Caltech | Center for Technology & Management Education

Post Graduate Program in DevOps

Powered by simplilearn

DevOps and Cloud Computing

Caltech | Center for Technology & Management Education

Configuration Management with Ansible and Terraform

Powerd by simplilearn

Working with Ansible Roles

Learning Objectives

By the end of this lesson, you will be able to:

- Analyze Ansible Roles
- Define role directory structure
- Write a Role
- Describe Ansible galaxy

Ansible Roles

Introduction

Ansible role is an independent component that allows reuse of common configuration steps.

It helps in automatically loading certain vars_files, tasks, and handlers based on a known file structure.

Roles provide a framework for fully independent or interdependent collections of variables, tasks, files, templates, and modules.

Ansible Roles

Following are some points to remember about Ansible roles:

1. Ansible role must be used within a playbook.
2. Roles are defined using YAML files with a predefined directory structure.
3. Ansible role is a set of tasks to configure a host to serve a certain purpose.
4. It makes the reusability of codes easy for anyone if the role is suitable to them.
5. It can be easily modified and therefore, reduces syntax errors.

Features of Roles

Top-level playbooks serve as a link between the hosts in the inventory file and the roles that should be assigned to them.

Each role is limited to the desired output, with all the necessary actions to achieve that result either within the same role or in other roles listed as dependencies.

Roles are not playbooks. They are small functions that can be used individually within the playbooks.

Ansible Role Structure

Creating a Role

Roles require a Directory structure so that Ansible can find and use them.

Role Structure

- The roles have a structured layout on the file system.
- One can change the default structure of the roles as well.

Each role is a directory tree in itself. So, the role name is the directory name within the **/roles** directory.

Role Directory Structure

A role directory structure contains the following directories: defaults, vars, tasks, files, templates, meta, and handlers.

Each directory must contain a main.yml file that contains relevant content.

Components of a directory:

defaults	It contains default variables for the role. Variables in *default* have the lowest priority so they are easy to override.

Role Directory Structure

vars	It contains variables for the role.
tasks	It contains the main list of steps to be executed by the role.
files	It contains files that can be deployed via a role.

Role Directory Structure

templates	It contains file templates that can be deployed via a role.
meta	It contains metadata of roles like an author, support platforms, and dependencies.
handlers	It contains handlers that can be invoked by “notify” directives and are associated with service.

Role Directory Structure

Given below is an example of a simple role directory structure.

Example

```
site.yml
webservers.yml
xyzservers.yml
roles/
 common/
  tasks/
  handlers/
  files/
  templates/
  vars/
  defaults/
  meta/
 webservers/
  tasks/
  defaults/
  meta/
```

Role Directory Structure

It is a common practice to have platform-specific tasks included from the ***tasks/main.yml*** file

```
# roles/example/tasks/main.yml
-  name: added in 2.4, previously you used 'include'
   import_tasks: xyz.yml
   when: ansible_os_platform|lower == 'centos'
-  import_tasks: debian.yml
   when: ansible_os_platform|lower == 'debian'
# roles/example/tasks/xyz.yml
-  yum:
       name: "httpd"
       state: present
# roles/example/tasks/debian.yml
-  apt:
       name: "apache2"
       state: present
```

Roles may also include modules and other plugin types.

Role's Location

By default, Ansible looks for roles in two locations:

1. In a directory called ***roles/***, relative to the playbook file
2. In ***/etc/ansible/roles***

If the role is stored in a different location, the role path option must be set so that Ansible can find the roles.

Role's Location

Checking shared roles into a single location makes them easier to use in multiple playbooks.

Alternatively, users can call a role with a fully qualified path:

```
---
 - hosts: webservers
   roles:
     - role: '/path/to/my/roles/common'
```

Using Roles

Roles can be used in three ways:

01 At the play level with the roles option

02 At the tasks level with include_role

03 At the tasks level with import_role

Using Roles at Play Level

The roles option at the play level, for each role ‘x’:

If roles/x/tasks/main.yml exists	Ansible adds the tasks in that file to the play.
If roles/x/handlers/main.yml exists	Ansible adds the handlers in that file to the play.
If roles/x/vars/main.yml exists	Ansible adds the variables in that file to the play.
If roles/x/defaults/main.yml exists	Ansible adds the variables in that file to the play.
If roles/x/meta/main.yml exists	Ansible adds any role dependencies in that file to the list of roles.

Any copy, script, template or include tasks (in the role) can reference files in **roles/x/{files, templates, tasks}/** without having to path them relatively or absolutely.

Using Roles at Play Level

When the roles are used at the play level, Ansible treats them as static imports and processes them during playbook parsing.

Ansible executes your playbook in this order:

Any pre_tasks defined in the play

Any handlers triggered by pre_tasks

Using Roles at Play Level

Each role listed in roles

Any tasks defined in the play

Any handlers triggered by the roles or tasks

Any post_tasks defined in the play

Any handlers triggered by post_tasks

Using Roles Dynamically

Users can reuse roles dynamically anywhere in the **tasks** section of a play using **include_role**.

- Included roles run in the order they are defined.
- If there are other tasks before an **include_role** task, the other tasks will run first.

Using Roles Dynamically

Example

```
---

- hosts: webservers
    tasks:
      - name: Print a message
        ansible.builtin.debug:
            msg: "this task runs before the example role“

      - name: Include the example role
         include_role:
            name: example

      - name: Print a message
        ansible.builtin.debug:
             msg: “this task runs after the example role”
```

Using Roles Statically

Users can reuse roles statically anywhere in the **tasks** section of a play using **import_role**.

The behavior is the same as using the **roles** keyword.

Note

When a tag is added to an import_role statement, Ansible applies the tag to all tasks within the role.

Using Roles Statically

Example

```
---

 - hosts: webservers
    tasks:
         - name: Print a message
            ansible.builtin.debug:
                  msg: "before we run our role“

          - name: Import the example role
             import_role:
                  name: example

          - name: Print a message
             ansible.builtin.debug:
                    msg: "after we ran our role"
```

Multiple Executions of a Role

Running a role multiple times in one playbook

Ansible only executes each role once, even if it is defined multiple times.

Roles are executed multiple times only if the parameters defined on the role are different for each definition.

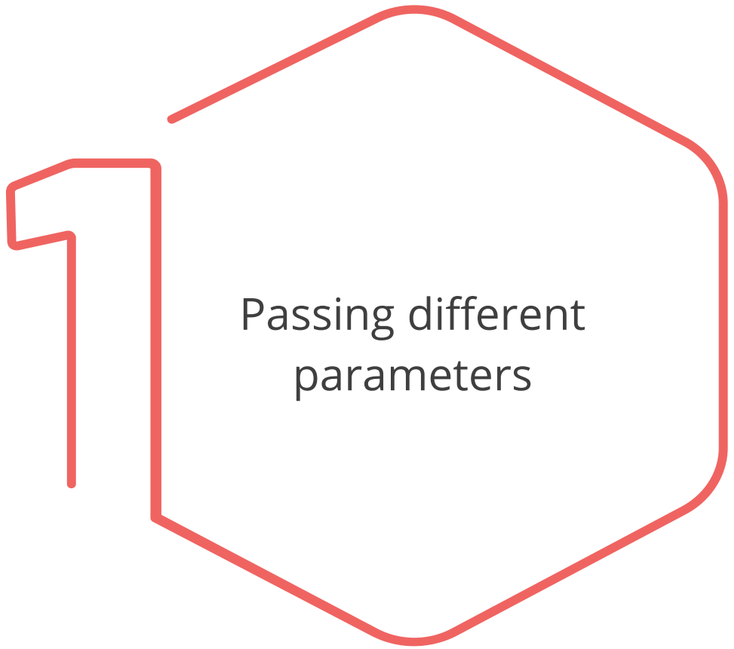
Ansible only runs the role **xyz** once in this play.

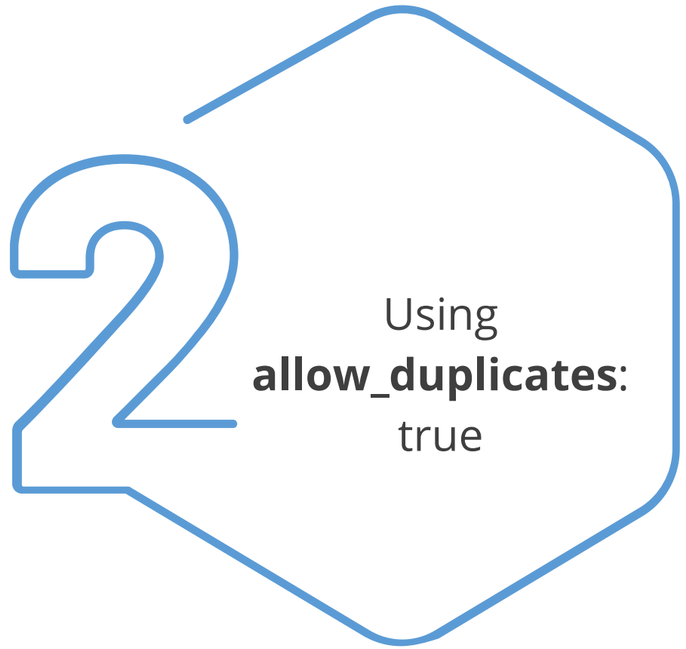
Example

```
---
- hosts: webservers
      roles:
          - xyz
          - abc
          - xyz
```

Multiple Executions of a Role

There are two options to force Ansible to run a role more than once.





Passing Different Parameters

If different parameters are passed in each role definition, Ansible runs the role more than once.

Providing different variable values is not the same as passing different role parameters.

To achieve this, the **roles** keyword is used since **import_role** and **include_role** do not accept the role parameters.

Passing Different Parameters

This playbook runs the **xyz** role twice:

Ansible runs **xyz** twice because each role definition has different parameters.

Example

```
---
-  hosts: webservers
        roles:
            role: xyz
             message: "first"

            role: xyz
              message: "second"
```

Using Duplicates

Ansible can run a role more than once using **allow_duplicates: true.**

Add **allow_duplicates: true** to the **meta/main.yml** file for the role:

In this example, Ansible runs xyz twice because we have explicitly enabled it to do so.

Example

```
# playbook.yml
---
- hosts: webservers
  roles:
   - foo
   - foo
# roles/foo/meta/main.yml
---
allow_duplicates: true
```

Creating a Role

Ansible galaxy commands have a template to create an Ansible role.

The following command will create roles under the default directory **/etc/ansible/roles**.

```
ansible-galaxy -h
```

Creating a Role

```
ansible-galaxy init /etc/ansible/roles/myrole --offline
```

In the above command:

- **ansible-galaxy** is the command to create roles using the templates.
- **init** is to initialize the role.
- **myrole** is the name of the role.
- **offline** creates an offline mode of role rather than getting it from an online repository.

Ansible Galaxy

Ansible Galaxy

Ansible Galaxy refers to a free Galaxy website for finding, downloading, and sharing community-developed roles.

- Galaxy provides pre-packaged units of work such as roles and collections.

- On Galaxy, a user can find roles for provisioning infrastructure, deploying applications, and all the tasks that are done every day in an organization.

Ansible Galaxy

The following are some of the most common Ansible Galaxy commands:

`ansible-galaxy list`	To display the list of installed roles with version numbers
`ansible-galaxy remove [role]`	To remove an installed role
`ansible-galaxy init`	To create a role template suitable for submission to Ansible Galaxy

Galaxy: Collection

Collections are the distribution formats for the Ansible content.

On Ansible version 2.8, users can get the unique feature of the collections.

Collections can be used to package and distribute roles, modules, playbooks, and plugins.

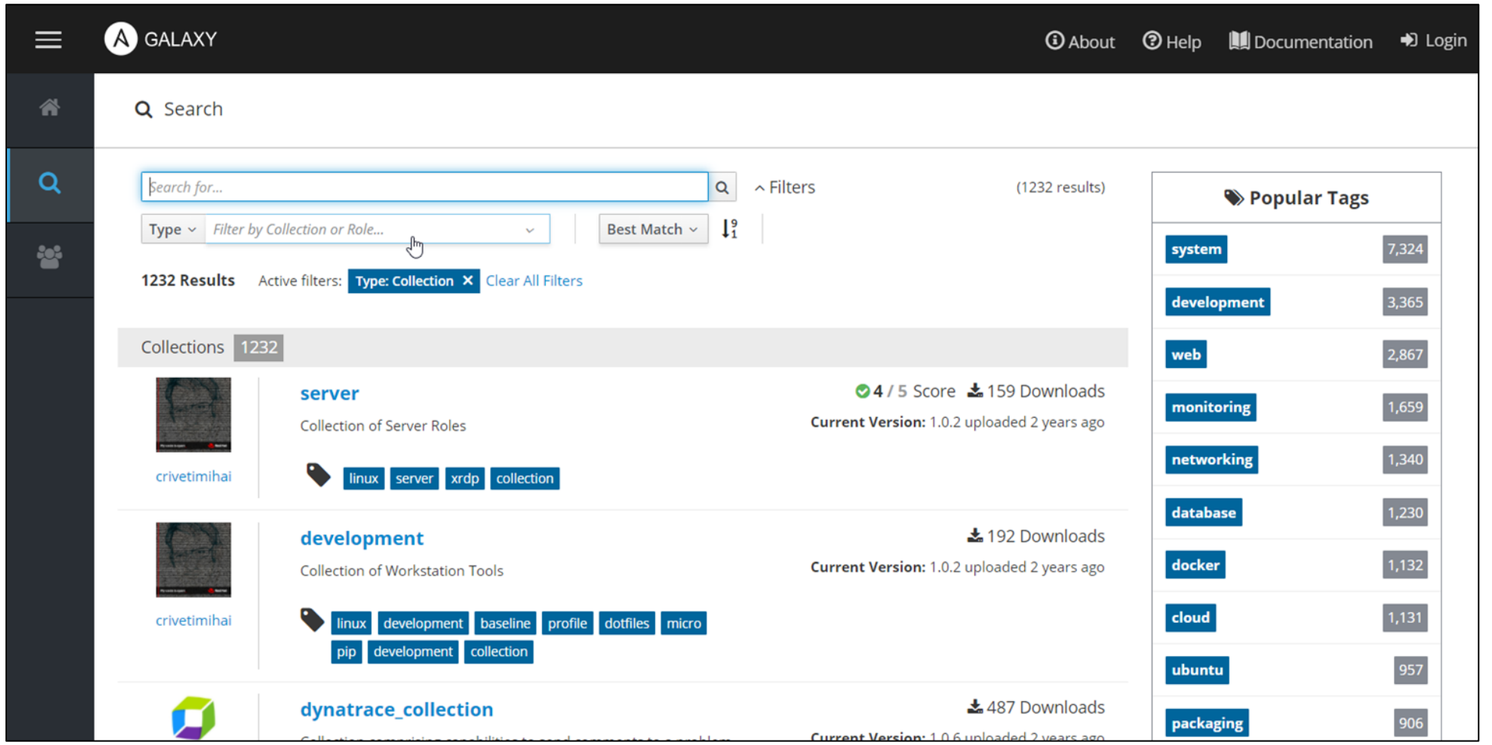
Galaxy: Collection

To find collections on Galaxy:

Click on the **Search** icon in the left-hand navigation.

Set the filter to the ***collection***.

Set the other filters and press **enter.**



Install a Collection

To install a collection from Galaxy:

```
ansible-galaxy collection install collection_name
```

The command by default uses **https://galaxy.ansible.com** as the Galaxy server.

To upgrade a collection to the latest available version from the Galaxy server, users can use the **--upgrade** option after the collection name.

Install a Collection: Automation Hub

Users can download collections from the automation hub from the command line.

Automation Hub content is available to subscribers only, so one must download an API token and configure the local environment, before downloading collections.

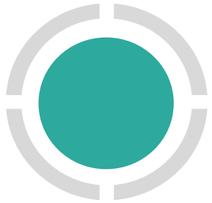
Steps to download collection from Automation Hub:

1. Get your Automation Hub API token from **https://cloud.redhat.com/ansible/automation-hub/token/.**
2. Configure Red Hat Automation Hub server in the **server_list** option under the **[galaxy]** section in **ansible.cfg** file.

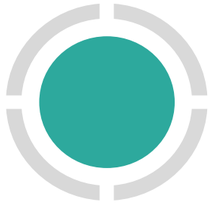
Install a Collection : GitHub

Collections can also be installed from the git repository.

Installing collections from a git repository offers the developer to review the collection before they build and publish it.



The repository must contain a galaxy.yml or MANIFEST.json file.



This file provides metadata such as the version number and namespace of the collection.

Collection Structure

A collection is a simple data structure. Unless users have specific content that belongs in one of the directories, none of them are required.

- A collection requires a **galaxy.yml** file at the root level of the collection.

- This file contains all the metadata that Galaxy and other tools need to package, build, and publish the collection.

Collection Structure

A collection follows the following simple structure of directories and files:

```
collection/
├── docs/
├── galaxy.yml
├── plugins/
│   ├── modules/
│   │   └── module1.py
│   ├── inventory/
│   └── .../
├── README.md
├── roles/
│   ├── role1/
│   ├── role2/
│   └── .../
├── playbooks/
│   ├── files/
│   ├── vars/
│   ├── templates/
│   └── tasks/
└── tests/
```

Collection Range Identifiers

By default, **ansible-galaxy** installs the latest available version.

Version range identifiers are used to install a specific version of the collection.

Multiple range identifiers can be specified by separating each with a **“,”**.

Collection Range Identifiers

The following is an example to install the most recent version that is greater than or equal to 3.1.0 and less than 2.0.0:

```
ansible-galaxy collection install 'my_namespace.my_collection:>=3.1.0,<2.0.0'
```

Collection Range Identifiers

The following range identifiers are:

*	The most recent version
!=	Not equal to the version specified
==	Exactly the version specified
>=	Greater than or equal to the version specified
>	Greater than the version specified
<=	Less than or equal to the version specified
<	Less than the version specified

Galaxy: Roles

The Ansible Galaxy is a large public repository of Ansible roles.

Roles come with READMEs detailing the roles' use and variables.

Ansible Galaxy contains a wide variety of roles that are continuously evolving and expanding.

Galaxy: Roles

The Galaxy can use Git to add other role sources like GitHub.

By default, roles are placed into the directory:
/etc/ansible/roles.

Note

Roles must be downloaded before using them in the playbooks.

Install a Role

The **ansible-galaxy** command can be used to install roles from Galaxy or directly from a Git-based SCM.

To install a role from Galaxy :

```
ansible-galaxy install namespace.role_name
```

The command by default uses https://galaxy.ansible.com as the Galaxy server.

Install a Role

The following are options to save the roles in a different path:

1. By setting the environment variable **ANSIBLE_ROLES_PATH** in the session
2. By using the **--roles-path** option for the ansible-galaxy command
3. By defining **roles_path** in an ansible.cfg file

Install a Role

A particular version of a role is downloaded by specifying one of the imported tags, that corresponds to the version to install.

To check the available versions for a role:

1. Locate the role on the Galaxy search page.

2. Click on the name to view more details, including the available versions.

Install a Role

To install a specific version of a role from Galaxy, append a comma and the value of a GitHub release tag.

For example:

```
$ ansible-galaxy install mywebserver.apache,1.0.0
```

It is also possible to point directly to the git repository and specify a branch name or commit hash as the version.

```
$ ansible-galaxy install git+<Github_Repository_URL>,0b7cd353c0250e87a26e0499e59e
```

Roles Installation

Each role in the file will have one or more of the following attributes:

src	The source of the role
scm	The SCM is specified
version	The version of the downloaded role
name	Download the role to a specific name

Multiple Roles Installation

Multiple roles can be installed by including the roles in a requirements.yml file.

The following command installs roles included in the requirements.yml:

```
$ ansible-galaxy install -r requirements.yml
```

Note

Roles and Collections can be installed from the same requirements file.

Role Dependencies

Roles can also be dependent on other roles, and when a user installs a role that has dependencies, those dependencies will automatically be installed to the roles_path.

There are two ways to define the dependencies of a role:

Using **meta/requirements.yml**

Using **meta/main.yml**

Role Dependencies

1

Users can create the file **meta/requirements.yml** and define dependencies in the same format as used for **requirements.yml**.
From there a user can import the specified roles in their tasks.

2

Users can specify role dependencies in the **meta/main.yml** file by providing a list of roles under the dependencies section.

Key Takeaways

- Roles are not playbooks. They are small functions that can be used individually within the playbooks.
- Ansible Galaxy refers to a free Galaxy website for finding, downloading, and sharing community-developed roles and collections.
- Collections can be used to package and distribute roles, modules, playbooks, and plugins.
- Roles can be dependent on other roles, and when a user installs a role that has dependencies, those dependencies will automatically be installed to the roles_path.

Creating an Apache Server on Ubuntu Using Ansible Roles

Duration: 25 Min.

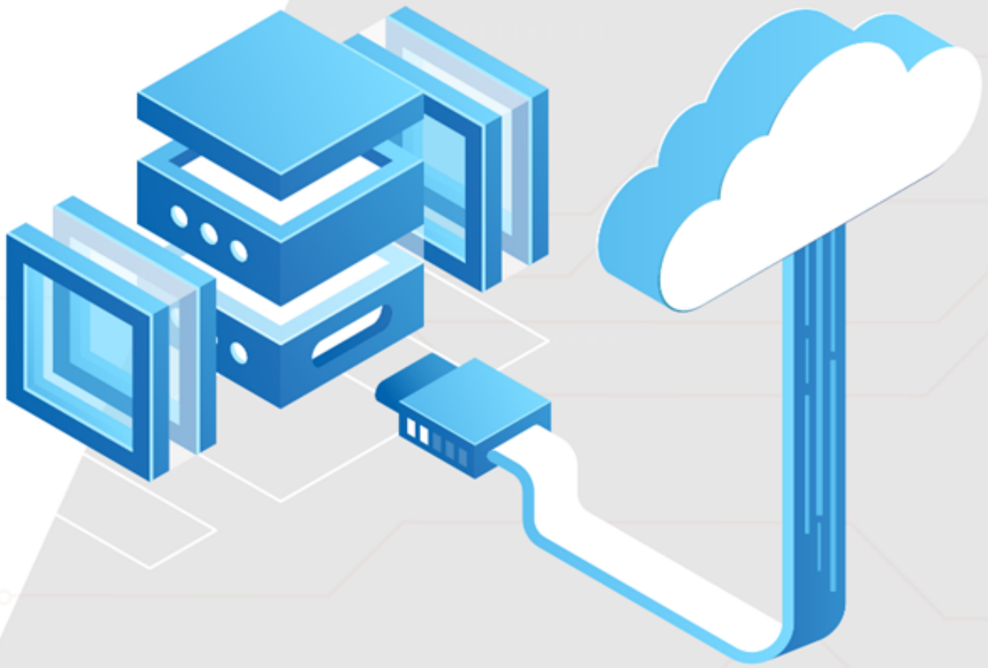
Project agenda: To create an Apache server on Ubuntu using Ansible Roles

Description: Create a playbook that automates setting up a remote Nginx server to host a static HTML website on Ubuntu 20.04.

Perform the following:

- Setting up the HTML files
- Creating a playbook
- Executing the playbook

LESSON-END PROJECT



Thank you