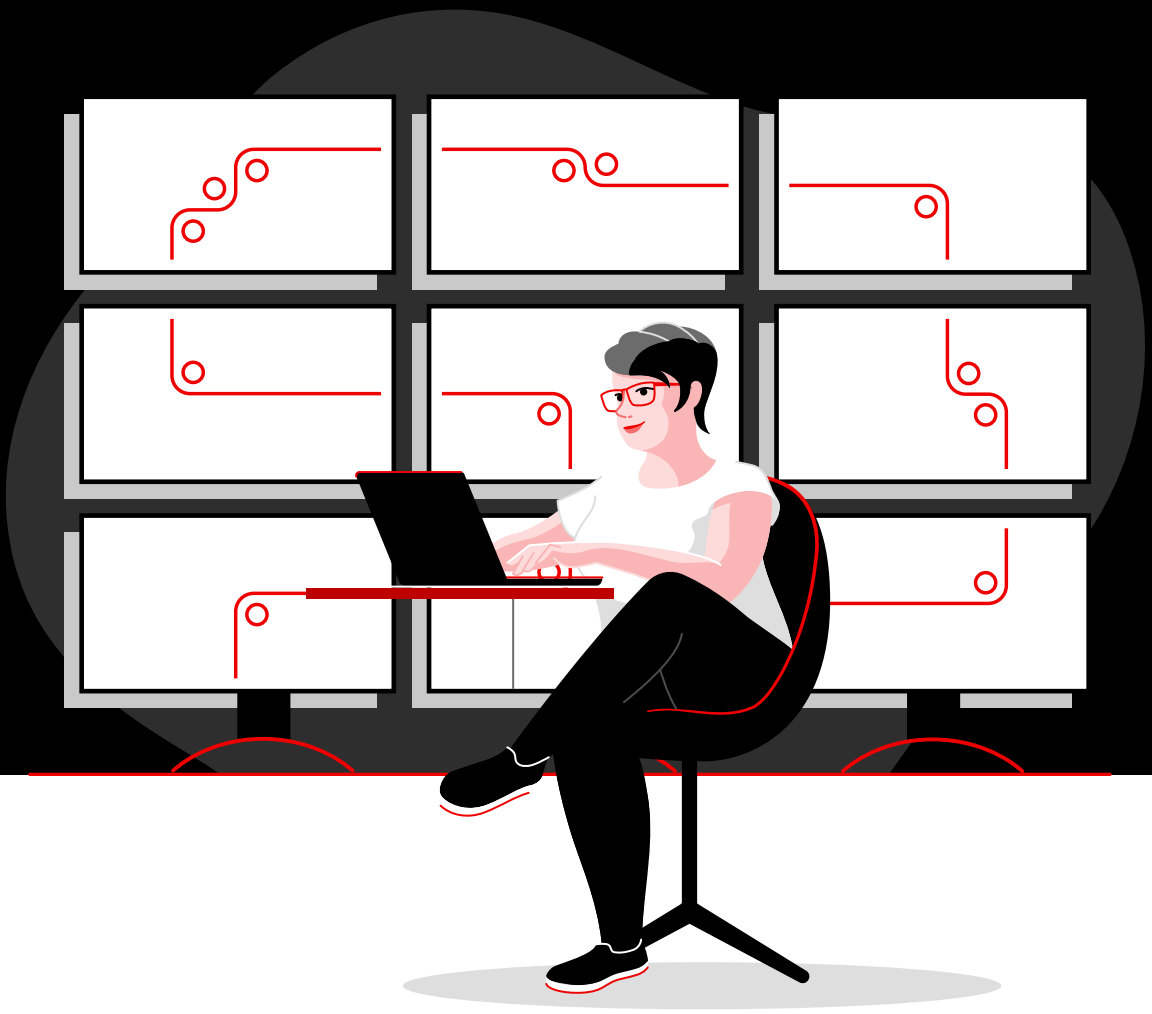
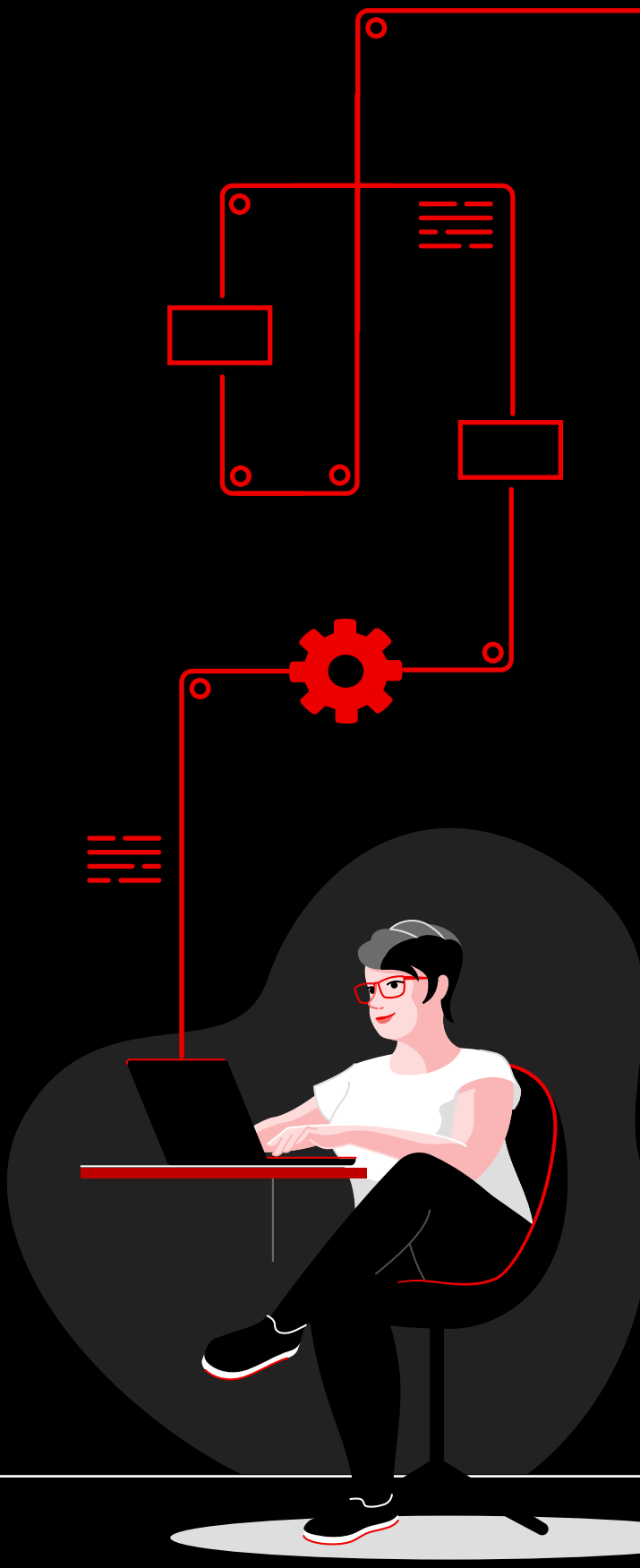
Red Hat
Ansible Automation
Platform



Automate your network with Red Hat

A technical handbook for implementing common network automation tasks with Red Hat Ansible Automation Platform

Contents



Speed operations with network automation

Traditional, manual approaches to network configuration and updates are too complicated and error-prone to effectively support today's rapidly shifting application and data transfer requirements. Programmable, software-based automation technologies can help your team better support and scale your organization's digital initiatives.

With network automation, network operations (NetOps) teams can quickly respond to dynamic needs for capacity, application security, load balancing, and multicloud integrations. They can also implement self-service and on-demand network activities.

As a result, NetOps teams can become as agile and flexible as applications and infrastructure teams to support modern business demands.

Key resources

Check out these resources to learn the basics of Ansible Automation Platform:

- Online training: **Ansible Basics: Automation Technical Overview**
- E-book: **Network automation for everyone**

Speed operations with Red Hat Ansible Automation Platform

With Red Hat® Ansible® Automation Platform, Red Hat brings the popular community Ansible project to the enterprise, adding the features and functionality needed for team-based automation at scale. This powerful IT automation platform combines a simple, easy-to-read automation language with a flexible architecture and security-focused sharing and collaboration capabilities. Because programming skills are not required, teams across your organization can readily use Ansible Automation Platform.

Ansible Automation Platform helps you streamline and manage complex datacenter environments, from servers and networks to applications and DevOps. It provides support for multivendor network infrastructure devices across campus, branch, cloud, and edge environments so you can automate everything using a single tool.

This e-book explains how to get started with common network automation tasks.

NOTE: The commands in this e-book are written for Ansible Automation Platform 2.x and are not applicable for Ansible Automation Platform 1.0 and previous versions.

Install and configure Red Hat Ansible Automation Platform

Install Ansible Automation Platform

Installing and setting up Ansible Automation Platform is easy and fast.

Step **1a**

Install command-line Ansible using yum

Run the following command:

```
$ sudo yum install ansible
```

Read the Ansible installation guide for complete instructions.

Step **1b**

Install Ansible Automation Platform using the installation tool

1. Make sure you have the latest edition or download a free trial at red.ht/try_ansible.
2. Unpack the tar file (version and name may be different):

```
$ tar xvzf ansible-automation-platform-setup-bundle-1.2.1-1.tar.gz
$ cd ansible-automation-platform-setup-bundle-<version>
```

3. Open the inventory file with a text editor:

```
$vi inventory
```

4. Set up your IP address or fully qualified domain name (FQDN) for the `[automationhub]` and `[automationcontroller]` hosts and define your passwords:
   - `admin_password` for administration
   - `pg_password` for database

5. Run the setup script.

```
$sudo ./setup.sh
```

6. Once installation has completed, navigate to your Ansible Automation Platform host, using Google Chrome or Mozilla Firefox by using either the hostname or IP address. Log in using the administrative credentials defined in the inventory file.
7. Add your subscription via the settings menu in the web user interface.

Read the Ansible Automation Platform installation guide and the Automation controller quick setup guide for complete instructions.

Step 2

Install Ansible Content Collections for networking

Red Hat offers certified, supported Ansible Content Collections for a wide variety of network devices, tools, and infrastructure through Ansible automation hub. Each collection exists within a namespace that contains one or more collections. Use the `ansible-galaxy` command to install these collections:

```
$ ansible-galaxy collection install namespace.collection_name
```

Follow the instructions in the Ansible Automation Platform documentation to configure Ansible automation hub to access and manage your Collections. You can find community-supported collections on Ansible Galaxy.

Step 3

Create your execution environments

Execution environments are containers that include all of the required dependencies to perform network automation. They replace Python virtual environments and let you easily port your automation across systems. You can create execution environments using `ansible-builder`.

Install `ansible-builder`:

```
$ pip install ansible-builder
```

Read the Ansible Builder documentation to learn more about requirements, installation, and use.

Automation developers can use `ansible-navigator` to test and run playbooks locally within an execution environment. You can also import the same container into your automation controller to run network automation job templates.

Follow the instructions in the Ansible Automation Platform documentation to create and consume execution environments.

Set up your network environment

Configure your network environment for Ansible Automation Platform according to these best practices.

Ensure connectivity to your network environment

Configure an Ansible Automation Platform service account on your authentication, authorization, and accounting (AAA) systems for login. Ansible Automation Platform supports enterprise authentication methods like Terminal Access Controller Access-Control System Plus (TACACS+), Remote Access Dial-In User Service (RADIUS), and lightweight directory access protocol (LDAP). Learn more in the Setting up enterprise authentication section of the documentation.

Create your playbook repository

Connect Ansible Automation Platform to your source control management (SCM) tool by setting up a project in the web interface, which gives you access to all playbooks in that project repository.

Configure your inventory

Create an inventory of the network devices you want to automate. Ansible Automation Platform can manage multiple inventories. You can dynamically load inventories from popular tools like Amazon Web Services (AWS) EC2, Microsoft Azure Resource Center, and VMware vCenter, using inventory plug-ins. You can also load inventories from an Ansible Automation Platform project. The Build your inventory chapter of this e-book describes more about building and using inventories.

Set your network firewall rules

Set your firewall rules to allow Ansible Automation Platform to connect to routers and switches, using the default secure shell (SSH) port 22. If desired, you can change this port number using the `ansible_port` host variable.

Set your Ansible Automation Platform passwords

Create a credential for holding your passwords. You can grant users and teams the ability to use credentials without actually exposing the credential to the user.

Create an Ansible job template

Create a job template to connect your inventory, credential, and project. Job templates define sets of parameters for running automation jobs, allowing you to execute the same set of tasks many times and reuse content across teams. Each job template includes:

- A project from which to load Ansible Playbooks.
- An inventory or list of automation targets like network switches.
- A credential for logging into and automating the devices in your inventory.
- An execution environment to pull the required dependencies for the automation job.

Run your first command and playbook

Get to know playbooks

Playbooks are Ansible's configuration, deployment, and orchestration language. They consist of sets of human-readable instructions called plays that define automation across an inventory of hosts. Each play includes 1 or more tasks that target 1, many, or all hosts in an inventory. Each task calls an Ansible module that performs a specific function, such as collecting useful information, backing up network files, managing network configurations, running commands, or validating connectivity.

Ansible Playbooks can be shared and reused by multiple teams to create repeatable automation.

Ansible Playbooks

Anatomy of a playbook

This example shows the common parts of an Ansible playbook.

```
1  ---
2  - name: Add VLANs
3    hosts: arista
4    gather_facts: false
5
6    vars:
7      vlans:
8        - name: desktops
9          vlan_id: 20
10       - name: servers
11         vlan_id: 30
12       - name: DMZ
13         vlan_id: 50
14
15   tasks:
16     - name: Add VLAN configuration
17       arista.eos.eos_vlans:
18         state: merged
19         config: "{{ vlans }}"
```

Indicates the start of a playbook

Calls a device or group of devices named `arista`

Optional parameter for retrieving facts

Variable definition

In this playbook, we define variable values directly.

If you are using the Ansible Automation Platform web interface, you can also create a survey to prompt users for variable values when they run your playbook. In that case, replace lines 9, 11, and 13 with:

```
# {{variable_name}} input
```

Learn more in the Create an Ansible Automation Platform survey section on page 7.

Tasks

Tasks and Ansible modules have a 1:1 correlation. This section calls modules to configure virtual local area networks (VLANs) for each of the 3 variables defined in the `vars` section.

Create a Red Hat Ansible survey

Surveys set extra variables for your playbook in a user-friendly, question-and-answer way. To create a survey:

1. Click the ***Add survey*** button in the Ansible Automation Platform web interface.
2. For each question, fill out the following information:
   - **Question**: The question to ask the user
   - **Description (optional)**: A description of what is being asked
   - **Answer variable name**: The Ansible variable name in which the response will be stored
   - **Answer type**: The format—single or multiline text, password, multiple choice, or integer or decimal number—of the response
   - **Default answer (optional)**: The default value of the variable
   - **Minimum and maximum length (optional)**: The minimum and maximum allowed length of the answer
   - **Required**: Whether or not the question is optional
3. Click the ***+*** button to add the question to the survey.
4. Repeat step 3 to add more questions to the survey.
5. Click the ***Save*** button to save the survey when you are finished.

Read the Surveys section of the Ansible Automation Platform documentation to learn more.

Run your playbook

Running a playbook is simple, but the process is different for command line Ansible and the Ansible Automation Platform web interface.

Command line Ansible

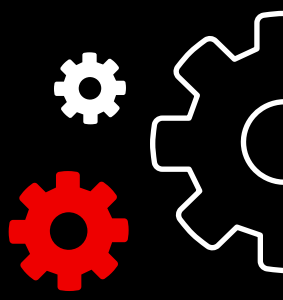
Run the following command:

```
ansible-navigator run <playbook name> -i <inventory file>
```

Red Hat Ansible Automation Platform web interface

Press the Launch job (rocket) button next to your template in the Ansible Automation platform web interface.

Build your inventory



Understand inventories

An inventory is a collection of hosts that may be acted on using Ansible commands and playbooks. Inventory files organize hosts into groups and can serve as a source of trust for your network. These files can be formatted as simple INI or YAML. Many organizations choose to write their inventories in YAML for consistency with their playbooks. Using an inventory file, a single playbook can maintain hundreds of network devices with a single command.

This chapter explains how to build an inventory file.

Create a basic INI-formatted inventory

First, group your inventory logically. Best practices are to group servers and network devices by their ***what*** (application, stack, or microservice), ***where*** (datacenter or region), and ***when*** (development stage).

- **What**: `db, web, leaf, spine`
- **Where**: `east, west, floor_19, building_A`
- **When**: `dev, test, staging, prod`

This example code, in INI format, illustrates a basic group structure for a small datacenter. You can group groups using the syntax `[metagroupname:children]` and listing groups as members of the metagroup.

Here, the group `network` includes all leafs and all spines. The group `datacenter` includes all network devices plus all webservers.

Read the Build your inventory section of the Ansible documentation to learn more. You can also find a sample inventory report playbook on GitHub.

```
[leafs]
leaf01
leaf02

[spines]
spine01
spine02

[network:children]
leafs
spines

[webservers]
webserver01
webserver02

[datacenter:children]
network
webservers
```

Anatomy of a YAML-formatted inventory

```
1  ---
2  all:
3    vars:
4      ansible_user: admin
5      ansible_password: password123
6      ansible_become_pass: password123
7      ansible_become: True
8      ansible_become_method: enable
9      ansible_network_cli_ssh_type: libssh
10   children:
11     routers:
12       children:
13         arista:
14         cisco:
15         juniper:
16     arista:
17       hosts:
18         rtr2:
19           ansible_host: 172.16.100.2
20         rtr4:
21           ansible_host: 172.16.100.4
22       vars:
23         ansible_network_os: arista.eos.eos
24         ansible_connection: ansible.netcommon.network_cli
25     cisco:
26       hosts:
27         rtr1:
28           ansible_host: 172.16.100.1
29       vars:
30         ansible_network_os: cisco.ios.ios
31         ansible_connection: ansible.netcommon.network_cli
32     juniper:
33       hosts:
34         rtr3:
35           ansible_host: 172.16.100.3
36       vars:
37         ansible_network_os: junipernetworks.junos.junos
38         ansible_connection: ansible.netcommon.netconf
```

Indicates the start of a playbook

Defines variables that apply to all hosts within the inventory, regardless of group

Group hierarchy

Lines 10-15 identify the host groups within this inventory. In this case, the group `routers` contains 3 subgroups: `arista`, `cisco`, and `juniper`.

Group definition

The `hosts` command defines which hosts belong to each group. In this example, the group `arista` contains 2 hosts that are identified by IP address.

Group variables

Each group can have its own set of variables. This inventory defines the operating system and connection type for each group. Both of these variables point to items contained in content collections.

Learn more about these variables in the Group your variables section on page 11.

Group inventory by platform

As your inventory grows, you may want to group devices by platform so you can easily define platform-specific variables for all devices on that platform. Lines 10-15 of the example inventory identify the hierarchy of host groups for this inventory. The `routers` group contains 3 children or subgroups categorized by platform: `arista`, `cisco`, and `juniper`. Each of these subgroups contains 1 or more hosts, defined by IP address in lines 16-21, 25-28, and 32-35.

Read the Group inventory by platform section of the Ansible documentation to learn more.

```
10    children:
11      routers:
12        children:
13          arista:
14          cisco:
15          juniper:
16      arista:
17        hosts:
18          rtr2:
19            ansible_host: 172.16.100.2
20          rtr4:
21            ansible_host: 172.16.100.4
```

Set your variables

You can set values for many of the variables you needed in your first Ansible command in the inventory, so you can skip them in the `ansible-playbook` command. Lines 2-9 of the example inventory define variables that apply to all hosts listed in the inventory, regardless of which group they belong to.

You can set and store variables in several different files. As a best practice, set variables used to connect to devices—like login information or IP address—in inventory files or playbooks. Set variables related to device configuration in separate files stored in the `group_vars` directory. Read the Organizing host and group variables section of the Ansible documentation to learn more.

```
2   all:
3     vars:
4       ansible_user: admin
5       ansible_password: password123
6       ansible_become_pass: password123
7       ansible_become: True
8       ansible_become_method: enable
9       ansible_network_cli_ssh_type: libssh
```

Group your variables

When devices in a group share the same variable values, such as operating system (OS) or SSH user, you can reduce duplication and simplify maintenance by consolidating these into group variables. Group variables are set within their respective group definitions. Lines 22-24, 29-31, and 36-38 of the example inventory set group variable values for each of the 3 host groups.

```
22        vars:
23          ansible_network_os: arista.eos.eos
24          ansible_connection: ansible.netcommon.network_cli
```

This example defines network operating system (NOS) and connection type variables for each of the 3 subgroups. In this case, these variables point to items contained in Ansible Content Collections. Ansible Content Collection items are formatted as `namespace.collection_name.item`. For example, `arista.eos.eos` points to the EOS operating system plug-in within the EOS Collection delivered through the Arista namespace, while `ansible.netcommon.network_cli` points to the network CLI plug-in within the Netcommon Collection delivered through the Ansible namespace.

Variable syntax

The syntax for variable values is different in inventory, in playbooks, and in group_vars files. Even though playbook and group_vars files are both written in YAML, you use variables differently in each.

INI-style inventory files

Use the syntax `key=value` for variable values:

```
ansible_network_os=cisco.ios.ios
```

Group_vars and playbook files

Use the full key name:

```
ansible_network_os: cisco.ios.ios
```

Files with .YML and .YAML extensions

Use YAML syntax:

```
key: value
```

Read the Variable syntax section of the Ansible documentation to learn more.

Protect sensitive variables

Best practices are to use additional protection for sensitive variables, such as passwords.

Ansible Automation Platform provides credential management for passwords and key information. Using the ***Credentials*** page in the web interface, you can grant users and teams the ability to use credentials without exposing the credential to the user. Read the **Credentials section** of the Ansible documentation to learn more.

Note that Ansible Automation Platform can run on systems with **Federal Information Processing Standards (FIPS) mode** enabled.

Implement common use cases

This chapter shows sample playbooks for common network automation use cases, including adding a VLAN, gathering facts, retrieving resource information, and backing up configurations.

Add a VLAN

Configuring VLANs that span multiple network devices is an ongoing activity for NetOps. Ansible makes it easy to create a VLAN and propagate it across your network.

```
---
- name: Add VLANs
  hosts: arista
  gather_facts: false
  vars:
    vlans:
      - name: desktops
        vlan_id: 20
      - name: servers
        vlan_id: 30
      - name: DMZ
        vlan_id: 50

  tasks:
    - name: Add VLAN configuration
      arista.eos.eos_vlans:
        state: merged
        config: “{{ vlans }}”
```

Gather facts

Most networks contain many different platforms and devices. Ansible can query, store, and report on network data, including software versions and interface information.

```
---
- name: Use facts module
  hosts: cisco
  gather_facts: false

  tasks:
    - name: Retrieve facts
      cisco.ios.ios_facts:

    - name: Display version
      debug:
        msg: "{{ ansible_net_version }}"

    - name: Display serial number
      debug:
        msg: "{{ ansible_net_serialnum }}"
```

Retrieve resource information

Ansible network resource modules simplify and standardize how you manage different network devices. Any resource module can use `state: gathered` to retrieve information about network resources.

```
---
- name: Retrieve interface information
  hosts: cisco
  gather_facts: false

  tasks:
    - name: Use state gathered
      cisco.ios.ios_interfaces:
        state: gathered
      register: interfaces_info

    - name: Print out interfaces information
      debug:
        msg: "{{ interfaces_info }}"
```

Back up configurations

Storing backups of configurations is a critical activity for NetOps. Ansible Automation Platform makes it easy to pull parts of or an entire configuration from a network device.

```
---
- hosts: cisco
  gather_facts: false

  tasks:
  - name: Back up config
    cisco.ios.ios_config:
      backup: yes
```

Access content to use with Ansible Automation Platform

You can access ready-to-use automation content to expedite the adoption of Red Hat Ansible Automation Platform.

Ansible Content Collections

An Ansible Content Collection is a standardized distribution format for Ansible content that can include playbook examples, roles, modules, plug-ins, and more. Ansible Automation Platform offers certified, supported Ansible Content Collections to extend platform capabilities, integrate with third-party technologies, expand automation across domains, and ease adoption. You can also access validated Content Collections that provide tested, opinionated references, including routing protocol, interfaces, access lists, and networking base configurations. You can install fully supported, Red Hat Ansible Certified Content Collections and access validated reference content from Ansible automation hub, available with your Ansible Automation Platform subscription.

Ansible roles

Ansible roles bundle automation content to make it reusable. Instead of creating long playbooks with hundreds of tasks, you can use roles to organize and break tasks apart into smaller, more discrete units of work. A role includes all of the tasks, variables, and handlers needed to complete a unit of work. Roles are distributed either as standalone entities or as part of Ansible Content Collections.

Ansible automation hub

Ansible automation hub gives Red Hat Ansible Automation Platform subscribers access to fully supported and certified Content Collections developed, tested, and maintained by Red Hat and its technology partners. It gives you a secure portal to Ansible Content Collections, along with a private network for internal and third-party automation content. Automation hub is the de facto repository for content used in production automation environments.

Ansible Galaxy

Ansible Galaxy houses all community Ansible Collections and existing standalone roles. You can also contribute the collections and roles you create to the community through Ansible Galaxy.

Take your automation even further

Create advanced automation workflows

Once you've gained experience and are confident managing your network with Ansible Automation Platform, you can explore advanced use cases with event-driven automation. Event-driven automation is the next logical step in the journey to end-to-end IT and network automation. It lets you respond in a predetemined way to changing network conditions, without manual intervention. Event-Driven Ansible is included with Ansible Automation Platform and takes advantage of the same constructs as playbooks, so it's easy to get started whenever you're ready. For example, you can use Event-Driven Ansible to automatically gather facts to enhance service tickets, handle user administration tasks like password resets, or perform basic troubleshooting actions.

You can also use Red Hat Ansible Lightspeed with IBM watsonx Code Assistant to help your automation teams learn, create, and maintain Ansible Automation Platform content, such as playbooks, more efficiently. With this generative artificial intelligence (AI) service, you can build, find, understand, and optimize your automation content rapidly.

Find more information

Red Hat provides many resources—including detailed documentation, articles, videos, and discussions—for Ansible Automation Platform. Most are located at **ansible.com** and on the **Red Hat Customer Portal**.

- Product website:
  Red Hat Ansible Network Automation
- Hands-on labs:
  Interactive Ansible Automation Platform labs
- Documentation:
  Network platform index
  Ansible community documentation
- User guides:
  Inventories and variables
  Surveys
  Credentials
- E-book:
  Network automation guide
- Training classes:
  Ansible Basics: Automation Technical Overview
  Ansible for Network Automation
- Free trial:
  Ansible Automation Platform trial download

Ready to automate your network?

Using an intuitive, human-readable language, Red Hat Ansible Automation Platform gives you a simple, powerful path to modern network operations while supporting your current processes and existing infrastructure. With a flexible, scalable automation framework, you can improve infrastructure availability, staff productivity, network security, and configuration compliance more easily.

Try Ansible Automation Platform for free: **red.ht/try_ansible**

Deploy faster with Red Hat experts

Automating your network may seem like a daunting task, but Red Hat Consulting can help. All Red Hat Consulting engagements begin with a half-day, no-cost, and on-site discovery session. During these sessions, Red Hat experts work with you to identify your most pressing business challenges, viable approaches for overcoming them, and desired outcomes for implementing network automation.

Schedule a complimentary discovery session: redhat.com/consulting

612699_1123_KVM

Red Hat