have been saved.

In the UI, you can configure these settings in the **Jobs Settings** window.



Configure collections from source management to run in a project

Automation controller supports project-specific Ansible collections in job runs.

If you specify a collections requirements file in the SCM at `collections/requirements.yml`, automation controller installs collections in that file in the implicit project synchronization before a job run.

Automation controller has a system-wide setting that enables collections to be dynamically downloaded from the `collections/requirements.yml` file for SCM projects. You can turn off this setting in the **Job Settings** screen from the navigation panel **Settings > Automation Execution > Job**, by unchecking the **Enable Collection(s) Download** box.

Roles and collections are locally cached for performance reasons, and you select **Update revision on launch** in the project **Options** to ensure this:

NOTE:
If you also have collections installed in your execution environment, the collections specified in the project's `requirements.yml` file will take precedence when running a job. This precedence applies regardless of the version of the collection. For example, if the collection specified in `requirements.yml` is older than the collection within the execution environment, the collection specified in `requirements.yml` is used.

Use collections with automation hub

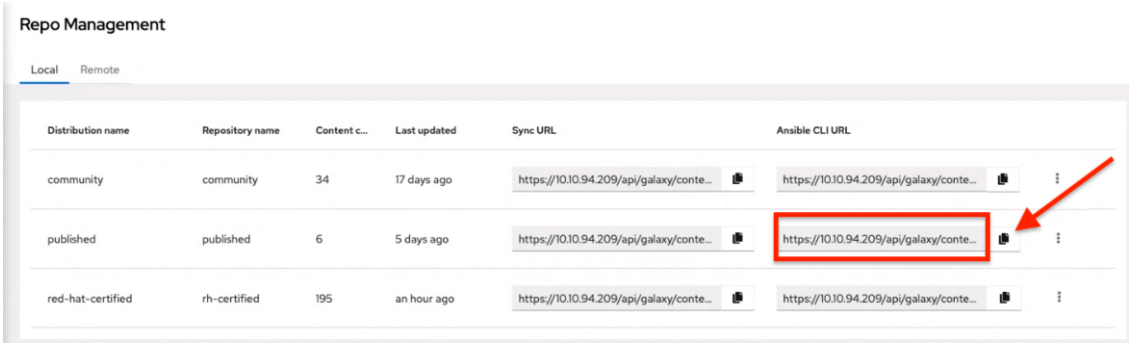
Before automation controller can use automation hub as the default source for collections content, you must create an API token in the automation hub UI. You then specify this token in automation controller.

Use the following procedure to connect to private automation hub or automation hub, the only difference is which URL you specify.

Procedure

1. Go to https://console.redhat.com/ansible/automation-hub/token.

2. Click **Load token**.
3. Click the copy icon to copy the API token to the clipboard.
4. Create a credential by choosing one of the following options:
    a. To use automation hub, create an automation hub credential by using the copied token and pointing to the URLs shown in the **Server URL** and **SSO URL** fields of the token page:
        - **Galaxy Server URL** = `https://console.redhat.com/ansible/automation-hub/token`
    b. To use private automation hub, create an automation hub credential using a token retrieved from the **Repo Management** dashboard of your private automation hub and pointing to the published repository URL as shown:



    You can create different repositories with different namespaces or collections in them. For each repository in automation hub you must create a different credential.
    c. Copy the **Ansible CLI URL** from the UI in the format of `/https://$<hub_url>/api/galaxy/content/<repo you want to pull from>` into the **Galaxy Server URL** field of **Create Credential**:

    For UI specific instructions, see Red Hat Certified, validated, and Ansible Galaxy content in automation hub.
5. Go to the organization for which you want to synchronize content from and add the new credential to the organization. This enables you to associate each organization with the credential, or repository, that you want to use content from.

    Example

    You have two repositories:

    - *Prod*: `Namespace 1` and `Namespace 2`, each with collection `A` and `B` so: `namespace1.collectionA:v2.0.0` and `namespace2.collectionB:v2.0.0`
    - *Stage*: `Namespace 1` with only collection `A` so: `namespace1.collectionA:v1.5.0`
    on , you have a repository URL for *Prod* and *Stage*.
    You can create a credential for each one.

    Then you can assign different levels of access to different organizations. For example, you can create a `Developers` organization that has access to both repository, while an Operations organization just has access to the **Prod** repository only.

For UI specific instructions, see Configuring user access for container repositories in private automation hub.

6. If automation hub has self-signed certificates, use the toggle to enable the setting **Ignore Ansible Galaxy SSL Certificate Verification** in **Job Settings**. For automation hub, which uses a signed certificate, use the toggle to disable it instead. This is a global setting:
7. Create a project, where the source repository specifies the necessary collections in a requirements file located in the `collections/requirements.yml` file.
8. In the **Projects** list view, click the sync icon to update this project. Automation controller fetches the Galaxy collections from the `collections/requirements.yml` file and reports it as changed. The collections are installed for any job template using this project.

NOTE:
If updates are required from Galaxy or Collections, a sync is performed that downloads the required roles, consuming that much more space in your /tmp file. In cases where you have a large project (around 10 GB), disk space on `/tmp` may be an issue.

Manage playbooks manually

Manage Ansible playbooks and directories directly on the automation controller filesystem. This approach helps ensure proper file ownership and permissions when you cannot use source control.

- Create one or more directories to store playbooks under the Project Base Path, for example, `/var/lib/awx/projects/`.
- Create or copy playbook files into the playbook directory.
- Ensure that the playbook directory and files are owned by the same UNIX user and group that the service runs as.
- Ensure that the permissions are appropriate for the playbook directories and files.
- If you have not added any Ansible Playbook directories to the base project path, an error message is displayed. Choose one of the following options to troubleshoot this error:
  - Create the appropriate playbook directories and check out playbooks from your SCM.
  - Copy playbooks into the appropriate playbook directories.

Configure project permissions

The set of permissions assigned to a project (role-based access controls) that provide the ability to read, change, and administer projects, inventories, job templates, and other elements are privileges.

To access the project permissions, select the **User Access** or **Team Access** tab of the **Projects** page. This screen displays a list of users that currently have permissions to this project.

You can sort and search this list by **Username**, **First Name**, or **Last Name**.

Add project permissions

Manage the permissions that users and teams have to access a project.

Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. Select the project that you want to update and click the **User Access** or **Team Access** tab.
3. Click **Add roles**.
4. Select a user or team to add and click **Next**.
5. Select one or more users or teams from the list by clicking the checkbox next to the name to add them as members.
6. Click **Next**.
7. Select the roles you want the selected users or teams to have. Different resources have different options available.
8. Click **Finish** to apply the roles to the selected users or teams and to add them as members. The updated roles assigned for each user and team are displayed.

Remove permissions from a project

Remove existing roles assigned to users and teams within an automation controller project. This action helps ensure that access is restricted only to the necessary functions and data.

Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. Select the project that you want to update and click the **User Access** or **Team Access** tab.
3. Click the ✖ icon next to the user role in the **Roles** column.
4. Click **Delete** in the confirmation window to confirm the disassociation.

Add project permissions

Manage the permissions that users and teams have to access a project.

Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. Select the project that you want to update and click the **User Access** or **Team Access** tab.
3. Click **Add roles**.
4. Select a user or team to add and click **Next**.
5. Select one or more users or teams from the list by clicking the checkbox next to the name to add them as members.
6. Click **Next**.
7. Select the roles you want the selected users or teams to have. Different resources have different options available.
8. Click **Finish** to apply the roles to the selected users or teams and to add them as members. The updated roles assigned for each user and team are displayed.

Remove permissions from a project

Remove existing roles assigned to users and teams within an automation controller project. This action helps ensure that access is restricted only to the necessary functions and data.

Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. Select the project that you want to update and click the **User Access** or **Team Access** tab.
3. Click the ✖ icon next to the user role in the **Roles** column.
4. Click **Delete** in the confirmation window to confirm the disassociation.

Enforce project integrity with signing and verification

Use project signing and verification in your project directory to sign files. You can then verify whether or not that content has changed in any way, or files have been added to, or removed from the project unexpectedly.

To do this, you require a private key for signing and a matching public key for verifying.

About project signing

Automation controller supports content signing for projects to ensure the integrity and authenticity of project files. This feature helps to prevent tampering with project content by ensuring that any changes to files are detected during project synchronization.

For project maintainers, the supported way to sign content is to use the `ansible-sign` utility, using the *command-line interface* (CLI) supplied with it.

The CLI aims to make it easy to use cryptographic technology such as *GNU Privacy Guard* (GPG) to validate that files within a project have not been tampered with in any way. Currently, GPG is the only supported means of signing and validation.

Automation controller is used to verify the signed content. After a matching public key has been associated with the signed project, automation controller verifies that the files included during signing have not changed, and that files have been added or removed unexpectedly. If the signature is not valid or a file has changed, the project fails to update, and jobs making use of the project will not launch. Verification status of the project ensures that only secure, untampered content can be run in jobs.

If the repository has already been configured for signing and verification, the usual workflow for altering the project becomes the following:

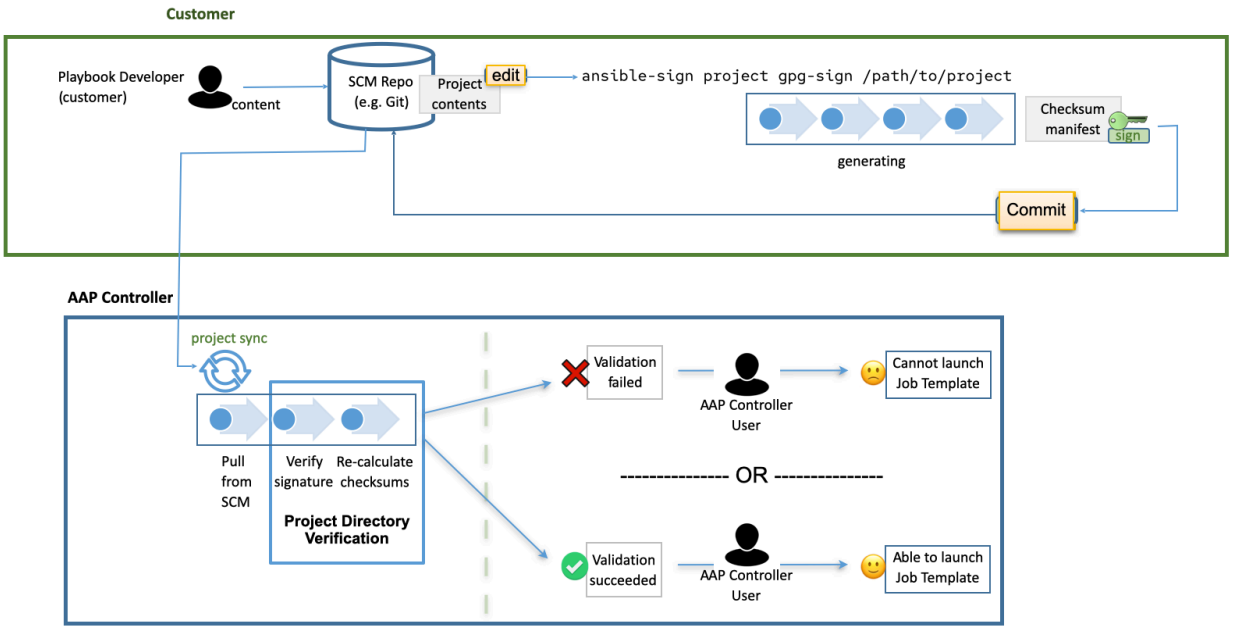
1. You have a project repository set up already and want to make a change to a file.
2. You make the change, and run the following command:

   ```
   ansible-sign project gpg-sign /path/to/project
   ```

   This command updates a checksum manifest and signs it.
3. You commit the change, the updated checksum manifest, and the signature to the repository.

When you synchronize the project, automation controller pulls in the new changes, checks that the public key associated with the project in automation controller matches the private key that the checksum manifest was signed with (this prevents tampering with the checksum manifest itself), then re-calculates the checksums of each file in the manifest to ensure that the checksum matches (and thus that no file has changed). It also ensures that all files are accounted for:

Files must be included in, or excluded from, the `MANIFEST.in` file. For more information on this file, see Sign a project. If files have been added or removed unexpectedly, verification fails



Sign content in automation projects

Prepare your Red Hat Enterprise Linux systems by enabling the correct repositories and subscriptions needed to install components for project signing. This step helps ensure that your environment is properly configured for content verification.

- RHEL nodes must properly be subscribed to:
  - RHEL subscription with **baseos** and **appstream** repositories must be enabled.
  - Your Red Hat Ansible Automation Platform subscription and the proper channel must be enabled:

```
ansible-automation-platform-2.5-for-rhel-8-x86_64-rpms for RHEL 8
ansible-automation-platform-2.5-for-rhel-9-x86_64-rpms for RHEL 9
```

- You require a valid GPG public or private keypair for signing content. For more information, see How to create GPG keypairs.
  For more information about GPG keys, see the GnuPG documentation.

  Verify that you have a valid GPG keypair in your default GnuPG keyring, with the following command:

```
gpg --list-secret-keys
```

  If this command produces no output, or one line of output that states, `trustdb was created`, then you do not have a secret key in your default keyring. In this case, see How to

create GPG keypairs to learn how to create a new keypair before proceeding. If it produces any other output, you have a valid secret key and are ready to use `ansible-sign`.

Add a GPG key to automation controller

You can add a GPG public key to automation controller and enable content signing and validation for your projects.

To use the GPG key for content signing and validation in automation controller, add it by running the following command in the CLI:

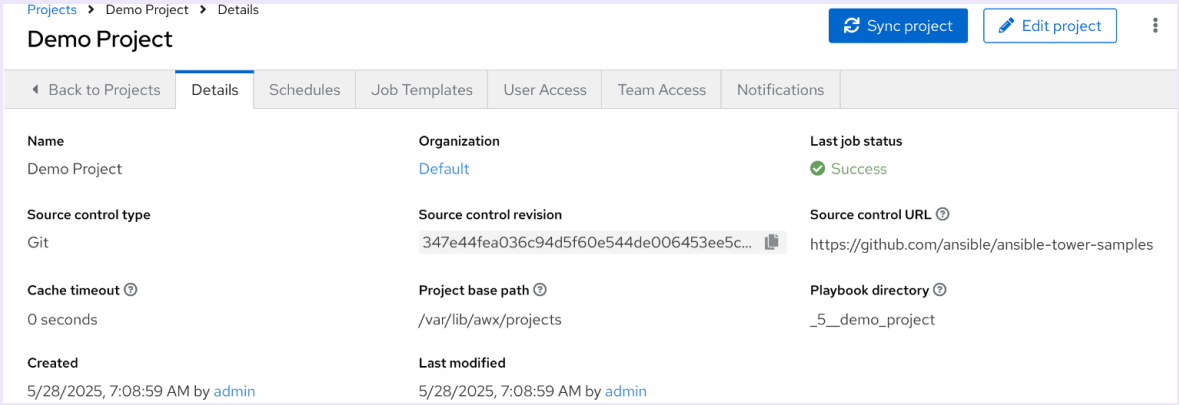
```
$ gpg --list-keys
$ gpg --export --armour <key fingerprint> > my_public_key.asc
```

Procedure

1. From the navigation panel, select **Automation Execution > Infrastructure > Credentials**.
2. Click **Create credential**.
3. Give a meaningful name for the new credential, for example, "Infrastructure team public GPG key".
4. In the **Credential type** field, select **GPG Public Key**.
5. Click **Browse** to locate and select the public key file, for example, `my_public_key.asc`.
6. Click **Create credential**.

   You can select this credential in `projects <ug_projects_add>`, and content verification automatically takes place on future project synchronizations.

NOTE:
Use the project cache SCM timeout to control how often you want automation controller to re-validate the signed content. When a project is configured to update on launch (of any job template configured to use that project), you can enable the cache timeout setting, which sets it to update after `N` seconds have passed since the last update. If validation is running too often, you can slow down how often project updates occur by specifying the time in the **Cache Timeout** field of the **Options Details** view of the project.



Install the ansible-sign CLI utility

Use the `ansible-sign` utility to offer options for you to sign and verify whether the project is signed.

Procedure

1. Run the following command to install `ansible-sign`:

```
$ dnf install ansible-sign
```

2. Verify that `ansible-sign` was successfully installed using the following command:

```
$ ansible-sign --version
```

Output similar to the following indicates that you have successfully installed `ansible-sign`:

```
ansible-sign 0.1
```

Sign a project

Signing a project involves an Ansible project directory.

The following sample project has a very simple structure: an inventory file, and two small playbooks under a playbooks directory:

```
$ cd sample-project/
$ tree -a .
.
├── inventory
└── playbooks
    └── get_uptime.yml
    └── hello.yml

    1 directory, 3 files
```

NOTE:
The commands used assume that your working directory is the root of your project. `ansible-sign project` commands take the project root directory as their last argument.

Use `.` to indicate the current working directory.

`ansible-sign` protects content from tampering by taking checksums (SHA256) of all of the secured files in the project, compiling those into a checksum manifest file, and then signing that manifest file.

To sign content, create a `MANIFEST.in` file in the project root directory that tells `ansible-sign` which files to protect.

Internally, `ansible-sign` uses the `distlib.manifest` module of Python's distlib library, therefore `MANIFEST.in` must follow the syntax that this library specifies. For an explanation of the `MANIFEST.in` file directives, see the Python Packaging User Guide.

In the sample project, two directives are included, resulting in the following `MANIFEST.in` file:

```
include inventory
recursive-include playbooks *.yml
```

With this file in place, generate your checksum manifest file and sign it. Both of these steps are achieved in a single `ansible-sign` command:

```
$ ansible-sign project gpg-sign .
```

Successful execution displays output similar to the following:

```
[OK   ] GPG signing successful!
[NOTE ] Checksum manifest: ./.ansible-sign/sha256sum.txt
[NOTE ] GPG summary: signature created
```

The project has now been signed.

Note that the `gpg-sign` subcommand is under the `project` subcommand.

For signing project content, every command starts with `ansible-sign project`.

Every `ansible-sign project` command takes the project root directory `.` as its final argument.

`ansible-sign` makes use of your default keyring and looks for the first available secret key that it can find, to sign your project. You can specify a specific secret key to use with the `--fingerprint` option, or even a completely independent GPG home directory with the `--gnupg-home` option.

NOTE:
If you are using a desktop environment, GnuPG automatically prompts you for your secret key's passphrase.

If this functionality does not work, or you are working without a desktop environment, for example, through SSH, you can use the `-p --prompt-passphrase` flag after `gpg-sign`, which causes `ansible-sign` to prompt for the password instead.

Note that an `.ansible-sign` directory was created in the project directory. This directory contains the checksum manifest and a detached GPG signature for it.

```
$ tree -a .
.
├── .ansible-sign
│   ├── sha256sum.txt
│   └── sha256sum.txt.sig
├── inventory
├── MANIFEST.in
└── playbooks
    ├── get_uptime.yml
    └── hello.yml
```

Related information
Python Packaging User Guide

Verify your project

To verify that a signed Ansible project has not been altered, you can use `ansible-sign` to check whether the signature is valid and that the checksums of the files match what the checksum manifest says they should be.

The `ansible-sign project gpg-verify` command can be used to automatically verify both of these conditions.

```
$ ansible-sign project gpg-verify .
[OK   ] GPG signature verification succeeded.
[OK   ] Checksum validation succeeded.
```

NOTE:
By default, `ansible-sign` makes use of your default GPG keyring to look for a matching public key. You can specify a keyring file with the `--keyring` option, or a different GPG home with the `--gnugpg-home` option.

If verification fails for any reason, information is displayed to help you debug the cause. More verbosity can be enabled by passing the global `--debug` flag, immediately after `ansible-sign` in your commands.

NOTE:
When a GPG credential is used in a project, content verification automatically takes place on future project synchronizations.

Automate signing

In environments with highly-trusted *Continuous Integration* (CI) environments such as OpenShift or Jenkins, it is possible to automate the signing process.

For example, you can store your GPG private key in a CI platform of choice as a secret, and import that into GnuPG in the CI environment. You can then run through the signing workflow within the normal CI environment.

When signing a project by using GPG, the environment variable `ANSIBLE_SIGN_GPG_PASSPHRASE` can be set to the passphrase of the signing key. This can be injected and masked or secured in a CI pipeline.

Depending on the scenario, `ansible-sign` returns with a different exit-code, during both signing and verification. This can also be useful in the context of CI and automation, as a CI environment can act differently based on the failure. For example, it can send alerts for some errors, but fail silently for others.

These are the current exit codes used in `ansible-sign`, which can be considered stable:

Exit code	Approximate meaning	Example scenarios
0	Success	• Signing was successful • Verification was successful
1	General failure	• The checksum manifest file contained a syntax error during verification • The signature file did not exist during verification • `MANIFEST.in` did not exist during signing
2	Checksum verification failure	• The checksum hashes calculated during verification differed from what was in the signed checksum manifest, for example, a project file was changed but the signing process was not re-completed.
3	Signature verification failure	• The signer's public key was not in the user's GPG keyring • The wrong GnuPG home directory or keyring file was specified • The signed checksum manifest file was modified in some way
4	Signing process failure	• The signer's private key was not found in the GPG keyring • The wrong GnuPG home directory or keyring file was specified

Launch automation templates from self-service automation portal

Learn how to access and authenticate to the self-service automation portal using your Ansible Automation Platform credentials.

Sign in to self-service automation portal

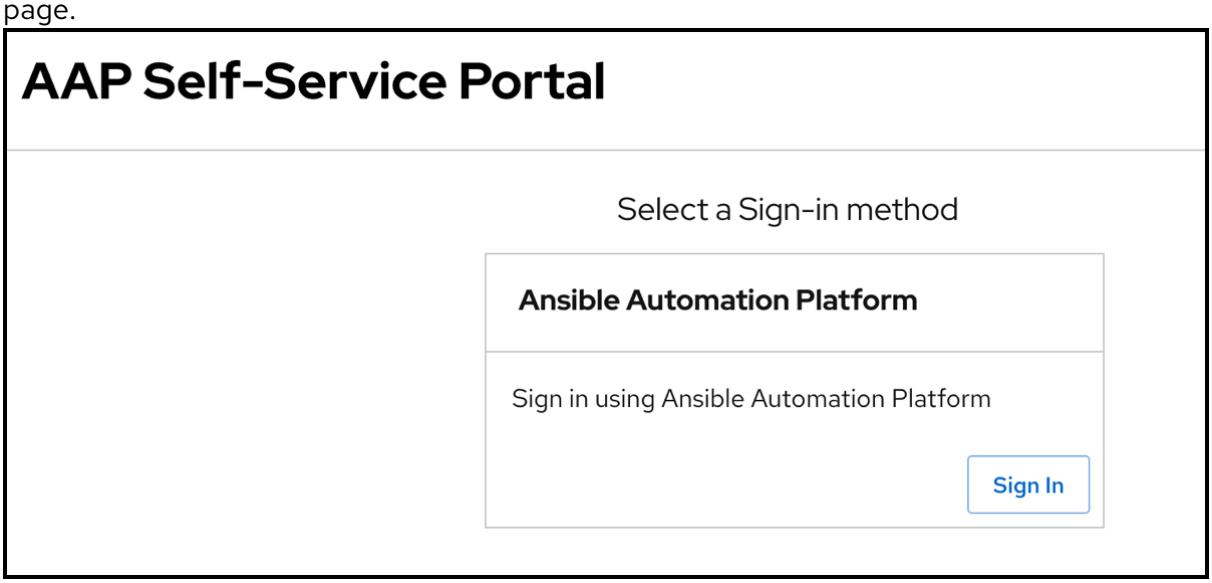
Log in to the deployed self-service automation portal using your existing Ansible Automation Platform credentials. The portal uses these credentials for authentication.

Before you begin

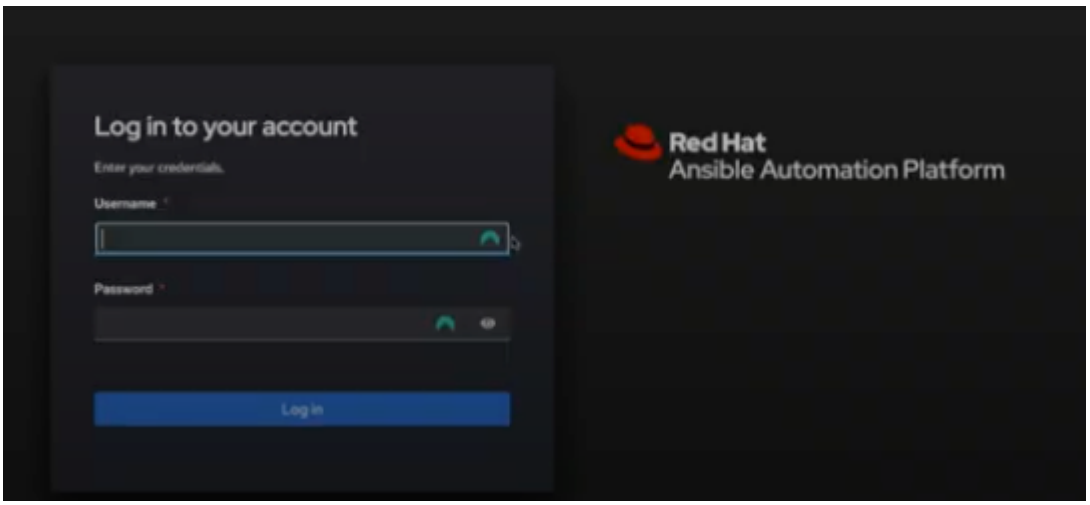
- You have configured an OAuth application in Ansible Automation Platform for self-service automation portal.
- You have configured a user account in Ansible Automation Platform.

Procedure

1. In a browser, navigate to the URL for self-service automation portal to open the sign-in page.



2. Click **Sign In**.
3. The sign-in page for Ansible Automation Platform appears:



4. Enter your Ansible Automation Platform credentials and click **Log in**.
5. The self-service automation portal web console opens.

If you are using custom or self-signed SSL certificates and when attempting to log in to self-service automation portal, it displays the error:

```
Login failed; caused by Error: Failed to send POST request: fetch failed
```

This error indicates that self-service automation portal cannot verify the SSL certificate from your Ansible Automation Platform instance.

To resolve this issue, configure self-service automation portal to trust your custom CA certificate.

NOTE:
While you can disable SSL validation by setting `checkSSL: false` in the Helm chart configuration, this approach is not recommended for production environments as it reduces security. Instead, configure self-service automation portal to trust your custom CA certificate.

View templates

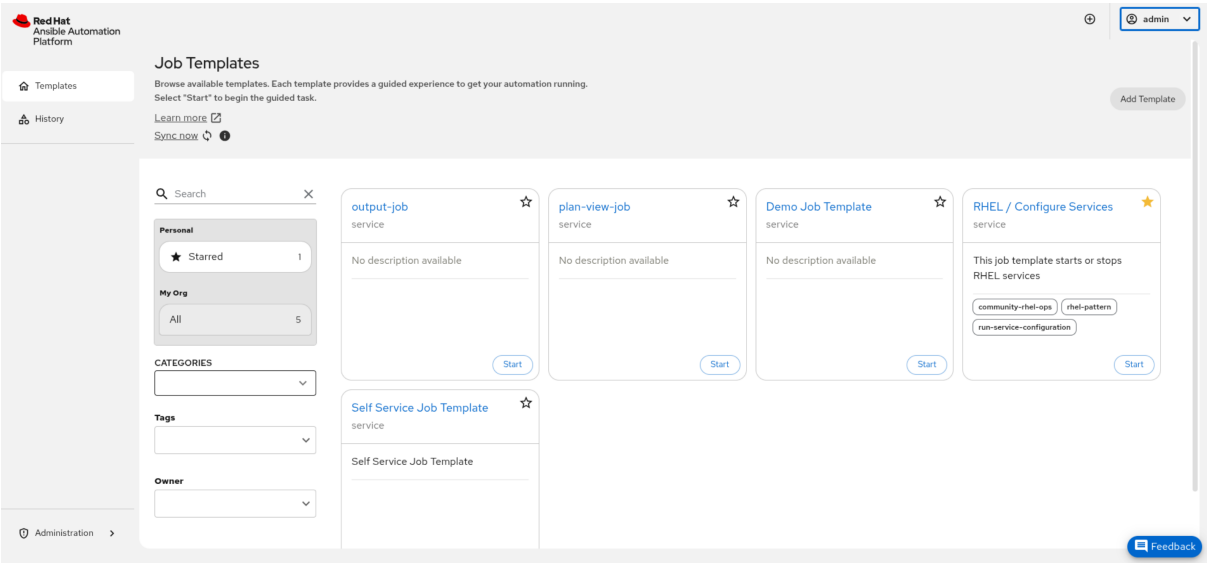
View your accessible automation templates, displayed as tiles, on the **Templates** landing page of the self-service automation portal console.

Before you begin

- You have signed in to self-service automation portal.

Procedure

1. In a browser, sign in to self-service automation portal.
2. In the navigation pane, select **Templates** to open a landing page where tiles are displayed, representing the templates that you have access to.



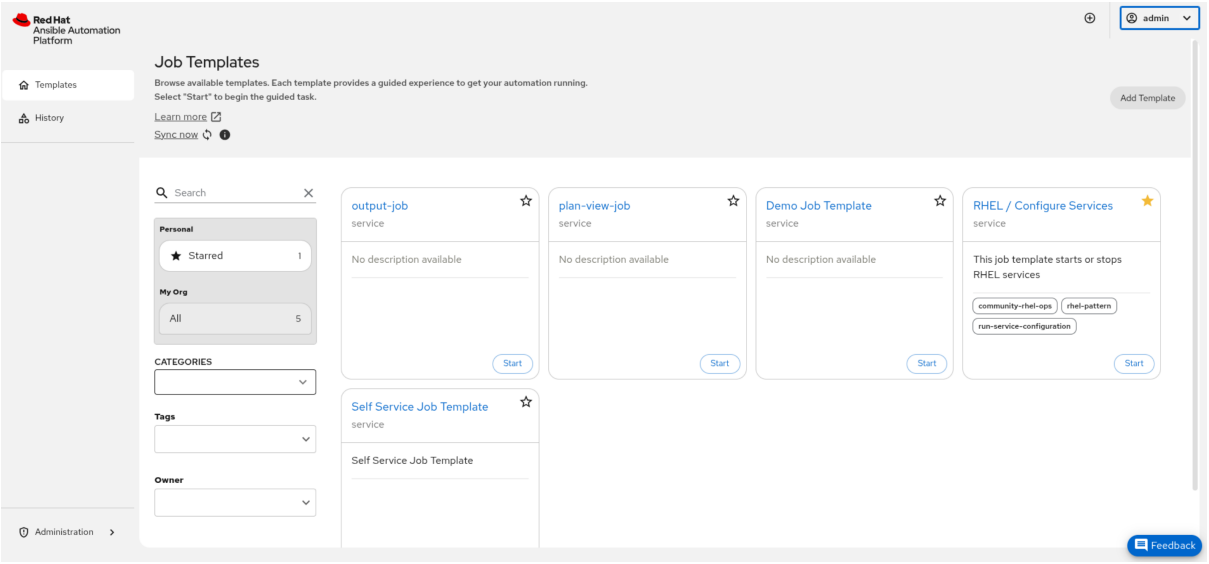
Synchronize auto-generated templates

The auto-generated self-service templates are synchronized from the job templates in Ansible Automation Platform to self-service automation portal. You can manually trigger a sync from the **Templates** page.

NOTE:
This synchronization fetches updates for auto-generated self-service templates only.

Procedure

1. In a browser, sign in to self-service automation portal.
2. In the navigation pane, select **Templates** to display the templates that you have access to.



3. Select **Sync now** to launch a synchronization.

Configure custom SSL certificates for self-service automation portal

If your Ansible Automation Platform instance uses custom or self-signed SSL certificates, you must configure self-service automation portal to trust those certificates to avoid SSL verification errors.

Before you begin

- You have administrator access to your OpenShift Container Platform cluster.
- You have the custom Certificate Authority (CA) certificate file used by your Ansible Automation Platform instance.
- Self-service automation portal is installed in your OpenShift Container Platform cluster.

Procedure

1. Obtain the CA certificate file from your Ansible Automation Platform instance.

   If you do not have the CA certificate file, you can extract it from your Ansible Automation Platform server:

   ```
   openssl s_client -showcerts -connect <aap-hostname>:443 </dev/null 2>/dev/null
   | openssl x509 -outform PEM > aap-ca-cert.pem
   ```

   Replace `<aap-hostname>` with your Ansible Automation Platform hostname.
2. Log in to your OpenShift Container Platform cluster with administrator privileges.
3. Create a ConfigMap containing your custom CA certificate:

   ```
   oc create configmap custom-ca-bundle \
     --from-file=ca-bundle.crt=aap-ca-cert.pem \
     -n <namespace>
   ```

   Replace `<namespace>` with the namespace where self-service automation portal is installed.
4. Update your self-service automation portal Helm chart values to mount the custom CA certificate:

```
upstream:
  backstage:
    extraEnvVarsSecrets:
      - custom-ca-bundle
    extraVolumes:
      - name: custom-ca
        configMap:
          name: custom-ca-bundle
    extraVolumeMounts:
      - name: custom-ca
        mountPath: /etc/pki/ca-trust/source/anchors/
        readOnly: true
```

5. Apply the updated configuration by upgrading the self-service automation portal Helm chart:

```
helm upgrade <release-name> <chart-name> \
  -f values.yaml \
  -n <namespace>
```

   Replace `<release-name>` with your Helm release name and `<chart-name>` with the self-service automation portal chart name.

6. Wait for the self-service automation portal pods to restart with the new configuration.

Result

1. Verify that the self-service automation portal pods are running:

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

   All self-service automation portal pods should show a status of `Running`.

2. Attempt to sign in to self-service automation portal using your Ansible Automation Platform credentials.
   If the SSL certificate configuration is correct, you can authenticate successfully without SSL verification errors.

3. Check the self-service automation portal logs for SSL-related errors:

```
oc logs -n <namespace> <pod-name> | grep -i ssl
```

If you see no SSL verification errors, the custom CA certificate is trusted correctly.

If you continue to experience SSL verification errors after following this procedure:

- Verify that the CA certificate file contains the complete certificate chain.
- Ensure that the certificate file is in PEM format.
- Confirm that the Ansible Automation Platform hostname in your configuration matches the hostname in the SSL certificate.
- Check that the `checkSSL` parameter in your Helm values is set to `true` (the default). Setting it to `false` disables SSL verification entirely, which is not recommended for production environments.

Set up permissions for custom self-service templates

Custom self-service templates that you set up in self-service automation portal use a different set of RBAC rules than the RBAC rules for auto-generated self-service templates that are synchronized from Ansible Automation Platform.

- **Auto-generated self-service templates**: The RBAC settings for these templates are synchronized from Ansible Automation Platform.
- **Custom self-service templates**: You must set up RBAC for these templates in self-service automation portal.

Set up RBAC for custom self-service templates

By default, Ansible Automation Platform administrators can define self-service automation portal RBAC roles.

Before you begin

- You have created a user, for example `example-user`.
- You have added this user as a member of a team, for example `example-team`.

This procedure describes how to create a role in self-service automation portal that allows only a selected team to view and execute particular custom self-service templates.

Custom self-service templates in self-service automation portal are associated with job templates in Ansible Automation Platform. This association is set in the `steps.actions` section of the YAML file for the custom self-service template.

If you assign permissions to a particular team to launch a custom self-service template from self-service automation portal, then you must make sure that that team has permission to run the associated job templates in Ansible Automation Platform.

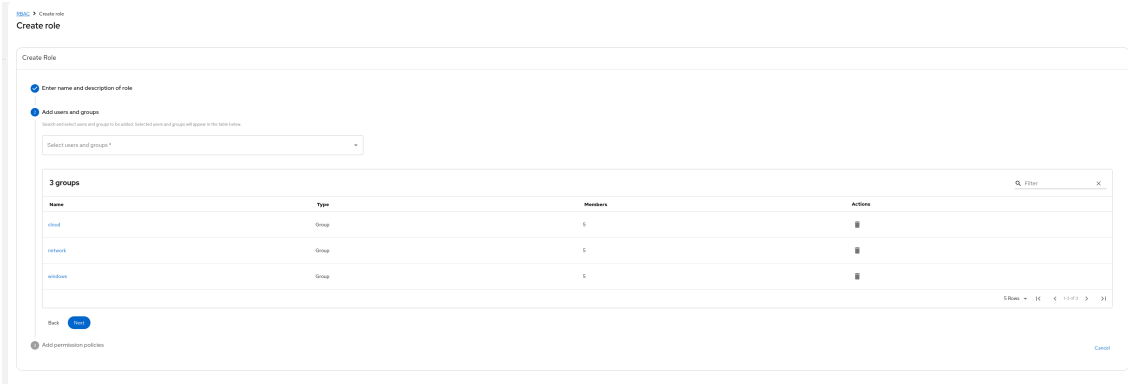
Procedure

1. In a browser, log in to your self-service automation portal instance as an Ansible Automation Platform user with Ansible Automation Platform administrator privileges.
2. In the navigation panel, select **Administration > RBAC**.
3. In the **RBAC** view, click **Create**.

   The **Create Role** view appears.

   a. Enter a name for the role.

   b. In the **Users and Groups** section, select the Ansible Automation Platform teams and users to assign to this role.



> **NOTE:**
> The **Members** column displays the total count of users in each team, including both regular team members and administrators.

   c. In the **Add Permission policies** section, select the plug-ins that you want to enable for the role.

   d. Select **Permission** in the list of plug-ins to configure the fine-grained permission policies for the role.

4. Click **Next**.
5. Review the settings that you have selected for the role.
6. Click **Create** to create the role.

Related information
Creating a user
Adding users to a team

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.

Verify RBAC

This procedure describes how to verify that the role you set up is working correctly.

Procedure

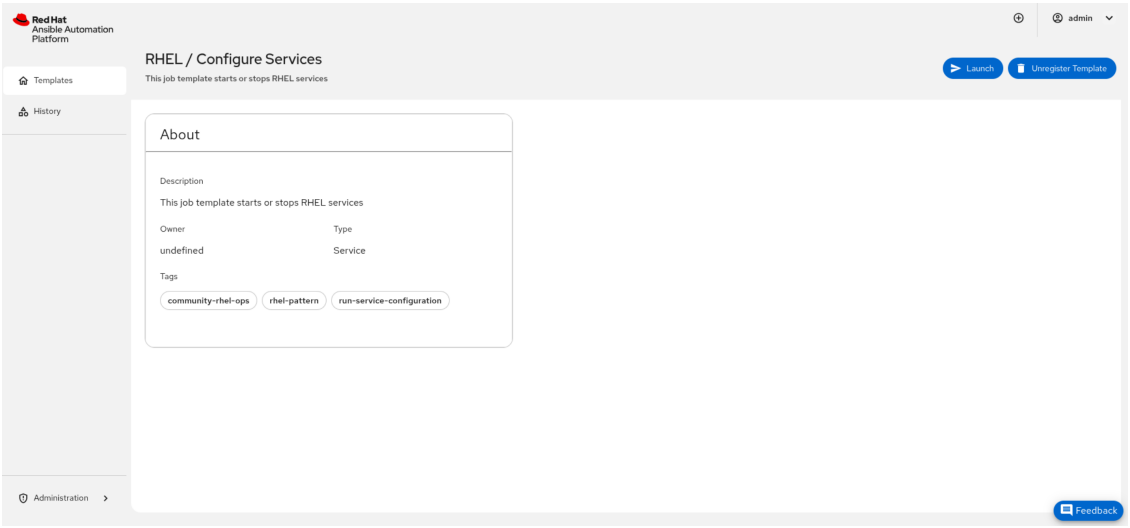
1. Verify that users with permissions can use a template:
    a. Log in to self-service automation portal as a user who is a member of a team that has been enabled to use a role.
    b. Verify that RBAC is applied and that the user can use the templates that you enabled for the role.
2. Log out of self-service automation portal.
3. Verify that users without permissions can not see or use a template:
    a. Log in to self-service automation portal as a user who is not a member of the new team that has been enabled to use the role.
    b. Verify that RBAC is applied and that the user cannot use the templates that you enabled for the role.
4. Log out of self-service automation portal.

Deregister custom self-service templates

You can deregister custom self-service templates. Deregistering templates deletes them from the **Templates** view in the self-service automation portal console.

Procedure

1. In a browser, log in to self-service automation portal as a user with administrative privileges.
2. Select **Templates** to display the self-service templates.
3. For each custom self-service template that you want to delete, execute the following steps:
    a. Click a custom self-service template to open the **Template detail** view. The navigation bar contains the **Unregister Template** option.
    b. Click **Unregister Template**.



c. In the dialog, confirm that you want to deregister the template.

d. Click **Delete Entity** to unregister the template.

Result

In a browser, navigate to the **Templates** view for your self-service automation portal instance. Verify that the custom self-service templates that you deregistered have been deleted.

Understanding auto-generated templates

Self-service automation portal automatically generates templates from Ansible Automation Platform Job Templates. Each Ansible Automation Platform Job Template with the appropriate configuration becomes a template that users can execute from self-service automation portal.

NOTE:
Templates in self-service automation portal use Backstage Software Templates as the underlying technology. For details on supported usage, see the self-service automation portal support policy.

Auto-generated templates include:

- Form fields generated from Ansible Automation Platform Job Template Surveys and "Prompt on Launch" options.
- Metadata (name, description, labels) mapped from Ansible Automation Platform Job Template properties.
- A single step that launches the Ansible Automation Platform Job Template using the `rhaap:launch-job-template` action.
- Output that displays the job execution results to the user.

Users only see and execute templates for Ansible Automation Platform Job Templates they have Job Template Execute permission in Ansible Automation Platform.

Related information
Backstage Software Templates
Self-service automation portal support policy

What users see

Auto-generated templates appear in the template catalog alongside custom templates. Users browse or search the catalog and select a template to run.

When a user opens an auto-generated template, self-service automation portal checks the Ansible Automation Platform Job Template for **Prompt on Launch** options and Survey questions. If the Job Template has **Prompt on Launch** options or Survey questions, self-service automation portal displays a form. Users select Ansible Automation Platform resources by name (such as Inventories or Credentials) and answer survey questions (such as application name or environment). Self-service automation portal handles authentication automatically, so credentials are not visible on the form. If the Job Template has no **Prompt on Launch** options or Survey questions, the Job Template starts running automatically without displaying a form.

After the user submits the form or execution starts automatically, self-service automation portal launches the Ansible Automation Platform Job Template and displays an output page. The output page shows the job ID, job status, and optional links such as a direct link to the job in Ansible Automation Platform.

Users do not need to understand YAML or the template structure. The form collects the required input, and the output page displays the results.

Auto-generated template example

The following example shows a template that self-service automation portal auto-generates from an existing Ansible Automation Platform Job Template.

Every field in the generated YAML maps directly to an Ansible Automation Platform artifact:

- The Ansible Automation Platform Job Template name and description become `metadata` fields.
- Each **Prompt on Launch** option becomes an `AAPResourcePicker` field.
- Each Ansible Automation Platform Survey question becomes a standard form field.
- The `steps` section launches the original Ansible Automation Platform Job Template using the `rhaap:launch-job-template` action.

```
apiVersion: scaffolder.backstage.io/v1beta3
kind: Template
metadata:
  # Namespace is hardcoded to "default"
  namespace: default
  # AAP artifact: Job Template name -> lowercase with hyphens
  name: deploy-application
  # AAP artifact: Job Template name -> title (copied directly)
  title: Deploy Application
  # AAP artifact: Job Template description -> description (copied directly)
  description: Deploy application to target environment
  # AAP artifact: Job Template labels -> tags (lowercase)
  tags:
    - automation
    - aap
  # AAP artifact: Job Template URL -> annotations
  annotations:
    backstage.io/techdocs-ref: url:https://aap.example.com/#/templates/job_template/
42
    backstage.io/source-location: url:https://aap.example.com/#/templates/
job_template/42


spec:
  type: service


  parameters:
    - title: "Please enter the following details"
      required:
        - token
        - inventory
        - app_name
      properties:
        # Always present: OAuth token for AAP authentication (hidden from user)
        token:
          title: Token
          type: string
          description: Oauth2 token
```

```
          ui:field: AAPTokenField
          ui:widget: hidden
          ui:backstage:
            review:
              show: false
          ui:options:
            disabled: true
            hidden: true

        # AAP artifact: "Prompt on Launch" -> Inventory
        inventory:
          title: Inventory
          description: Please enter the inventory you want to use the services on
          resource: inventories
          ui:field: AAPResourcePicker
          default: Production Servers

        # AAP artifact: Survey question -> "Application Name"
        app_name:
          title: Application Name
          description: Name of the application to deploy
          type: string
          default: my-app

steps:
  - id: launch-job
    # AAP artifact: Job Template name used as the step name
    name: Deploy Application
    action: rhaap:launch-job-template
    input:
      token: ${{ parameters.token }}
      values:
        # AAP artifact: Job Template name identifies which template to launch
        template: Deploy Application
        # "Prompt on Launch" values passed as launch parameters
        inventory: ${{ parameters.inventory }}
```

```
          # Survey answers passed as extra variables to AAP
          extraVariables:
            app_name: ${{ parameters.app_name }}


  output:
    text:
      - title: Deploy Application template executed successfully
        content: |
          **Job ID:** ${{ steps['launch-job'].output.data.id }}
          **Job status:** ${{ steps['launch-job'].output.data.status }}
```

Field mapping

The following table maps each generated field to its Ansible Automation Platform source:

Ansible Automation Platform source	Ansible Automation Platform value	Generated field	Transformation
N/A	`default`	`metadata.namespace`	Hardcoded to `default`.
Job Template > Name	Deploy Application	`metadata.name`	Lowercase with hyphens.
Job Template > Name	Deploy Application	`metadata.title`	Copied directly.
Job Template > Description	Deploy application to target environment	`metadata.description`	Copied directly.
Job Template > Labels	`automation`, `aap`	`metadata.tags`	Lowercase, special characters replaced with hyphens.
Always present	OAuth2 token	`parameters.token`	Auto-populated and hidden from user.
Prompt on Launch > Inventory	Production Servers	`parameters.inventory`	`AAPResourcePicker`; user selects by name, resolved to Ansible Automation Platform ID at launch.
Survey > Question 1	app_name (text, required, default: "my-app")	`parameters.app_name`	Variable, title, type, and default copied from Survey spec.

Ansible Automation Platform source	Ansible Automation Platform value	Generated field	Transformation
Job Template > Name	Deploy Application	`input.values.template`	Identifies which Ansible Automation Platform Job Template to launch.
Prompt on Launch value	`${{ parameters.inventory }}`	`input.values.inventory`	Name resolved to Ansible Automation Platform ID at launch.
Survey answers	`${{ parameters.app_name }}`	`input.values.extraVariables`	Passed as `extra_vars` to the Ansible Automation Platform Job.

How templates are generated

When synchronization runs, self-service automation portal reads Ansible Automation Platform Job Template configurations and generates corresponding templates.

Metadata mapping

The following table shows how Ansible Automation Platform Job Template properties are mapped to template metadata fields:

Ansible Automation Platform Job Template property	Generated template field	Transformation
Name	`metadata.name`	Converted to lowercase with hyphens
Name	`metadata.title`	Copied directly
Description	`metadata.description`	Copied directly
Labels	`metadata.tags`	Converted to lowercase, special characters replaced with hyphens
N/A	`metadata.namespace`	Hardcoded to `default`

Parameter mapping

The following table shows how Ansible Automation Platform Job Template sources are mapped to template parameters:

Ansible Automation Platform Job Template source	Generated template parameter type
Survey questions	Standard form fields (`type: string` , `enum` , `ui:widget`)
"Prompt on Launch" options	`AAPResourcePicker` fields for Ansible Automation Platform resources

Field order

Self-service automation portal generates form fields from two sources in the Ansible Automation Platform Job Template configuration. Fields appear in the following order:

1. **OAuth token (hidden):** Auto-populated with the user's authentication token. This field is always present and always hidden.
2. **"Prompt on Launch" fields:** Each enabled "Prompt on Launch" option becomes an `AAPResourcePicker` field. Users select Ansible Automation Platform resources by name (Inventories, Credentials, Execution Environments). Self-service automation portal resolves names to Ansible Automation Platform internal IDs at launch time.
3. **Survey questions:** Each survey question becomes a form field with the matching input type (`text` , `dropdown` , `password` , `textarea` , `integer` , `float` , `multiplechoice` , `multiselect`).

Defaults and required fields

- Survey question defaults and required/optional settings are preserved from the Ansible Automation Platform Job Template Survey.
- "Prompt on Launch" fields use the current Ansible Automation Platform Job Template settings for defaults and required/optional status.
- Ansible Automation Platform resource picker fields display resource names, not internal IDs.

Add and launch custom self-service templates

Custom self-service templates are stored as YAML files in repositories in GitHub or Gitlab. When a user launches a software template from self-service automation portal, they must fill in a form with the values needed to run the associated job template in Ansible Automation Platform.

The custom self-service template YAML file must have a `token:` section that includes a `ui:field:` key for the authentication token for Ansible Automation Platform. This generates a field for the token in the form that appears when the user launches the template in self-service automation portal. The user enters the token: it is used to authenticate job template execution in Ansible Automation Platform.

The following example shows the `token:` section in a template. For security reasons, set the value of `token.ui:backstage.review.show` to `false` to ensure that the token is not visible to the user.

```
spec:
  ...
  parameters:
    ...
      properties:
        token:
          title: Token
          type: string
          description: Oauth2 token
          ui:field: AAPTokenField
          ui:widget: hidden
          ui:backstage:
            review:
              show: false
```

NOTE:
Setting the `ui:widget: hidden` field hides the Red Hat Ansible Automation Platform token input in the form.

Add a template to self-service automation portal

You can add a custom self-service template to the **Templates** view of your self-service automation portal instance. Custom self-service templates are stored in git repositories. self-service automation portal supports GitLab and GitHub Source Control Management (SCM).

Before you begin

- You have created repositories in your Git SCM for the templates that you want to use.
- In the git repository for your custom templates, ensure that the `metadata.name` field is unique and does not match an existing auto-generated template or another custom self-service template. For example, append `*-custom` to the value of the `metadata.name` key.

```
metadata:
  name: provision-database-custom
```

- You must be logged in to self-service automation portal as an Ansible Automation Platform platform administrator.

NOTE:
Names for custom self-service templates must be unique. Custom self-service templates must have a different name to auto-generated job templates and also to other custom self-service templates.

Procedure

1. In a browser, navigate to your self-service automation portal instance and sign in with your Ansible Automation Platform credentials.
2. Navigate to the **Templates** Page.
3. Click **Add template**.
4. Enter a valid Git SCM URL for the template that you want to add.
5. Click **Analyze** to fetch the template.
6. After the template has been fetched, review the list of what will be imported and added to the catalog.
7. Click **Import**.

Result

After the import is complete, return to the **Templates** page to view the newly created template. You can now launch your template.

Next steps

- You must configure RBAC for your imported custom templates to allow users to view and run them. To do this, you must be logged into self-service automation portal as a platform administrator.
  For more information, see Setting up RBAC for custom self-service templates.

Launch a template

This procedure describes how to launch a template from a tile in the **Templates** view of your self-service automation portal instance.

Before you begin

- You have configured RBAC in Ansible Automation Platform for templates that are associated with Ansible Automation Platform job templates.

Procedure

1. In a browser, navigate to your self-service automation portal instance and sign in with your Ansible Automation Platform credentials.
2. Navigate to the **Templates** page. The templates you have set up are displayed as tiles on the page.
3. In the template that you want to launch, click **Start**.
   A description of the template is displayed.
4. Click **Launch** to begin configuring the parameters for running the template.
5. Fill out the required fields.
6. Click **Next**.
7. Review the entered information.
8. Click **Create** to launch the template.
9. The progress for the template execution is displayed.

Use custom actions and UI components in Backstage Software Templates

The self-service automation portal is built on a Red Hat Developer Hub base image. It uses Backstage Software Templates, which are YAML-based workflow definitions that provide user forms to execute automation tasks in Ansible Automation Platform.

Overview

Learn how to use the custom actions, filters, and UI components provided by the **Ansible Backstage Plugins** to create and manage custom software templates for the self-service automation portal.

IMPORTANT:
Red Hat does not support the use of the Red Hat Developer Hub image for standalone purposes outside the scope of the Ansible Automation Platform self-service automation portal functionality. Refer to the official support policy for details.

Ansible backstage plugins

The portal's capabilities are delivered through **Ansible Backstage Plugins** that extend Red Hat Developer Hub functionality; the base Red Hat Developer Hub image is not customized. These plugins provide additional Backstage actions and filters that you use to create custom software templates.

Plugin	Functionality
auth-backend-module-rhaap-provider	Provides OAuth 2.0 authentication with Ansible Automation Platform.
catalog-backend-module-rhaap	Synchronizes Ansible Automation Platform job templates as Backstage Software Templates.
scaffolder-backend-module-backstage-rhaap	Provides the `rhaap:launch-job-template` action.
backstage-rhaap-common	Contains shared libraries and utilities for Ansible Automation Platform integration.
self-service	Provides the user interface for all listed functionality.

Custom Backstage actions

The following actions enable you to manage Ansible Automation Platform resources within a software template workflow.

`rhaap:create-project`

Create an Ansible Automation Platform project that links to a source control repository containing Ansible content.

Input parameters

Parameter	Type	Required	Description
`token`	string	Yes	OAuth2 token for Ansible Automation Platform authentication.
`deleteIfExist`	boolean	No	If `true`, the action deletes the project if it already exists before creating a new one.
`values`	object	Yes	The project configuration object. See the "values" Object Structure table.

“values” Object Structure

Field	Type	Required	Description
`projectName`	string	Yes	Name of the project.
`projectDescription`	string	No	Description of the project.
`organization`	object	Yes	Organization object with `id` (number, required) and `name` (string, optional).
`credentials`	object	No	Credential object with `id` (number, required), `name` (string, optional), and `kind` (string, optional).
`scmUrl`	string	Yes	Source control URL (for example, GitHub/GitLab repository URL).
`scmBranch`	string	No	Source control branch, tag, or commit.

Field	Type	Required	Description
`scmUpdateOnLaunch`	boolean	No	If `true`, updates the project revision before each job run.

Output parameters

Parameter	Type	Description
`project`	object	Created project details.
`project.id`	number	Project ID in Ansible Automation Platform (AAP).
`project.name`	string	Project name.
`project.description`	string	Project description.
`project.url`	string	Ansible Automation Platform URL for the project.

Example

```
steps:
  - id: create-aap-project
    name: Create AAP Project
    action: rhaap:create-project
    input:
      token: ${{ parameters.AAP_TOKEN }}
      deleteIfExist: true
      values:
        projectName: ${{ parameters.projectName }}
        organization: ${{ parameters.organization }}
        scmUrl: https://github.com/my-org/ansible-playbooks
        scmBranch: main
        scmUpdateOnLaunch: true
        credentials: GitHub Token
```

`rhaap:create-execution-environment`

Create an execution environment in Ansible Automation Platform that defines the container image used to run Ansible jobs.

Input parameters

Parameter	Type	Required	Description
`token`	string	Yes	OAuth2 token for Ansible Automation Platform authentication.
`deleteIfExist`	boolean	No	If `true`, the action deletes the execution environment if it already exists.
`values`	object	Yes	The execution environment configuration object.

“values” Object Structure

Field	Type	Required	Description
`environmentName`	string	Yes	Name of the execution environment.
`environmentDescription`	string	No	Description of the execution environment.
`organization`	object	Yes	Organization object with required `id` (number) and optional `name` (string).
`image`	string	Yes	Full image location, including registry, image name, and tag (for example, `quay.io/ansible/creator-ee:latest`).
`pull`	string	No	Image pull policy: `always`, `missing`, or `never`. Default is `missing`.

Output parameters

Parameter	Type	Description
`executionEnvironment`	object	Created execution environment details.
`executionEnvironment.id`	number	Execution environment ID.

Parameter	Type	Description
`executionEnvironment.name`	string	Execution environment name.
`executionEnvironment.description`	string	Execution environment description.
`executionEnvironment.url`	string	Ansible Automation Platform (AAP) URL for the execution environment.

Example

```
steps:
  - id: create-ee
    name: Create Execution Environment
    action: rhaap:create-execution-environment
    input:
      token: ${{ parameters.AAP_TOKEN }}
      deleteIfExist: false
      values:
        environmentName: Custom EE
        environmentDescription: Execution environment with custom collections
        organization: ${{ parameters.organization }}
        image: quay.io/ansible/creator-ee:v0.16.0
        pull: missing
```

`rhaap:create-job-template`

Create a job template in Ansible Automation Platform that defines a reusable configuration for running Ansible playbooks.

Input parameters

Parameter	Type	Required	Description
`token`	string	Yes	OAuth2 token for Ansible Automation Platform authentication.

Parameter	Type	Required	Description
`deleteIfExist`	boolean	No	If `true`, the action deletes the job template if it already exists.
`values`	object	Yes	The job template configuration object.

“values” Object Structure

Field	Type	Required	Description
`templateName`	string	Yes	Name of the job template.
`templateDescription`	string	No	Description of the job template.
`project`	object	Yes	Project object with required `id` (number) and optional `name` (string).
`organization`	object	No	Organization object with `id` (number, required) and `name` (string, optional).
`jobInventory`	object	Yes	Inventory object with required `id` (number) and optional `name` (string).
`playbook`	string	Yes	Path to the playbook file to execute.
`templateDescription`	string	No	Description of the job template.
`scmType`	string	No	Source control type (for example, `Github` or `Gitlab`).
`executionEnvironment`	object	No	Execution environment object with required `id` (number) and optional `name` (string).
`extraVariables`	object	No	Extra variables to pass to the playbook (key-value pairs).

Output parameters

Parameter	Type	Description
`template`	object	Created job template details.
`template.id`	number	Job template ID.
`template.name`	string	Job template name.

Parameter	Type	Description
`template.description`	string	Job template description.
`template.url`	string	Ansible Automation Platform (AAP) URL for the job template.

Example

```
steps:
  - id: create-job-template
    name: Create Job Template
    action: rhaap:create-job-template
    input:
      token: ${{ parameters.AAP_TOKEN }}
      deleteIfExist: true
      values:
        templateName: Deploy Application
        templateDescription: Deploys the application to production
        project: ${{ steps['create-aap-project'].output.project }}
        organization: ${{ parameters.organization }}
        jobInventory: ${{ parameters.jobInventory }}
        playbook: deploy.yml
        executionEnvironment: ${{ steps['create-ee'].output.executionEnvironment }}
        extraVariables:
          app_version: 1.0.0
          environment: production
```

`rhaap:launch-job-template`

Launch an existing job template in Ansible Automation Platform.

Input parameters

Parameter	Type	Required	Description
`token`	string	Yes	OAuth2 token for Ansible Automation Platform authentication.

Parameter	Type	Required	Description
`values`	object	Yes	The job launch configuration object.

“values” Object Structure

Field	Type	Required	Description
`template`	string	Yes	Job template name to launch.
`inventory`	object	No	Override inventory with `id` (number) and `name` (string).
`credentials`	array	No	Array of credential objects to use.
`extraVariables`	object	No	Extra variables to pass to the job (key-value pairs).
`limit`	string	No	Host pattern to constrain which hosts the job runs against.
`jobType`	string	No	Job type: `run` or `check`.
`executionEnvironment`	object	No	Override execution environment with `id` (number) and `name` (string).
`verbosity`	object	No	Verbosity level object with `id` and `name`.
`forks`	number	No	Number of parallel processes (default: `5`).
`jobSliceCount`	number	No	Divide the job into N slices.
`timeout`	number	No	Job timeout in seconds (`0` = no timeout).
`diffMode`	boolean	No	Enable diff mode to show changes.
`jobTags`	string	No	Comma-separated tags to run only specific tasks.
`skipTags`	string	No	Comma-separated tags to skip specific tasks.

Output parameters

Parameter	Type	Description
`data`	object	Job execution details.
`data.id`	number	Job ID.
`data.status`	string	Job status.
`data.url`	string	Ansible Automation Platform (AAP) URL for the job.

Example

```
steps:
  - id: launch-job
    name: Launch Job Template
    action: rhaap:launch-job-template
    input:
      token: ${{ parameters.AAP_TOKEN }}
      values:
        template: Deploy Application
        inventory: ${{ parameters.jobInventory }}
        credentials: ${{ parameters.credentials }}
        extraVariables:
          app_version: v1.0.1
          deploy_env: production
        jobType: run
        verbosity: Normal
        diffMode: true
        timeout: 3600
```

Example with all backstage actions present

The following example displays how you can use multiple actions together.

```
apiVersion: scaffolder.backstage.io/v1beta3
kind: Template
metadata:
```

```
  name: ansible-complete-setup
  title: Complete Ansible Setup
  description: Creates an Ansible project, execution environment, job template, and
launches it
spec:
  owner: platform-team
  type: service


  parameters:
    - title: Project Information
      required:
        - projectName
        - repoUrl
      properties:
        projectName:
          title: Project Name
          type: string
          description: Name of the Ansible project
        repoUrl:
          title: Repository URL
          type: string
          description: Git repository URL containing Ansible content


    - title: AAP Configuration
      required:
        - organization
        - jobInventory
        - credentials
      properties:
        organization:
          title: Organization
          type: string
          description: AAP Organization name
          default: 1-org
          ui:field: hidden  # To hide this field in the UI
        jobInventory:
          title: Inventory name
```

```
        type: string
        description: AAP Inventory name
        resource: inventories
        ui:field: AAPResourcePicker
        default: my-inventory
      credentials:
        title: Credentials
        description: Select SCM credential for Private repositories
        resource: credentials
        ui:field: AAPResourcePicker
        default: my-machine-credential
steps:
  # Step 1: Create AAP Project
  - id: create-project
    name: Create AAP Project
    action: rhaap:create-project
    input:
      token: ${{ parameters.AAP_TOKEN }}
      deleteIfExist: true
      values:
        projectName: ${{ parameters.projectName }}
        organization: ${{ parameters.organization }}
        scmUrl: ${{ parameters.repoUrl }}
        scmBranch: main
        scmUpdateOnLaunch: true

  # Step 2: Create Execution Environment
  - id: create-ee
    name: Create Execution Environment
    action: rhaap:create-execution-environment
    input:
      token: ${{ parameters.AAP_TOKEN }}
      values:
        environmentName: ${{ parameters.projectName }}-ee
        organization: ${{ parameters.organization }}
        image: quay.io/ansible/creator-ee:latest
```

```
          pull: missing


    # Step 3: Create Job Template
    - id: create-template
      name: Create Job Template
      action: rhaap:create-job-template
      input:
        token: ${{ parameters.AAP_TOKEN }}
        values:
          templateName: ${{ parameters.projectName }}-deploy
          templateDescription: Deployment job template
          project: ${{ steps['create-project'].output.project }}
          organization: ${{ parameters.organization }}
          jobInventory: ${{ parameters.jobInventory }}
          playbook: site.yml
          executionEnvironment: ${{ steps['create-
ee'].output.executionEnvironment }}


    # Step 4: Launch the Job
    - id: launch-job
      name: Launch Job
      action: rhaap:launch-job-template
      input:
        token: ${{ parameters.AAP_TOKEN }}
        values:
          template: ${{ parameters.projectName }}-deploy
          inventory: ${{ parameters.jobInventory }}
          credentials: ${{ parameters.credentials }}
          jobType: run
          extraVariables:
            created_by: backstage
            timestamp: ${{ '' | now }}


  output:
    links:
      - title: AAP Project
        url: ${{ steps['create-project'].output.project.url }}
```

```
      - title: Job Template
        url: ${{ steps['create-template'].output.template.url }}
      - title: Job Execution
        url: ${{ steps['launch-job'].output.data.url }}
```

Custom UI components and filters

The **Ansible Backstage Plugins** provide custom UI components that enhance the software template experience by integrating directly with Ansible Automation Platform resource selection and authentication.

`AAPTokenField`

The `AAPTokenField` is a secure authentication field used in backstage scaffolder templates. It automatically fetches and stores an Ansible Automation Platform OAuth2 token, which is then available for all rhaap:* actions, enabling seamless authentication.

AAPTokenField Properties

The following table details the field's properties for use in a template's properties section.

Property	Type	Description
`title`	string	The label displayed in the UI (for example, "AAP Token"). Defaults to "AAP Token".
`description`	string	A short help text displayed below the input field.
`ui:field`	string	Must be set to `AAPTokenField`. This setting instructs Backstage to render a custom react component instead of a default input field.
`ui:backstage .review.show`	boolean	If `true`, this field appears in the **Review** step before scaffolding executes. The default value is `true`.

Authentication flow and token management

All `rhaap:*` actions require an OAuth2 token for authenticating with Ansible Automation Platform. The field manages the token through the following process:

- Token Source: The token is automatically obtained from the Ansible Automation Platform OAuth2 authentication provider.

- Storage: The token is stored securely in Backstage secrets or fetched through the `@ansible/backstage-plugin-auth-backend-module-rhaap-provider`.
- Usage: The token is passed to each action using the `token` input parameter.

When the RHAAP auth provider is used, the token is injected automatically and can be referenced in the workflow steps as shown:

```
- id: create-project
  action: rhaap:create-project
  input:
    token: ${{ parameters.AAP_TOKEN }}
    # ... other inputs
```

Example

The following example shows how to declare and reference the AAPTokenField within a backstage template. Note that `ui:widget: hidden` and `ui:backstage: review: show: false` are used to ensure the token is not exposed in the UI.

```
apiVersion: scaffolder.backstage.io/v1beta3
kind: Template
metadata:
  name: my-AAP-template
  title: Example AAP Template
spec:
  parameters:
    - title: Authentication
      properties:
        token:
          title: AAP Authentication Token
          type: string
          description: Oauth2 token
          ui:field: AAPTokenField
          ui:widget: hidden
          ui:backstage:
            review:
              show: false
  steps:
    - id: launch-job
      name: Launch AAP Job Template
      action: rhaap:launch-job-template
      input:
        token: ${{ parameters.token }}
        ...
```

Error and validation handling

All `rhaap:*` actions include built-in validation and user-friendly error reporting:

- Validation: If the token is missing or invalid, the action throws the error: "`Authorization token not provided`."
- Error Messages: Actions catch API client errors, extracting and surfacing meaningful messages without exposing stack traces.
- Workflow Safety: If a step fails due to authentication, subsequent steps are automatically skipped, ensuring a safe and predictable workflow.

`AAPResourcePicker`

`AAPResourcePicker` is a dynamic field for Backstage scaffolder templates. It fetches Ansible Automation Platform resources (like inventories or credentials) via the Ansible Automation Platform API, allowing users to select resources for their automation workflows. Data about Data:

AAPResourcePicker Properties

The following table details the essential properties for configuring the resource picker in a template's `properties` section.

Property	Type	Description
`title`	string	The label displayed in the UI (for example, "Inventory").
`description`	string	A short help text shown below the field.
`ui:field`	string	Must be set to `AAPResourcePicker`.
`resource`	string	The specific Ansible Automation Platform (AAP) resource type to fetch and display (for example, `inventories`, `credentials`, or `organizations`).
`idKey`	string	The property name used to retrieve the resource ID (default: "id").
`nameKey`	string	The property name used to display the resource name in the list (default: "name").
`type`	string	Set to "array" for a multi-select field; omit this property for a single-select field.

Example

The following example demonstrates how to use the `AAPResourcePicker` to create a single-select field for choosing an **Inventory**.

```
apiVersion: scaffolder.backstage.io/v1beta3
kind: Template
metadata:
  name: my-AAP-template
  title: Example AAP Template
spec:
  parameters:
    - title: Authentication
      properties:
        jobInventory:
          title: Inventory
          description: Select inventory
          resource: inventories
          ui:field: AAPResourcePicker
          default: DemoInventory
```

Custom filters

The plugins provide custom filters to extract specific properties from resource objects, which is essential for passing data between backstage steps.

Filter	Purpose	Example Usage
`resourceFilter`	Extracts a single, specific property from a resource object.	`${{ parameters.organization | resourceFilter('name') }}`
`multiResourceFilter`	Extracts a specific property from multiple resource objects (when the input is an array).	`${{ parameters.organization | multiResourceFilter('name') }}`

Standard UI widgets

Standard UI widgets provide enhanced components for user forms to capture various types of input.

Textarea widget

The `textarea` widget renders a multi-line text input field for capturing long-form user input such as descriptions or configuration content.

```
properties:
  description:
    type: string
    title: Description
    ui:widget: textarea
    ui:options:
      rows: 5
      placeholder: "Enter description..."
```

Select widget

The `select` widget displays a dropdown menu that enables users to choose a single value from a predefined list of options.

```
properties:
  environment:
    type: string
    title: Environment
    enum: ['dev', 'staging', 'prod']
    ui:widget: select
    default: 'dev'
```

Checkboxes widgets

The `checkboxes` widget renders a group of checkboxes, enabling the selection of multiple options from a list. The original plural "Checkboxes Widgets" is corrected to the singular form for style consistency.

```
properties:
  features:
    type: array
    title: Base Images
    items:
      type: string
      enum: ['baseImage1', 'baseImage2', 'baseImage3']
    ui:widget: checkboxes
    uniqueItems: true
```

Hidden widget

The `hidden` widget holds a fixed or prefilled value that is not visible to the user but is included in the submitted data. Use this widget for passing non-user-editable data, such as authentication tokens.

```
kind: Template
metadata:
  name: my-AAP-template
  title: Example AAP Template
spec:
  parameters:
    - title: Authentication
      properties:
        token:
          title: AAP Authentication Token
          type: string
          description: Oauth2 token
          ui:field: AAPTokenField
          ui:widget: hidden
          ui:backstage:
            review:
              show: false
  steps:
    - id: launch-job
      name: Launch AAP Job Template
      action: rhaap:launch-job-templat
      input:
        token: ${{ parameters.token }}
      ...
```

Standard UI widgets

Standard UI widgets provide enhanced components for user forms to capture various types of input.

Textarea widget

The `textarea` widget renders a multi-line text input field for capturing long-form user input such as descriptions or configuration content.

```
properties:
  description:
    type: string
    title: Description
    ui:widget: textarea
    ui:options:
      rows: 5
      placeholder: "Enter description..."
```

Select widget

The `select` widget displays a dropdown menu that enables users to choose a single value from a predefined list of options.

```
properties:
  environment:
    type: string
    title: Environment
    enum: ['dev', 'staging', 'prod']
    ui:widget: select
    default: 'dev'
```

Checkboxes widgets

The `checkboxes` widget renders a group of checkboxes, enabling the selection of multiple options from a list. The original plural "Checkboxes Widgets" is corrected to the singular form for style consistency.

```
properties:
  features:
    type: array
    title: Base Images
    items:
      type: string
      enum: ['baseImage1', 'baseImage2', 'baseImage3']
    ui:widget: checkboxes
    uniqueItems: true
```

Hidden widget

The `hidden` widget holds a fixed or prefilled value that is not visible to the user but is included in the submitted data. Use this widget for passing non-user-editable data, such as authentication tokens.

```
kind: Template
metadata:
  name: my-AAP-template
  title: Example AAP Template
spec:
  parameters:
    - title: Authentication
      properties:
        token:
          title: AAP Authentication Token
          type: string
          description: Oauth2 token
          ui:field: AAPTokenField
          ui:widget: hidden
          ui:backstage:
            review:
              show: false
  steps:
    - id: launch-job
      name: Launch AAP Job Template
      action: rhaap:launch-job-templat
      input:
        token: ${{ parameters.token }}
      ...
```

Standard UI widgets

Standard UI widgets provide enhanced components for user forms to capture various types of input.

Textarea widget

The `textarea` widget renders a multi-line text input field for capturing long-form user input such as descriptions or configuration content.

```
properties:
  description:
    type: string
    title: Description
    ui:widget: textarea
    ui:options:
      rows: 5
      placeholder: "Enter description..."
```

Select widget

The `select` widget displays a dropdown menu that enables users to choose a single value from a predefined list of options.

```
properties:
  environment:
    type: string
    title: Environment
    enum: ['dev', 'staging', 'prod']
    ui:widget: select
    default: 'dev'
```

Checkboxes widgets

The `checkboxes` widget renders a group of checkboxes, enabling the selection of multiple options from a list. The original plural "Checkboxes Widgets" is corrected to the singular form for style consistency.

```
properties:
  features:
    type: array
    title: Base Images
    items:
      type: string
      enum: ['baseImage1', 'baseImage2', 'baseImage3']
    ui:widget: checkboxes
    uniqueItems: true
```

Hidden widget

The `hidden` widget holds a fixed or prefilled value that is not visible to the user but is included in the submitted data. Use this widget for passing non-user-editable data, such as authentication tokens.

```
kind: Template
metadata:
  name: my-AAP-template
  title: Example AAP Template
spec:
  parameters:
    - title: Authentication
      properties:
        token:
          title: AAP Authentication Token
          type: string
          description: Oauth2 token
          ui:field: AAPTokenField
          ui:widget: hidden
          ui:backstage:
            review:
              show: false
  steps:
    - id: launch-job
      name: Launch AAP Job Template
      action: rhaap:launch-job-templat
      input:
        token: ${{ parameters.token }}
      ...
```

Standard UI widgets

Standard UI widgets provide enhanced components for user forms to capture various types of input.

Textarea widget

The `textarea` widget renders a multi-line text input field for capturing long-form user input such as descriptions or configuration content.

```
properties:
  description:
    type: string
    title: Description
    ui:widget: textarea
    ui:options:
      rows: 5
      placeholder: "Enter description..."
```

Select widget

The `select` widget displays a dropdown menu that enables users to choose a single value from a predefined list of options.

```
properties:
  environment:
    type: string
    title: Environment
    enum: ['dev', 'staging', 'prod']
    ui:widget: select
    default: 'dev'
```

Checkboxes widgets

The `checkboxes` widget renders a group of checkboxes, enabling the selection of multiple options from a list. The original plural "Checkboxes Widgets" is corrected to the singular form for style consistency.

```
properties:
  features:
    type: array
    title: Base Images
    items:
      type: string
      enum: ['baseImage1', 'baseImage2', 'baseImage3']
    ui:widget: checkboxes
    uniqueItems: true
```

Hidden widget

The `hidden` widget holds a fixed or prefilled value that is not visible to the user but is included in the submitted data. Use this widget for passing non-user-editable data, such as authentication tokens.

```
kind: Template
metadata:
  name: my-AAP-template
  title: Example AAP Template
spec:
  parameters:
    - title: Authentication
      properties:
        token:
          title: AAP Authentication Token
          type: string
          description: Oauth2 token
          ui:field: AAPTokenField
          ui:widget: hidden
          ui:backstage:
            review:
              show: false
  steps:
    - id: launch-job
      name: Launch AAP Job Template
      action: rhaap:launch-job-templat
      input:
        token: ${{ parameters.token }}
      ...
```

Standard UI widgets

Standard UI widgets provide enhanced components for user forms to capture various types of input.

Textarea widget

The `textarea` widget renders a multi-line text input field for capturing long-form user input such as descriptions or configuration content.

```
properties:
  description:
    type: string
    title: Description
    ui:widget: textarea
    ui:options:
      rows: 5
      placeholder: "Enter description..."
```

Select widget

The `select` widget displays a dropdown menu that enables users to choose a single value from a predefined list of options.

```
properties:
  environment:
    type: string
    title: Environment
    enum: ['dev', 'staging', 'prod']
    ui:widget: select
    default: 'dev'
```

Checkboxes widgets

The `checkboxes` widget renders a group of checkboxes, enabling the selection of multiple options from a list. The original plural "Checkboxes Widgets" is corrected to the singular form for style consistency.

```
properties:
  features:
    type: array
    title: Base Images
    items:
      type: string
      enum: ['baseImage1', 'baseImage2', 'baseImage3']
    ui:widget: checkboxes
    uniqueItems: true
```

Hidden widget

The `hidden` widget holds a fixed or prefilled value that is not visible to the user but is included in the submitted data. Use this widget for passing non-user-editable data, such as authentication tokens.

```
kind: Template
metadata:
  name: my-AAP-template
  title: Example AAP Template
spec:
  parameters:
    - title: Authentication
      properties:
        token:
          title: AAP Authentication Token
          type: string
          description: Oauth2 token
          ui:field: AAPTokenField
          ui:widget: hidden
          ui:backstage:
            review:
              show: false
  steps:
    - id: launch-job
      name: Launch AAP Job Template
      action: rhaap:launch-job-templat
      input:
        token: ${{ parameters.token }}
      ...
```

Parameter types and field options

The `parameters` section in a template defines the form fields that users complete before execution. Each property in `parameters.properties` renders as a form field.

This section documents the available types, widgets, and validation options you can use when building custom templates.

Supported parameter types

Templates support standard Backstage parameter types and widgets. Self-service automation portal adds Ansible Automation Platform-specific field types for resource selection and authentication.

Templates support standard Backstage parameter types (`string` , `number` , `integer` , `boolean` , `array`) and widgets (`textarea` , `select` , `checkboxes` , `hidden`). For a complete reference of standard types and widgets, see the Backstage Software Templates documentation.

Self-service automation portal adds the following Ansible Automation Platform-specific field types:

Type	Widget	Description	Example use case
`ui:field: AAPTokenField`	Hidden token field	OAuth2 authentication token, auto-populated and hidden from the user.	Always present in auto-generated templates.
`resource` + `ui:field: AAPResourcePicker`	Ansible Automation Platform resource picker (single-select)	Select one Ansible Automation Platform resource by name.	Inventory, organization.
`type: array` + `ui:field: AAPResourcePicker`	Ansible Automation Platform resource picker (multi-select)	Select multiple Ansible Automation Platform resources by name.	Credentials.

These Ansible Automation Platform-specific fields are documented in detail in See "AAP resource picker fields" on page 1 243.

AAP resource picker fields

Use `ui:field: AAPResourcePicker` to let users select Ansible Automation Platform resources by name. Self-service automation portal queries the Ansible Automation Platform API and displays available resources in a picker.

Set the `resource` property to specify the Ansible Automation Platform resource type.

The following table lists supported `resource` values:

`resource` value	Ansible Automation Platform resource type	Selection mode
`inventories`	Inventories	Single-select
`credentials`	Credentials	Multi-select (`type: array`)
`organizations`	Organizations	Single-select

Single-select example (inventory)

```
inventory:
  title: Inventory
  description: Select target inventory.
  resource: inventories
  ui:field: AAPResourcePicker
  default: Production Servers
```

Multi-select example (credentials)

```
credentials:
  title: Credentials
  description: Select credentials for accessing the target hosts.
  type: array
  resource: credentials
  ui:field: AAPResourcePicker
  default:
    - SSH Key
    - AWS Credentials
```

Dynamic fields

Dynamic fields appear or disappear based on user selections. Use the `dependencies` keyword with `allOf` and `if` / `then` to define conditional field visibility.

This lets you build forms where selecting a value in one field reveals additional fields relevant to that selection.

Dynamic fields are useful for:

- Showing advanced options only when the user enables them.
- Displaying different configuration fields based on a resource type or category selection.
- Requesting a reason or justification when the user selects a specific option.

Show or hide fields based on a toggle

Use a `boolean` toggle field to show or hide a group of additional fields.

In this example, selecting "Show advanced options" reveals one additional field. When the toggle is off, the field is hidden and not submitted.

```
parameters:
  - title: Time server settings
    required:
      - ntpServers
    properties:
      ntpServers:
        title: NTP servers
        type: string
        default: 0.rhel.pool.ntp.org, 1.rhel.pool.ntp.org
      showAdvanced:
        title: Show advanced options?
        type: boolean
        default: false
    dependencies:
      showAdvanced:
        allOf:
          - if:
              properties:
                showAdvanced:
                  const: true
            then:
              properties:
                maxSyncDelay:
                  type: number
                  title: Max sync delay (ms)
                  default: 500
```

When `showAdvanced` is `true`, the form displays `maxSyncDelay`. When `showAdvanced` is `false`, the field is hidden.

Show different fields based on a selection

Use an `enum` field with `dependencies` and `allOf` to display different configuration fields for each option.

In this example, selecting a database type reveals a version dropdown specific to that engine:

```
properties:
  dbType:
    title: Database type
    type: string
    enum:
      - PostgreSQL
      - MySQL
dependencies:
  dbType:
    allOf:
      - if:
          properties:
            dbType:
              const: PostgreSQL
        then:
          properties:
            version:
              type: number
              enum: [13, 14, 15]
              title: PostgreSQL version
              default: 15
      - if:
          properties:
            dbType:
              const: MySQL
        then:
          properties:
            version:
              type: string
              enum: ['5.7', '8.0']
              title: MySQL version
              default: '8.0'
```

Each `if` / `then` block in the `allOf` array matches one `enum` value. Selecting "PostgreSQL" shows the PostgreSQL version list; selecting "MySQL" replaces it with the MySQL version list.

Chain multiple dynamic dependencies

You can define multiple `dependencies` in the same parameter step. Each dependency operates independently.

In this example, selecting "Yes" for the restart confirmation reveals a reason field, while enabling advanced options reveals additional technical fields:

```
    dependencies:
      showAdvanced:
        allOf:
          - if:
              properties:
                showAdvanced:
                  const: true
            then:
              properties:
                maxSyncDelay:
                  type: number
                  title: Max sync delay (ms)
                  default: 500
      serviceRestart:
        allOf:
          - if:
              properties:
                serviceRestart:
                  const: 'Yes'
            then:
              properties:
                restartReason:
                  type: string
                  title: Reason for restart
                  description: Provide a reason for restarting the service.
                  default: Applying new configuration.
                  minLength: 10
                  errorMessage:
                    minLength: 'Provide a more detailed reason (at least 10
characters).'
```

Each key under `dependencies` corresponds to a property name in the same parameter step. When the user changes a field value, only the matching `if` / `then` branch is displayed.

Configure the output section

The `output` section defines what users see after a template runs. You can display text messages, reference user input from `parameters`, show job execution data from `steps`, and provide external links.

The `output` section supports two types of content:

- **`text`:** Displays text blocks with titles and markdown content.
- **`links`:** Displays clickable buttons that open external URLs.

Output text

Use `output.text` to display information to the user after the template runs. Each text block has a `title` and `content` field. The `content` field supports markdown formatting.

```
output:
  text:
    - title: Request submitted
      content: |
        Your request has been submitted.
```

You can include multiple text blocks:

```
output:
  text:
    - title: Request submitted
      content: |
        Your deployment of **${{ parameters.app_name }}** has been submitted.

        **Job ID:** ${{ steps['launch-job'].output.data.id }}
        **Status:** ${{ steps['launch-job'].output.data.status }}

    - title: Configuration summary
      content: |
        - Inventory: ${{ parameters.inventory }}
        - Application: ${{ parameters.app_name }}
```

Output links

Use `output.links` to display clickable buttons. Each link has a `title` and `url` field. You can optionally set an `icon` field.

```
output:
  links:
    - title: Check Request Status
      url: https://portal.example.com/requests/status
      icon: help
```

You can reference step output data in link URLs:

```
output:
  links:
    - title: View job in Ansible Automation Platform
      url: ${{ steps['launch-job'].output.data.url }}
```

You can include multiple links:

```
output:
  links:
    - title: View job in Ansible Automation Platform
      url: ${{ steps['launch-job'].output.data.url }}
    - title: RHEL documentation
      url: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/
      icon: docs
```

Reference parameters in output

You can reference any value the user entered in the `parameters` section by using the `${{ parameters.<field_name> }}` syntax.

The following table lists common parameter references:

Reference	Description	Example output
`${{ parameters.app_name }}`	User-entered text field value	`my-app`
`${{ parameters.inventory }}`	User-selected Ansible Automation Platform resource name	`Production Servers`
`${{ parameters.credentials }}`	User-selected credentials (array)	`["SSH Key", "AWS Credentials"]`
`${{ parameters.environment }}`	User-selected enum value	`production`

Displaying user selections in the output

```
output:
  text:
    - title: Request submitted
      content: |
        Your request for **${{ parameters.app_name }}** has been submitted.

        **Configuration:**
        - Inventory: ${{ parameters.inventory }}
        - Environment: ${{ parameters.environment }}
```

Reference step output in output

After the `rhaap:launch-job-template` action runs, the step output includes job execution data. Reference this data using the `${{ steps['<step-id>'].output.data.<field> }}` syntax.

The `rhaap:launch-job-template` action returns the following output fields:

Reference	Type	Description
`${{ steps['launch-job'].output.data.id }}`	number	Ansible Automation Platform Job ID.
`${{ steps['launch-job'].output.data.status }}`	string	Ansible Automation Platform Job status (for example, `pending`, `running`, `successful`).
`${{ steps['launch-job'].output.data.url }}`	string	Direct URL to the job in Ansible Automation Platform.

Displaying job execution data in the output

```
output:
  text:
    - title: Job launched
      content: |
        **Job ID:** ${{ steps['launch-job'].output.data.id }}
        **Status:** ${{ steps['launch-job'].output.data.status }}
  links:
    - title: View job in Ansible Automation Platform
      url: ${{ steps['launch-job'].output.data.url }}
```

Streamline development by integrating Red Hat Developer Hub plug-ins

You can use Ansible plug-ins for Red Hat Developer Hub (RHDH) to learn Ansible, create projects, and access tools for testing code. Through the RHDH UI, you can navigate to your Ansible Automation Platform instance to configure and run your automation jobs.

This document describes how to use the Ansible plug-ins for Red Hat Developer Hub. It presents a worked example of developing a playbook project for automating updates to your firewall configuration on RHEL systems.

Optional requirement

The Ansible plug-ins for Red Hat Developer Hub link to Learning Paths on the Red Hat developer portal, developers.redhat.com/learn.

To access the Learning Paths, you must have a Red Hat account and you must be able to log in to developers.redhat.com.

Related information
developers.redhat.com/learn
developers.redhat.com

Streamline development by integrating Red Hat Developer Hub plug-ins

You can use Ansible plug-ins for Red Hat Developer Hub (RHDH) to learn Ansible, create projects, and access tools for testing code. Through the RHDH UI, you can navigate to your Ansible Automation Platform instance to configure and run your automation jobs.

This document describes how to use the Ansible plug-ins for Red Hat Developer Hub. It presents a worked example of developing a playbook project for automating updates to your firewall configuration on RHEL systems.

Optional requirement

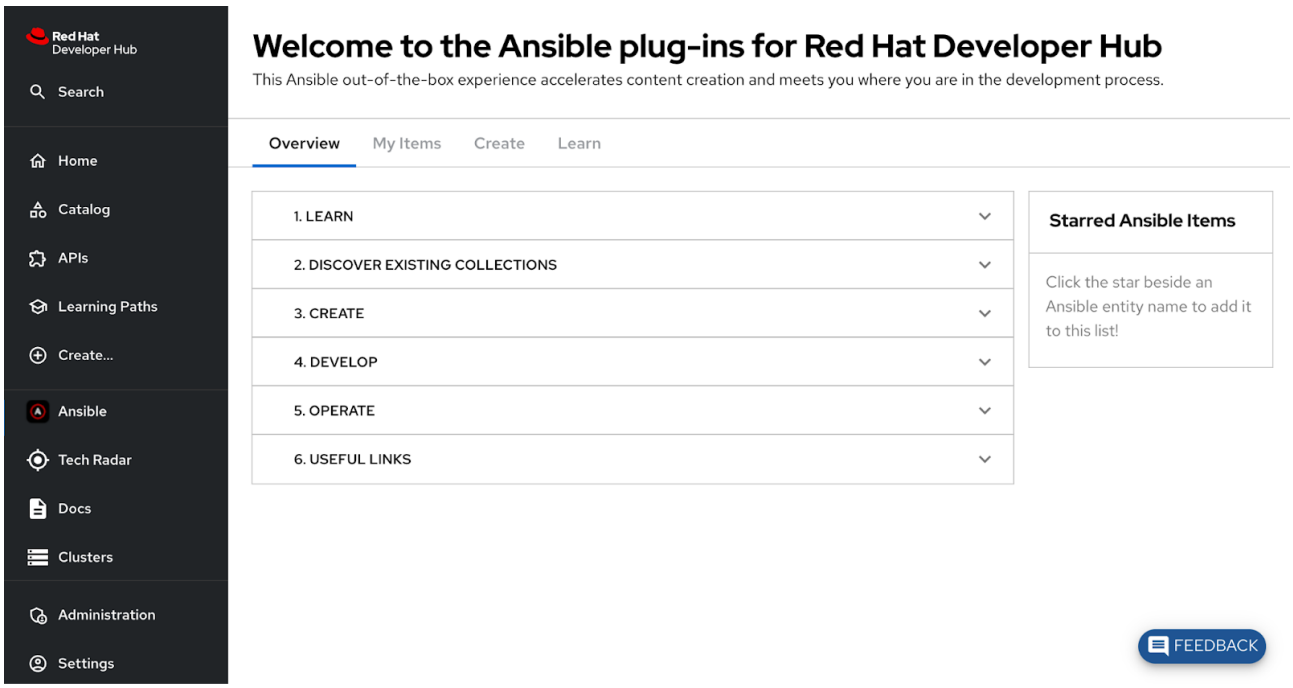
The Ansible plug-ins for Red Hat Developer Hub link to Learning Paths on the Red Hat developer portal, developers.redhat.com/learn.

To access the Learning Paths, you must have a Red Hat account and you must be able to log in to developers.redhat.com.

Related information
developers.redhat.com/learn
developers.redhat.com

Dashboard navigation

After logging into Red Hat Developer Hub (RHDH), use the Ansible navigation panel item to view the plug-in dashboard. The dashboard shows the full automation workflow, from learning to deploying jobs on Ansible Automation Platform.



The plug-in dashboard illustrates the steps you need to take from learning about Ansible to deploying automation jobs from Ansible Automation Platform:

- **Overview** displays the main dashboard page.
- **Learn** provides links to resources curated by Red Hat that introduce you to Ansible and provide step-by-step examples to get you started. For more information, see Learning about Ansible.
- **Discover existing collections** links to private automation hub, if configured in the plug-ins, or to automation hub hosted on the Red Hat Hybrid Cloud Console. Automation hub stores existing collections and execution environments that you can use in your projects. For more information, see Discovering existing collections.
- **Create** creates new projects in your configured Source Control Management platforms such as GitHub. For more information, see Creating a project.
- **Develop** links you to OpenShift Dev Spaces, if configured in the Ansible plug-ins installation. OpenShift Dev Spaces provides on-demand, web-based Integrated Development Environments (IDEs), where you can develop automation content. For more information, see Developing projects.
- **Operate** connects you to Ansible Automation Platform, where you can create and run automation jobs that use the projects you have developed. For more information, see Setting up a controller project to run your playbook project.

Access learning paths, labs, and collections

To learn more about getting started with automation, click **Learn** from the **Overview** page of the plug-in dashboard. The **Learn** page provides the following options for learning:

- **Learning Paths** lists a curated selection of learning tools hosted on developers.redhat.com that guide you through the foundations of working with Ansible, the Ansible VS Code extension, and using YAML.
  You can select other Ansible learning paths from the **Useful links** section.
- **Labs** are self-led labs that are designed to give you hands-on experience in writing Ansible content and using Ansible development tools.

Discover existing collections

From the **Overview** page in the Ansible plug-ins dashboard on Red Hat Developer Hub, click **Discover Existing Collections**.

The links in this pane provide access to the source of reusable automation content collections that you configured during plug-in installation.

If you configured private automation hub when installing the plug-in, you can click **Go to Automation Hub** to view the collections and execution environments that your enterprise has curated.

If you did not configure a private automation hub URL when installing the plug-in, the **Discover existing collection** pane provides a link to Red Hat automation hub on console.redhat.com. You can explore certified and validated Ansible content collections on this site.

Create a project in the Red Hat Developer Hub UI

Create a new Ansible Playbook project in the Red Hat Developer Hub by logging in, navigating to the Ansible section, and selecting the project creation template. You must have the correct access (RBAC) assigned by your administrator to view the templates.

Before you begin

- You have the correct access (RBAC) to view the templates in Red Hat Developer Hub. Ask your administrator to assign access to you if necessary.

Procedure

1. Log in to your Red Hat Developer Hub UI.
2. Click the Ansible `A` icon in the Red Hat Developer Hub navigation panel.
3. Navigate to the **Overview** page.
4. Click **Create**.

5. Click **Create Ansible Git Project**. The **Available Templates** page opens.
6. Click **Create Ansible Playbook project**.
7. In the **Create Ansible Playbook Project** page, enter information for your new project in the form.

   You can see sample values for this form in the Example project.

Field	Description
Source code repository organization name or username	The name of your source code repository username or organization name
Playbook repository name	The name of your new Git repository
Playbook description (Optional)	A description of the new playbook project
Playbook project's collection namespace	The new playbook Git project creates an example collection folder for you. Enter a value for the collection namespace.
Playbook project's collection name	The name of the collection
Catalog Owner Name	The name of the Developer Hub catalog item owner. This is a Red Hat Developer Hub field.
Source code repository organization name or username	The name of your source code repository username or organization name
Playbook repository name	The name of your new Git repository
Playbook description (Optional)	A description of the new playbook project
System (Optional)	This is a Red Hat Developer Hub field

NOTE:
Collection namespaces must follow Python module naming conventions. Collections must have short, all lowercase names. You can use underscores in the collection name if it improves readability.

8. Click **Review**.

Develop and execute projects in Dev Spaces

OpenShift Dev Spaces provides a web-based Integrated Development Environment that includes the Ansible VS Code extension and command line tools. You can use this environment to develop and test your automation code.

The OpenShift Dev Spaces instance provides a default configuration that installs the Ansible VS Code extension and provides the Ansible command line tools.

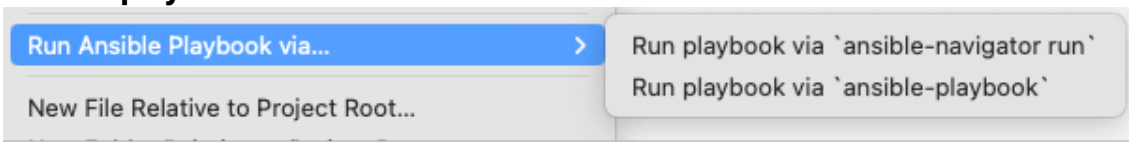
- Activate Ansible Lightspeed in the Ansible VS Code extension. For more information, refer to the Red Hat Ansible Lightspeed with IBM watsonx Code Assistant User Guide.

> **IMPORTANT:**
> OpenShift Dev Spaces is not included with your Ansible Automation Platform subscription or the Ansible plug-ins for Red Hat Developer Hub.

Execute automation tasks in Dev Spaces

The default configuration for Dev Spaces provides access to the Ansible command line tools.

- To execute an automation task in Dev Spaces from the VSCode user interface:
  a. Right-click a playbook name in the list of files.
  b. Select **Run Ansible Playbook via ansible-navigator run** or **Run playbook via ansible-playbook**.



Connect your project to Ansible Automation Platform

Access Ansible Automation Platform through the Red Hat Developer Hub to create a project for your playbook repository, then set up a job template that uses the playbook. You can go directly to your automation controller instance if it was not configured during the plug-in installation.

Procedure

1. The Ansible plug-ins provide a link to Ansible Automation Platform.
2. Log in to your Red Hat Developer Hub UI.
3. Click the Ansible `A` icon in the Red Hat Developer Hub navigation panel.
4. Click **Operate** to display a link to your Ansible Automation Platform instance.

   If automation controller was not included in your plug-in installation, a link to the product feature page is displayed.
5. Click **Go to Ansible Automation Platform** to open your platform instance in a new browser tab.

   Alternatively, if your platform instance was not configured during the Ansible plug-in installation, navigate to your automation controller instance in a browser and log in.
6. Log in to automation controller.
7. Create a project in Ansible Automation Platform for the GitHub repository where you stored your playbook project. Refer to Projects for more information.
8. Create a job template that uses a playbook from the project that you created. Refer to Workflow job templates for more information.

Example: Automate a Red Hat Linux firewall configuration

This example demonstrates how the Ansible plug-ins can help Ansible users of all skill levels create quality Ansible content.

As an infrastructure engineer new to Ansible, you have been tasked to create a playbook to configure a Red Hat Enterprise Linux (RHEL) host firewall.

The following procedures show you how to use the Ansible plug-ins and Dev Spaces to develop a playbook.

Learn more about playbooks

The first step is to learn more about Ansible playbooks using the available learning paths.

Procedure

1. Click the Ansible `A` icon in the Red Hat Developer Hub navigation panel.
2. Click **Learn** and select the **Getting Started with Ansible Playbooks** learning path. This redirects you to the Red Hat Developer website.
3. If you are prompted to log in, create a Red Hat Developer account, or enter your details.

4. Complete the learning path.

Discover existing Ansible content for RHEL system roles

Red Hat recommends that you use trusted automation content that has been tested and approved by Red Hat or your organization.

Automation hub is a central repository for discovering, downloading, and managing trusted content collections from Red Hat and its partners. Private automation hub provides an on-premise solution for managing content collections.

Procedure

1. Click on the Ansible `A` icon in the Red Hat Developer Hub navigation panel.
2. Click **Discover existing collections**.
3. Click **Go to Automation Hub**.
   - If private automation hub has been configured in the Ansible plug-ins, you are redirected to your **PrivateHubName** instance.
   - If private automation hub has not been configured in the Ansible plug-ins installation configuration, you will be redirected to the Red Hat Hybrid Console (RHCC) automation hub.

   In this example, you are redirected to the RHCC automation hub.
4. If you are prompted to log in, provide your Red Hat Customer Portal credentials.
5. Filter the collections with the `rhel firewall` keywords.

   The search returns the `rhel_system_roles` collection.

   The RHEL System Roles collection contains certified Ansible content that you can reuse to configure your firewall.

Create a new playbook project to configure a firewall

Create a new Ansible Playbook project quickly using the plug-in templates. This sets up the basic structure needed to configure your Red Hat Enterprise Linux firewall successfully.

Procedure

1. Click the Ansible `A` icon in the Red Hat Developer Hub navigation panel.
2. From the **Create** dropdown menu on the landing page, select **Create Ansible Git Project**.
3. Click **Choose** in the **Create Ansible Playbook Project** software template.

4. Fill in the following information in the **Create Ansible Playbook Project** page:

Field	Required	Description	Example value
Source code repository organization name or username	Yes	The name of your source code repository username or organization name.	`my_github_username`
Playbook repository name	Yes	The name of your new Git repository.	`rhel_firewall_config`
Playbook description	No	A description of the new playbook project.	`This playbook configures firewalls on Red Hat Enterprise Linux systems`
Playbook project's collection namespace	Yes	The new playbook Git project creates an example collection folder for you. Enter a value for the collection namespace.	`my_galaxy_username`
Playbook project's collection name	Yes	This is the name of the example collection.	`rhel_firewall_config`
Catalog Owner Name	Yes	The name of the Developer Hub catalog item owner. It is a Red Hat Developer Hub field.	`my_rhdh_username`
System	No	This is a Red Hat Developer Hub field.	`my_rhdh_linux_system`

5. Click **Review**.
6. Click **Create** to provision your new playbook project.
7. Click **Open in catalog** to view your project.

Create a new playbook to automate the firewall configuration

Create a new playbook and use the RHEL System Role collection to automate your Red Hat Enterprise Linux firewall configuration.

Procedure

1. In your Dev Spaces instance, click **File > New File**.
2. Enter `firewall.yml` for the filename and click **OK** to save it in the root directory.
3. Add the following lines to your `firewall.yml` file:

```
---
- name: Open HTTPS and SSH on firewall
  hosts: rhel
  become: true
  tasks:
    - name: Use rhel system roles to allow https and ssh traffic
      vars:
        firewall:
          - service: https
            state: enabled
            permanent: true
            immediate: true
            zone: public
          - service: ssh
            state: enabled
            permanent: true
            immediate: true
            zone: public
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.firewall
```

NOTE:
You can use Ansible Lightspeed with IBM watsonx Code Assistant from the Ansible VS Code extension to help you generate playbooks. For more information, refer to the Ansible Lightspeed with IBM watsonx Code Assistant User Guide.

Edit your firewall playbook project

The Ansible plug-ins integrate OpenShift Dev Spaces to edit your Ansible projects. OpenShift Dev Spaces provides on-demand, web-based Integrated Development Environments (IDEs).

Ansible Git projects provisioned using the Ansible plug-ins include best practice configurations for OpenShift Dev Spaces. These configurations include installing the Ansible VS Code extension and providing access from the IDE terminal to Ansible development tools, such as Ansible Navigator and Ansible Lint.

NOTE:
OpenShift Dev Spaces is optional and it is not required to run the Ansible plug-ins. It is a separate Red Hat product and it is not included in the Ansible Automation Platform or Red Hat Developer Hub subscription.

This example assumes that OpenShift Dev Spaces has been configured in the Ansible plug-ins installation.

- In the **catalog item** view of your playbook project, click **Open Ansible project in OpenShift Dev Spaces**.

  A VS Code instance of OpenShift Dev Spaces opens in a new browser tab. It automatically loads your new Ansible Playbook Git project.

Use the REST API to browse, query, filter, and authenticate

Representational State Transfer (REST) relies on a stateless, client/server, and cacheable communications protocol, usually the HTTP protocol.

You might find it helpful to see which API calls the user interface makes in sequence. To do this, you can use the UI from Firebug or Chrome with developer plugins.

Another option is to use Charles Proxy. This offers a visualizer that you might find helpful. While it is commercial software, it can insert itself as an operating system X proxy and intercept both requests from web browsers, curl and other API consumers.

Further options include the following:

- Fiddler
- mitmproxy
- Paros

Related information
Charles Proxy
Fiddler
mitmproxy

Paros

Browse the REST API

Access and use the platform gateway REST API through a web browser. You can find the recommended current version (v2), view documentation for endpoints, and use GET, PUT, and POST methods with JSON formatting.

Procedure

1. Go to the REST API in a web browser at:

   `https://<gateway server name>/api/controller/v2`
2. Click the **v2** link next to **current versions** or **available versions**.

   The API supports version 2.
3. Perform a `GET` with the `/api/` endpoint to get the `current_version`, which is the recommended version.
4. Click the question mark icon on the navigation menu for documentation on the access methods for that particular API endpoint and what data is returned when using those methods.
5. Use the `PUT` and `POST` verbs on the specific API pages by formatting JSON in the various text fields.

> **NOTE:**
> You can also view changed settings from factory defaults at the `/api/gateway/v1/settings/` endpoint. It reflects changes you made in the API browser, not changed settings that come from static settings files.

Find resources with the search query string parameter

The API is versioned for compatibility reasons. You can see what API versions are available by querying `/api/`.

You might have to specify the content or type on `POST` or `PUT` requests:

- `PUT`: Update a specific resource (by an identifier) or a collection of resources. You can also use `PUT` to create a specific resource if you know the resource identifier before-hand.
- `POST`: Create a new resource. Also acts as a catch-all verb for operations that do not fit into the other categories.

All URIs not ending with "/" receive a 301 redirect.

NOTE:
The formatting of `extra_vars` attached to Job Template records is preserved. YAML is returned as YAML with formatting and comments preserved, and JSON is returned as JSON.

Sort the API

Learn about managing ordering and sorting in the API.

To give you examples that are easy to follow, we use the following URL throughout this section:

```
https://<gateway server name>/api/controller/v2/groups/
```

Filter resources through the API

The system recognizes a collection as a "queryset". You can filter this by using various operators.

Advanced queries in the API

You can use additional query string parameters used to filter the list of results returned to those matching a given value. You can only use fields and relations that exist in the database for filtering.

Ensure that any special characters in the specified value are URL-encoded. For example:

```
?field=value%20xyz
```

Fields can also span relations, only for fields and relationships defined in the database:

```
?other__field=value
```

To exclude results matching certain criteria, prefix the field parameter with `not__`:

```
?not__field=value
```

By default, all query string filters are AND'ed together, so only the results matching all filters are returned. To combine results matching any one of multiple criteria, prefix each query string parameter with `or__`:

```
?or__field=value&or__field=othervalue
?or__not__field=value&or__field=othervalue
```

The default AND filtering applies all filters simultaneously to each related object being filtered across database relationships. The chain filter instead applies filters separately for each related object. To use this, prefix the query string parameter with `chain__`:

```
?chain__related__field=value&chain__related__field2=othervalue
?chain__not__related__field=value&chain__related__field2=othervalue
```

If you write the first query as `?related`*`field=value&related`*`field2=othervalue`, it returns only the primary objects where the same related object satisfied both conditions. As written by using the chain filter, it would return the intersection of primary objects matching each condition.

Field lookups

Field lookups allow you to filter API results based on specific criteria. You can use field lookups for more advanced queries, by appending the lookup to the field name:

```
?field__lookup=value
```

The following field lookups are supported:

- exact: Exact match (default lookup if not specified, see the following note for more information).
- iexact: Case-insensitive version of exact.
- contains: Field contains value.
- icontains: Case-insensitive version of contains.
- startswith: Field starts with value.
- istartswith: Case-insensitive version of startswith.
- endswith: Field ends with value.
- iendswith: Case-insensitive version of endswith.
- regex: Field matches the given regular expression.

- iregex: Case-insensitive version of regular expression.
- gt: Greater than comparison.
- gte: Greater than or equal to comparison.
- lt: Less than comparison.
- lte: Less than or equal to comparison.
- isnull: Check whether the given field or related object is null; expects a boolean value.
- in: Check whether the given field's value is present in the list provided; expects a list of items.
- You can specify boolean values as `True` or `1` for true, `False` or `0` for false (both case-insensitive).

For example, `?created__gte=2023-01-01` provides a list of items created after 1/1/2023.

You can specify null values as `None` or `Null` (both case-insensitive), though we recommend using the `isnull` lookup to explicitly check for null values.

You can specify lists (for the `in` lookup) as a comma-separated list of values. Filtering based on the requesting user's level of access by query string parameter:

- `role_level`: Level of role to filter on, such as `admin_role`

NOTE:
Earlier releases of Ansible Automation Platform returned queries with **_exact** results by default. As a workaround, set the `limit` to `?limit_exact` for the default filter. For example, `/api/controller/v2/jobs/?limit_exact=example.domain.com` results in:

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
...
```

View collection responses that are not shown by default

The API paginates responses for collections. This means that while a collection might contain tens or hundreds of thousands of objects, in each web request, only a limited number of results are returned for API performance reasons.

Access resource objects through API identifiers

The platform gateway API uses a primary key to access individual resource objects. You can access resources by using resource-specific, human-readable identifiers through the named URL feature.

The following example shows the named URL path where you can access a resource object without an auxiliary query string:

```
/api/controller/v2/hosts/host_name++inv_name++org_name/
```

API configuration values

There are two named-URL-related configuration settings available under `/api/controller/v2/settings/named-url/: NAMED_URL_FORMATS` and `NAMED_URL_GRAPH_NODES`.

`NAMED_URL_FORMATS` is a read only key-value pair list of all available named URL identifier formats. The following shows an example `NAMED_URL_FORMATS`:

```
"NAMED_URL_FORMATS": {

"organizations": "<name>",

"teams": "<name>++<organization.name>",

"credential_types": "<name>+<kind>",

"credentials": "<name>++<credential_type.name>+<credential_type.kind>+
+<organization.name>",

"notification_templates": "<name>++<organization.name>",

"job_templates": "<name>++<organization.name>",

"projects": "<name>++<organization.name>",

"inventories": "<name>++<organization.name>",

"hosts": "<name>++<inventory.name>++<organization.name>",

"groups": "<name>++<inventory.name>++<organization.name>",

"inventory_sources": "<name>++<inventory.name>++<organization.name>",

"inventory_scripts": "<name>++<organization.name>",

"instance_groups": "<name>",

"labels": "<name>++<organization.name>",

"workflow_job_templates": "<name>++<organization.name>",

"workflow_job_template_nodes": "<identifier>++<workflow_job_template.name>+
+<organization.name>",

"applications": "<name>++<organization.name>",

"users": "<username>",

"instances": "<hostname>"

}
```

For each item in `NAMED_URL_FORMATS`, the key is the API name of the resource to have named URL. The value is a string indicating how to form a human-readable unique identifier for that resource. `NAMED_URL_FORMATS` only lists the resources that can have named URL, any resource not listed there has no named URL. If a resource can have named URL, its objects must have a `named_url` field that represents the object-specific named URL. That field is only visible under detail view, not list view. You can access specified resource objects by using accurately generated named URL. This the object and its related URLs. For example, if `/api/controller/v2/res_name/obj_slug/` is valid, `/api/controller/v2/res_name/obj_slug/related_res_name/` is also valid.

`NAMED_URL_FORMATS` are instructive enough to compose human-readable unique identifiers and named URLs themselves. For ease-of-use, every object of a resource that can have named URL has a related field `named_url` that displays that object's named URL. You can copy and paste that field for your own custom use. For more information, see the help text of the API browser if a resource object has named URL.

You can manually decide the named URL label, for example with ID 5. To compose a named URL for this specific resource object by using `NAMED_URL_FORMATS`, first look up the labels field of `NAMED_URL_FORMATS` to get the identifier format `<name>++<organization.name>`:

- The first part of the URL format is `<name>`, which indicates that you can find the label resource detail in `/api/controller/v2/labels/5/`, and look for `name` field in returned JSON. If you have the `name` field with value `Foo`, then the first part of the unique identifier is `Foo`.
- The second part of the format is double plus sign `++`. That is the delimiter that separates different parts of a unique identifier. Append them to the unique identifier to get `Foo++`.
- The third part of the format is `<organization.name>`, which indicates that the field is not in the current label object under investigation, but in an organization that the label object points to. As the format indicates, look up the organization in the related field of the current returned JSON. That field might not exist. If it exists, follow the URL given in that field, for example, `/api/gateway/v1/organizations/3/`, to get the details of the specific organization, extract its name field, for example, "Default", and append it to the current unique identifier. Since `<organizations.name>` is the last part of the format, it generates the following named URL: `/api/controller/v2/labels/Foo++Default/`.
  In the case where an organization does not exist in the related field of the label object detail, append an empty string instead. This does not alter the current identifier. Therefore, `Foo++` becomes the final unique identifier and the resulting generated named URL becomes `/api/controller/v2/labels/Foo++/`.

An important aspect of generating a unique identifier for named URL has to do with reserved characters. As the identifier is part of a URL, the following reserved characters by URL standard are encoded by percentage symbols: `;/?:@=&[]`. For example, if an organization is named `;/?:@=&[]`, its unique identifier should be `%3B%2F%3F%3A%40%3D%26%5B%5D`. Another special reserved character is `` `, which is not reserved by URL standard but used by named URL to link different parts of an identifier. It is encoded by `[] ``. For example, if an organization is named `[]`, its unique identifier is `%5B[]%5D`, where original `[` and `]` are percent encoded and `` ` is converted to `[] ``.

Although you cannot manually change `NAMED_URL_FORMATS`, modifications do occur automatically and expand over time, reflecting underlying resource modification and expansion. Consult the `NAMED_URL_FORMATS` on the same cluster where you want to use the named URL feature.

`NAMED_URL_GRAPH_NODES` is another read-only list of key-value pairs that exposes the internal graph data structure used to manage named URLs. This is not intended to be human-readable but must be used for programmatically generating named URLs. An example script for generating named URL given the primary key of arbitrary resource objects that can have a named URL, using info provided by `NAMED_URL_GRAPH_NODES`, can be found in GitHub.

Understand how to identify resources by name

The API uses a consistent protocol to generate human-readable unique identifiers for resources that can be addressed by name in the API and the web interface.

Resources are identifiable by their unique keys, which are tuples of resource fields. Every resource is guaranteed to have its primary key number alone as a unique key, but there might be many other unique keys. A resource can generate an identifier format and, therefore, have a named URL if it has at least one unique key that satisfies the following rules:

1. The key must contain only fields that are either the `name` field, or text fields with a finite number of possible choices (such as credential type resource's `kind` field).
2. The only permitted exceptional field that breaks the preceding rule is a many-to-one related field relating to a resource other than itself, which is also allowed to have a slug.

If there are resources `Foo` and `Bar`, both `Foo` and `Bar` contain a name field and a choice field that can only have values "yes" or "no". Additionally, resource `Foo` has a many-to-one field (a foreign key) relating to `Bar`, for example `fk`. `Foo` has a unique key tuple (`name`, `choice`, `fk`) and `Bar` has a unique key tuple (`name`, `choice`). `Bar` can have named URL because it satisfies the preceding first rule. `Foo` can also have named URL, even though it breaks the first rule, the extra field breaking rule number one is the `fk` field, which is many-to-one-related to `Bar` and `Bar` can have named URL.

For resources satisfying rule number one, their human-readable unique identifiers are combinations of foreign key fields, delimited by `` `. In specific, resource `Bar` in the preceding example has slug format `<name><choice>` ``.

Note that the field order matters in slug format and the `name` field always comes first if present, followed by the remaining fields arranged in lexicographic order of field name. For example, if `Bar` also has an `a_choice` field satisfying rule one and the unique key becomes (`name`, `choice`, `a_choice`), its slug format becomes `<name><a_choice><choice>`.

For resources satisfying rule number two, if traced back through the extra foreign key fields, the result is a tree of resources that identify objects of that resource. To generate the identifier format, each resource in the traceback tree generates its own part of the standalone format, using all fields but the foreign keys. Finally, all parts are combined by `++` in the following order:

- Put standalone format as the first identifier part.
- Recursively generate unique identifiers for each resource. The underlying resource is pointing to using a foreign key (a child of a traceback tree node).
- Treat generated unique identifiers as the rest of the identifier components. Sort them in lexicographic order of corresponding foreign keys.
- Combine all components together using `++` to generate the final identifier format.

When generating an identifier format for resource `Foo`, the API generates the standalone formats, `` <name><choice>` for `Foo` and `<fk.name><fk.choice> `` for `Bar`, then combines them together to be `<name><choice>+<fk.name>+<fk.choice>`.

When generating identifiers according to the given identifier format, there are cases where a foreign key might point to nowhere. In this case, the API substitutes the part of the format corresponding to the resource the foreign key should point to with an empty string. For example, if a `Foo` object has the `name ="alice"`, `choice ="yes"`, but `fk field = None`, its resulting identifier is `alice+yes++`.

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

Build automation faster with Red Hat Ansible Lightspeed

Learn about Red Hat Ansible Lightspeed with IBM watsonx Code Assistant, its benefits, key features, process, and data gathered to train the IBM watsonx Code Assistant models.

Key features of Red Hat Ansible Lightspeed

Red Hat Ansible Lightspeed offers the following key features:

- **Ansible-specific IBM watsonx Code Assistant models**
  Red Hat Ansible Lightspeed with IBM watsonx Code Assistant uses Ansible-specific IBM watsonx Granite models unique to your organization, which are provided, managed, and maintained by IBM.
- **Red Hat Ansible Lightspeed cloud service and on-premise deployments**
  Red Hat Ansible Lightspeed is available both as a cloud service and as an on-premise deployment. Red Hat Ansible Lightspeed on-premise deployments provide the Red Hat Ansible Automation Platform customers more control over their data and supports compliance with enterprise security policies. For example, organizations in sensitive industries with data privacy or air-gapped requirements can use on-premise deployments of both Red Hat Ansible Lightspeed and IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data. Red Hat Ansible Lightspeed on-premise deployments are supported on Red Hat Ansible Automation Platform version 2.4 and later.
- **Red Hat Ansible Lightspeed trial**
  Existing Ansible users can now start a free 90-day Red Hat Ansible Lightspeed cloud service trial. You can create single-task and multitask recommendations, generate playbooks, and view playbook explanations with a trial account.

  To start your Red Hat Ansible Lightspeed trial, you need a trial or paid subscription to the Red Hat Ansible Automation Platform; however, you do not need a trial or paid subscription to IBM watsonx Code Assistant.
- **Playbook and task generation**
  This includes the following capabilities:

  - **Playbook generation and explanations**
    Using the Ansible VS Code extension, you can create Ansible playbooks using a natural language interface in English. Red Hat Ansible Lightspeed with IBM watsonx Code Assistant reads the natural language prompts and generates an entire playbook recommendation based on your intent. You can also view the explanations for new or existing playbooks. The playbook explanations describe what the playbook or task within the playbook does and contextualize its impact.
  - **Single and multitask generation**
    Using natural language prompts, you can generate single task or multiple task recommendations for Ansible task files and playbooks. To request multitask code recommendations, you can enter a sequence of natural language task prompts in a YAML file comment separated by ampersand (**&**) symbols.

    Currently, Red Hat Ansible Lightspeed supports user prompts in English language only. However, there could be instances where the training data that was used to train the IBM watsonx Code Assistant models included non-English language. In such

scenarios, the model can generate code recommendations for prompts made in the same non-English language, but the generated code recommendations might or might not be accurate.

- **Content source matching**
  For each generated code recommendation, Red Hat Ansible Lightspeed lists content source matches, including details such as potential source, content author, and relevant licenses. You can use this data to gain insight into potential training data sources used to generate the code recommendations.
- **Post-processing capabilities**
  Red Hat Ansible Lightspeed offers post-processing capabilities that augment IBM watsonx Code Assistant and improve the quality and accuracy of code recommendations.
- **Content maintenance and modernization**
  The Ansible code bot scans existing content collections, roles, and playbooks through Git repositories, and proactively creates pull requests whenever best practices or quality improvement recommendations are available. The bot automatically submits pull requests to the repository, which proactively alerts the repository owner to a recommended change to their content.
- **Telemetry data collection on the Admin dashboard**
  Red Hat Ansible Lightspeed now collects Admin dashboard telemetry data that provides insight into how your organization users are using the Ansible Lightspeed service, and displays the metrics on the Admin dashboard. If you no longer want to collect and manage the Admin dashboard telemetry, you can disable it for your organization.

Related information
Start a trial of Red Hat Ansible Lightspeed

Red Hat Ansible Lightspeed Overview

Red Hat Ansible Lightspeed with IBM watsonx Code Assistant is a generative AI service that helps automation teams create, adopt, and maintain Ansible content more efficiently. It uses natural language prompts to generate code recommendations for automation tasks based on Ansible best practices.

Red Hat Ansible Lightspeed is the cloud service that enables integration of generative AI into Ansible Automation Platform. This document specifically describes the integration of Red Hat Ansible Lightspeed with IBM watsonx Code Assistant.

Red Hat Ansible Lightspeed uses IBM watsonx Code Assistant models trained on subject matter expertise across the Ansible ecosystem, which includes Galaxy, GitHub, and Ansible certified and validated content. For ease of use, Red Hat Ansible Lightspeed is integrated with your existing Ansible developer workflows. For example, you can use your existing Git repositories (both public and private) to train your IBM watsonx Code Assistant models. You can also access Lightspeed content suggestions in VS Code through the Ansible VS code extension.

Access Red Hat Ansible Lightspeed with IBM watsonx Code Assistant

This section contains information about accessing both Red Hat Ansible Lightspeed cloud service and on-premise deployment.

- **Red Hat Ansible Lightspeed cloud service**
  To use the Red Hat Ansible Lightspeed cloud service, you must meet **one** of the following requirements:
  - Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
  - Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.

    NOTE:
    A Red Hat Ansible Lightspeed trial account does not require an IBM watsonx Code Assistant subscription.

- **Red Hat Ansible Lightspeed on-premise deployment**
  To use an on-premise deployment of Red Hat Ansible Lightspeed, your organization must have the following subscriptions:
  - A trial or paid subscription to the Red Hat Ansible Automation Platform
  - An installation of IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data

Benefits of using Red Hat Ansible Lightspeed

Red Hat Ansible Lightspeed with IBM watsonx Code Assistant offers the following benefits:

- **Reduces the onboarding learning period for Ansible developers**
  With just a basic understanding of YAML syntax, Ansible developers can use natural language prompts in English language to describe the automation goal. Red Hat Ansible Lightspeed then offers Ansible code recommendations to help achieve the automation goal more efficiently. This combination of content and best practice suggestions reduces the learning curve and offers a smoother onboarding experience for new Ansible users.

  For example, to get a multitask code recommendation, you can enter the prompt `Install postgresql-server & run postgresql-setup command`. The Ansible Lightspeed service

reads the text, interacts with IBM watsonx Code Assistant, and generates code recommendations to automate a multitask that installs a PostgreSQL server and sets up a PostgreSQL database. You can then view and accept the code recommendations to create tasks in an Ansible YAML file.

- **Increases productivity with quality content creation**
  Red Hat Ansible Lightspeed offers automation code recommendations that adhere to Ansible best practices, and IBM watsonx Code Assistant provides model fine-tuning features to improve the accuracy of suggested content based on your organization's existing Ansible content. Therefore, the AI-generated code recommendations are more accurate, more reliable, and integrated with your existing automation development workflows.
- **Extends trust with AI-generated code recommendations**
  The AI-generated code recommendations enable you to extend trust, with an automation code base that adheres to accepted Ansible best practices and significant data safeguards.

Key features of Red Hat Ansible Lightspeed

Red Hat Ansible Lightspeed offers the following key features:

- **Ansible-specific IBM watsonx Code Assistant models**
  Red Hat Ansible Lightspeed with IBM watsonx Code Assistant uses Ansible-specific IBM watsonx Granite models unique to your organization, which are provided, managed, and maintained by IBM.
- **Red Hat Ansible Lightspeed cloud service and on-premise deployments**
  Red Hat Ansible Lightspeed is available both as a cloud service and as an on-premise deployment. Red Hat Ansible Lightspeed on-premise deployments provide the Red Hat Ansible Automation Platform customers more control over their data and supports compliance with enterprise security policies. For example, organizations in sensitive industries with data privacy or air-gapped requirements can use on-premise deployments of both Red Hat Ansible Lightspeed and IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data. Red Hat Ansible Lightspeed on-premise deployments are supported on Red Hat Ansible Automation Platform version 2.4 and later.
- **Red Hat Ansible Lightspeed trial**
  Existing Ansible users can now start a free 90-day Red Hat Ansible Lightspeed cloud service trial. You can create single-task and multitask recommendations, generate playbooks, and view playbook explanations with a trial account.

  To start your Red Hat Ansible Lightspeed trial, you need a trial or paid subscription to the Red Hat Ansible Automation Platform; however, you do not need a trial or paid subscription to IBM watsonx Code Assistant. For more information, see Starting a trial of Red Hat Ansible Lightspeed.
- **Playbook and task generation**

This includes the following capabilities:

  - **Playbook generation and explanations**
    Using the Ansible VS Code extension, you can create Ansible playbooks using a natural language interface in English. Red Hat Ansible Lightspeed with IBM watsonx Code Assistant reads the natural language prompts and generates an entire playbook recommendation based on your intent. You can also view the explanations for new or existing playbooks. The playbook explanations describe what the playbook or task within the playbook does and contextualize its impact.

  - **Single and multitask generation**
    Using natural language prompts, you can generate single task or multiple task recommendations for Ansible task files and playbooks. To request multitask code recommendations, you can enter a sequence of natural language task prompts in a YAML file comment separated by ampersand (**&**) symbols.

    Currently, Red Hat Ansible Lightspeed supports user prompts in English language only. However, there could be instances where the training data that was used to train the IBM watsonx Code Assistant models included non-English language. In such scenarios, the model can generate code recommendations for prompts made in the same non-English language, but the generated code recommendations might or might not be accurate.

- **Content source matching**
  For each generated code recommendation, Red Hat Ansible Lightspeed lists content source matches, including details such as potential source, content author, and relevant licenses. You can use this data to gain insight into potential training data sources used to generate the code recommendations.

- **Post-processing capabilities**
  Red Hat Ansible Lightspeed offers post-processing capabilities that augment IBM watsonx Code Assistant and improve the quality and accuracy of code recommendations.

- **Content maintenance and modernization**
  The Ansible code bot scans existing content collections, roles, and playbooks through Git repositories, and proactively creates pull requests whenever best practices or quality improvement recommendations are available. The bot automatically submits pull requests to the repository, which proactively alerts the repository owner to a recommended change to their content.

- **Telemetry data collection on the Admin dashboard**
  Red Hat Ansible Lightspeed now collects Admin dashboard telemetry data that provides insight into how your organization users are using the Ansible Lightspeed service, and displays the metrics on the Admin dashboard. If you no longer want to collect and manage the Admin dashboard telemetry, you can disable it for your organization.

Related information
Starting a trial of Red Hat Ansible Lightspeed

Prerequisites

Review the following licensing and connectivity requirements for setting up Red Hat Ansible Lightspeed.

To use the Red Hat Ansible Lightspeed cloud service, you must meet **one** of the following requirements:

- Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
- Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.

> **NOTE:**
> A Red Hat Ansible Lightspeed trial account does not require an IBM watsonx Code Assistant subscription.

To use an on-premise deployment of Red Hat Ansible Lightspeed, your organization must have the following subscriptions:

- A trial or paid subscription to Red Hat Ansible Automation Platform
- An installation of IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data

You must also install the following components:

- VS Code version 1.70.1 or later
- The Ansible extension for VS Code version 2.8 or later

Connectivity requirements

To generate code recommendations, the Ansible Lightspeed service in Visual Studio (VS) Code editor requires access to the outbound domain *https://c.ai.ansible.redhat.com*. The outbound connections are encrypted on TCP protocol port 443.

Data gathered to train the IBM watsonx Code Assistant models

This topic provides information about the data that is collected to train the IBM watsonx Code Assistant models.

Models

Red Hat Ansible Lightspeed with IBM watsonx Code Assistant uses Ansible-specific IBM watsonx Granite models unique to your organization. These models are provided, managed, and maintained by IBM.

Data sources

IBM watsonx Code Assistant models are trained on Ansible content from Ansible Galaxy, data from public Git repositories, and Red Hat Ansible subject matter expert examples.

If you publish content to Ansible Galaxy and want to restrict your Ansible Galaxy content from being used to train the models, you can opt out of sharing your Ansible Galaxy data in the Ansible Galaxy namespace configuration.

Data telemetry

Red Hat Ansible Lightspeed collects the following telemetry data by default:

- Operational telemetry data
- Admin dashboard telemetry data

NOTE:
No telemetry data is collected in an Red Hat Ansible Lightspeed on-premise deployment.

Telemetry data collection notice for the Admin dashboard

In connection with your use of this Red Hat offering, Red Hat may collect telemetry data about your use of the software. This data allows Red Hat to monitor the software and to improve Red Hat offerings and support, including identifying, troubleshooting, and responding to issues that impact users.

The telemetry data may also be used to enable you to track your entitlements to Red Hat subscriptions and take advantage of future Red Hat purchasing programs. It may also allow Red Hat to assist you in implementing upgrades to minimize service impact. The data may be shared internally within Red Hat to improve the user experience. If you are evaluating Red Hat software, the data will help Red Hat determine if you need assistance.

What information does Red Hat collect?

Tools within the software monitor various metrics and this information is transmitted to Red Hat. The following metrics are monitored:

- **Operational telemetry data**
  This is the data that is required to operate and troubleshoot the Ansible Lightspeed service. For more information, refer the Enterprise Agreement. You cannot disable the collection of operational telemetry data.

  This includes the following data:

  - Organization you are logged into (Organization ID, account number)
  - Large language model (or models) that you are connected to

- **Admin dashboard telemetry data**
  This is the data that provides insight into how your organization users are using the Ansible Lightspeed service, and the metrics are displayed on the Admin dashboard.

  This includes the following data:

  - Prompts and content suggestions, including accept or reject of the content suggestions
  - User sentiment feedback
    You can also disable the Admin dashboard telemetry if you no longer want to collect and monitor the telemetry data.

NOTE:
No telemetry data is collected in an Red Hat Ansible Lightspeed on-premise deployment.

Personal Data

Red Hat does not intend to collect personal information. If Red Hat discovers that personal information has been inadvertently received, Red Hat will delete such information.

- Retention
  Red Hat retains and stores telemetry data only for as long as it's needed for the purposes described above or as otherwise required or permitted by law.
- Data security
  Red Hat employs technical and organizational measures designed to protect the telemetry data. Data stored in the Red Hat cloud is being protected, where possible, through encryption. Data is also segmented, and therefore is not accessible across organizations.
- Data sharing
  Red Hat may share telemetry data with its business partners in an aggregated form that does not identify customers. This data helps the partners better understand their markets and their customers' use of Red Hat offerings. The data also helps to ensure the successful integration of products jointly supported by those partners.
- Third Party Service Providers
  Red Hat may engage certain service providers to assist in the collection and storage of the telemetry data.
- User control/ enabling and disabling Admin Dashboard telemetry collection
  You cannot disable collection of operational telemetry data. Operational telemetry data includes only data that is necessary to operate and troubleshoot the service. However, you can disable the collection of Admin Dashboard telemetry data.

Related information
Red Hat Privacy Statement

Quick start for administrators

This section shows how to get started with Red Hat Ansible Lightspeed as an administrator.

Administrators can leverage Red Hat Ansible Lightspeed's powerful automation capabilities to streamline system management tasks, reduce human error, and maintain consistency across complex environments. Red Hat Ansible Lightspeed helps administrators and enables developers to work more efficiently, improving productivity and the overall quality of their work.

As an organization administrator, perform the following tasks to set up and manage Red Hat Ansible Lightspeed for your organization:

1. Choose and configure your deployment
2. Administer the Ansible Lightspeed service

Choose and configure your deployment

As an organization administrator, you can set up a Red Hat Ansible Lightspeed cloud service, as an on-premise deployment, or as a hybrid deployment.

Configure the Red Hat Ansible Lightspeed cloud service or on-premise deployment and connect it to your IBM watsonx Code Assistant instance. You can also set up a hybrid deployment, wherein Red Hat Ansible Lightspeed is an on-premise deployment, while IBM watsonx Code Assistant model is a cloud deployment.

You must have a trial or paid subscription to both Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.

Administer the Ansible Lightspeed Service

As an organization administrator, you can manage the Ansible Lightspeed service after deploying Red Hat Ansible Lightspeed cloud service or on-premise deployment.

Perform one of the following tasks:

- Log in to the Ansible Lightspeed admin portal
  Use the Ansible Lightspeed admin portal to access and manage the Ansible Lightspeed service.
- View and manage the Admin dashboard telemetry data
  View the operational telemetry and Admin dashboard telemetry data from the Ansible Lightspeed Admin dashboard. If you no longer want to collect the telemetry data for your organization, you can disable the Admin dashboard telemetry.

Related information
Setting up the cloud service

Overview

Red Hat Ansible Lightspeed's powerful automation capabilities enable administrators to streamline system management, reduce errors, and maintain consistency across complex environments. It also helps administrators and developers improve efficiency, productivity, and work quality.

As an organization administrator, perform the following tasks to set up and manage Red Hat Ansible Lightspeed for your organization:

1. Choose and configure your deployment

2. Administer the Ansible Lightspeed service

Choosing and configuring your deployment

As an organization administrator, you can set up a Red Hat Ansible Lightspeed cloud service, as an on-premise deployment, or as a hybrid deployment.

Configure the Red Hat Ansible Lightspeed cloud service or on-premise deployment and connect it to your IBM watsonx Code Assistant instance. You can also set up a hybrid deployment, wherein Red Hat Ansible Lightspeed is an on-premise deployment, while IBM watsonx Code Assistant model is a cloud deployment.

You must have a trial or paid subscription to both Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.

Administering the Ansible Lightspeed Service

As an organization administrator, you can manage the Ansible Lightspeed service after deploying Red Hat Ansible Lightspeed cloud service or on-premise deployment.

Perform one of the following tasks:

- Log in to the Ansible Lightspeed admin portal
  Use the Ansible Lightspeed admin portal to access and manage the Ansible Lightspeed service.
- View and manage the Admin dashboard telemetry data
  View the operational telemetry and Admin dashboard telemetry data from the Ansible Lightspeed Admin dashboard. If you no longer want to collect the telemetry data for your organization, you can disable the Admin dashboard telemetry.

Related information
Setting up the cloud service

Quick start for developers

This section shows how to get started with Red Hat Ansible Lightspeed as a developer.

Red Hat Ansible Lightspeed simplifies writing and managing infrastructure as code with intuitive automation and integrations, enabling fast, efficient coding. Developers can focus on applications, not infrastructure management, due to its simplified workflows.

After the organization administrators have set up Red Hat Ansible Lightspeed, automation developers can use the Ansible Lightspeed service to develop and use custom automation content.

Perform the following tasks to develop Ansible content:

- Install and configure the Ansible VS Code extension
- Develop Ansible content

Install and configuring Ansible VS Code extension

To access Red Hat Ansible Lightspeed, all Ansible users must install and configure the Ansible VS Code extension in their VS Code editor. For the procedure, see Install and configure the Ansible VS Code extension.

Develop Ansible content

After installing and configuring the Ansible VS Code extension, developers can create custom Ansible content.

You can perform the following tasks:

- Start a free 90-day trial of Red Hat Ansible Lightspeed
  This is especially useful as you **do not need** a trial or paid subscription to IBM watsonx Code Assistant. After your 90-day trial expires, you must set up the cloud service or on-premise deployment to continue accessing the Ansible Lightspeed service.
- Create single-task or multitask code recommendations
  You can generate single-task or multitask code recommendations for your task intent using the Ansible VS Code extension.
- Create Ansible playbooks and view playbook explanations
  You can create Ansible playbooks and view explanations for new or existing playbooks by using the Ansible VS Code extension.
- Create roles within a collection and view role explanations
  You can generate roles within collections using the Ansible VS Code extension and view explanations for existing roles.

Related information
Install and configure the Ansible VS Code extension

Quick start for developers

Red Hat Ansible Lightspeed simplifies writing and managing infrastructure as code with intuitive automation and integrations, enabling fast, efficient coding. Developers can focus on applications, not infrastructure management, due to its simplified workflows.

After the organization administrators have set up Red Hat Ansible Lightspeed, automation developers can use the Ansible Lightspeed service to develop and use custom automation content.

Perform the following tasks to develop Ansible content:

- Install and configure the Ansible VS Code extension
- Develop Ansible content

Installing and configuring Ansible VS Code extension

To access Red Hat Ansible Lightspeed, all Ansible users must install and configure the Ansible VS Code extension in their VS Code editor. For the procedure, see Install and configure the Ansible VS Code extension.

Developing Ansible content

After installing and configuring the Ansible VS Code extension, developers can create custom Ansible content.

You can perform the following tasks:

- Start a free 90-day trial of Red Hat Ansible Lightspeed
  This is especially useful as you **do not need** a trial or paid subscription to IBM watsonx Code Assistant. After your 90-day trial expires, you must set up the cloud service or on-premise deployment to continue accessing the Ansible Lightspeed service.
- Create single-task or multitask code recommendations
  You can generate single-task or multitask code recommendations for your task intent using the Ansible VS Code extension.
- Create Ansible playbooks and view playbook explanations
  You can create Ansible playbooks and view explanations for new or existing playbooks by using the Ansible VS Code extension.
- Create roles within a collection and view role explanations
  You can generate roles within collections using the Ansible VS Code extension and view explanations for existing roles.

Start a trial of Red Hat Ansible Lightspeed

Red Hat Ansible Lightspeed cloud service provides a free 90-day trial for existing Ansible users.

Set up Red Hat Ansible Lightspeed for your organization

As a Red Hat customer portal administrator, you must configure Red Hat Ansible Lightspeed to connect to your IBM watsonx Code Assistant instance. This chapter provides information about configuring both the Red Hat Ansible Lightspeed cloud service and on-premise deployment.

Configuration requirements

Ensure that you meet the licensing and setup requirements specified below before you begin setting up Red Hat Ansible Lightspeed.

Licensing requirements

- **Red Hat Ansible Lightspeed cloud service**
  To use the Red Hat Ansible Lightspeed cloud service, you must meet **one** of the following requirements:
    - Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
    - Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.

      NOTE:
      A Red Hat Ansible Lightspeed trial account does not require an IBM watsonx Code Assistant subscription.

- **Red Hat Ansible Lightspeed on-premise deployment**
  To use an on-premise deployment of Red Hat Ansible Lightspeed, your organization must have the following subscriptions:
    - A trial or paid subscription to the Red Hat Ansible Automation Platform

- An installation of IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data

Setup requirements

To set up Red Hat Ansible Lightspeed for your organization, you need the following IBM watsonx Code Assistant information:

- **API key**
  A unique API key authenticates all requests made from Red Hat Ansible Lightspeed to IBM watsonx Code Assistant. Each Red Hat organization with a valid Ansible Automation Platform subscription must have a configured API key. An authenticated RH-SSO user creating a task in Red Hat Ansible Lightspeed is authenticated to IBM watsonx Code Assistant through the user's Red Hat organization API key.
- **Model ID**
  A unique model ID identifies an IBM watsonx Code Assistant model in your IBM Cloud account. The model ID that you configure in the Ansible Lightspeed administrator portal is used as the default model, and can be accessed by all Ansible Lightspeed users within your organization.

IMPORTANT:
You must configure both the API key and the model ID when you are initially configuring Red Hat Ansible Lightspeed.

Set up Red Hat Ansible Lightspeed cloud service

As a Red Hat customer portal administrator, you must configure Red Hat Ansible Lightspeed cloud service to connect to your IBM watsonx Code Assistant instance.

NOTE:
The IBM Cloud service instance of IBM watsonx Code Assistant is available in the following data centers:

- Dallas (`us-south`)
- Frankfurt (`eu-de`)
- Sydney (`au-syd`) (Essentials plan only)

Ansible Lightspeed cloud deployments are configured to connect exclusively to the US (Dallas) IBM data center. Attempts to connect from non-US data centers will result in connection failure. If you want to use a non-Dallas IBM data center, then you must set up Ansible Lightspeed in the hybrid deployment model.

Related information
Setting up your IBM watsonx Code Assistant for Red Hat Ansible Lightspeed service

Log in to the Ansible Lightspeed administrator portal

Use the Ansible Lightspeed administrator portal to connect Red Hat Ansible Lightspeed to IBM watsonx Code Assistant.

Before you begin

- You have organization administrator privileges to a Red Hat Customer Portal organization with a valid Red Hat Ansible Automation Platform subscription.

Procedure

1. Log in to the Ansible Lightspeed portal as an organization administrator.
2. Click **Log in > Log in with Red Hat**.
3. Enter your Red Hat account username and password. The Ansible Lightspeed Service uses Red Hat Single Sign-On (RH-SSO) for authentication.

   As part of the authentication process, the Ansible Lightspeed Service checks whether your organization has an active Ansible Automation Platform subscription. On successful authentication, the login screen is displayed along with your username and your assigned user role.
4. From the login screen, click **Admin Portal**.

Result

You are redirected to the Red Hat Ansible Lightspeed with IBM watsonx Code Assistant administrator portal where you can connect Red Hat Ansible Lightspeed to your IBM watsonx Code Assistant instance.

Configure Red Hat Ansible Lightspeed cloud service

Use this procedure to configure the Red Hat Ansible Lightspeed cloud service.

Before you begin

- You have obtained an API key and a model ID from IBM watsonx Code Assistant that you want to use in Red Hat Ansible Lightspeed.
  For information about how to obtain an API key and model ID from IBM watsonx Code Assistant, see the IBM watsonx Code Assistant documentation.

Procedure

1. Log in to the Ansible Lightspeed portal as an organization administrator.
2. From the login screen, click **Admin Portal**.
3. Specify the API key of your IBM watsonx Code Assistant instance:
   a. Under **IBM Cloud API Key**, click **Add API key**. A screen to enter the **API Key** is displayed.
   b. Enter the API Key.
   c. Optional: Click **Test** to validate the API key.
   d. Click **Save**.
4. Specify the model ID of the model that you want to use:
   a. Click **Model Settings**.
   b. Under **Model ID**, click **Add Model ID**. A screen to enter the **Model Id** is displayed.
   c. Enter the **Model ID** that you obtained in the previous procedure as the default model for your organization.
   d. Optional: Click **Test model ID** to validate the model ID.
   e. Click **Save**.

   When the API key and model ID is successfully validated, Red Hat Ansible Lightspeed is connected to your IBM watsonx Code Assistant instance.

Related information
Troubleshooting Red Hat Ansible Lightspeed configuration errors

Set up Red Hat Ansible Lightspeed on-premise deployment

As an administrator, you can deploy Ansible Lightspeed on-premise and connect it to IBM watsonx Code Assistant. Once the deployment is complete, you can use the Ansible Lightspeed service through the Ansible Visual Studio (VS) Code extension.

NOTE:

- Red Hat Ansible Lightspeed on-premise deployments are supported on Red Hat Ansible Automation Platform version 2.4 and later.
- The IBM Cloud service instance of IBM watsonx Code Assistant is available in the following data centers:
  - Dallas (`us-south`)
  - Frankfurt (`eu-de`)
  - Sydney (`au-syd`) (Essentials plan only)

Ansible Lightspeed cloud deployments are configured to connect exclusively to the US (Dallas) IBM data center. Attempts to connect from non-US data centers will result in connection failure. If you want to use a non-Dallas IBM data center, then you must set up Ansible Lightspeed in hybrid deployment model. For more information about IBM's supported data centers, see the topic Setting up your watsonx Code Assistant for Red Hat Ansible Lightspeed service in *IBM watsonx Code Assistant* documentation.

Related information
Setting up your watsonx Code Assistant for Red Hat Ansible Lightspeed service

Overview

This section provides information about the system requirements, prerequisites, and the process for setting up a Red Hat Ansible Lightspeed on-premise deployment.

Deployment models

You can use one of the following modes of deployment:

- **On-premise deployment**

Both Red Hat Ansible Lightspeed and the IBM watsonx Code Assistant model (IBM Cloud Pak for Data) are on-premise deployments. Telemetry data is not collected for an on-premise mode of deployment.

- **Hybrid deployment**
  Red Hat Ansible Lightspeed is an on-premise deployment, while IBM watsonx Code Assistant model is a cloud deployment. Telemetry data is not collected for hybrid deployments.

  A hybrid deployment model provides the following benefits:

  - Enables you to set up an on-premise deployment of Red Hat Ansible Lightspeed, with IBM watsonx Code Assistant model on a cloud environment.
  - Provides the freedom and flexibility to choose an environment that best suits your organizational needs.
  - Enables organizations to use the Ansible Automation Platform for user authentication, instead of logging into the Red Hat cloud.
  - Enables organizations to deploy the Ansible Automation Platform in their preferred region.

System requirements

Your system must meet the following minimum system requirements to install and run the Red Hat Ansible Lightspeed on-premise deployment.

Requirement	Minimum requirement
RAM	5 GB
CPU	1
Local disk	40 GB

To see the rest of the Red Hat Ansible Automation Platform system requirements, see the System requirements section of *Planning your installation*.

NOTE:
You must also have installed IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data. The installation includes a base model that you can use to set up your Red Hat Ansible Lightspeed on-premise deployment. For installation information, see the watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data documentation.

Prerequisites

- You have installed Red Hat Ansible Automation Platform on your Red Hat OpenShift Container Platform environment.
- You have administrator privileges for Red Hat Ansible Automation Platform.
- You have installed IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data.
- Your system meets the minimum system requirements to set up Red Hat Ansible Lightspeed on-premise deployment.
- You have obtained an API key and a model ID from IBM watsonx Code Assistant.
  For information about obtaining an API key and model ID from IBM watsonx Code Assistant, see the IBM watsonx Code Assistant documentation. For information about installing IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data, see the watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data documentation.
- You have an existing external PostgreSQL database configured for the Red Hat Ansible Automation Platform, or have a database created for you when configuring the Red Hat Ansible Lightspeed on-premise deployment.

Related information
System requirements
watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data documentation
IBM watsonx Code Assistant documentation
watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data documentation

Configure Ansible VS Code extension for Red Hat Ansible Lightspeed on-premise deployment

To access the on-premise deployment of Red Hat Ansible Lightspeed, all Ansible users within your organization must install the Ansible Visual Studio (VS) Code extension in their VS Code editor, and configure the extension to connect to the on-premise deployment.

Before you begin

- You have installed VS Code version 1.70.1 or later.

Procedure

1. Obtain the URL of your Ansible Lightspeed instance:

a. In Red Hat OpenShift Container Platform, select **Networking > Routes** and locate the Red Hat Ansible Lightspeed instance that was created.

b. From the **Location** column, copy the URL of your Ansible Lightspeed instance.

The URL will be in the following format: `https://<lightspeed_route>/complete/aap/`

2. Open the VS Code application.
3. From the **Activity** bar, click the **Extensions** icon.
4. From the **Installed Extensions** list, select **Ansible**.
5. From the **Ansible** extension page, click the **Settings** icon () and select **Extension Settings**.
6. Select **Ansible Lightspeed** settings and specify the following information:

- In the **URL for Ansible Lightspeed** field, enter the **Route URL** of the Red Hat Ansible Lightspeed on-premise deployment. Ansible users must have Ansible Automation Platform controller credentials.
  After configuring Ansible VS Code extension to connect to Red Hat Ansible Lightspeed on-premise deployment, you must log in to Ansible Lightspeed through the Ansible VS Code extension.

> **NOTE:**
> If your organization recently subscribed to the Red Hat Ansible Automation Platform, it might take a few hours for Red Hat Ansible Lightspeed to detect the new subscription. In VS Code, use the **Refresh** button in the Ansible extension from the Activity bar to check again.

Connect to a different IBM watsonx Code Assistant model

After you have set up the Red Hat Ansible Lightspeed on-premise deployment successfully, you can modify the deployment if you want to connect to another IBM watsonx Code Assistant model.

Before you begin

- You have set up a Red Hat Ansible Lightspeed on-premise deployment.
- You have obtained an API key and a model ID of the IBM watsonx Code Assistant model you want to connect to.
- You have created a new model configuration secret for the IBM watsonx Code Assistant model that you want to connect to. For information about creating a model configuration secrets, see Creating a model configuration secret.

For example, you connected to the default IBM watsonx Code Assistant model but now want to connect to a custom model instead. To connect to another IBM watsonx Code Assistant model, you must create new connection secrets, and then update the connection secrets and parameters on an existing Ansible Automation Platform operator.

Procedure

1. Go to the Red Hat OpenShift Container Platform.
2. Select **Operators › Installed Operators**.
3. From the list of installed operators, select the **Ansible Automation Platform** operator.
4. Locate and select the **Ansible Automation Platform** custom resource, and then click the required app.
5. Select the **YAML** tab.
6. Scroll the text to find the `spec` section under `Lightspeed` category. For example:

```
spec:
  lightspeed:
    disabled: false
    model_config_secret_name: <Name of the model configuration secret that you
recently created.>
```

7. Replace the `model_config_secret_name` value with the name of the IBM watsonx Code Assistant that you want to connect to.
8. Click **Save**.

   The new Ansible Lightspeed pods are created. After the new pods are running successfully, the old Ansible Lightspeed pods are terminated.

Monitor your Red Hat Ansible Lightspeed on-premise deployment

After the Red Hat Ansible Lightspeed on-premise deployment is successful, use the following procedure to monitor the metrics on an API endpoint `/metrics`.

Procedure

1. Create a **system auditor** user:
   a. Create a user with a **system auditor** role in the Red Hat Ansible Automation Platform. For the procedure, see the Creating a user section of *Getting started with Ansible Automation Platform*.

b. Verify that you can log in to the Ansible Lightspeed portal for on-premise deployment (`https://<lightspeed_route>/`) as the newly-created system auditor user, and then log out.

2. Create a token for the **system auditor** user:
   a. Log in to the Ansible Lightspeed portal for on-premise deployment (`https://<lightspeed_route>/admin`) as an administrator by using the following credentials:
      - Username: **admin**
      - Password: The secret that is named as `<lightspeed-custom-resource-name>-admin-password` in the Red Hat OpenShift Container Platform cluster namespace where Red Hat Ansible Lightspeed is deployed.
   b. On the Django administration window, select **Users** from the Users area. A list of users is displayed.
   c. Verify that the user with the system auditor role is listed in the **Users** list.
   d. From the Django Oauth toolkit area, select **Access tokens > Add**.
   e. Provide the following information and click **Save**:
      - **User**: Use the magnifying glass icon to search and select the user with the system auditor role.
      - **Token**: Specify a token for the user. Copy this token for later use.
      - **Id token**: Select the token ID.
      - **Application**: Select **Ansible Lightspeed for VS Code**.
      - **Expires**: Select the date and time when you want the token to expire.
      - **Scope**: Specify the scope as **read write**.
        An access token is created for the user with a system auditor role.
   f. Log out from the Ansible Lightspeed portal for on-premise deployment.
3. Monitor your Red Hat Ansible Lightspeed on-premise deployment, by using the authorization token of the user with the system auditor role, to access the metrics endpoint `https://<lightspeed_route>/metrics` .

Use the Ansible Lightspeed REST API

As the platform administrator, you can configure and use the Ansible Lightspeed REST API to build a custom automation development and tooling workflow outside of VS Code. For information about the Ansible Lightspeed REST API, see Ansible AI Connect. 1.0.0 (v1) in the API catalog.

Before you begin

- Ensure that you are using the Red Hat Ansible Automation Platform operator patch version 2.5-20250305.9 or later and Red Hat Ansible Lightspeed operator version 2.5.250225 or later.

NOTE:
The Ansible Lightspeed REST API is available for Ansible Automation Platform 2.5 and later.

Procedure

1. Select the platform user for whom you want to grant REST API access.

   You can select an existing user or create a platform user in the Red Hat Ansible Automation Platform. For the procedure, see the Creating a user section of *Getting started with Ansible Automation Platform*.

2. Verify that you can log in to the Ansible Lightspeed portal for on-premise deployment (`https://<lightspeed_route>/`) as the platform user you selected or created, and then log out.

3. Create a token for the platform user:

   a. Log in to the Ansible Lightspeed portal for on-premise deployment (`https://<lightspeed_route>/admin`) as an administrator by using the following credentials:

      - Username: **admin**
      - Password: The secret that is named as `<lightspeed-custom-resource-name>-admin-password` in the Red Hat OpenShift Container Platform cluster namespace where Red Hat Ansible Lightspeed is deployed.

   b. On the Django administration window, select **Users** from the Users area. A list of users is displayed.

   c. Verify that the platform user is listed in the **Users** list.

   d. From the Django Oauth toolkit area, select **Access tokens > Add**.

   e. Provide the following information and click **Save**:

      - **User**: Use the magnifying glass icon to search and select the newly-created or existing user for whom you want to grant API access.
      - **Token**: Specify a token for the user. Copy this token for later use.
      - **Id token**: Select the token ID.
      - **Application**: Select **Ansible Lightspeed for VS Code**.
      - **Expires**: Select the date and time when you want the token to expire.
      - **Scope**: Specify the scope as **read write**.
        An access token is created for the user.

   f. Log out from the Ansible Lightspeed portal for on-premise deployment.

4. Make a direct call to the Ansible Lightspeed REST API by specifying the newly-created token in the authorization header:

```
curl -H "Authorization: Bearer <token>"
https://<lightspeed_route>/api/v1/me/
```

Install the Red Hat Ansible Automation Platform operator

Use this procedure to install the Ansible Automation Platform operator on the Red Hat OpenShift Container Platform.

Before you begin

- You have installed and configured automation controller.

Procedure

1. Log in to the Red Hat OpenShift Container Platform as an administrator.
2. Create a namespace:
    a. Go to **Administration › Namespaces**.
    b. Click **Create Namespace**.
    c. Enter a unique name for the namespace.
    d. Click **Create**.
3. Install the operator:
    a. Go to **Operators › OperatorHub**.
    b. Select the namespace where you want to install the Red Hat Ansible Automation Platform operator.
    c. Search for the Ansible Automation Platform operator.
    d. From the search results, select the Ansible Automation Platform (provided by Red Hat) tile.
    e. Select an **Update Channel**. You can select either **stable-2.x** or **stable-2.x-cluster-scoped** as the channel.
    f. Select the destination namespace if you selected “stable-2.x” as the update channel.
    g. Select **Install**. It takes a few minutes for the operator to be installed.
4. Click **View Operator** to see the details of your newly installed Red Hat Ansible Automation Platform operator.

Create a model configuration secret

You must create a configuration secret to connect to an IBM watsonx Code Assistant model, which can be either an on-premise deployment or a cloud deployment.

Before you begin

- You have installed the Ansible Automation Platform operator 2.5.0-0.1753402603 or later on the Red Hat OpenShift Container Platform.
- You have created an OAuth application in the automation controller.
- You have obtained an API key and a model ID from IBM watsonx Code Assistant.
  For information about obtaining an API key and model ID from IBM watsonx Code Assistant, see the IBM watsonx Code Assistant documentation. For information about installing IBM watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data, see the watsonx Code Assistant for Red Hat Ansible Lightspeed on Cloud Pak for Data documentation.

Procedure

1. Go to the Red Hat OpenShift Container Platform.
2. Select **Workloads › Secrets**.
3. Click **Create › Key/value secret**.
4. From the **Projects** list, select the namespace that you created when you installed the Red Hat Ansible Automation Platform operator.
5. Click **Create › Key/value secret**.
6. In **Secret name**, enter a unique name for the secret. For example, `model-aiconnect`.
7. Add the following keys and their associated values individually:

Key	Value
`username`	*For on-premise deployment only* Enter the username you use to connect to an IBM Cloud Pak for Data deployment.
`model_type`	Enter one of the following values per your IBM watsonx Code Assistant model: • For on-premise deployment (IBM Cloud Pak for Data): `wca-onprem` • For cloud deployment (IBM Cloud): `wca`

Key	Value
`model_url`	Enter the URL of the IBM watsonx Code Assistant model. For cloud deployment, the model URL could be `https://api.dataplatform.cloud.ibm.com`.
`model_api_key`	Enter the API key of your IBM watsonx Code Assistant model that was generated during the model installation.
`model_id`	Enter the ID of your IBM watsonx Code Assistant model.
`model_verify_ssl`	*Optional, and supported on Ansible Automation Platform 2.5 and later* This key controls whether the SSL certificate of the IBM watsonx Code Assistant model is verified. Default = `true`
`model_enable_anonymization`	*Optional and supported on Ansible Automation Platform 2.5.250730 and later* This key controls whether the anonymization of Personally Identifiable Information (PII) is enabled. PII information includes passwords, IP addresses, email addresses, and other sensitive data. When is enabled, users' personal information is modified to some generic values to protect their data and reduce the risk of data leaks. You can turn off anonymization by setting the value to `false` to retain all original information entered by users and improve the quality of the answers. Disabling anonymization for Ansible Lightspeed hybrid deployments (the model is in IBM watsonx Code Assistant on IBM Cloud) results in users' PII being sent to IBM Cloud. Default = `true`

IMPORTANT:
Ensure that you do not accidentally add any whitespace characters (extra line, space, and so on) to the value fields. If there are any extra or erroneous characters in the secret, the connection to IBM watsonx Code Assistant fails.

8. Click **Create**.

After you create the model configuration secret, you must update the YAML file of the Ansible Automation Platform operator.

Update the YAML file of the Ansible Automation Platform operator

After you create the model configuration secret, you must update the YAML file of the Ansible Automation Platform operator to use the secret.

Procedure

1. Go to the Red Hat OpenShift Container Platform.
2. Select **Operators › Installed Operators**.
3. From the list of installed operators, select the **Ansible Automation Platform** operator.
4. Locate and select the **Ansible Automation Platform** custom resource, and then click the required app.
5. Select the **YAML** tab.
6. Scroll the text to find the `Lightspeed` category, and add the following details under the `spec:` section:

```
spec:
  lightspeed:
    disabled: false
    model_config_secret_name: <Name of the model configuration secret that you
recently created.>
```

7. Click **Save**. The Ansible Lightspeed service takes a few minutes to set up.

Develop Ansible content

As an automation developer, you can use Red Hat Ansible Lightspeed to implement your organization's automation strategy. Red Hat Ansible Lightspeed can help you create and use custom automation content.

This section provides information about how to get set up as an automation developer on Red Hat Ansible Lightspeed, with details on how to:

- Access the Ansible Lightspeed portal as an automation developer
- Install and configure the VS Code
- Create task recommendations
- Create playbooks and view playbook explanations

- Provide feedback on the Ansible Lightspeed service

Access the Ansible Lightspeed portal as an automation developer

You can access the Red Hat Ansible Lightspeed portal by authenticating through your Red Hat Single Sign-On (RH-SSO) account. After you are logged in, your assigned user role is displayed on the Ansible Lightspeed portal login screen.

User login scenarios

Scenario	Result
You are a RH-SSO user. This is the typical scenario for accessing Red Hat Ansible Lightspeed as an Ansible user.	You are routed to the Red Hat Ansible Lightspeed paid commercial offering.
You are a RH-SSO user, but your organization administrator has not configured Red Hat Ansible Lightspeed to connect with IBM watsonx Code Assistant.	You are routed to the Red Hat Ansible Lightspeed paid commercial offering with a message that your organization administrator has not configured a model for your organization.

Related information
Ansible Lightspeed portal

Log in to the Ansible Lightspeed portal as an automation developer

Use the following procedure to log into the Ansible Lightspeed portal with your Red Hat account.

Procedure

1. Go to the Ansible Lightspeed portal login page.
2. Click **Log in > Log in with Red Hat**.
3. Enter your Red Hat account username and password.

   On successful authentication, the login screen is displayed along with your username and your assigned user role.

Install and configure the Ansible VS Code extension

Red Hat Ansible Lightspeed with IBM watsonx Code Assistant integrates with the Ansible Visual Studio (VS) Code extension. When enabled, the extension automatically collects recommendations, usage telemetry, and Ansible YAML file state through automated events.

To access Red Hat Ansible Lightspeed, all Ansible users must install and configure the Ansible VS Code extension in their VS Code. The Ansible VS Code extension uses the Ansible-specific IBM watsonx Granite model configured in the Red Hat Ansible Lightspeed administrator portal as the default mode for all users in your organization.

Connectivity requirements

To generate code recommendations, the Ansible Lightspeed service in Visual Studio (VS) Code editor requires access to the following outbound domain:

- Red Hat Ansible Lightspeed with IBM watsonx Code Assistant, this is linked below.

The outbound connections are encrypted on TCP protocol port 443.

Related information
Red Hat Ansible Lightspeed with IBM watsonx Code Assistant

Install and configure the Ansible VS Code extension

Red Hat Ansible Lightspeed with IBM watsonx Code Assistant integrates with the Ansible Visual Studio (VS) Code extension. When enabled, the extension automatically collects recommendations, usage telemetry, and Ansible YAML file state through automated events.

To access Red Hat Ansible Lightspeed, all Ansible users must install and configure the Ansible VS Code extension in their VS Code. The Ansible VS Code extension uses the Ansible-specific IBM watsonx Granite model configured in the Red Hat Ansible Lightspeed administrator portal as the default mode for all users in your organization.

Connectivity requirements

To generate code recommendations, the Ansible Lightspeed service in Visual Studio (VS) Code editor requires access to the following outbound domain:

- Red Hat Ansible Lightspeed with IBM watsonx Code Assistant, this is linked below.

The outbound connections are encrypted on TCP protocol port 443.

Related information
Red Hat Ansible Lightspeed with IBM watsonx Code Assistant

Install the Ansible VS Code extension

Use the following procedure to install the Ansible Lightspeed extension in VS Code.

Before you begin

- VS Code version 1.70.1 or later.

NOTE:
You can also install VScode derivatives, such as VScode Insider or VS Codium.

Procedure

1. Open the VS Code application.
2. From the navigation menu, click the **Extensions** icon.
3. In the **Search** field, enter **Ansible**.
4. Select **Ansible** to choose the Ansible language support extension published by Red Hat.
5. Click **Install**.
6. After installation is complete, verify your VSCode installation:
    a. Create a new YAML file using the `.yml` or `.yaml` file extension.
    b. From the **Status** toolbar, click the language indicator and select **Ansible** to associate the Ansible language type with the new YAML file.
    c. Start writing a test playbook. Contextual aids are displayed as you start creating your content.

Configure the Ansible VS Code extension

Configure third-party LLM providers, such as IBM watsonx Code Assistant or Google Gemini, within the Ansible VS Code extension.

Before you begin

- You have installed the Ansible VS Code extension v25.12.3.
- You have obtained a valid API key for your chosen third-party LLM provider.

Procedure

1. Open the VS Code application.
2. From the Activity bar, click the **Extensions** icon  .
3. From the Installed Extensions list, select **Ansible**.
4. From the **Ansible** extension page, click the **Settings** icon and select **Extension Settings**.
5. Select **Ansible Lightspeed** settings, and specify the following information:

UI field	Description
Ansible Lightspeed: Enabled	Select this checkbox to enable the Red Hat Ansible Lightspeed service.
Ansible Lightspeed: Provider	Select the active AI service for code generation. Choose the AI provider from the following options: • `wca` : (Default setting) Uses IBM watsonx Code Assistant as the AI provider. • `google` : Uses Google Gemini as the AI provider.
Ansible Lightspeed: Model Name	Specify the specific AI model version to use for code generation: • For IBM watsonx Code Assistant: (Required) The IBM watsonx Code Assistant model name or ID that you want to use for code generation. • For Google Gemini: (Optional) The system applies a recommended default model if left blank.

UI field	Description
Ansible Lightspeed: Api Key	Specify the secret credential required to authenticate requests with third-party model providers. • For IBM watsonx Code Assistant: (Optional) This field is not used for the IBM WCA provider. IBM watsonx Code Assistant relies on a separate OAuth2 login flow via the Red Hat portal. • For Google Gemini: (Required) Paste your active Google Gemini API key into this field. This token authorizes the extension to send prompts to Google's servers.
Ansible Lightspeed: Api Endpoint	Specify the destination URL for network requests sent by the Ansible VS Code extension. • For IBM watsonx Code Assistant: (Required) This field allows modification of the service URL for IBM watsonx Code Assistant connections. The default URL is `https://c.ai.ansible.redhat.com`. • For Google Gemini: (Not configurable) This setting is not configurable when using the Google provider. The extension automatically manages the correct endpoint URL for Google services internally.
Ansible Lightspeed Suggestions: Enabled	Toggle the automatic display of inline code completions within the Ansible VS Code editor. Inline suggestions are currently available for the IBM watsonx Code Assistant provider only.
Ansible Lightspeed: Timeout	Define the maximum duration the Ansible VS Code extension waits for a server response. The default timeout for API calls is 3000 milliseconds.

Result

Your settings are automatically saved in VS Code.

NOTE:
If your organization recently subscribed to the Red Hat Ansible Automation Platform, it might take a few hours for Red Hat Ansible Lightspeed to detect the new subscription. In VS Code, use the **Refresh** button in the Ansible extension from the Activity bar to check again.

Related information
Troubleshooting Ansible Visual Studio Code extension errors

Log in to Ansible Lightspeed through the Ansible VS Code extension

After installing and configuring the VS Code extension, you can log in to the Ansible Lightspeed service.

Red Hat Ansible Lightspeed provides different sign-in methods depending on whether you are using the cloud service or the on-premise deployment.

- **Ansible Lightspeed on-premise deployments**
  Users are authenticated using your Red Hat Ansible Automation Platform login.

  To sign in, you can use the **Connect** button in the Ansible Lightspeed view, or the **Sign in with Ansible Lightspeed to use Ansible** option in the **Accounts** menu. Once prompted in the browser, select **Log in with Ansible Automation Platform**, and log in with the authorization mechanism that your automation controller is configured with.

- **Ansible Lightspeed cloud service**
  Users are authenticated using Red Hat Single Sign-On (RH-SSO).

  To sign in from VS Code, you can use the **Connect** button in the Ansible Lightspeed view, or the **Sign in with Ansible Lightspeed to use Ansible** option in the **Accounts** menu. Follow the on-screen prompts to log in and access the Ansible Lightspeed service using your RH-SSO.

> **NOTE:**
> If you are using a cloud development environment at a domain unknown by Ansible Lightspeed, such as on-premise Red Hat OpenShift Dev Spaces, your **Accounts** sign-in menu provides the option **Sign-in with Red Hat to use Ansible**.
>
> This option uses a device code flow to successfully complete the sign-in process and requires the Red Hat Authentication extension v0.2.0 or later. If you require this authentication flow but don't see the **Sign-in with Red Hat to use Ansible** option, ensure you are using the Ansible VS Code extension v24.5.2 or later.

Procedure

1. Open the VS Code application.
2. Sign in using either the **Connect** button in the Ansible Lightspeed view or the **Accounts** menu.
   - **Sign in using the Connect button**:
     a. From the VS Code activity bar, click the **Ansible** icon.
     b. In the **Ansible Lightspeed** view, click **Connect**.
     c. Follow the on-screen prompts to sign in to Ansible Lightspeed.

- **Sign in using the Accounts menu**:
  a. From the VS Code activity bar, click the **Accounts** menu.
  b. Sign in with Ansible Lightspeed to use Ansible or sign in with Red Hat to use Ansible, depending on the sign-in option you are presented with.

> **NOTE:**
> - The sign-in options are displayed when the VS Code extension is in an active state. The extension is activated after you open the Ansible side panel or after you open an Ansible file in the VS Code editor. If you do not see this option, use the **Connect** button to link to the Ansible Lightspeed service.
> - If you are using a cloud development environment at a domain unknown by Ansible Lightspeed, such as on-premise Red Hat OpenShift Dev Spaces, your **Accounts** sign-in menu provides the option **Sign-in with Red Hat to use Ansible**. This option uses a device code flow to successfully complete the sign-in process and requires the Red Hat Authentication extension v0.2.0 or later. If you require this authentication flow but don't see the **Sign-in with Red Hat to use Ansible** option, ensure you are using the Ansible VS Code extension v24.5.2 or later.

  c. Follow the on-screen prompts to sign in to Red Hat Ansible Lightspeed.

Result

On successful authentication, the login screen is displayed along with your username and your assigned user role.

Create task recommendations

Red Hat Ansible Lightspeed is integrated into Visual Studio (VS) Code through the Ansible VS Code extension. You can request code recommendations for your task intent by using Ansible VS Code extension.

You can perform the following tasks from the Ansible VS Code extension:

- **Create single task or multitask requests by using natural language prompts**
  - Create a single task prompt
    Write a description of your task in the `- name:` key of a new task line in your Ansible file. For example, to automate a task of installing PostgreSQL server, you can enter the prompt `- name: Install postgresql-server`.
  - Create a multitask prompt
    Place your cursor on a new line in your Ansible YAML file at the correct indentation, and start your prompt with a Pound key (#).

Write the descriptions of your tasks, separating each prompt by using Ampersand symbols (&). For example, to automate a multitask of installing PostgreSQL server and running the initial PostgreSQL setup command, you can enter the prompt `# Install postgresql-server & run postgresql-setup command`.

The Ansible Lightspeed service reads the text, interacts with the IBM watsonx Code Assistant model, and generates Ansible task recommendations based on your natural language prompt.

> **NOTE:**
> Currently, Red Hat Ansible Lightspeed supports user prompts in English language only. However, there could be instances where the training data that was used to train the IBM watsonx Code Assistant models included non-English language. In such scenarios, the model can generate code recommendations for prompts made in the same non-English language, but the generated code recommendations might or might not be accurate.

- **View the content source matching results**
  For each generated code recommendation, Red Hat Ansible Lightspeed lists content source matches, including details such as potential source, content author, and relevant licenses. You can use this data to gain insight into potential training data sources used to generate the code recommendations.
- **Provide feedback on the Ansible Lightspeed service**
  The Ansible Lightspeed service learns your organizational patterns and improves the code recommendation experience over time. You can provide feedback on whether the generated code recommendations were suitable for your task intent. This feedback enables Red Hat Ansible Lightspeed with IBM watsonx Code Assistant to improve on the quality of its suggestions.

Best practices to improve the recommended guidance

Follow these guidelines to improve the likelihood of a quality code recommendation.

- Ensure that your YAML file is properly formatted.
- Avoid context switching within a single playbook file.
  The Ansible Lightspeed service attempts to correlate earlier tasks to the active recommendation, and the entire contents of the file before the cursor position are used as context by the model. If the earlier task is not relevant to your prompt, VS code provides inline suggestions instead of code recommendations.
- Reword your natural language prompts to get code recommendations that match your task intent.

If you get a recommendation that does not align with the intent of your task name, then rewording your prompt to provide more information about what is desired can lead to improved results.

- Use descriptive prompts and provide additional content to improve the code recommendations.
  Red Hat Ansible Lightspeed reads the full Ansible YAML file when generating a code recommendation. Using descriptive prompts and having additional YAML file content related to the desired task improves the code recommendation. For example, you can add the previous Ansible tasks and appropriate playbook and variable names to improve the code recommendations.

Related information
Ansible YAML syntax guidelines

Create single task recommendations

You can request code recommendations for a single task by entering natural language prompts in Ansible VS Code extension. The Ansible Lightspeed service reads the text, interacts with the IBM watsonx Code Assistant model, and generates the code recommendations.

Before you begin

- You meet **one** of the following requirements:
  - Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
  - Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.
- You have installed and configured the Ansible VS Code extension.

Procedure

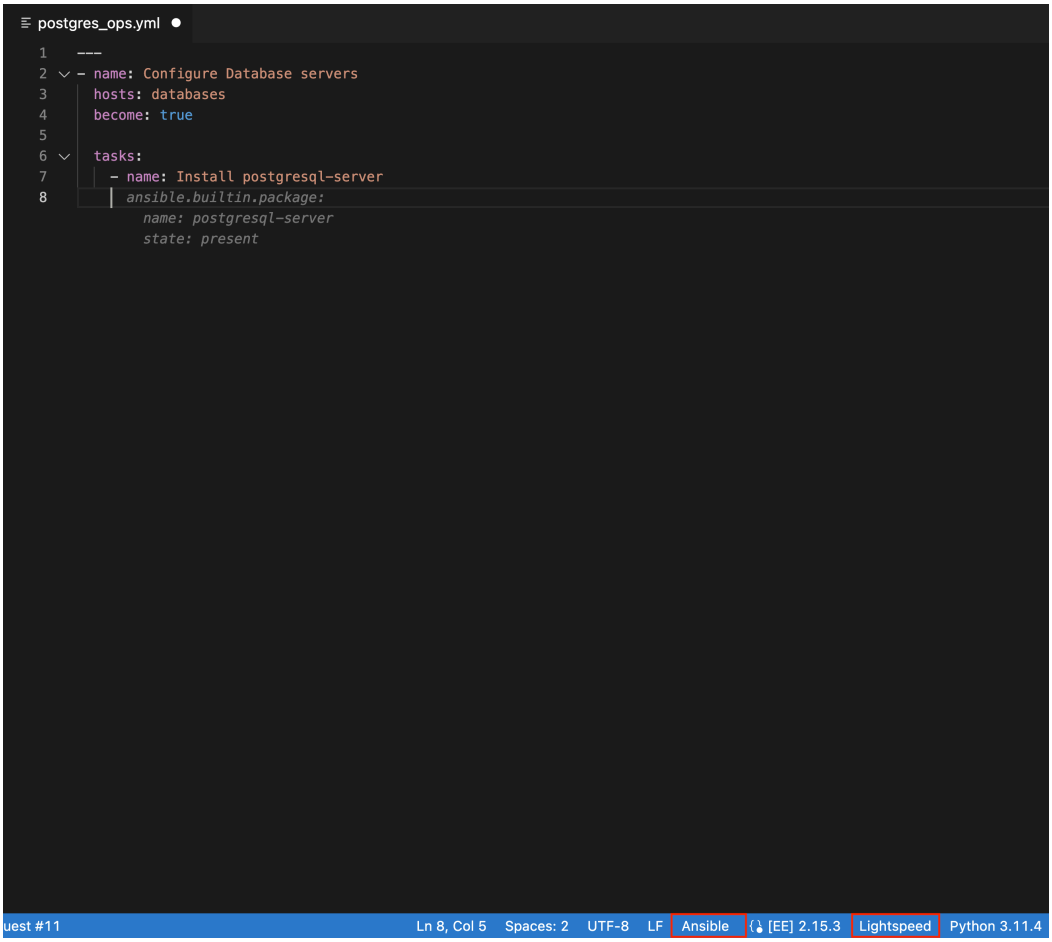
1. Log in to VS Code with your Red Hat account.
2. Create a new YAML file or use an existing YAML file:
   - Create a YAML file:
     a. Select **File > New Text File**.
     b. From the lower right of the screen, click **Plain Text**, and in the language mode, select **Ansible**.
     c. Save the file as a YAML file format extension (`.yml` or `.yaml`).
   - Use an existing YAML file:
     a. On the bottom right of the screen, click the existing language mode, and in the language mode settings, select **Ansible**.

NOTE:
If you do not see the language mode section in your VS Code editor, from the Command Palette, select **Configure Language Mode > Ansible**.

3. Verify that you see an entry for **Lightspeed** on the status bar at the lower right of VS Code.

   If **Ansible** is already selected as the desired language but the **Lightspeed** entry is not displayed, re-select **Ansible** as the language mode. The following illustration shows **Lightspeed** and **Ansible** entries on the VS Code status bar.

Ansible and Lightspeed set as selected language mode



4. Optional: If you see an error message about missing Ansible lint, you can install the missing module or disable it. Perform any one of the following tasks:

   - Install Ansible lint: For installation information, see the Installing section of the Ansible Lint documentation.
   - Disable Ansible lint:

     a. From the Activity bar, click the **Extensions** icon  .

     b. From the **Installed** extensions list, select **Ansible**.

     c. From the **Ansible** extension page, click the **Settings** icon and select **Extension Settings**.

d. Clear the **Ansible > Validation > Lint: Enabled** checkbox.

5. Create a playbook or use an existing playbook.

   For more information, see the Getting started with playbooks guide.

6. In the playbook, provide the following information to request code recommendations for a single task:

   a. Add a new Ansible task by starting a new line with `- name:` at the correct indentation.

   b. Add a detailed natural language prompt in the task description after `- name:` on the same line. For example, you can specify the following single task prompt: `- name: Install postgresql-server`

   c. Press **Enter** directly after the task description. Keep the cursor at the same location in your file, and wait for the code recommendation results to populate.

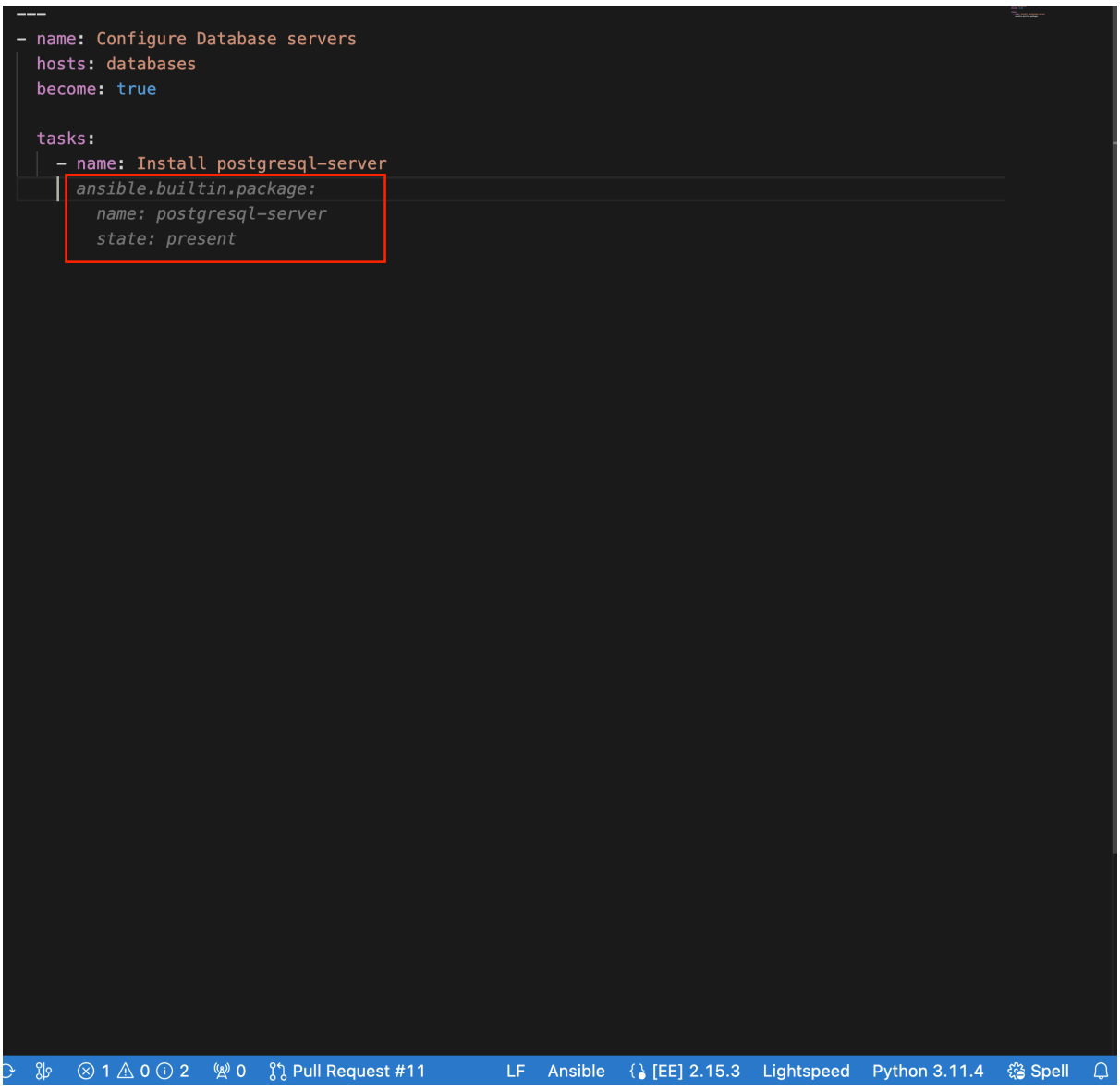
   The Ansible Lightspeed service is engaged, and it starts generating code recommendations for a single task.

> **IMPORTANT:**
> Ansible Lightspeed service takes around 5 seconds per task to populate the code recommendations. If you are using a multitask prompt, the Ansible Lightspeed service takes a bit longer (number of tasks times 5 seconds) to populate the results. Do not move your cursor or press any key while the code recommendation is being generated. If you change the cursor location or press any key, Ansible VS Code extension cancels the request and the Ansible Lightspeed service does not process your request.

When the Ansible Lightspeed service is engaged, a **Lightspeed** processing status indicator is displayed in the lower right of the screen to denote that your code recommendation is being generated.

7. View your code recommendations and ensure that the recommendations match your task intent.

   The following illustration shows the code recommendations generated by the Ansible Lightspeed service for the single task **Install postgresql-server**:



8. Accept or reject the code recommendations:

   - To accept a code recommendation, press **Tab**.
   - To reject a code recommendation, press **Esc**.

> **NOTE:**
> If you reject a recommendation, you can modify the prompt and review the generated code recommendations once again to match your task intent.

9. After you accept the code recommendation, click the **ANSIBLE** tab to see the content source matching results.

   For each generated code recommendation, Ansible Lightspeed lists content source matches, including details such as potential source, content author, and relevant licenses. You can use this data to gain insight into potential training data sources used to generate the code recommendations.

10. Click **Save** to save the code recommendation changes in your Ansible YAML file.

Related information
Troubleshooting Ansible Visual Studio Code extension errors
Troubleshooting Ansible code bot errors

Create multitask recommendations

You can request multitask code recommendations by entering a sequence of natural language task prompts in Ansible VS Code extension. In a YAML file, start your prompt with a pound symbol (#), and separate each prompt by using the ampersand symbol (&).

Before you begin

- You meet **one** of the following requirements:
  - Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
  - Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.
- You have installed and configured the Ansible VS Code extension.

Example of a multitask prompt

```
# Install postgresql-server & run postgresql-setup command
```

For better readability, you can split your multitask inline prompts over multiple lines. To achieve this, end your current line with an ampersand symbol (&) and start the next line with the hash symbol (#).

Example of a multitask prompt split over multiple lines

```
# Create a keypair called lightspeed-keypair & create a vpc & create vpc_id var &
# create a security group that allows SSH & create subnet with 10.0.1.0/24 cidr &
# create an internet gateway & create a route table
```

The Ansible Lightspeed service reads the text, interacts with the IBM watsonx Code Assistant model, and generates the code recommendations.

NOTE:
While entering a multitask prompt, the Ansible VS Code extension might display a warning if you have long lines in your prompt based on your ansible-lint settings. This warning is a minor readability error and does not impact the quality of your code recommendation output. To resolve the error, you can either ignore it or fix it by splitting your multitask inline prompt over multiple lines.

Procedure

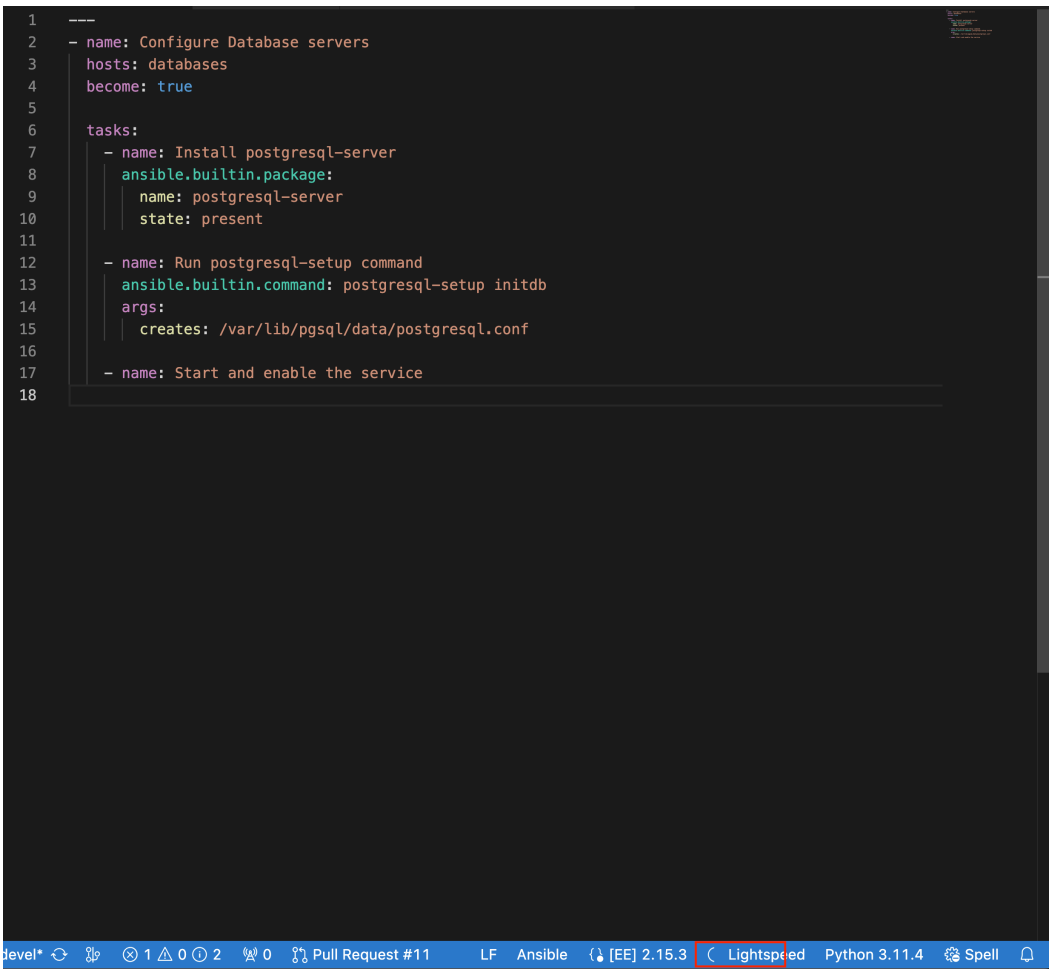
1. Log in to VS Code with your Red Hat account.
2. Create a new YAML file or use an existing YAML file.
   - Create a YAML file:
     a. Select **File > New Text File**.
     b. From the lower right of the screen, click **Plain Text**, and in the language mode, select **Ansible**.
     c. Save the file as a YAML file format extension (`.yml` or `.yaml`).
   - Use an existing YAML file:
     a. On the bottom right of the screen, click the existing language mode, and in the language mode settings, select **Ansible**.

     NOTE:
     If you do not see the language mode section in your VS Code editor, from the Command Palette, select **Configure Language Mode > Ansible**.

3. Verify that you see an entry for **Lightspeed** on the status bar at the lower right of VS Code.

   If **Ansible** is already selected as the desired language but the **Lightspeed** entry is not displayed, re-select **Ansible** as the language mode. The following illustration shows **Lightspeed** entry on the VS Code status bar.

   Ansible and Lightspeed set as selected language mode



4. Optional: If you see an error message about missing Ansible lint, you can install the missing module or disable it. Perform any one of the following tasks:

   - Install Ansible lint: For installation information, see the Installing section of the Ansible Lint documentation.
   - Disable Ansible lint:

     a. From the Activity bar, click the **Extensions** icon  .

     b. From the **Installed** extensions list, select **Ansible**.

     c. From the **Ansible** extension page, click the **Settings** icon and select **Extension Settings**.

     d. Clear the **Ansible > Validation > Lint: Enabled** checkbox.

5. Create a playbook or use an existing playbook.

   For more information, see the Getting started with playbooks guide.

6. In the playbook, provide the following information to request multitask code recommendations:

   a. Start a new YAML file comment by entering a pound symbol (#) at the correct indentation.

b. Add a detailed natural language prompt in a sequence, separating each task by using the ampersand symbol (&).

Example of a multitask prompt

```
# Install postgresql-server & run postgresql-setup command
```

For better readability, split your multitask inline prompts over multiple lines. To achieve this, end your current line with an ampersand symbol (&) and start the next line with the hash symbol (#).

Example of a multitask prompt split over multiple lines

```
# Create a keypair called lightspeed-keypair & create a vpc & create
vpc_id var &

# create a security group that allows SSH & create subnet with
10.0.1.0/24 cidr &

# create an internet gateway & create a route table
```

c. Press **Enter** directly after the task description. Keep the cursor at the same location in your file, and wait for the code recommendation results to populate.

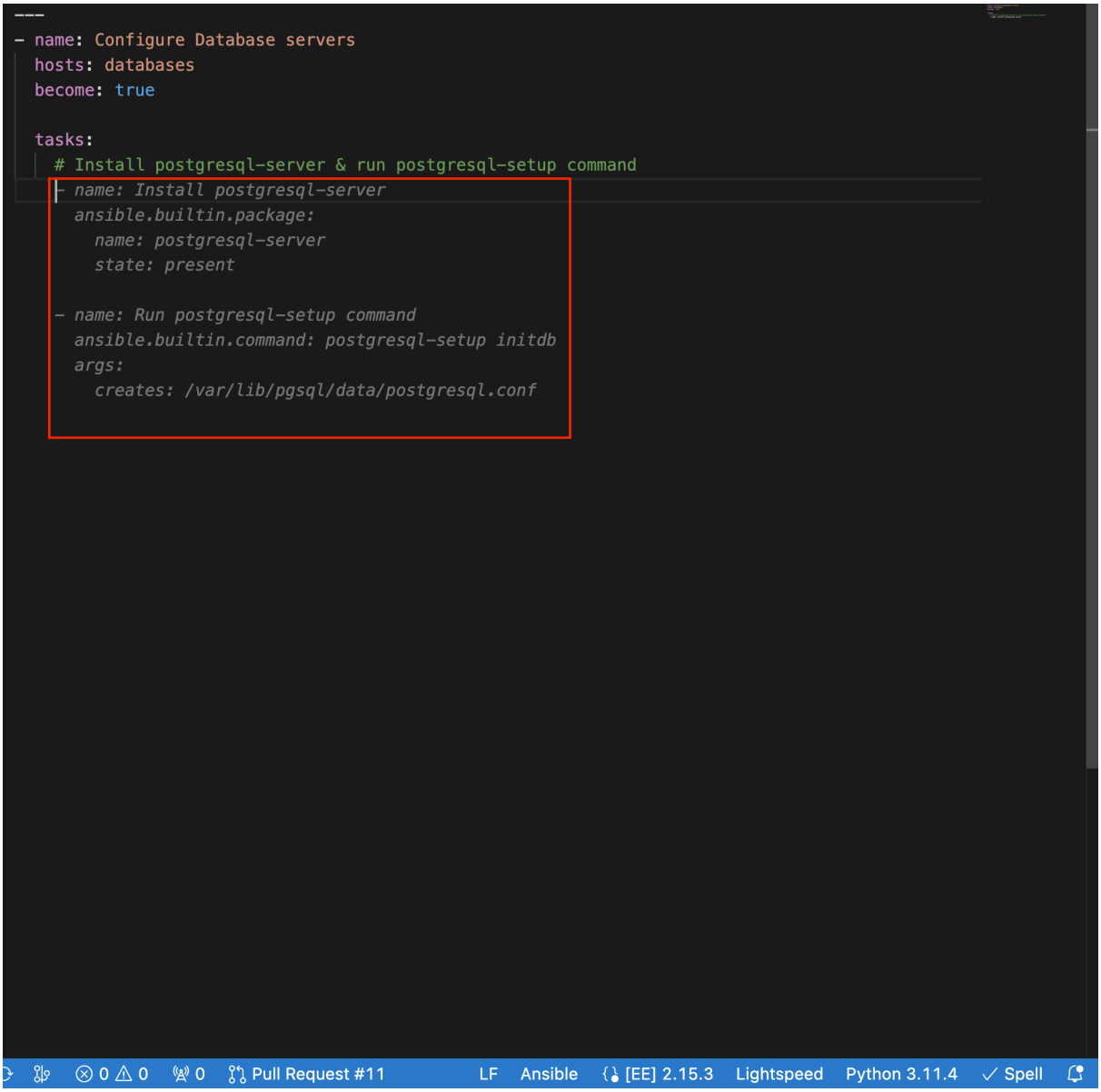
The Ansible Lightspeed service is engaged, and it starts generating code recommendations for multiple tasks.

IMPORTANT:
Ansible Lightspeed service takes around 5 seconds per task to populate the code recommendations. If you are using a multitask prompt, the Ansible Lightspeed service takes a bit longer (number of tasks times 5 seconds) to populate the results. Do not move your cursor or press any key while the code recommendation is being generated. If you change the cursor location or press any key, Ansible VS Code extension cancels the request and the Ansible Lightspeed service does not process your request.

When the Ansible Lightspeed service is engaged, a **Lightspeed** processing status indicator is displayed in the lower right of the screen to denote that your code recommendation is being generated.

7. Optional: If multitask code recommendations are not being generated, log out of VS Code and log in again using your Red Hat account.
8. View your code recommendations and ensure that the recommendations match your task intent.

The following illustration shows the code recommendations generated by the Ansible Lightspeed service for the multitask prompt **Install postgresql-server & run postgresql-setup command**: :



9. Accept or reject the code recommendations:

    - To accept a code recommendation, press **Tab**.
    - To reject a code recommendation, press **Esc**.

> **NOTE:**
> If you reject a recommendation, you can modify the prompt and review the generated code recommendations once again to match your task intent.

10. After you accept the code recommendation, click the **ANSIBLE** tab to see the content source matching results.

For each generated code recommendation, Ansible Lightspeed lists content source matches, including details such as potential source, content author, and relevant licenses. You can use this data to gain insight into potential training data sources used to generate the code recommendations.

11. Click **Save** to save the code recommendation changes in your Ansible YAML file.

Related information
Troubleshooting Ansible Visual Studio Code extension errors
Troubleshooting Ansible code bot errors

View the Ansible Lightspeed training matches

The Red Hat Ansible Lightspeed with IBM watsonx Code Assistant machine learning model is trained on the following content:

- Existing public or private Git repositories
- Content from Ansible Galaxy

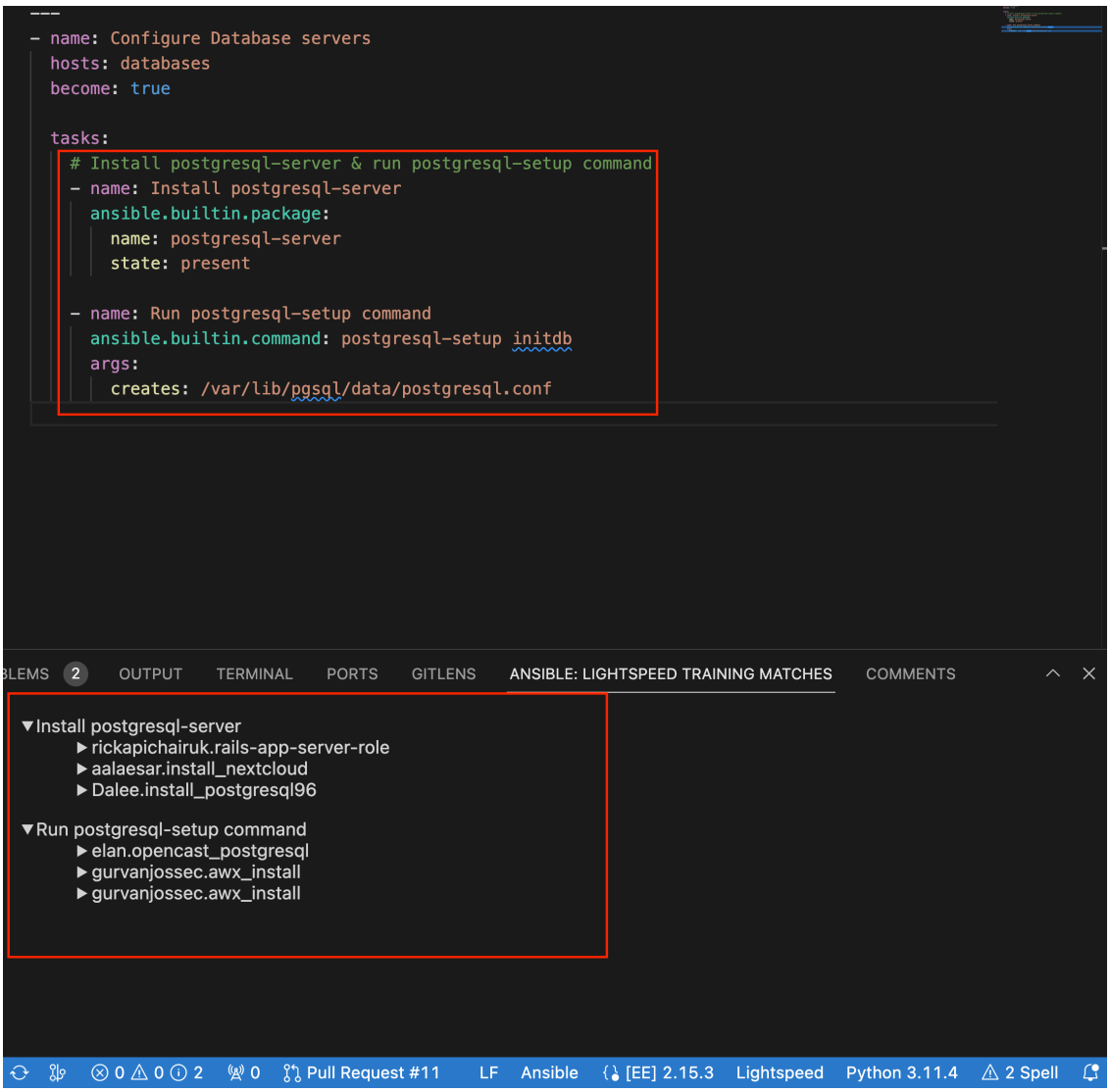
IBM watsonx Code Assistant uses generative AI technology and various types of Ansible content to train the model. Therefore, it is not possible to trace the specific training data that produced a given code recommendation.

For each generated code recommendation, Red Hat Ansible Lightspeed lists the content source matches, including details such as potential source, content author, and relevant licenses. You can use this data to gain insight into potential training data sources used to generate the code recommendations.

After you enter a natural language prompt in VS Code and see the generated code recommendations, you can view the content source matches on the **ANSIBLE: LIGHTSPEED TRAINING MATCHES** tab.

For example, the following illustration shows the training matches for the multitask recommendation **Install postgresql-server & run postgresql-setup command**:

Figure: Training matches for a multitask recommendation



This capability enables you to find out the open source license terms that are associated with related training data. However, it is unlikely that either the training data used in fine-tuning the code or the output recommendations themselves are protected by copyright, or that the output reproduces training data that is controlled by copyright licensing terms.

NOTE:
Red Hat does not claim any copyright or other intellectual property rights in the suggestions generated by Red Hat Ansible Lightspeed with IBM watsonx Code Assistant.

Create playbooks and view playbook explanations

Using the Ansible VS Code extension, you can create Ansible playbooks using a natural language interface in English.

Red Hat Ansible Lightspeed with IBM watsonx Code Assistant reads the natural language prompts and generates an entire playbook recommendation based on your intent. You can also view the explanations for new or existing playbooks. The playbook explanations describe what the playbook does and contextualize its impact.

These capabilities enable Ansible developers to use natural language prompts to create new Ansible playbooks quickly and more efficiently and also get an explanation for existing Ansible playbook, thereby reducing the overall onboarding learning period.

NOTE:
You can create playbooks and view playbook explanations when connecting to the Red Hat Ansible Lightspeed cloud service and on-premise deployments.

Best practices to create playbooks

Follow these guidelines for the highest quality of a playbook recommendation.

- Ensure that the goal statements directly specify what the playbook must do.
  Your statement should start with the goal of the playbook, for example, `Apply security patches to RHEL9`. Avoid starting statements with `Create a playbook that`, `Please prepare a playbook that`, or `I need help with`.
- Ensure that the goal statement does not contain new lines.
- Ensure that the goal statement is not more than one sentence.
  You might have to repeat the details in the goal statement to produce the best results. It is recommended that you use the generated outline as feedback about whether your goal statement might benefit from more or less details, and then modify the goal statement as necessary.
- Ensure the following when you edit the outline:
  - Do not restate the goal of the playbook.
  - Verify that the steps considered capture the key steps in the playbook. The steps need not reflect each and every task that is expected in the playbook.
  - Keep the step description in one sentence without adding new lines to the outline.

Related information
Get started with Ansible Playbooks

Generate Ansible playbooks

You can use the natural language interface in the Ansible VS Code extension to generate an entire Ansible playbook.

Before you begin

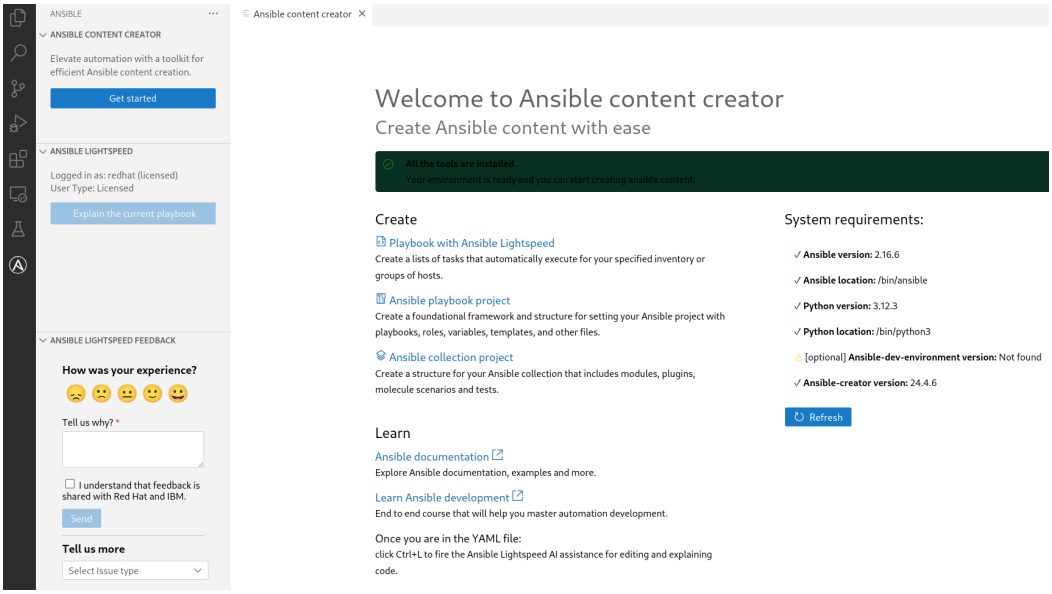
- You meet **one** of the following requirements:
  - Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
  - Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.
- You have installed and configured the Ansible VS Code extension.

Procedure

1. Log in to VS Code with your Red Hat account.
2. From the **Activity** bar, click the **Ansible** icon.
3. Under **Ansible Creator**, click **Get started**. The **Ansible Content Creator** page is displayed.

   The following illustration displays the **Ansible Content Creator** page:

Settings to create Ansible playbooks



4. Select the **Playbook with Ansible Lightspeed** tile. The **Create a playbook** page is displayed.
5. In the **What do you want the playbook to accomplish?** field, enter the prompts to create a playbook and click **Analyze**.

After a few seconds, the recommended steps for your playbook intent are displayed in the **Review the suggested steps for your playbook and modify as needed** field.

6. Perform one of the following tasks:
   - If the steps match your intent: Click **Generate Playbook**.
   - If any modifications are required: Click the editor and update the tasks or steps to suit your intent.
   - If the task suggestions do not match your intent: Click **Back** to change the original prompt and start over.
   - If you want to restore the original task suggestions: Click **Reset** and proceed to the next step.
7. After you verify the steps, click **Generate playbook**.

   It takes a few seconds for the playbook to generate, and **The following playbook was generated for you** field displays the newly generated playbook.
8. Click **Open editor**. The generated playbook opens as an untitled YAML file in the VS Code editor.
9. Save the untitled YAML file.

View the playbook explanations

You can request explanations for a newly created playbook as well as an existing Ansible playbook.

Before you begin

- You meet **one** of the following requirements:
  - Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
  - Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.
- You have installed and configured the Ansible VS Code extension.
- You have opened the playbook whose explanation you want to view.

Procedure

1. Log in to VS Code with your Red Hat account.
2. Open an Ansible playbook YAML file in VS Code.
3. Use one of the following methods to view the playbook explanation:
   - **From an active playbook YAML file**:
     a. Place your cursor anywhere within the playbook file.

b. Right-click and select **Explain the playbook with Ansible Lightspeed**.

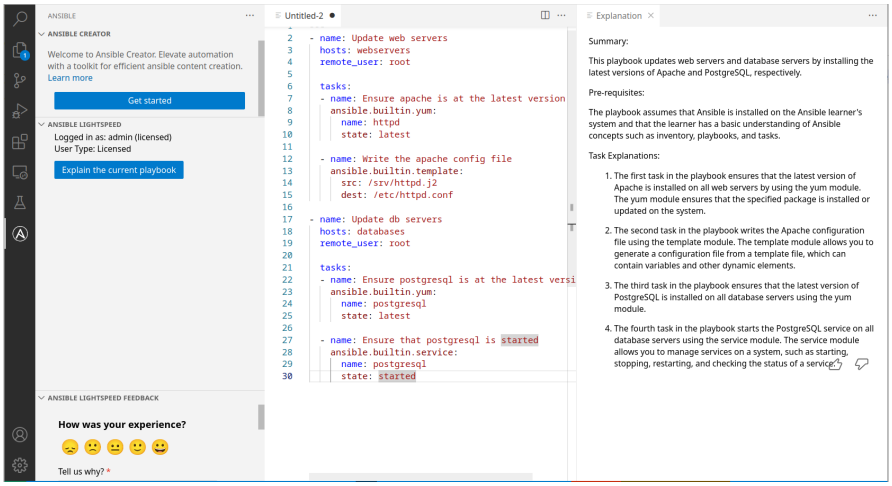
- **From the Ansible panel**:

  a. From the navigation menu, click the **Ansible** icon.

  b. Select **Explain the current playbook**.
  The playbook explanation is displayed on the right panel of the VS Code screen.

  The following illustration shows an example of a playbook explanation:

Example of a playbook explanation



Create roles and view roles explanations

You can create roles within Ansible collections using the Ansible VS Code extension. To create roles, use the Ansible VS Code extension, select the **Role Generation** option, and then enter the natural language prompts in English language.

Red Hat Ansible Lightspeed reads the natural language prompts and creates a role recommendation based on your intent. You can also view the explanations for new or existing roles. The role explanations describe what the role does and contextualize its impact.

These capabilities enable Ansible developers to use natural language prompts to create Ansible roles quickly and more efficiently, as well as get an explanation for existing Ansible roles. In addition to generating playbooks, role generation can now further reduce your team's overall onboarding learning period. For information about Ansible roles, see *Bundle content with Ansible roles* in the *Related Links* section.

NOTE:
You can create roles and view role explanations when connecting to the Red Hat Ansible Lightspeed cloud service only.

Related information

Create roles within collections

You can use the natural language interface in the Ansible VS Code extension to create one or more roles within an Ansible collection.

Before you begin

- You meet **one** of the following requirements:
  - Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
  - Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.
- You have installed and configured the Ansible VS Code extension v25.3.0 or later. For the procedure, see Installing and configuring the Ansible VS Code extension.
- You have an existing Ansible environment configured with a valid collection directory within the Ansible VS Code extension.

Procedure

1. Log in to VS Code with your Red Hat account.
2. From the **Activity** bar, click the **Ansible** icon.
3. Use one of the following methods to create a role:
   - **From the Ansible panel**:
     a. From the navigation menu, click the **Ansible** icon.
     b. Click **Generate a role**.
   - **From the Command Palette**:
   - From the Command Palette of the VS Code editor, click **View > Command Palette**, and then enter **> Ansible Lightspeed: Role generation**.
     The **Create a role with Ansible Lightspeed** page is displayed on the right panel of the VS Code screen.
4. From the **Select the collection to create role in** list, choose the collection where you want to create the role. You must have a collection inside your workspace to create a role.

   If you do not have a collection, you must create it by using one of the following methods:

   - By using the following command:
     `ansible-creator init collection myns.mycollection my_directory`

     After you run the command, open the new directory with VS Code.

- By using the Ansible VS Code extension
  For more information, see the topic About content collections in the *Getting started with Ansible Automation Platform* guide.

5. In the **Describe what you want to achieve in natural language** field, enter the prompts to create a role and then click **Analyze**.

   After a few seconds, the recommended steps for your role intent are displayed in the **Review the suggested steps for your role and modify as needed** field.

6. Perform the following tasks:
   a. Review and optionally change the role name.
   b. Review the collection where the role will be created.
   c. Verify that the suggested steps match your intent and then click **Continue**.

      It takes a few seconds for the role to generate, and the newly generated role is displayed along with the list of files where the role was generated.

> **NOTE:**
> - If you want to modify the steps: Click the editor field, update the prompts or steps to suit your intent, and then click **Continue**.
> - If the role suggestions do not match your intent: Click **Back** to change the original prompt and start over.
> - If you want to restore to the original suggested steps: Click **Reset** and then click **Continue** to proceed to the next step.

7. Click **Save files**. A list of files is displayed that includes the new role.
8. Click the files to open them in the VS Code editor directly.

View the role explanations

You can request explanations for a newly created role as well as an existing Ansible role.

Before you begin

- You meet **one** of the following requirements:
  - Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
  - Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.
- You have installed and configured the Ansible VS Code extension v25.3.0 or later. For the procedure, see Installing and configuring the Ansible VS Code extension.

- You have opened the role whose explanation you want to view in the VS Code editor.

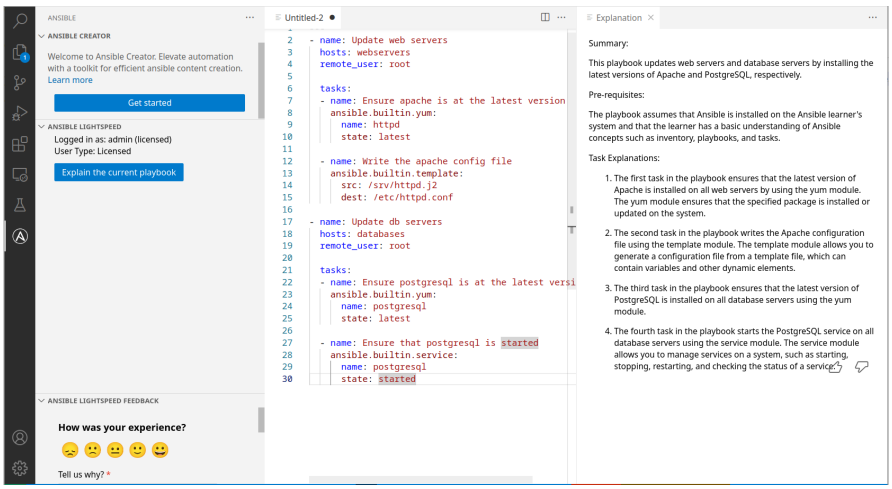
Procedure

1. Log in to VS Code with your Red Hat account.
2. Open an Ansible role YAML file within the roles directory in VS Code.
3. Use one of the following methods to view the playbook explanation:

   - **From an active role YAML file**:
     a. Place your cursor anywhere within the playbook file.
     b. Right-click and select **Explain the role with Ansible Lightspeed**.
   - **From the Ansible panel**:
     a. From the navigation menu, click the **Ansible** icon.
     b. Select **Explain the current playbook**.
   - **From the Command Palette**:
     - From the Command Palette of the VS Code editor, enter **Explain the role with Ansible Lightspeed**.
       The role explanation is displayed on the right panel of the VS Code screen.

       The following illustration shows an example of a role explanation:

Example of a role explanation



View the audit logs

Use the steps described in this section to view the audit logs.

Provide feedback on the Ansible Lightspeed service

Use the steps described in this section to provide feedback on code suggestions of the Ansible Lightspeed service.

Administer the Ansible Lightspeed Service

Organization administrators can use Red Hat Ansible Lightspeed to manage the Ansible Lightspeed service, enabling users and teams to create and use custom automation content.

This chapter provides information about how to get set up as an organization administrator on Red Hat Ansible Lightspeed, with details on how to:

- Access the Ansible Lightspeed portal as an organization administrator
- View and manage the Admin dashboard telemetry data

NOTE:
If you are using a free 90-day trial account, you need a trial or paid subscription to the Red Hat Ansible Automation Platform, but you do not need a trial or paid subscription to IBM watsonx Code Assistant. This means that you do not need to configure the API key or model ID when setting up or using a trial account.

Log in to the Ansible Lightspeed administrator portal

Use the Ansible Lightspeed administrator portal to connect Red Hat Ansible Lightspeed to IBM watsonx Code Assistant.

Before you begin

- You have organization administrator privileges to a Red Hat Customer Portal organization with a valid Red Hat Ansible Automation Platform subscription.

Procedure

1. Log in to the Ansible Lightspeed portal as an organization administrator.
2. Click **Log in > Log in with Red Hat**.

3. Enter your Red Hat account username and password. The Ansible Lightspeed Service uses Red Hat Single Sign-On (RH-SSO) for authentication.

   As part of the authentication process, the Ansible Lightspeed Service checks whether your organization has an active Ansible Automation Platform subscription. On successful authentication, the login screen is displayed along with your username and your assigned user role.

4. From the login screen, click **Admin Portal**.

Result

You are redirected to the Red Hat Ansible Lightspeed with IBM watsonx Code Assistant administrator portal where you can connect Red Hat Ansible Lightspeed to your IBM watsonx Code Assistant instance.

View and manage Admin dashboard telemetry

Red Hat Ansible Lightspeed collects the following telemetry data by default:

- **Operational telemetry data**
  This is the data that is required to operate and troubleshoot the Ansible Lightspeed service. For more information, refer the Enterprise Agreement. You cannot disable the collection of operational telemetry data.

  This includes the following data:

  - Organization you are logged into (Organization ID, account number)
  - Large language model (or models) that you are connected to

- **Admin dashboard telemetry data**
  This is the data that provides insight into how your organization users are using the Ansible Lightspeed service, and the metrics are displayed on the Admin dashboard.

  This includes the following data:

  - Prompts and content suggestions, including accept or reject of the content suggestions
  - User sentiment feedback
    You can also disable the Admin dashboard telemetry if you no longer want to collect and monitor the telemetry data.

NOTE:
Viewing telemetry data on the Admin dashboard is not yet supported on Red Hat Ansible Lightspeed on-premise deployments.

Prerequisites

To view and manage the Admin dashboard telemetry data, ensure that you have the following:

- You have organization administrator privileges to a Red Hat Customer Portal organization with a valid Red Hat Ansible Automation Platform subscription.
- You have installed the Ansible VS Code extension v2.13.148 that is required to collect Admin dashboard telemetry.

IMPORTANT: Red Hat Ansible Lightspeed does not collect users' personal information, such as usernames or passwords. If any personal information is inadvertently received, the data is deleted. For more information about Red Hat Ansible Lightspeed's privacy practices, see the Telemetry Data Collection Notice for the Admin dashboard.

What telemetry data is collected?

Following is the list of telemetry data that Red Hat Ansible Lightspeed collects:

- Details of the organization that you are logged into, such as organization ID and account number
- Large language models that you are connected to
- Inline suggestions that were accepted, rejected, or ignored by your organization users
- User sentiment feedback
- Top 10 modules returned in code recommendations

Related information
Telemetry Data Collection Notice for the Admin dashboard

View the Admin dashboard telemetry

The Admin dashboard displays the analytics telemetry data that you can use to gain insight into how your organization users are using the Ansible Lightspeed service.

The Admin dashboard displays the following charts:

- **Inline suggestions accepted, rejected, or ignored by users**
  This graph tracks the number of inline suggestions that were accepted, rejected, or ignored by users in your organization. Use this graph to gain insight into how your organization users are using the Ansible Lightspeed service.

- **User sentiment**
  This graph measures the users' feedback (feelings, opinions). Use this graph to gain insight into the overall user experience with Red Hat Ansible Lightspeed.
- **Top 10 modules returned in code recommendations**
  This graph displays the top 10 modules returned in code recommendations. Use this metric to determine which modules are being suggested the most to your organization's automation developers.

Procedure

1. Log in to the Ansible Lightspeed with IBM watsonx Code Assistant Hybrid Cloud Console as an organization administrator.
2. From the navigation panel, select **Ansible Lightspeed > Admin Dashboard**.

   The Admin dashboard displays a graphical representation of analytics telemetry data for the last 30 days by default.
3. Use the following filters to refine your telemetry data:

   - To view the telemetry data for a specific time period or for a custom date range, select the date range from the **Quick Date Range** list.
   - To view the telemetry data for a specific IBM watsonx Code Assistant model only, select the model ID from the **Model Name** list. By default, the Admin dashboard displays telemetry data for all models.

Disable the Admin dashboard telemetry

Red Hat Ansible Lightspeed collects Admin dashboard telemetry by default to gain insight into how your organization uses the service. You can disable this telemetry to stop collecting analytics data.

Before you begin

- You have organization administrator privileges to a Red Hat Customer Portal organization with a valid Red Hat Ansible Automation Platform subscription.

After you disable the Admin dashboard telemetry, the Ansible Lightspeed service no longer collects the analytics telemetry data for your organization. The earlier telemetry data is still available on the Admin dashboard, but no latest data is displayed. If you re-enable the Admin dashboard telemetry, the Ansible Lightspeed service starts collecting data for your organization, and the metrics are displayed on the Admin dashboard after 24 hours.

Procedure

1. Log in to the Ansible Lightspeed portal as an organization administrator.
2. From the login screen, click **Admin Portal**.
3. Under Admin Portal, click **Telemetry**.

4. To disable the Admin dashboard telemetry, select **Operational telemetry data only**.

> **NOTE:**
> To re-enable the Admin dashboard telemetry, select **Admin dashboard telemetry data**.

5. Click **Save**.

Install and configure the Ansible code bot

Ansible code bot scans GitHub repositories (collections, roles, playbooks) and proactively creates pull requests with best practice or quality improvement recommendations.

> **IMPORTANT:**
> The Ansible code bot was deprecated on October 1, 2025 and will be retired anytime after December 31, 2025. Red Hat is no longer actively maintaining or supporting the component.

Ansible code bot scans your code repositories to recommend code quality improvements. It promotes Ansible best practices while avoiding common errors that can lead to bugs or make code harder to maintain. The bot automatically submits pull requests to the repository, which proactively alerts the repository owner to a recommended change to their content. You can configure Ansible code bot to scan your existing Git repositories (both public and private). Your organization must have an active subscription to Red Hat Ansible Automation Platform to use the Ansible code bot. However, IBM watsonx Code Assistant is not required to use the Ansible code bot.

After the Ansible code bot is installed, it automatically scans the selected repositories that are in Jinja format. Once the scanning is complete, the code bot generates an initial PR for each repository; the initial PR also contains the scan schedule configured to run weekly. You must review the initial PR for the suggested changes and merge the PR. Once the initial PR is merged, the scan schedule is triggered, and the subsequent repository scans are performed weekly. If required, you can change the scan schedule to a daily or monthly cadence.

You can access the Ansible code bot dashboard that displays all your repositories that have the bot installed along with their scan status. From the dashboard, you can start a manual scan, view the scan history, and view the repository. From GitHub, you can configure a schedule to scan your repository at regular intervals, and add or remove a repository from being scanned.

IMPORTANT:
Ansible code bot is supported on the following GitHub versions:

- GitHub.com
- GitHub Enterprise Cloud
  Ansible code bot is not supported on GitHub Enterprise Server.

The following examples are code recommendations that the Ansible code bot can suggest:

- Available alternatives for deprecated legacy syntax or implementation patterns
- Module version changes and updates, such as:
  - Adding any new required parameters
  - Flagging deprecated parameters
  - Removing unused parameters
- Applying YAML best practices
- Adding comment blocks
- Fixing casing issues in name fields

Related information
GitHub's plans

Install the Ansible code bot

Install the Ansible code bot to get code recommendations for your repositories, and then log in to the Ansible code bot dashboard to monitor and manage your repository scans.

Procedure

1. Log in to GitHub by using the account associated with your organization.
2. Go to the Ansible code bot GitHub app.
3. Select the Ansible repositories that you want the app to access:
   - **All repositories**: Provides access to read the metadata of all repositories.
   - **Only select repositories**: Provides access to read the metadata of only the repositories that you select.
4. Optional: If you selected **Only select repositories** in the previous step, select the repositories that you want the Ansible code bot to access from the **Select repositories** list.

5. Click **Install & Authorize**. A message is displayed that specifies the following permissions are granted automatically to the bot during installation:
    - Read access to metadata
    - Read and write access to code and pull requests
6. When prompted, log in to your Red Hat Single Sign-On account as an organization administrator.
7. Log in to the Ansible code bot dashboard:
    a. On the **Authorize Ansible code bot** page, verify your account and repository permissions.
    b. Click **Authorize Ansible**.

    From the **Authorize Ansible code bot** page, the following actions occur:

    - Ansible code bot verifies that you are a part of an organization that has an active subscription to Red Hat Ansible Automation Platform.
    - GitHub requests read permissions to access the repositories associated with your account.
      On successful authorization, you are logged in to Ansible code bot dashboard. The dashboard displays all your repositories that have the Ansible code bot installed along with their scan status.

Result

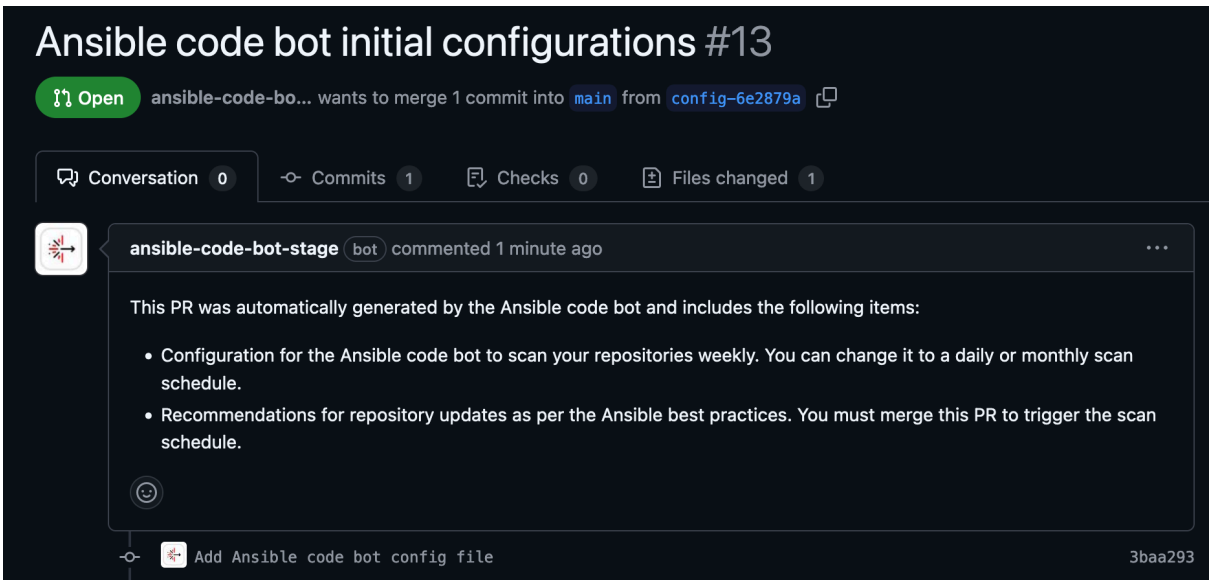
After the Ansible code bot is installed, it automatically scans the selected repositories that are in Jinja format. When the scanning is complete, the code bot generates an initial PR for each repository; the initial PR also contains the scan schedule configured to run weekly.

Perform the following tasks:

1. Review the initial PR for the suggested changes, and merge the PR.
   After you merge the initial PR, the configured scan schedule is triggered, and the subsequent repository scans are performed weekly.

> **NOTE:**
> If you do not merge the initial PR, the weekly scan schedule is not triggered and the Ansible code bot dashboard displays the repositories without any associated scan history.

The following illustration is an example of an initial PR being created:



2. Optional: If required, you can manually scan your repositories or change the scan schedule to a daily or monthly cadence.
3. Modify the scanned repositories.

Related information
Troubleshooting Ansible code bot errors

Uninstall the Ansible code bot

If you no longer want to use the Ansible code bot, you can uninstall it from GitHub. After the code bot is uninstalled, you can still access the Ansible code bot dashboard but you cannot see the repositories on the dashboard or scan your repositories.

Procedure

1. Log in to GitHub by using the account associated with your organization.
2. In GitHub, click your profile photo > **Settings**.
3. Under Integrations, click **Applications** > **Installed GitHub Apps**.
4. Click **Configure** beside the Ansible code bot app.
5. Under the **Danger zone** area, click **Uninstall**.

   The Ansible code bot app is uninstalled from your GitHub account.

Manage repository scans

The Ansible code bot dashboard displays a list of your repositories where the code bot is installed, and indicates if the scan schedule is not set, or is set to manual or scheduled scan.

You can scan your Git repository by starting a manual scan, or configure a schedule to scan your repository at regular intervals. After the scan is completed, you can view the scan history. The scan history shows the start time, status, and type of scan. It also includes a link to the pull request if it was created, and the log message if the scan failed. You can also add new repositories for scanning or remove existing repositories from being scanned.

Manually scan the repository from GitHub

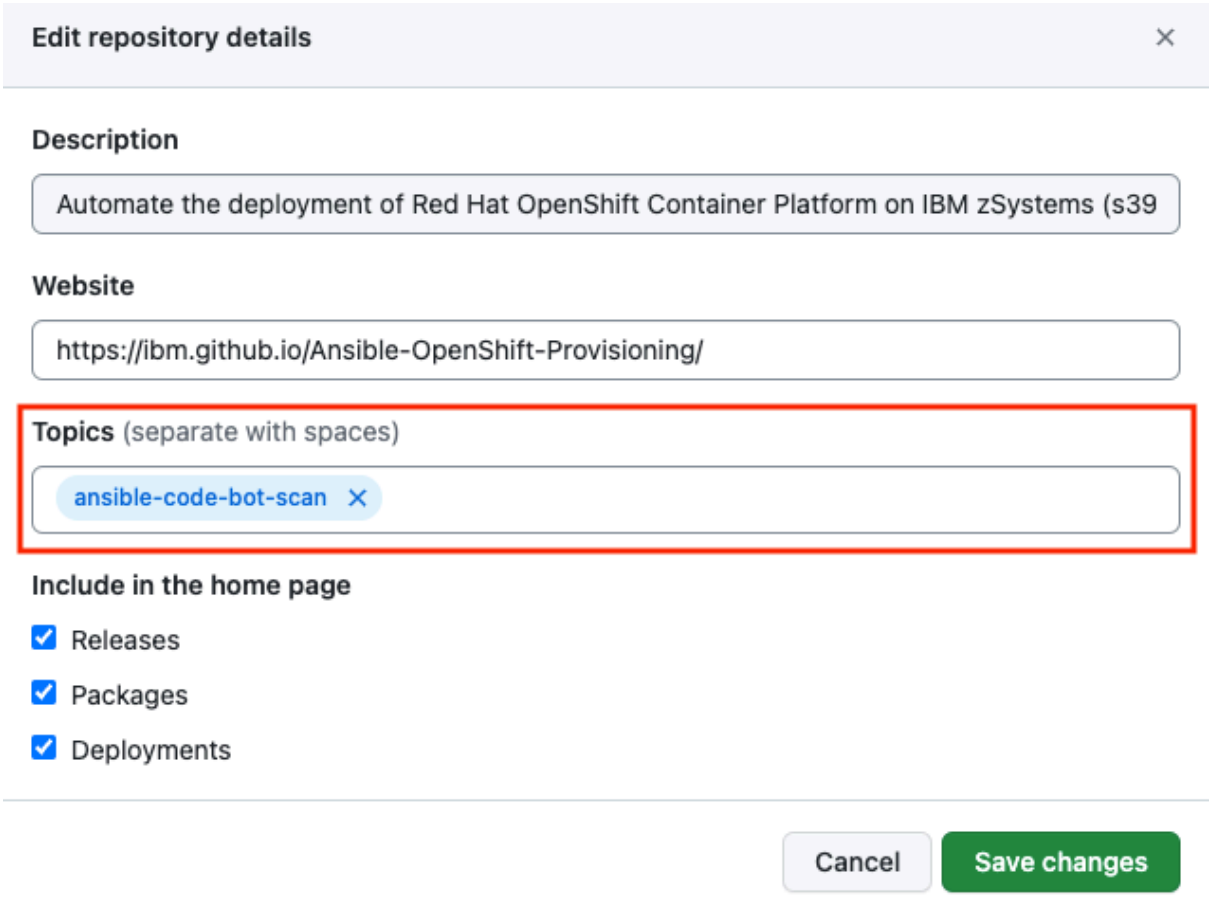
You can manually scan your Git repositories from GitHub if you did not set up a scanning schedule for your Ansible code bot or if you do not want to wait for the next scheduled scan.

If you manually scan your repository, and no pull request was created, it is likely so because a duplicate pull request already exists.

Procedure

1. In GitHub, go to the main page of the repository that you want to scan.
2. To modify the repository settings, click the **Settings** icon beside the **About** area.
3. In the **Topics** field, enter the keyword topic **ansible-code-bot-scan** to the repository.

   The following illustration shows the keyword topic for starting a manual scan:



4. Click **Save changes**.

   Based on the repository webhook event, Ansible code bot starts a manual scan of your repository. If the avoid duplicate pull requests condition is not met, then the manual scan results in a new pull request with all the necessary Ansible code bot recommendations.

Related information
Troubleshooting Ansible code bot errors

Manually scan the repository from the Ansible code bot dashboard

You can manually scan your Git repositories if you did not set up a scanning schedule for your Ansible code bot or if you do not want to wait for the next scheduled scan.

If you manually scan your repository, and no pull request was created, it is likely so because a duplicate pull request already exists.

Procedure

1. Log in to the Ansible code bot dashboard.

   The **Repositories** list displays a list of repositories that you selected for scanning.

NOTE:
If you do not see your repository in the **Repositories** list, you can add it for scanning.

2. To start a manual scan of your repository, click the **Ellipsis** icon ( ) beside the repository that you want to scan and select **Scan now**.
3. Click **Refresh** to view the status of the scan job.
4. To view more details about your repository scans, click the **Ellipsis** icon ( ) beside the repository and select **View scan history**.

   The repository's scan history is displayed along with the scan start time, scan status, type of scan (scheduled or manual), link to the pull request if it was created, and the log message if the scan failed.
5. To view your repository on GitHub, click the **Ellipsis** icon ( ) beside the repository and select **View repository**.

Configure the Ansible code bot to scan your repository at regular intervals

After installing the Ansible code bot, it automatically scans the selected repositories that are in Jinja format. Once the scanning is complete, the code bot generates an initial PR for each repository; the initial PR also contains the scan schedule configured to run weekly.

You must review the initial PR for the suggested changes and merge the PR. Once the initial PR is merged, the scan schedule is triggered, and the subsequent repository scans are performed weekly. If required, you can change the scan schedule to a daily or monthly cadence.

NOTE:
If you do not merge the initial PR, the weekly scan schedule is not triggered and the Ansible code bot dashboard displays the repositories without any associated scan history. In such a scenario, you must manually create a configuration file `ansible-code-bot.yml` and specify your scan schedule within the file.

How Ansible code bot handles duplicate pull requests

This section outlines the conditional logic used by the Ansible code bot to manage repository scanning and pull request generation.

- If Ansible code bot has created a pull request on the latest commit default branch, it does not scan the repository. The bot skips scanning the repository because the pull request was committed on the latest default branch, and no new commit was made after that pull request.
- If there is an existing pull request that is not on the latest commit default branch, the Ansible code bot does a pull request difference to compare the changes in both branches. The following scenarios are possible:
    - **There is no difference in the existing and new scan results**: Ansible code bot does not push the scan results as a new pull request.
    - **There are differences found in the existing and the new scan results**: the Ansible code bot creates a new pull request. The newly-created pull request does not close the existing pull request, against which the pull request difference was noted. This behavior makes it easier for the repository administrator to review only the latest pull request created by the Ansible code bot, and the administrator can avoid reviewing the older pull requests created by the bot. If required, the administrator can close the older pull requests.

Troubleshoot Red Hat Ansible Lightspeed configuration errors

This section provides information about errors when configuring the Red Hat Ansible Lightspeed and their workarounds.

Cannot access the Ansible Lightspeed administrator portal

The Ansible Lightspeed administrator portal can be accessed by the Red Hat organization administrator only.

If you are the Red Hat organization administrator, before you access the Ansible Lightspeed administrator portal, ensure that:

- You have a valid Ansible Automation Platform subscription.

Cannot save the API key

When you enter the IBM watsonx Code Assistant API key, authentication fails and shows the following error message:

```
IBM Cloud API key is invalid
```

Red Hat Ansible Lightspeed verifies the API key by generating an associated access token. To resolve the error, ensure that you have not accidentally included any extra spaces when obtaining the API key from IBM watsonx Code Assistant. If you still cannot upload the API key, contact IBM Support.

Cannot configure the model ID due to authentication failure

When you enter the model ID in the Red Hat Ansible Lightspeed administrator portal, the authentication fails.

To resolve the error, ensure that:

- You have configured a valid API key before you upload the model ID.
- You have not accidentally included any extra spaces when entering the model ID.

Cannot configure the model ID due to inference failure

While validating the model ID, Red Hat Ansible Lightspeed performs a test inference. If Red Hat Ansible Lightspeed detects an error, the validation fails and an `Inference failed` message is displayed.

To resolve the error, ensure that:

- You have a valid API key and model ID.
- You have not accidentally included any extra spaces when obtaining the API key and model ID from IBM watsonx Code Assistant.

Related information

IBM Support

Troubleshoot Red Hat Ansible Lightspeed on-premise deployment errors

This section provides information about errors when configuring the Red Hat Ansible Lightspeed on-premise deployment and their workarounds.

Cannot log out of the Ansible Lightspeed portal

After you log out from the Ansible Lightspeed portal, you are redirected to the automation controller API page instead of Ansible Lightspeed.

This error indicates that the logout redirect URI was not configured while setting up your Red Hat Ansible Lightspeed on-premise deployment. You must configure the logout redirect URI by adding the **LOGOUT_ALLOWED_HOSTS** entry to the YAML file.

Cannot connect to the Ansible Lightspeed service from the Ansible VS Code extension

The following scenarios are possible:

- The log-in attempt fails with the following error message:
  `Enable lightspeed services from settings to use the feature.`

  This error indicates that Ansible Lightspeed is not enabled in the Ansible VS Code extension. To resolve this error, perform the following tasks:

  a. Open the VS Code application.

  b. From the **Activity** bar, click the **Extensions** icon.

  c. From the **Installed Extensions** list, select **Ansible**.

  d. From the **Ansible** extension page, click the **Settings** icon ( ) and select **Extension Settings**.

  e. Select **Ansible Lightspeed** settings and then select the **Enable Ansible Lightspeed** checkbox.

- You are redirected to an incorrect Ansible Lightspeed instance on clicking the **Connect** button.
  This error indicates an incorrect route URL was used while configuring the Ansible Lightspeed service in Ansible VS Code extension. Ensure that you have configured the

correct value in the route URL without any suffix. For more information, see Configuring Ansible VS Code extension for Red Hat Ansible Lightspeed on-premise deployment.

- Cannot request code recommendations
  The following error message is displayed:

  `An error occurred attempting to complete your request. Please try again later.`

  This error indicates that the Ansible Lightspeed service is not running or is running with issues. Please check the Lightspeed service logs (the pod with suffix `-api`) for more details and error codes.

- Cannot request code recommendations
  The following error message is displayed:

  `The IBM watsonx Code Assistant is unavailable. Please try again later.`

  or:

  `IBM watsonx Code Assistant Model ID is invalid. Please contact your administrator.`

  This error indicates that the model secret contains incorrect values. To resolve this error, ensure that you have not accidentally added any whitespace characters (extra line, space, and so on) to the `username`, `model_url`, and `model_api_key` parameters in the model connection secret. For more information, see Creating connection secrets.

Cannot connect to the Ansible Lightspeed service due to SSL connection error

If you are using self-signed certificates on the model server, you might encounter SSL certification verification errors, causing the connection between Ansible Lightspeed service and the model server to fail. The following error message is displayed:

```
Caused by SSLError(SSLCertVerificationError(1, '[SSL: CERTIFICATE_VERIFY_FAILED]

certificate verify failed: self signed certificate in certificate chain
(_ssl.c:1006)'))
```

To resolve this error, use one of the following workarounds based on your Ansible Automation Platform version:

- For Red Hat Ansible Automation Platform 2.5 and later:

Specify the optional key/value pair as `model_verify_ssl=true` in the model secret to connect to an IBM watsonx Code Assistant model. For details about the procedure, see Creating connection secrets.

- For Red Hat Ansible Automation Platform 2.4:

You can disable the SSL protection between the model server and the Ansible Lightspeed service. For example, you can disable the SSL protection when you are on a testing environment. To disable the SSL protection, you must add the following extra setting in the Red Hat Ansible Lightspeed Custom Resource Definition (CRD) YAML file under the `spec:` section:

```
extra_settings:
    - setting: ANSIBLE_AI_MODEL_MESH_API_VERIFY_SSL
      value: false
```

IMPORTANT:
Reenabling the SSL protection You must re-enable the SSL protection when deploying on a production environment. To re-enable the SSL protection, simply remove the extra setting from the YAML file.

To reenable the SSL protection, perform the following tasks: . Go to the Red Hat OpenShift Container Platform. . Select **Operators › Installed Operators**. . From the **Projects** list, select the namespace that you created when you installed the Red Hat Ansible Automation Platform operator. . Locate and select the **Ansible Automation Platform** operator. . Locate and select the Ansible Automation Platform custom resource, and then click the required app. . Select the **YAML** tab. The editor switches to a YAML editor view. . Scroll and find the spec: section, and add the following parameter under the `spec:` section:

```
extra_settings:
    - setting: ANSIBLE_AI_MODEL_MESH_API_VERIFY_SSL
      value: false
```

1. Click **Save**.
2. Restart the automation controller pods to apply the revised YAML:
    a. From the Red Hat OpenShift Container Platform, select **Workloads › Pods**.
    b. Locate and select the Ansible Lightspeed pod that you updated.
    c. Click the **Edit** icon beside the pod and select **Delete Pod**.
    The select pod gets deleted and a new pod gets created.

Related information
Configuring Ansible VS Code extension for Red Hat Ansible Lightspeed on-premise deployment
Creating connection secrets
Creating connection secrets

Troubleshoot Ansible Visual Studio Code extension errors

Use this topic to troubleshoot issues with Ansible VS Code.

Cannot view the generated code recommendations using the Ansible VS Code extension

The following scenarios are possible:

- You receive a `403 error` message.
  To resolve this error, ensure that:
    - Your organization administrator has configured Red Hat Ansible Lightspeed for your organization.
    - You meet **one** of the following requirements:
        - Your organization has a trial or paid subscription to both the Red Hat Ansible Automation Platform and IBM watsonx Code Assistant.
        - Your organization has a trial or paid subscription to the Red Hat Ansible Automation Platform, and you have a Red Hat Ansible Lightspeed trial account.
- You have not configured the required Ansible VS code extension settings.
    - To resolve this error, ensure that you have enabled the **Lightspeed:Enabled** and **Lightspeed › Suggestions:Enabled** settings.
- You receive a `Failure on completion requests` error when you make inference requests in VS Code.
  This error occurs because your organization has a trial or paid subscription to both Ansible Automation Platform and IBM watsonx Code Assistant. However, your organization administrator has not configured an IBM watsonx Code Assistant model for your organization.
- You receive an `Ansible Lightspeed encountered an error. Try again after some time.` error message when you make single-task or multitask requests.
  This error occurs when you use a remote SSH extension with VS Code to request single or multitask recommendations in playbooks. However, the task recommendations are generated when using a role. This error occurs in workspaces that contain a large number of roles.
- Your VS Code Workspace settings override user settings.
  If your Workspace settings are configured, they can override our user settings even if you have configured the Ansible VS Code extension correctly. The Workspace settings can

disable your VS Code extension settings, and therefore you cannot access the Ansible Lightspeed service.

To resolve this error, ensure that there are no Workspace settings configured in VS Code.

- You entered a multitask prompt, but code recommendations were not generated.
  To resolve this error, log out of VS Code and log in again using your Red Hat account.
- You clicked a different location or switched to a different window; therefore, the populated code recommendations disappeared.
  The Red Hat Ansible Lightspeed service could take multiple seconds per task to populate the code recommendations. If you are using a multitask prompt, the Red Hat Ansible Lightspeed service takes a bit longer to populate the results. Do not move your cursor or press any key while the code recommendation is being generated. If you change the cursor location or press any key, the Ansible VS Code extension cancels the request and the Red Hat Ansible Lightspeed service does not process your request. In this scenario, you must get the cursor back to its original position and repopulate the results.

Cannot request code recommendations by using the Ansible VS Code extension

The following error message is displayed: `Your trial to the generative AI model has expired. Refer to your IBM Cloud Account to re-enable access to the IBM watsonx Code Assistant.`

To resolve this error, refer to your IBM Cloud account and select an upgrade option.

Cannot connect to Ansible VS code extension when using a proxy configuration or a self-signed certificate or both

You might encounter errors when connecting with to the Ansible VS code extension through a proxy. Connections to the outbound domain `https://c.ai.ansible.redhat.com` fail and a network error message is displayed.

To resolve this issue, you must add the URL `https://c.ai.ansible.redhat.com/` in VS Code proxy settings. If you are using Red Hat Single Sign-On (RH-SSO) to authenticate users, then you must also grant access to `https://sso.redhat.com` in VS code proxy settings.

To modify the proxy settings in VS code, perform the following tasks:

1. Open Visual Studio Code.
2. Navigate to **File > Preferences > Settings**.
3. Select **Application > Proxy** in the sidebar.
4. In the **Http: Proxy** field, add the following URLs to the proxy:
   - `https://c.ai.ansible.redhat.com/`

- `https://sso.redhat.com`, if you are using RH-SSO to authenticate users.

5. In the **http: Proxy Support** drop down list, select **Override**.
6. Search for and select the following configuration keys:
   - **Electron Fetch**
   - **System Certificates V2** if you are using your own Certificate Authority (CA).

For information about how to set up proxy support in VS Code, see the Related Links section below.

Cannot connect to Ansible VS code extension due to network issues

If you encounter network issues, use the **Network Proxy Test** extension to test the connection:

1. Install the VS code extension **Network Proxy Test**.
2. Use the **Network Proxy Test: Test Connection** action to target the server and the endpoint using the parameter `/check/status end-point`.
   For example:

   `https://c.ai.ansible.redhat.com/check/status/` to test the connection to Red Hat Ansible Lightspeed cloud service.

Related information
Configure the Ansible VS Code extension
Workspace settings
Proxy server support
Proxy settings and fallback
Network Proxy Test

Troubleshoot Ansible code bot errors

This section provides information about errors when using the Ansible code bot and their workarounds.

Cannot access Ansible code bot

After you install Ansible code bot and attempt to log in, you receive the following error message:

```
Your organization does not have a valid Red Hat Ansible Lightspeed subscription
```

After you install Ansible code bot, you are redirected to a page that shows an active subscription status, as shown in the following image:

Figure: Ansible code bot login screen with an active subscription

If the login screen displays an inactive subscription status, Ansible code bot does not scan your Git repositories. The error occurs because your organization does not have a valid Ansible Automation Platform subscription. To resolve this error, ensure that you are part of an organization that has a valid Red Hat Ansible Automation Platform subscription.

Cannot scan your Git repository using Ansible code bot

If the Ansible code bot is not configured correctly, it does not scan your Git repositories or does not create pull requests.

To resolve Ansible code bot errors, ensure that:

- You have selected all the Git repositories that you want to scan.
- You have a `.yml` configuration file named `ansible-code-bot.yml` in your repository `.github` folder. For example, `.github/ansible-code-bot.yml`.

Run a manual scan on your git repositories by adding the **ansible-code-bot-scan** topic to your repository. For more information, see Manually scan your Git repositories.

If the Ansible code bot still cannot scan your Git repository, the following scenarios are possible:

- The Ansible code bot did not identify any ansible-lint violations in the Git repository.
- The Ansible code bot does not have permission to scan the Git repository.
- Your organization does not have a valid Red Hat Ansible Automation Platform subscription.

Cannot create pull requests

You might encounter an error where the Ansible code bot cannot create pull requests after scanning your Git repositories.

To resolve this error, ensure that:

- You have verified that there are are no duplicate pull requests. For more information, see How Ansible code bot handles duplicate pull requests.
- You have deleted the branches after closing the pull requests created by the Ansible code bot. For more information, see Deleting a branch used for a pull request.

Related information
Manually scan your Git repositories
How Ansible code bot handles duplicate pull requests
Deleting a branch used for a pull request

Trigger automation with webhooks

Use webhooks to run specified commands between applications over the web. Automation controller currently provides webhook integration with GitHub and GitLab.

Set up a webhook using GitHub or GitLab, then view the payload output.

The webhook post-status-back functionality for GitHub and GitLab is designed to work only under certain CI events. Receiving another kind of event results in messages such as the following in the service log:

```
awx.main.models.mixins Webhook event did not have a status API endpoint associated, skipping.
```

Related information
Set up a GitHub webhook to trigger automation jobs
Set up a GitLab webhook
View the payload output

Set up a GitHub webhook to trigger automation jobs

Automation controller has the ability to run jobs based on a triggered webhook event coming in.

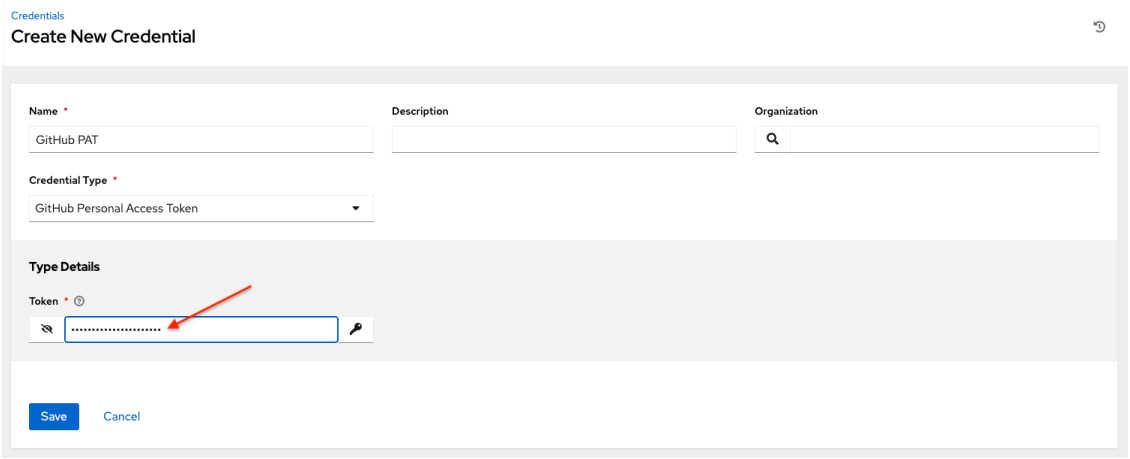
Job status information (pending, error, success) can be sent back only for pull request events. If you do not need automation controller to post job statuses back to the webhook service, go directly to step 3.

Procedure

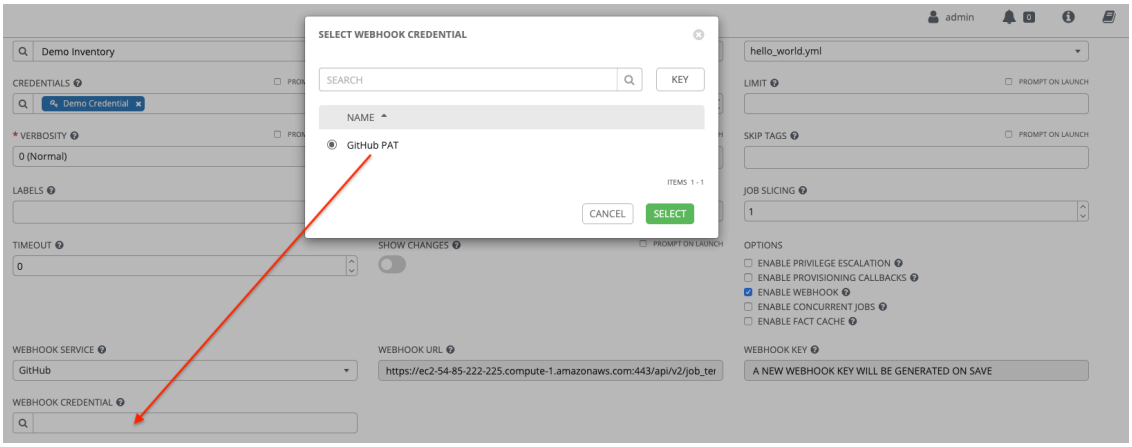
1. Generate a *Personal Access Token* (PAT) for use with automation controller:
    a. In the profile settings of your GitHub account, select **Settings**.
    b. From the navigation panel, select **<> Developer Settings**.
    c. On the **Developer Settings** page, select **Personal access tokens**.
    d. Select **Tokens (classic)**
    e. From the **Personal access tokens** screen, click **Generate a personal access token**.
    f. When prompted, enter your GitHub account password to continue.
    g. In the **Note** field, enter a brief description about what this PAT is used for.
    h. In the **Select scopes** fields, check the boxes next to **repo:status**, **repo_deployment**, and **public_repo**. The automation webhook only needs repository scope access, with the exception of invites. For more information, see Scopes for OAuth apps documentation.
    i. Click **Generate token**.

> **IMPORTANT:**
> When generating the token, ensure that you copy the PAT, as you need it in step 2. You cannot access this token again in GitHub.

2. Use the PAT to optionally create a GitHub credential:
    a. Go to your instance and create a new credential for the GitHub PAT, using the generated token.
    b. Make note of the name of this credential, as you use it in the job template that posts back to GitHub.



    c. Go to the job template with which you want to enable webhooks, and select the webhook service and credential you created in the preceding step.



d. Click **Save**. Your job template is set up to post back to GitHub.

3. Go to a GitHub repository where you want to configure webhooks and select **Settings**.
4. From the navigation panel, select **Webhooks › Add webhook**.
5. To complete the **Add webhook** page, you must check the **Enable Webhook** option in a job template or workflow job template. For more information, see step 3 in both Creating a job template and Creating a workflow job template.
6. Complete the following fields:

   - **Payload URL**: Copy the contents of the **Webhook URL** from the job template and paste it here. The results are sent to this address from GitHub.
   - **Content type**: Set it to **application/json**.
   - **Secret**: Copy the contents of the **Webhook Key** from the job template and paste it here.
   - **Which events would you like to trigger this webhook?**: Select the types of events you want to trigger a webhook. Any such event will trigger the job or workflow. To have the job status (pending, error, success) sent back to GitHub, you must select **Pull requests** in the **Let me select individual events** section.

- **Active**: Leave this checked.

7. Click **Add webhook**.
8. When your webhook is configured, it is displayed in the list of webhooks active for your repository, along with the ability to edit or delete it. Click a webhook, to go to the **Manage webhook** screen.
9. Scroll to view the delivery attempts made to your webhook and whether they succeeded or failed.

Related information
Webhooks documentation

Set up a GitLab webhook

Automation controller has the ability to run jobs based on a triggered webhook event coming in. Job status information (pending, error, success) can be sent back only for pull request events.

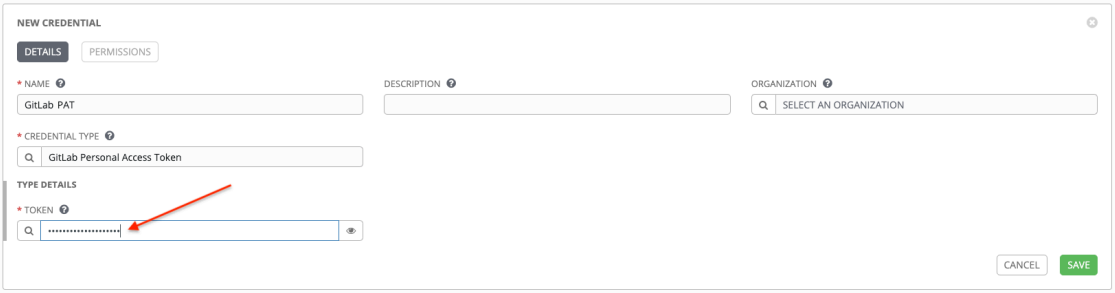
If automation controller is not required to post job statuses back to the webhook service, go directly to step 3.

Procedure

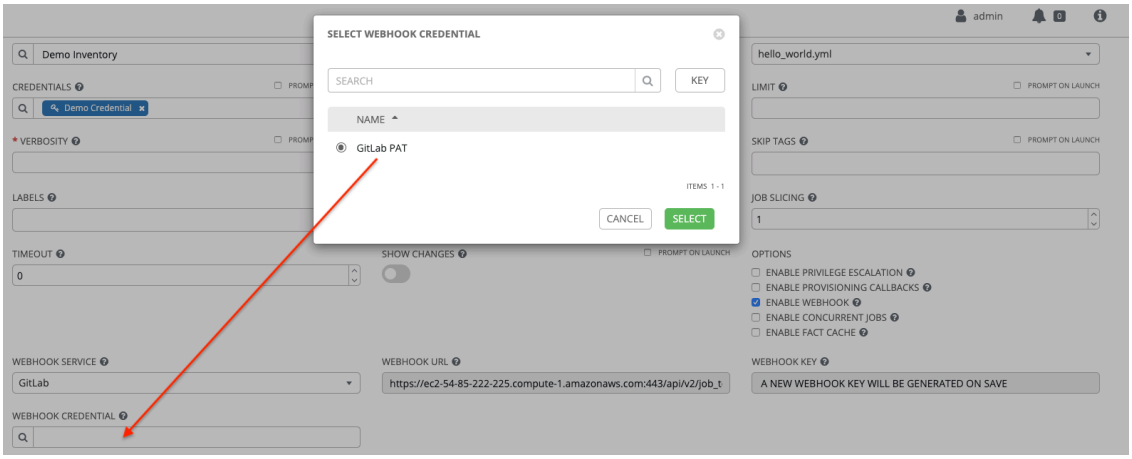
1. Generate a *Personal Access Token* (PAT) for use with automation controller:
   a. From the navigation panel in GitLab, select your avatar and **Edit profile**.
   b. From the navigation panel, select **Access tokens**.
   c. Complete the following fields:
      - **Token name**: Enter a brief description about what this PAT is used for.
      - **Expiration date**: Skip this field unless you want to set an expiration date for your webhook.
      - **Select scopes**: Select those that are applicable to your integration. For automation controller, **api** is the only selection necessary.
   d. Click **Create personal access token**.

> **IMPORTANT:**
> When the token is generated, ensure that you copy the PAT, as you need it in step 2. You cannot access this token again in GitLab.

2. Use the PAT to optionally create a GitLab credential:
   a. Go to your instance, and create a new credential for the GitLab PAT, using the generated token.
   b. Make note of the name of this credential, as you use it in the job template that posts back to GitLab.



   c. Go to the job template with which you want to enable webhooks, and select the webhook service and credential you created in the preceding step.



d. Click **Save**. Your job template is set up to post back to GitLab.

3. Go to a GitLab repository where you want to configure webhooks.
4. From the navigation panel, select **Settings › Integrations**.
5. To complete the **Add webhook** page, you must check the **Enable Webhook** option in a job template or workflow job template. For more information, see step 3 in both Creating a job template and Creating a workflow job template.
6. Complete the following fields:

   - **URL**: Copy the contents of the **Webhook URL** from the job template and paste it here. The results are sent to this address from GitLab.
   - **Secret Token**: Copy the contents of the **Webhook Key** from the job template and paste it here.
   - **Trigger**: Select the types of events you want to trigger a webhook. Any such event will trigger the job or workflow. To have job status (pending, error, success) sent back to GitLab, you must select **Merge request events** in the **Trigger** section.
   - **SSL verification**: Leave **Enable SSL verification** selected.

7. Click **Add webhook**.
8. When your webhook is configured, it is displayed in the list **Project Webhooks** for your repository, along with the ability to test events, edit or delete the webhook. Testing a webhook event displays the results on each page whether it succeeded or failed.

Related information
Webhooks

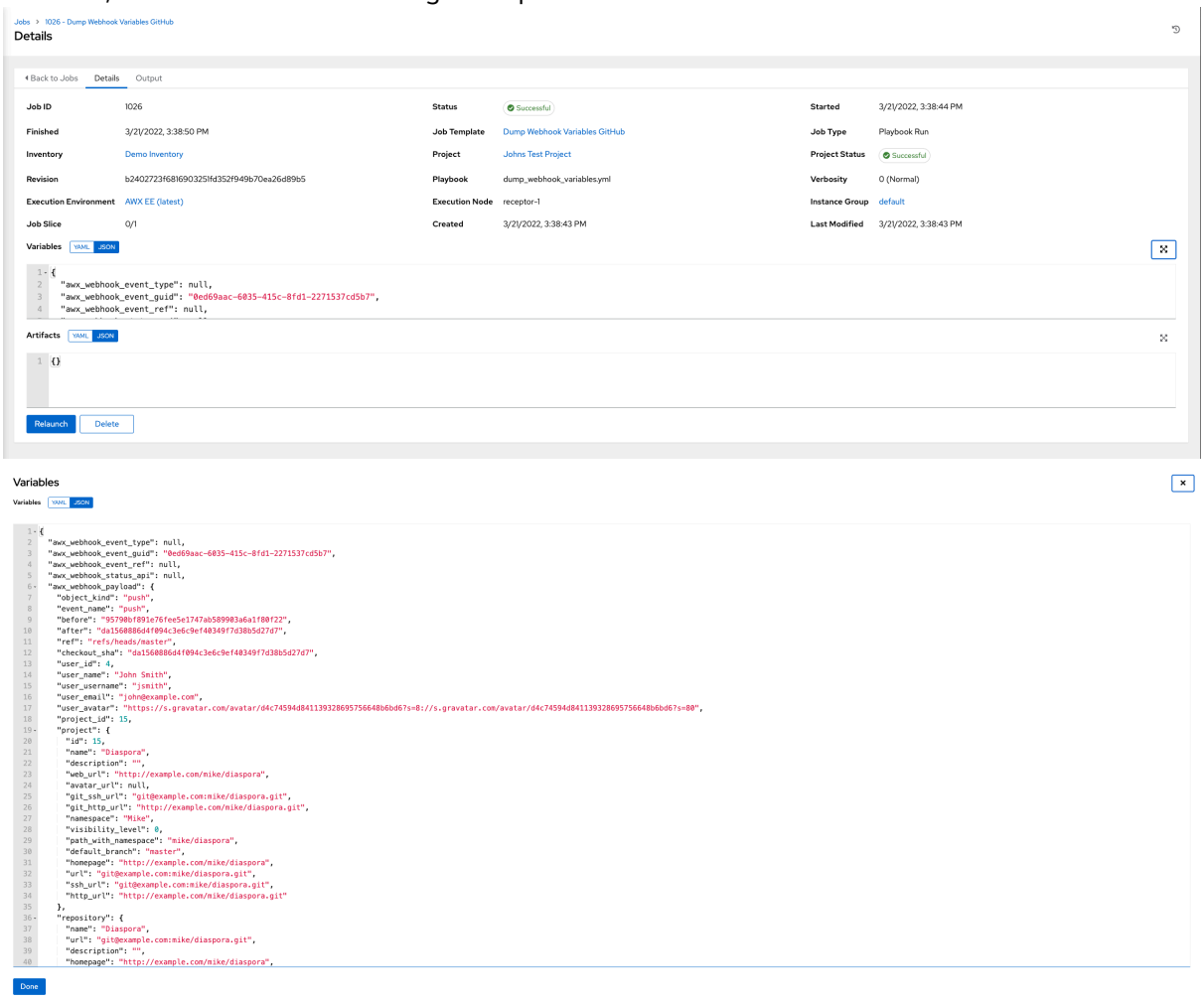
View the payload output

When a webhook is triggered in automation controller, the payload data is sent to the specified URL.

You can view the entire payload exposed as an extra variable.

Procedure

1. From the navigation panel, select **Automation Execution > Jobs**.
2. Select the job template with the webhook enabled.
3. Select the **Details** tab.
4. In the **Extra Variables** field, view the payload output from the `awx_webhook_payload` variable, as shown in the following example:



Best practices for automation execution

Adopt these best practices to help ensure scalable, maintainable, and highly efficient automation. This guidance focuses on using source control, dynamic inventories, and robust file structures within automation controller.

Use source control

Automation controller supports playbooks stored directly on the server. Therefore, you must store your playbooks, roles, and any associated details in source control.

This way you have an audit trail describing when and why you changed the rules that are automating your infrastructure. Additionally, it permits sharing of playbooks with other parts of your infrastructure or team.

Ansible file and directory structure

Follow recommended directory structures for Ansible projects to organize playbooks, inventories, and variables. A consistent layout improves maintainability and scaling.

To ensure reliable and consistent automation, follow these best practices for managing your content:

- Package reusable content, such as roles, modules, and plugins into Ansible Collections.
- Reference all necessary Collections for a project in the project's `requirements.yml` file. These dependencies are automatically installed into the execution environment (EE) at runtime, but only if they are not already present in the EE image.
- Do not import content from other projects or common file-system locations, such as `/opt`, at runtime. All content must be explicitly defined within the EE.
- Working directory: The playbook directory is used as the current working directory at runtime. However, always use the `playbook_dir` variable instead of relying on the current working directory path.

WARNING:
Automation controller does not support interactive features.

- Avoid using the `vars_prompt` feature, as automation controller does not permit interactive questions. For user input, use Surveys in job templates.
- Do not use the `pause` feature without a timeout. Automation controller does not permit canceling a pause interactively. If `pause` is necessary, you must set a timeout.

Use dynamic inventory sources

Define an inventory synchronization process by using dynamic sources, such as a cloud provider or CMDB, to help ensure your infrastructure inventory in automation controller stays consistently up-to-date.

NOTE:
Edits and additions to Inventory host variables persist beyond an inventory synchronization as long as `--overwrite_vars` is **not** set.

Variable management for inventory

Variables associated with hosts and groups in an inventory can be managed in several ways in automation controller.

Keep variable data with the hosts and groups definitions (see the inventory editor), rather than using `group_vars/` and `host_vars/`. If you use dynamic inventory sources, automation controller can synchronize such variables with the database while the **Overwrite Variables** option is not set.

Autoscale

Use the "callback" feature to permit newly booting instances to request configuration for auto-scaling scenarios or provisioning integration.

Larger host counts

Set "forks" on a job template to larger values to increase parallelism of execution runs.

Continuous integration / Continuous deployment

Continuous Integration (CI) and Continuous Deployment (CD) are development practices that require developers to integrate code into a shared repository several times a day.

Each integration can then be verified by an automated build and automated tests. CI/CD is a method to deliver applications to customers by introducing automation into the stages of app development.

The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment. Automation controller can be integrated with CI/CD systems to enable automated provisioning, configuration management, application deployment, and other IT tasks as part of the CI/CD pipeline.

For a Continuous Integration system, such as Jenkins, to create a job, it must make a `curl` request to a job template. The credentials to the job template must not require prompting for any particular passwords.

Red Hat product documentation legal notices

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

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

www.redhat.com