



# Automate your network with Red Hat

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

# See what's inside

Speed operations with network automation

## Chapter 1

Install and configure Red Hat Ansible Automation Platform

## Chapter 2

Run your first command and playbook

## Chapter 3

Build your inventory

## Chapter 4

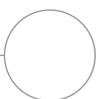
Implement common use cases

## Chapter 5

Access content to use with Red Hat Ansible Automation Platform

Find more information

Ready to get started?



# Speed operations with network automation

Traditional, manual approaches to network configuration and updates are too slow 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 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.

## 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 trusted, composable execution environment and security-focused sharing and collaboration capabilities. Because programming skills are not required, roles 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 legacy and open network infrastructure devices across multivendor virtual and physical environments so you can automate your entire network using a single tool.

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

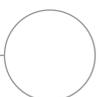
### Key resources

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

- ▶ Online training:  
[Ansible Essentials](#)
- ▶ E-book:  
[Network automation for everyone](#)



**NOTE:** The commands in this e-book are written for Ansible Automation Platform 1.x and are not applicable for Ansible Automation Platform 2.0 and later.



# Install and configure Red Hat Ansible Automation Platform

## Install Red Hat Ansible Automation Platform

Installing and setting up Red Hat 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](https://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
```

3. Set up your passwords:

- ▶ `admin_password` for administration
- ▶ `rabbitmq_password` for messaging
- ▶ `pg_password` for database

4. Run the setup script. 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.

Read the [Ansible Automation Platform quick installation guide](#) for complete instructions.

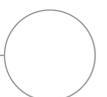
### Step 2

#### Install Ansible Automation Platform Content Collections for networking

Red Hat offers certified, supported [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 [Red Hat Ansible Automation Platform documentation](#) to configure Ansible Automation Hub to access and manage your Collections. You can find community-supported Collections on [Ansible Galaxy](#).



## Set up your network environment

We recommend you configure your network environment for Red Hat Ansible Automation Platform according to these best practices.



### Ensure connectivity to your network environment

Install an Ansible Automation Platform service account on your routers and switches for login and authentication. Ansible Automation Platform supports enterprise authentication methods like Terminal Access Controller Access-Control System Plus (TACACS+) and Remote Access Dial-In User Service (RADIUS). 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](#), giving 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 EC2, Microsoft Azure Resource Center, and VMware vCenter using [inventory plugins](#). You can also [load inventories](#) from an Ansible Automation Platform project. [Chapter 3](#) 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

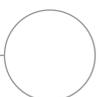
Create a [credential](#) for holding your passwords. You can grant users and teams the ability to use credentials without actually exposing the credential to the user.



### 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.

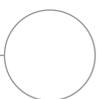
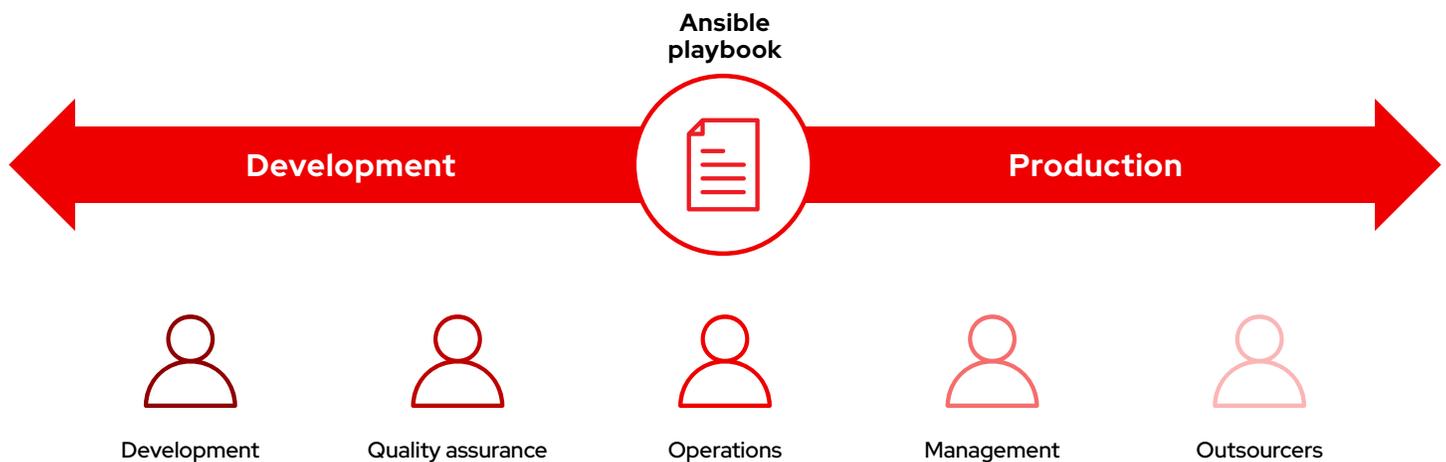


# Run your first command and playbook

## 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 one or more tasks that target one, many, or all hosts in an inventory. Each task calls an Ansible module that performs a specific function like collecting useful information, backing up network files, managing network configurations, or validating connectivity.

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



## 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.

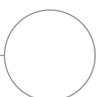
If you are using the Red Hat Ansible Automation Platform web interface, you can also **create a survey** to prompt users for variable values when they run your playbook. In that case, replace lines 9, 11, and 13 with:

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

Learn more about surveys in the **Create a Red Hat Ansible survey** section on page 6.

### 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 three 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 Red Hat Ansible Automation Platform web interface.
2. For each question fill out the following information:
  - ▶ **Name:** 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 – entered text, multiple choice, or number – of the response
  - ▶ **Default answer:** The default value of the variable
  - ▶ **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.

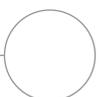
### Command-line Ansible

Run the following command:

```
ansible-playbook <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).

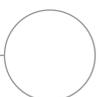
- ▶ **What:** `db`, `web`, `leaf`, `spine`
- ▶ **Where:** `east`, `west`, `floor_19`, `building_A`
- ▶ **When:** `dev`, `test`, `staging`, `prod`

This example code, in INI format, illustrates a basic group structure for a very 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 web servers.

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

```
1  [leafs]
2  leaf01
3  leaf02
4
5  [spines]
6  spine01
7  spine02
8
9  [network:children]
10 leafs
11 spines
12
13 [webserver]
14 webserver01
15 webserver02
16
17 [datacenter:children]
18 network
19 webserver
```



## 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 three subgroups: `arista`, `cisco`, and `juniper`.

### Group definition

The `hosts` command defines which hosts belong to each group. In this example, the group `arista` contains two 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.

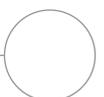
Learn more about these variables in the [Group your variables section](#) on page 10.

## 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 three children or subgroups categorized by platform: `arista`, `cisco`, and `juniper`. Each of these subgroups contains one or more hosts, defined by IP address in lines 16-21, 25-28, and 32-35.

```
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
```

Read the [Group inventory by platform section](#) of the Ansible documentation to learn more.



## 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.

```
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
```

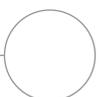
You can set and store variables in several different files. As a best practice, set variables used to connect to devices – like login information or IP address – in inventory files or playbooks. Set variables related to device configuration in separate files stored in the `group_vars` directory. Read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

## Group your variables

When devices in a group share the same variable values, like 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 three 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 three subgroups. In this case, these variables point to items contained in Red Hat Ansible Automation Platform Content Collections. Content Collection items are formatted as `namespace.collection_name.item`. For example, `arista.eos.eos` points to the EOS operating system plugin within the EOS Collection delivered through the Arista namespace, while `ansible.netcommon.network_cli` points to the network CLI plugin 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
```

### Files with .YML and .YAML extensions

Use YAML syntax:

```
key: value
```

### Group\_vars and playbook files

Use the full key name:

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

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 like passwords.

Red Hat 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.

## 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.

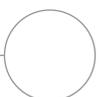
```
1 ---
2 - name: add vlans
3   hosts: arista
4   gather_facts: false
5   vars:
6     vlans:
7       - name: desktops
8         vlan_id: 20
9       - name: servers
10        vlan_id: 30
11       - name: DMZ
12        vlan_id: 50
13
14   tasks:
15     - name: add VLAN configuration
16       arista.eos.eos_vlans: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



## Gather facts

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

```
1 ---
2 - name: use facts module
3   hosts: cisco
4   gather_facts: false
5
6   tasks:
7     - name: retrieve facts
8       cisco.ios.ios_facts:
9
10    - name: display version
11      debug:
12        msg: "{{ ansible_net_version }}"
13
14    - name: display serial number
15      debug:
16        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.

```
1 ---
2 - name: retrieve interface information
3   hosts: cisco
4   gather_facts: false
5
6   tasks:
7     - name: use state gathered
8       cisco.ios.ios_interfaces:
9         state: gathered
10        register: interfaces_info
11
12    - name: print out interfaces information
13      debug:
14        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.

```
1 ---
2 - hosts: cisco
3   gather_facts: false
4
5   tasks:
6   - name: back up config
7     cisco.ios.ios_config:
8       backup: yes
```



# Access content to use with Red Hat Ansible Automation Platform

You can access ready-to-use automation content to make adoption of Red Hat Ansible Automation Platform even faster and easier.



## Ansible Content Collections

A **Content Collection** is a standardized distribution format for Ansible content that can include playbook examples, roles, modules, plugins, and more. This new format decouples the Ansible executable from most of the automation content, giving you more flexibility and portability. You can install community-supported Collections from **Ansible Galaxy** and fully supported, **certified Collections** from **Ansible Automation Hub**.



## 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 Content Collections.



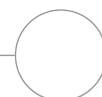
## Ansible Automation Hub

**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 Content Collections, as well as 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

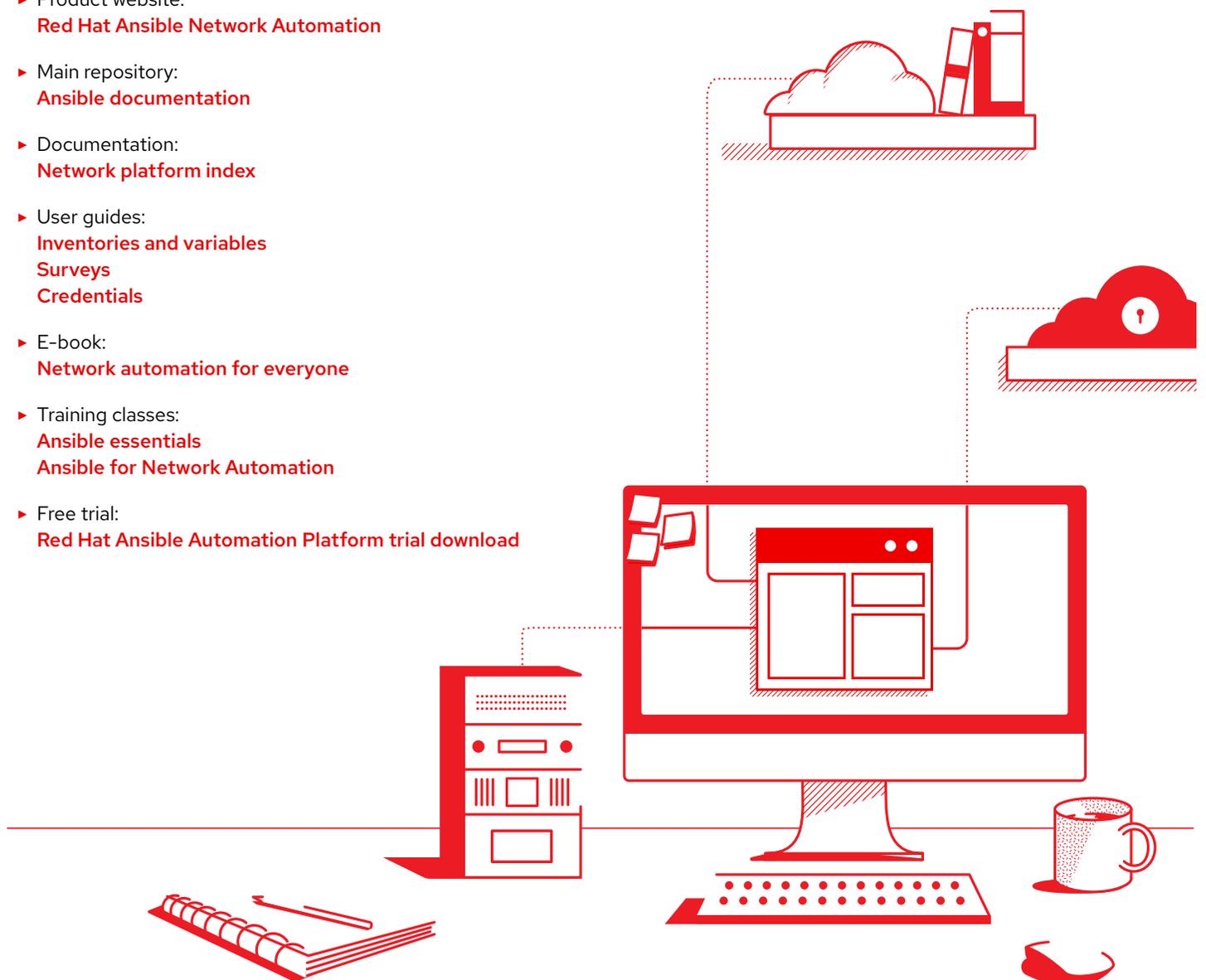
**Ansible Galaxy** houses all community Ansible Collections, as well as existing standalone roles. You can also contribute the Collections and roles you create to the community through Ansible Galaxy.



# Find more information

Red Hat provides many resources – including detailed documentation, articles, videos, and discussions – for Red Hat Ansible Automation Platform. Most are located at [ansible.com](https://www.ansible.com) and on the [Red Hat customer portal](#).

- ▶ Product website:  
[Red Hat Ansible Network Automation](#)
- ▶ Main repository:  
[Ansible documentation](#)
- ▶ Documentation:  
[Network platform index](#)
- ▶ User guides:  
[Inventories and variables](#)  
[Surveys](#)  
[Credentials](#)
- ▶ E-book:  
[Network automation for everyone](#)
- ▶ Training classes:  
[Ansible essentials](#)  
[Ansible for Network Automation](#)
- ▶ Free trial:  
[Red Hat 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 Red Hat Ansible Automation Platform for free:  
[red.ht/try\\_ansible](https://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 complimentary 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](https://redhat.com/consulting)