Documentation Ansible

Introduction to Ansible

This is the

latest

(stable) Ansible community documentation. For Red Hat Ansible Automation Platform subscriptions, see

Life Cycle

for version details.

Introduction to Ansible

Ansible provides open-source automation that reduces complexity and runs everywhere.

Using Ansible lets you automate virtually any task.

Here are some common use cases for Ansible:

Eliminate repetition and simplify workflows

Manage and maintain system configuration

Continuously deploy complex software

Perform zero-downtime rolling updates

Ansible uses simple, human-readable scripts called playbooks to automate your tasks.

You declare the desired state of a local or remote system in your playbook.

Ansible ensures that the system remains in that state.

As automation technology, Ansible is designed around the following principles:

Agent-less architecture

Low maintenance overhead by avoiding the installation of additional software across IT infrastructure.

Documentation Ansible

Simplicity

Automation playbooks use straightforward YAML syntax for code that reads like documentation. Ansible is also decentralized, using SSH with existing OS credentials to access to remote machines.

Scalability and flexibility

Easily and quickly scale the systems you automate through a modular design that supports a large range of operating systems, cloud platforms, and network devices.

Idempotence and predictability

When the system is in the state your playbook describes Ansible does not change anything, even if the playbook runs multiple times.

Ready to start using Ansible?

Get up and running in a few easy steps

.

Documentation Ansible

Start automating with Ansible

This is the

latest

(stable) Ansible community documentation. For Red Hat Ansible Automation Platform subscriptions, see

Life Cycle

for version details.

Start automating with Ansible

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

Install Ansible.

pip

install

ansible

Copy to clipboard

Create a project folder on your filesystem.

mkdir

ansible_quickstart

&&

cd

ansible_quickstart

Copy to clipboard

Using a single directory structure makes it easier to add to source control as well as to

reuse and share automation content.

Continue getting started with Ansible by

building an inventory

.

See also

Installing Ansible

Installation guide with instructions for installing Ansible on various operating systems

Ansible Demos

Demonstrations of different Ansible usecases

Ansible Labs

Labs to provide further knowledge on different topics

Ansible Communication Guide

Questions? Help? Ideas? Ask the community

Documentation Ansible

Building an inventory

This is the

latest

(stable) Ansible community documentation. For Red Hat Ansible Automation Platform subscriptions, see

Life Cycle

for version details.

Building an inventory

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.

To complete the following steps, you will 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.

Continue getting started with Ansible and build an inventory as follows:

Create a file named

inventory.ini

in the

ansible_quickstart

directory that you created in the

preceding step

.

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

Copy to clipboard

Verify your inventory.

ansible-inventory

-i

inventory.ini

--list

Copy to clipboard

Ping the

myhosts

group in your inventory.

ansible

myhosts

-m

ping

-i

inventory.ini

Copy to clipboard

Note

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"
}

Copy to clipboard

Congratulations, you have successfully built an inventory.

Continue getting started with Ansible by

creating a playbook

.

Inventories in INI or YAML format

You can create inventories in either

INI

files or in

YAML

.

In most cases, such as the example in the preceding steps,

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.

For example, the following is an equivalent of the

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

Documentation Ansible

Copy to clipboard

Tips for building inventories

Ensure that group names are meaningful and unique. Group names are also case sensitive.

Avoid spaces, hyphens, and 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: datacenter, region, floor, building.

When

Group hosts by stage, for example: development, test, staging, production.

Use metagroups

Documentation Ansible

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

metagroupname

:

children

:

Copy to clipboard

The following inventory illustrates a basic structure for a data center.

This example inventory contains a

network

metagroup that includes all network devices and a

datacenter

metagroup that includes the

network

group and all webservers.

leafs

:

hosts

:

leaf01

:

ansible_host

:

192.0.2.100

leaf02

Documentation Ansible

:

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

Documentation Ansible

:

webservers

:

hosts

:

webserver01

:

ansible_host

:

192.0.2.140

webserver02

:

ansible_host

:

192.0.2.150

datacenter

:

children

:

network

:

webservers

:

Copy to clipboard

Create variables

Documentation Ansible

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

Copy to clipboard

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

Copy to clipboard

See also

How to build your inventory

Learn more about inventories in

YAML

or

INI

format.

Adding variables to inventory

Find out more about inventory variables and their syntax.

Ansible Vault

Find out how to encrypt sensitive content in your inventory such as passwords and keys.

Documentation Ansible

Creating a playbook

This is the

latest

(stable) Ansible community documentation. For Red Hat Ansible Automation Platform subscriptions, see

Life Cycle

for version details.

Creating a playbook

Playbooks are automation blueprints, in

YAML

format, that Ansible uses to deploy and configure managed nodes.

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.

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.

Documentation Ansible

Complete the following steps to create a playbook that pings your hosts and prints a “Hello world” message:

Create a file named

playbook.yaml

in your

ansible_quickstart

directory, that you created earlier, with the following content:

-

name

:

My first play

hosts

:

myhosts

tasks

:

-

name

:

Ping my hosts

ansible.builtin.ping

:

-

name

:

Documentation Ansible

Print message

ansible.builtin.debug

:

msg

:

Hello world

Copy to clipboard

Run your playbook.

ansible-playbook

-i

inventory.ini

playbook.yaml

Copy to clipboard

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
```

Copy to clipboard

Documentation Ansible

In this output you can see:

The names that you give the play and each task.

You should always use descriptive names that make it easy to verify and troubleshoot playbooks.

The “Gathering 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.

Congratulations, you have started using Ansible!

See also

Ansible playbooks

Start building playbooks for real world scenarios.

Working with playbooks

Go into more detail with Ansible playbooks.

Ansible tips and tricks

Get tips and tricks for using playbooks.

Discovering variables: facts and magic variables

Learn more about the

gather_facts

keyword in playbooks.

Documentation Ansible

Ansible concepts

This is the

latest

(stable) Ansible community documentation. For Red Hat Ansible Automation Platform subscriptions, see

Life Cycle

for version details.

Ansible concepts

These concepts are common to all uses of Ansible.

You should understand them before using Ansible or reading the documentation.

Control node

Managed nodes

Inventory

Playbooks

Plays

Roles

Tasks

Handlers

Modules

Plugins

Collections

Control node

Documentation Ansible

The machine from which you run the Ansible CLI tools (

ansible-playbook

,

ansible

,

ansible-vault

and others).

You can use any computer that meets the software requirements as a control node - laptops, shared desktops, and servers can all run Ansible.

You can also run Ansible in containers known as

Execution Environments

.

Multiple control nodes are possible, but Ansible itself does not coordinate across them, see

AAP

for such features.

Managed nodes

Also referred to as ‘hosts’, these are the target devices (servers, network appliances or any computer) you aim to manage with Ansible.

Ansible is not normally installed on managed nodes, unless you are using

ansible-pull

, but this is rare and not the recommended setup.

Inventory

Documentation Ansible

A list of managed nodes provided by one or more ‘inventory sources’. Your inventory can specify information specific to each node, like IP address.

It is also used for assigning groups, that both allow for node selection in the Play and bulk variable assignment.

To learn more about inventory, see

the Working with Inventory

section. Sometimes an inventory source file is also referred to as a ‘hostfile’.

Playbooks

They contain Plays (which are the basic unit of Ansible execution). This is both an ‘execution concept’ and how we describe the files on which

ansible-playbook

operates.

Playbooks are written in YAML and are easy to read, write, share and understand. To learn more about playbooks, see

Ansible playbooks

.

Plays

The main context for Ansible execution, this playbook object maps managed nodes (hosts) to tasks.

The Play contains variables, roles and an ordered lists of tasks and can be run repeatedly.

It basically consists of an implicit loop over the mapped hosts and tasks and defines how to iterate over them.

Documentation Ansible

Roles

A limited distribution of reusable Ansible content (tasks, handlers, variables, plugins, templates and files) for use inside of a Play.

To use any Role resource, the Role itself must be imported into the Play.

Tasks

The definition of an ‘action’ to be applied to the managed host.

You can execute a single task once with an ad hoc command using

ansible

or

ansible-console

(both create a virtual Play).

Handlers

A special form of a Task, that only executes when notified by a previous task which resulted in a ‘changed’ status.

Modules

The code or binaries that Ansible copies to and executes on each managed node (when needed) to accomplish the action defined in each Task.

Each module has a particular use, from administering users on a specific type of database to managing VLAN interfaces on a specific type of network device.

You can invoke a single module with a task, or invoke several different modules in a playbook.

Documentation Ansible

Ansible modules are grouped in collections. For an idea of how many collections Ansible includes, see the

Collection Index

.

Plugins

Pieces of code that expand Ansible’s core capabilities.

Plugins can control how you connect to a managed node (connection plugins), manipulate data (filter plugins) and even control what is displayed in the console (callback plugins).

See

Working with plugins

for details.

Collections

A format in which Ansible content is distributed that can contain playbooks, roles, modules, and plugins.

You can install and use collections through

Ansible Galaxy

.

To learn more about collections, see

Using Ansible collections

.

Collection resources can be used independently and discretely from each other.