Exploring Ansible Galaxy

Welcome to our second chapter. Here, we are going to be looking at the ansible-galaxy command; we are going to be covering the features provided by the command and also discussing its importance in the development of Ansible over the last few years.

**Ansible Galaxy** is an online repository of community-contributed roles; we will discover some of the best roles available, how to use them, and how to create your own role and have it hosted on Ansible Galaxy.

By the end of the chapter, we will have worked through the following topics:

* The Ansible release life cycle
* Introduction to Ansible Galaxy
* What is a role?
* Publishing to and using Ansible Galaxy roles
* Ansible collections
* Ansible Galaxy commands

Before we start exploring Ansible Galaxy, let’s discuss the Ansible core release life cycle and how it has changed over the last few years, as these changes have made it an essential component of the Ansible ecosystem.

Technical requirements

In this chapter, we will again make use of **Multipass**, the tool that we covered in [*Chapter 1*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/1), *Installing and Running Ansible*, and the GitHub repository that accompanies this title, which can be found at https://github.com/PacktPublishing/Learn-Ansible-Second-Edition.

The Ansible release life cycle

In the previous chapter, when we installed Ansible, the keen-eyed among you may have spotted a few different Ansible packages installed using the sudo -H pip install ansible command.

What follows is an edited version of the output of installing Ansible using pip:

$ sudo -H pip install ansible

Collecting ansible

  Downloading ansible-8.2.0-py3-none-any.whl (45.1 MB)

Collecting ansible-core~=2.15.2

  Downloading ansible\_core-2.15.2-py3-none-any.whl (2.2 MB)

Installing collected packages: resolvelib, packaging, ansible-core, ansible

Successfully installed ansible-8.2.0 ansible-core-2.15.2 packaging-23.1 resolvelib-1.0.1

CopyExplain

As you can see, two main Ansible packages were installed: ansible-8.2.0 and ansible-core-2.15.2. Before we discuss the difference between the two packages, let’s quickly discuss how Ansible was maintained and packaged up until version 2.9 of Ansible.

Every version of Ansible before version 2.10 shipped with many modules baked into the release; while Ansible was new and its user base and functionality were focused on a few tasks, it was easy to manage and maintain the releases of these modules as part of the main Ansible code base, which was maintained by the Ansible team at the official GitHub repo, which you can find at https://github.com/ansible/ansible.

By the end of [*Chapter 1*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/1), *Installing and Running Ansible*, we had used a total of five modules, all of which are built into Ansible; these were the following:

* ansible.builtin.setup: A module that discovers information on the target host and makes it available during the playbook run
* ansible.builtin.service: A module that manages the state of a service on the target host
* ansible.builtin.debug: This module allows you to print statements during your playbook execution
* ansible.builtin.apt: This module manages packages on the target host using the apt package manager
* ansible.builtin.template: This module brings templating to Ansible, allowing you to output files to your target hosts

That was just a single playbook that did a single task; add to that the sheer number of modules to give you an idea of the numbers—there are currently 95 modules in the Amazon AWS namespace and 282 in the Microsoft Azure namespace—that’s over 370 modules that cover the basic functionality of two different namespaces. There are, at the time of writing, over 40 different namespaces.

You might be asking, *“Wait, what is a namespace?”* The modules are now grouped into collections, and each collection has its namespace; some examples are the following:

* **ansible.builtin**: As you might have already guessed, the modules within this namespace provide some of Ansible’s core functionality
* **amazon.aws**: These are the official Amazon Web Services modules
* **azure.azcollection**: Here, you will find the official Microsoft Azure modules
* **kubernetes.core**: If you want to work with Kubernetes, these will be your modules

Again, you might be thinking, *“That’s useful information, but what has this got to do with anything?”*; well, Ansible used to have a few major releases a year and had to ship a release that had slowly grown to include thousands of modules, plus their associated plugins meant that the release process quickly became unmanageable, as the team not only had to worry about the core Ansible code base but also about the modules that shipped with it.

Each namespace potentially has its dev team comprised of core Ansible contributors, community members, and, in the case of some namespaces, large corporations such as Amazon and Microsoft. Thus, trying to coordinate an Ansible release became a challenge, both in terms of logistics and timing. Some of the technology that Ansible supports changes very fast; for example, both Amazon Web Services and Microsoft Azure introduce new features and add functionality to existing services almost weekly. It didn’t make sense for Ansible to potentially wait up to six months to provide an update that adds compatibility issues, which is why the Ansible team decided it was time to decouple the release of what is now known as **Ansible Core**, which is the tooling needed to run **Ansible**; this now comprises over 85 name namespace collections, which are made up of well over 1,000 modules and plugins.

The life cycle of a release

The release cycle begins with introducing a new major version of ansible-core, such as ansible-core 2.11. Following this, the latest release of ansible-core and its two preceding versions, ansible-base 2.10 and Ansible 2.9, are actively maintained. Development then shifts to the **devel** branch, where new features for ansible-core are continuously worked on. During this phase, there’s a freeze on adding or updating **collections** in the **Ansible community** package.

Subsequently, a release candidate for the **Ansible community** package is introduced. This undergoes testing, and additional release candidates are rolled out if necessary.

Once finalized, a new major version of the **Ansible community** package is released, which aligns with the recent ansible-core; for instance, Ansible 4.0.0 would be based on ansible-core 2.11. After this release, only the latest version of the **Ansible community** package remains under active maintenance.

The focus then shifts to **collections**, where new features are developed. Individual **collections** instances have the flexibility to introduce several minor and major versions. On a regular schedule, minor updates are released every four weeks for the three supported ansible-core versions, such as 2.11.1, and for the single supported version of the **Ansible community** package, such as 4.1.0.

As the cycle progresses, there’s a feature freeze on ansible-core. This is followed by introducing a release candidate for ansible-core, which undergoes testing. If needed, more release candidates are introduced. Finally, the subsequent major version of ansible-core is released, marking the commencement of a new cycle.

The following chart provides an overview of the cycle:

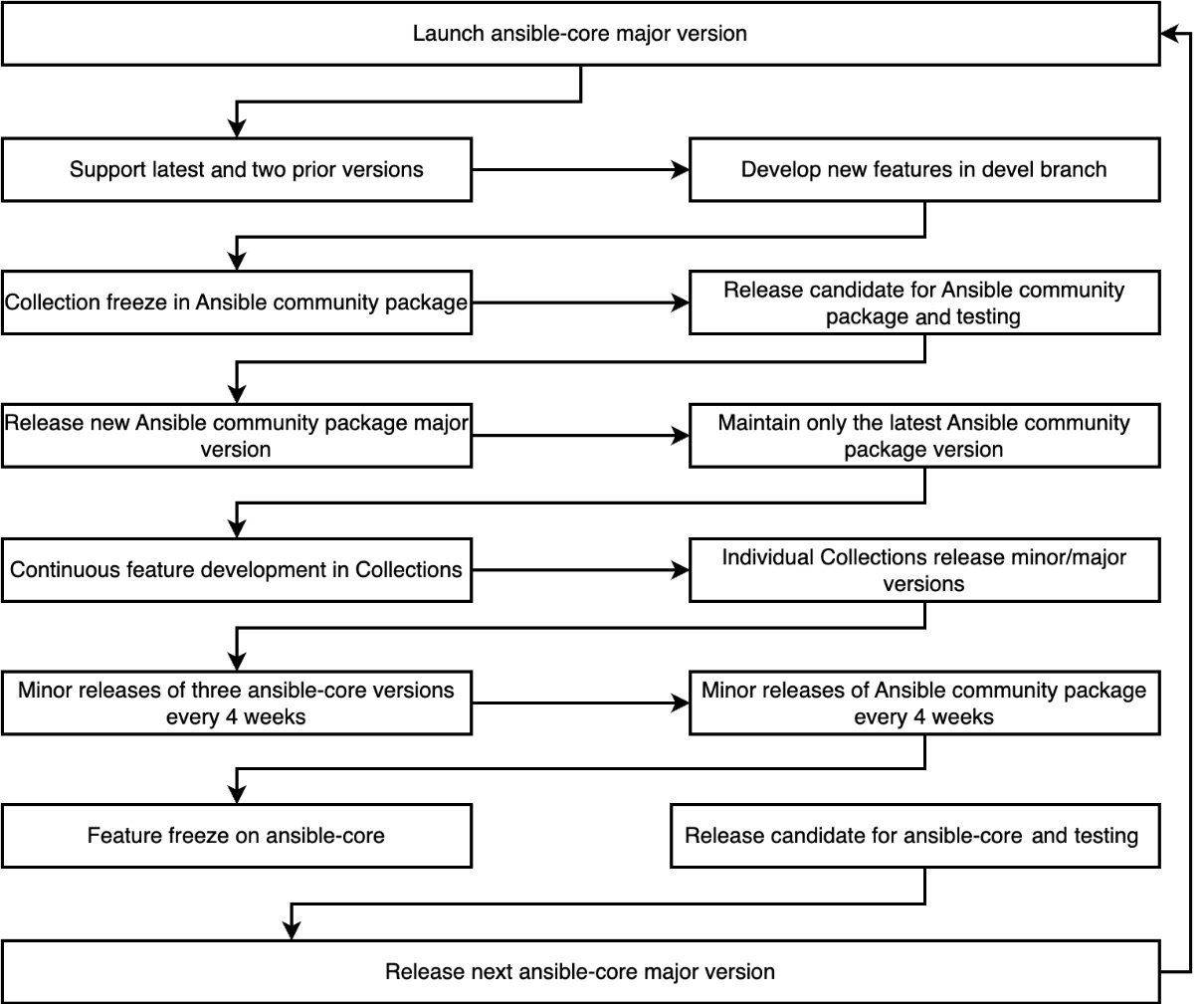


Figure 2.1 – An overview of the Ansible release cycle

As you can see, this approach allows the Ansible team to be a lot more flexible with their release schedules and allows a lot more concurrent work on the two different releases.

Now that we have an idea of how Ansible manages its release cycle and also how modules can be packaged, let’s take a look at Ansible Galaxy, which can be used to distribute collections and roles.

Introduction to Ansible Galaxy

Most people’s first exposure to Ansible Galaxy is the website hosted at https://galaxy.ansible.com/. The website is home to community-contributed roles and modules:

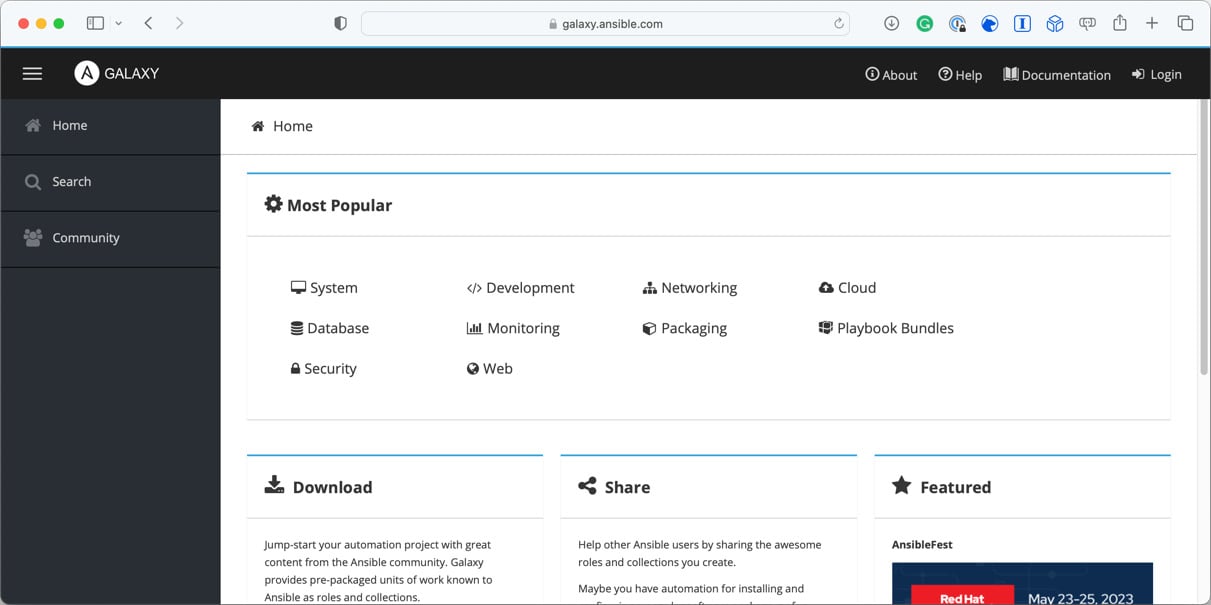


Figure 2.2 – The Ansible Galaxy home page

Throughout the remainder of this book, we will be writing custom roles that interact with the Ansible Core modules for use in our playbook.

More than 15,000 roles are published on Ansible Galaxy; these roles cover many tasks and support almost all the operating systems supported by Ansible.

Then, we have the ansible-galaxy command; this is a way of interacting with the Ansible Galaxy website from the comfort of your command line, as well as being able to bootstrap roles, which we will look at shortly; we can also use it to download, search, and publish our custom roles on Ansible Galaxy.

Finally, Red Hat has open-sourced the code for Ansible Galaxy, meaning you can also run a self-hosted version of the site should you need to distribute your roles behind a company firewall.

Before we look at publishing roles and using roles from Ansible Galaxy, let’s discuss what a role means.

What is a role?

Throughout the remainder of this book, we will be building custom roles of our own, so this will be just an overview of what a role is.

In [*Chapter 1*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/1), *Installing and Running Ansible*, our final playbook comprised some variables, a handler, four tasks, and a template.

Apart from the template file, all of the code was hardcoded into our playbook file, which, although this makes it easy to read when using a small number of tasks and variables, etc., taking this approach doesn’t make the code very re-useable. Additionally, in later chapters, we could potentially be executing over 50 tasks in a single playbook run, which will make for quite a large, unruly file.

To get around this, Ansible has the concept of roles; they allow you to structure your Ansible code in a way that makes sense logically, for example, grouping tasks that perform a single job, which, in [*Chapter 1*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/1), *Installing and Running Ansible*, was installing and configuring the NTPD service.

It also means you can drop a role into another playbook by copying the role folder, publishing it, and then pulling it down from Ansible Galaxy.

So, let’s look at creating a basic role based on the tasks, handler, variables, and template from the final playbook we ran at the end of [*Chapter 1*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/1), *Installing and Running Ansible*.

To start with, we will need to create the folder and file structure recommended by Ansible for a role; luckily, the ansible-galaxy command has us covered here; by running the following command in the folder where your playbook is going to be stored, it will bootstrap the folder and file structure, which is considered by Red Hat to be a best practice:

$ ansible-galaxy role init roles/learnansible-example-role

CopyExplain

The preceding command will create a folder called roles if one doesn’t already exist, and inside the roles folder, add a second called learnansible-example-role.

The learnansible-example-role folder contains all the best-practice folder layouts and files needed to be able to publish a role on Ansible Galaxy.

These are as follows:

* README.md: This file contains an outline for you to fill in to provide information on your role; you can use as much or as little as you want of the template. Please note its contents will appear on Ansible Galaxy if you decide to publish your role there, so make it as descriptive as possible.
* defaults/main.yml: This YAML file typically contains any default values for your role.
* files/: This empty folder holds any files that need to be copied for your role to the target hosts during playbook execution.
* handlers/main.yml: As you may have already guessed by the name of this folder, this YAML file is where you define any handlers your role needs.
* meta/main.yml: This YAML file, like the README.md file, is only used once the role is published to Ansible Galaxy; here, you can provide your details, any tags you want to add, and define the supported platform and the minimum version of Ansible version your role support.
* tasks/main.yml: This is the file we will spend most of the time in throughout the rest of the chapters; it is where all of the roles’ tasks are defined.
* templates/: This is another empty folder; this time, it is here to store your template files.
* tests/inventory and test.yml: Here, we have a file that contains two files, an inventory file and a test playbook; it is used to run tests on your role.
* vars/main.yml: Finally, this YAML file contains any variables that you may wish to use, and these override the contents of the defaults/main.yml file, should you need to do so.

To populate the role, I have taken the code from the final playbook and split it across the aforementioned various files; the only change that I made to the playbook itself was to remove the following task, as we don’t need it:

    - name: "Output some information on our host"

      ansible.builtin.debug:

        msg: "I am connecting to {{ ansible\_nodename }} which is running {{ ansible\_distribution }} {{ ansible\_distribution\_version }}"

CopyExplain

This leaves the roles/learnansible-example-role/tasks/main.yml file looking like the following code:

# tasks file for roles/learnansible-example-role

- name: "Update all packages to the latest version"

  ansible.builtin.apt:

    name: "\*"

    state: "latest"

    update\_cache: true

  tags:

    - "skip\_ansible\_lint"

- name: "Install packages"

  ansible.builtin.apt:

    state: "present"

    pkg:

      - "ntp"

      - "sntp"

      - "ntp-doc"

- name: "Configure NTP"

  ansible.builtin.template:

    src: "./ntp.conf.j2"

    dest: "/etc/ntp.conf"

    mode: "0644"

  notify: "Restart ntp"

CopyExplain

Notice that, as mentioned in [*Chapter 1*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/1), *Installing and Running Ansible*, we have --- at the top of the file to show that main.yml is a separate file. As it is in the tasks folder, we do not need to define that it contains tasks by using tasks, as we did in our original playbook.

This pattern is followed by the roles/learnansible-example-role/handlers/main.yml file, which looks like this:

# handlers file for roles/learnansible-example-role

- name: "Restart ntp"

  ansible.builtin.service:

    name: "ntp"

    state: "restarted"

CopyExplain

Additionally, this is followed by the roles/learnansible-example-role/vars/main.yml file, which contains the following:

---

# vars file for roles/learnansible-example-role

ntp\_servers:

  - "0.uk.pool.ntp.org"

  - "1.uk.pool.ntp.org"

  - "2.uk.pool.ntp.org"

  - "3.uk.pool.ntp.org"

CopyExplain

The roles/learnansible-example-role/vars/ntp.conf.j2 file is an exact copy of the template file we used in [*Chapter 1*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/1), *Installing and Running Ansible*.

The only addition outside of the README.md file is roles/learnansible-example-role/meta/main.yml. This file, as mentioned, contains all of the information needed to publish the role to Ansible Galaxy; in our example, this looks like the following:

galaxy\_info:

  role\_name: "ansible\_role\_learnansible\_example"

  namespace: "russmckendrick"

  author: "Russ McKendrick"

  description: "Example role to accompany Learn Ansible (Second Edition)"

  issue\_tracker\_url: "https://github.com/russmckendrick/ansible-role-learnansible-example/issues"

  license: "license (BSD-3-Clause)"

  min\_ansible\_version: "2.9"

  platforms:

    - name: "Ubuntu"

      versions:

        - "jammy"

  galaxy\_tags:

    - "ntp"

    - "time"

    - "example"

dependencies: []

CopyExplain

We will revisit this file in the next section of this chapter when it comes to publishing our role to Ansible Galaxy.

Now that we have everything that we need to run the role in place, we need a playbook to call it; in the Chapter02 folder for the repo, which accompanies this title, you will file the roles folder, as described earlier, and a playbook called playbook01.yml, which looks like the following:

- name: "Run the role locally"

  hosts: "ansible\_hosts"

  gather\_facts: true

  become: true

  become\_method: "ansible.builtin.sudo"

  roles:

    - learnansible\_example\_role

CopyExplain

As you can see, the start of the playbook looks precisely the same as the one we ran at the end of [*Chapter 1*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/1), *Installing and Running Ansible*. However, it is missing the vars, handlers, and tasks sections, and instead, we are just using a roles section, which contains the single role found at roles/learnansible-example-role.

In the Chapter02 folder, you will find all the files needed to launch a local virtual machine using Multipass.

Important note

When running the following commands, create a copy of the hosts.example file and call it hosts; once copied, update the newly created file with the IP address of the newly launched VM, as we did in [*Chapter 1*](https://subscription.packtpub.com/book/cloud-and-networking/9781835088913/1), *Installing and Running Ansible*.

To launch the virtual machine, get its IP address and run the playbook using the following commands:

$ multipass launch -n ansiblevm --cloud-init cloud-init.yaml

$ multipass info ansiblevm

$ ansible-playbook -i hosts playbook01.yml

CopyExplain

This should give you something like the following output:

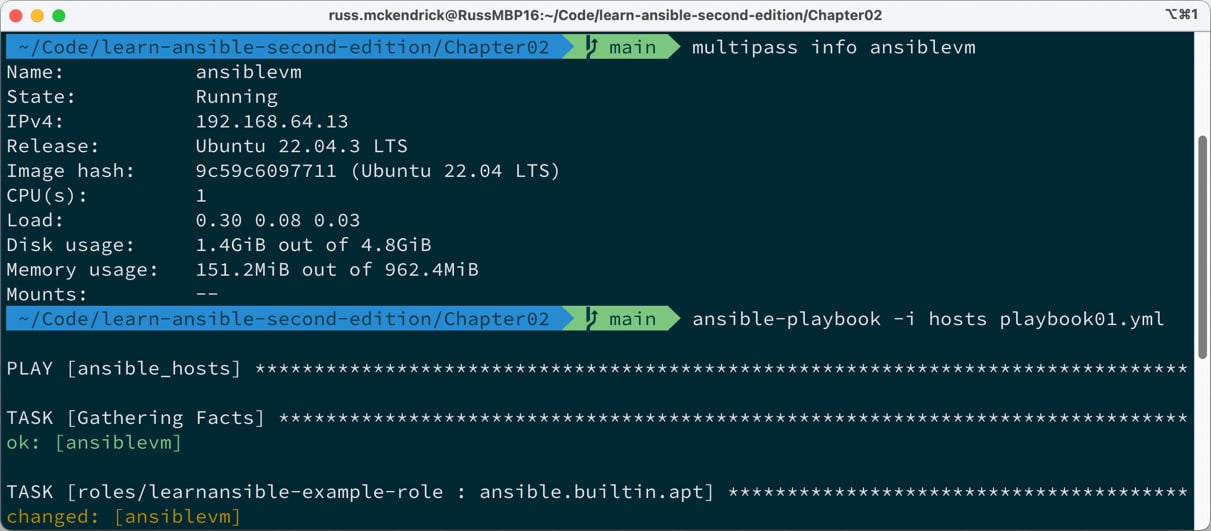


Figure 2.3 – Running the playbook using the newly created role

You can stop and delete the VM by using the following two commands:

$ multipass stop ansiblevm

$ multipass delete --purge ansiblevm

CopyExplain

Now that we know what a role is and have a fundamental one defined, let’s look at publishing our simple role to Ansible Galaxy and then use it—along with some others—in an Ansible playbook.

Publishing to and using Ansible Galaxy roles

Now that we know what a role is and have seen how using them can make our Ansible playbooks a little cleaner and repeatable, we should look at how we can publish our roles to Ansible Galaxy and use them from there in our playbooks.

Publishing your roles to Ansible Galaxy

You need two main prerequisites when publishing your role to Ansible Galaxy: an active GitHub account, which will be used to authenticate to Ansible Galaxy, and a public GitHub repository containing the code for your role.

In this example, I am going to be using my own GitHub account; you can find me at http://github.com/russmckendrick/. I will be using a repository that can be found at https://github.com/russmckendrick/ansible-role-learnansible-example/.

To publish your role, you need to take the following steps:

1. Go to the Ansible Galaxy website, which can be found at https://galaxy.ansible.com/, and click on the GitHub logo to log in:

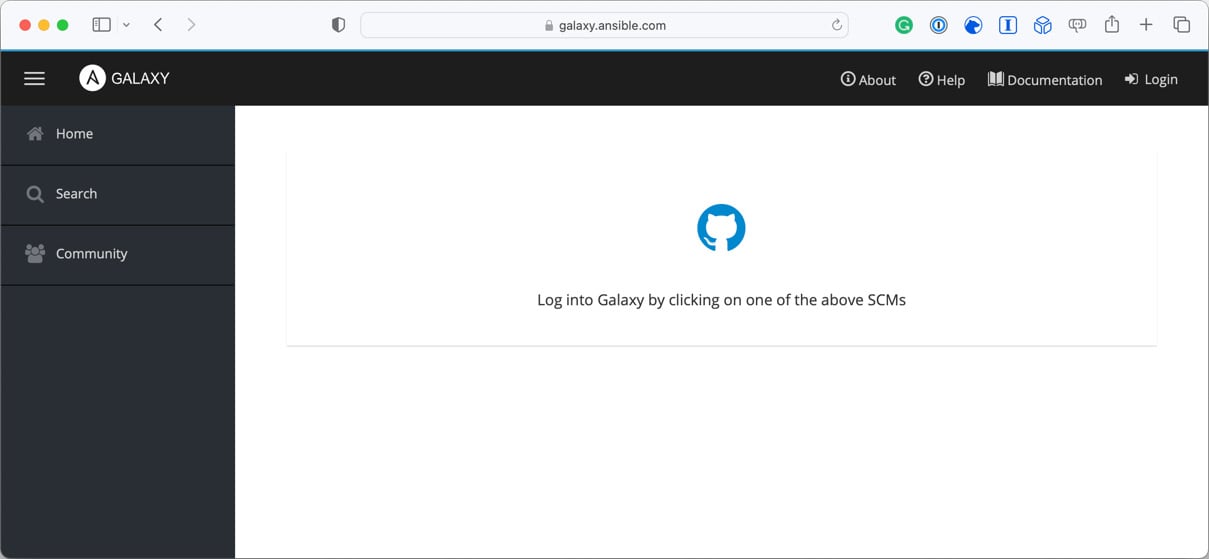


Figure 2.4 – Logging in to Ansible Galaxy

1. Once logged in, click on the **My Content** menu item, which is represented by the bulleted pointed list icon in the left-hand side menu as follows:

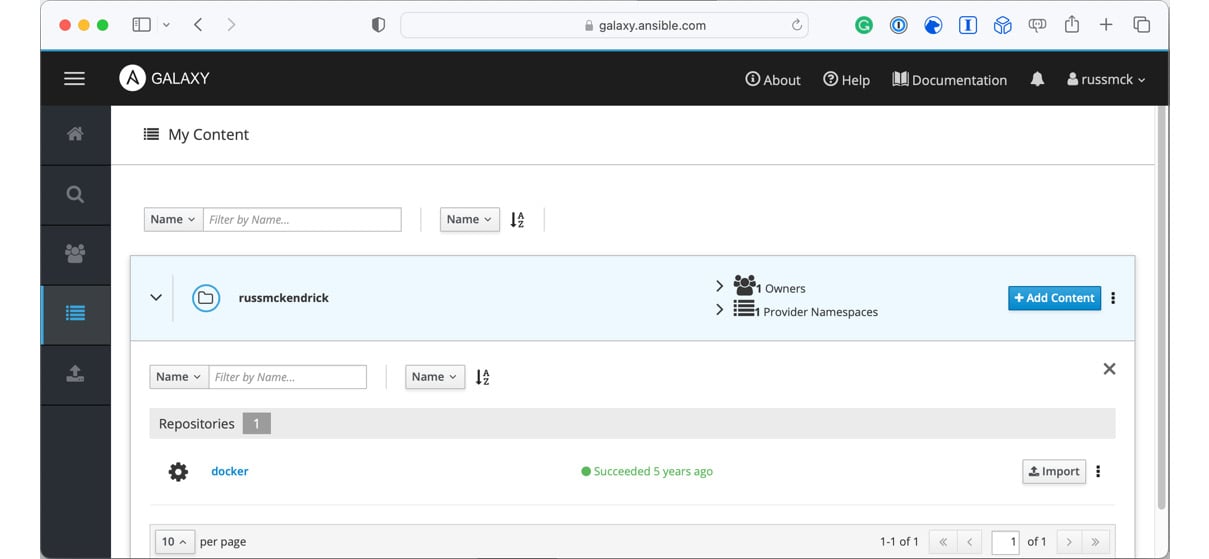


Figure 2.5 – Going to the My Content page

1. Once on the **My Content** page, click on the **+ Add Content** button; here, you will be given two options: **Import Role from GitHub** or **Upload New Collection**:

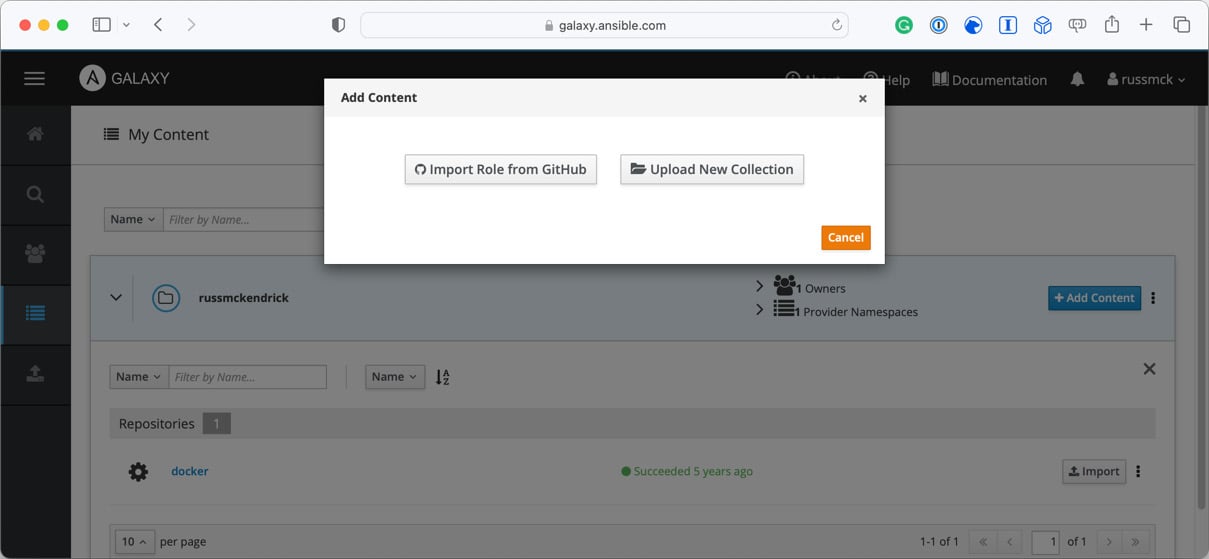


Figure 2.6 – Adding content options

1. Click on the **Import Role from GitHub** button, and you will be presented with a list of your repositories; select the repository containing the role you would like to publish and click on the **OK** button:

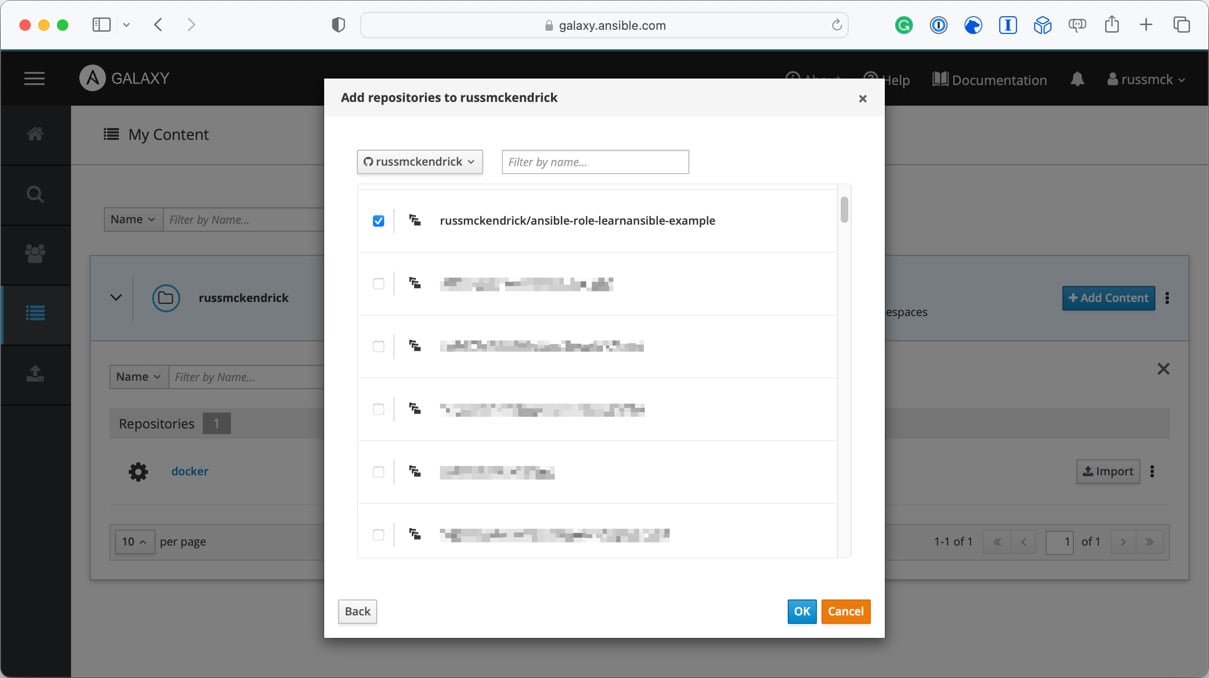


Figure 2.7 – Choosing the repository containing the role you want to publish

1. After a few moments, your role will be published, and you will be returned to the **My Content** page, which should now list your newly published role:

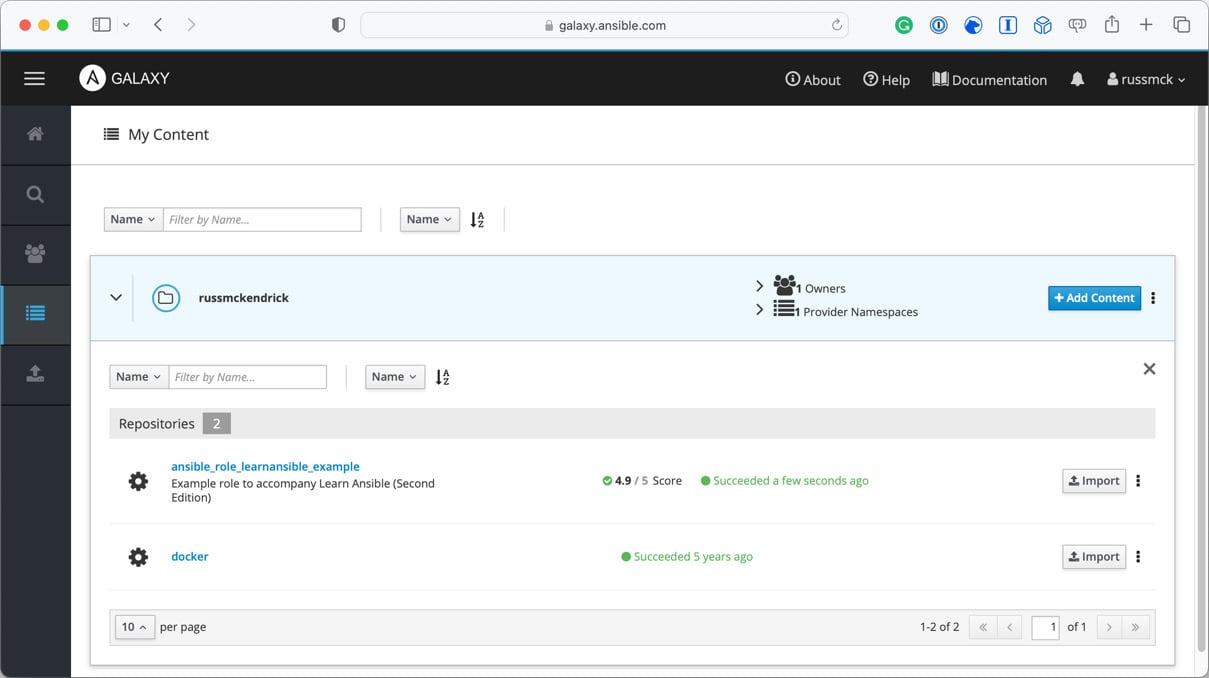


Figure 2.8 – Returning to the My Content page

1. Click on the newly published role name, which will take you to the Ansible Galaxy roles page:

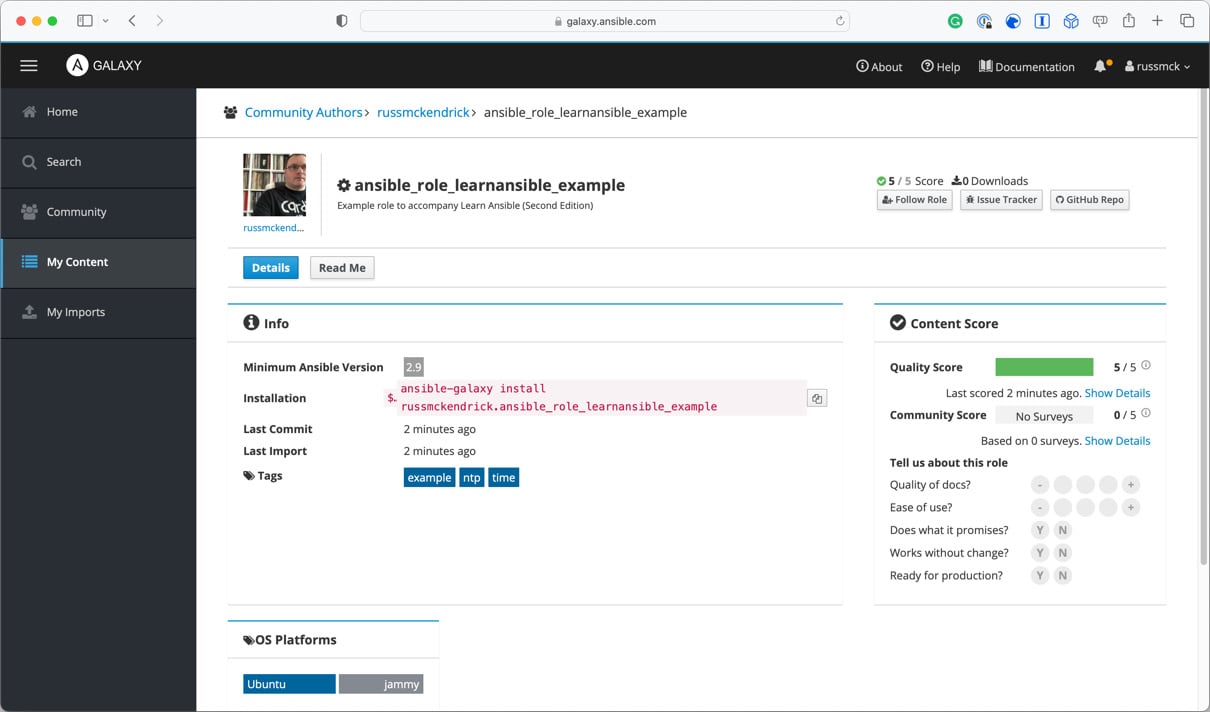


Figure 2.9 – The newly published Ansible Galaxy roles page

You can find a copy of the role I published during this walk-through at https://galaxy.ansible.com/russmckendrick/ansible\_role\_learnansible\_example.

As you can see, the details in the **Info** section of the Ansible Galaxy roles page contain the information we defined in the meta/main.yml file and clicking on the **Read Me** button on the page will display the rendered contents of the README.md file.

Now that the role has been published, how do we use it in our Ansible playbooks? Let’s find out.

Using roles from Ansible Galaxy

The first thing we need to do before we use the role in our playbook is to download the role; there are a few ways of doing this; first, you can download the role using the command given on the roles Ansible Galaxy page.

Running the following command will download the role to your Ansible configuration directory:

$ ansible-galaxy install russmckendrick.ansible\_role\_learnansible\_example

CopyExplain

The Ansible configuration directory is typically a hidden folder in your user’s home folder. The shorthand for this folder is ~/.ansible, or in my case, the full path to the folder is /Users/russ.mckendrick/.ansible, as you can see in the following shell output:

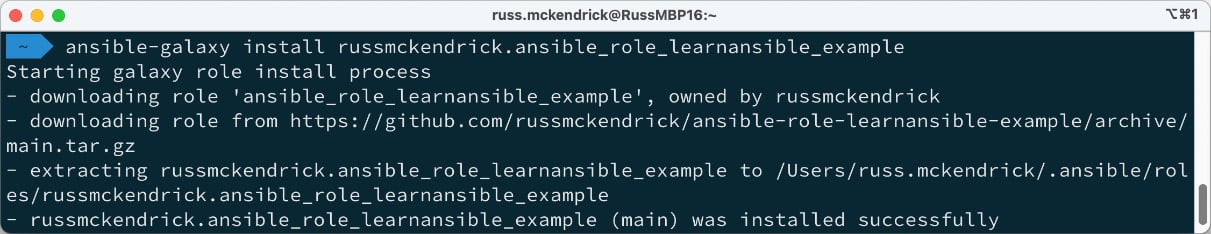


Figure 2.10 – Downloading the role from Ansible Galaxy

The second way to download roles from Ansible Galaxy is to create a requirements.yml file; this file should contain a list of the roles you wish to download, for example, the requirements.yml file in the Chapter02 folder of the repository, which accompanies this book and looks like the following:

- src: "itnok.update\_ubuntu"

- src: "geerlingguy.nginx"

- src: "russmckendrick.ansible\_role\_learnansible\_example"

CopyExplain

As you can see, there are three roles defined in there; to install all three, you can run the following command:

$ ansible-galaxy install -r requirements.yml

CopyExplain

The two others we will download are the following:

* itnok.update\_ubuntu: This role manages updates on Ubuntu hosts
* geerlingguy.nginx: This role helps you download, install, and configure NGINX on multiple Linux distributions

You can find links for the roles in the further reading section at the end of the chapter.

This will download only the missing roles; when I ran the command, I got the following output:

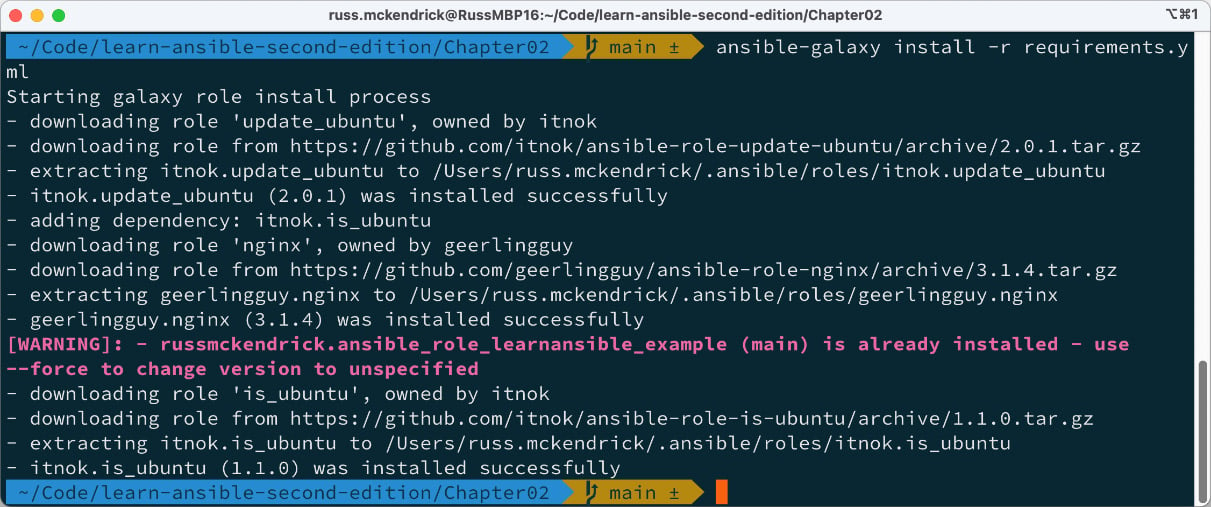


Figure 2.11 – Downloading the missing roles from Ansible Galaxy

As you can see, as russmckendrick.ansible\_role\_learnansible\_example was already present on my machine, it skipped downloading it.

The Ansible playbook called playbook02.yml, which can be found in the Chapter02 folder, calls the three roles defined in the requirements.yml file using the following code:

---

- name: "Run the remote roles"

  hosts: "ansible\_hosts"

  gather\_facts: true

  become: true

  become\_method: "ansible.builtin.sudo"

  roles:

    - "itnok.update\_ubuntu"

    - "geerlingguy.nginx"

    - "russmckendrick.ansible\_role\_learnansible\_example"

CopyExplain

As before, you can launch a VM using Multipass (ensuring that you update the IP address in the hosts file) and run the playbook using the following commands:

$ multipass launch -n ansiblevm --cloud-init cloud-init.yaml

$ multipass info ansiblevm

$ ansible-playbook -i hosts playbook02.yml

CopyExplain

As you can see from the playbook recap in the following screen, a lot more happened this time:

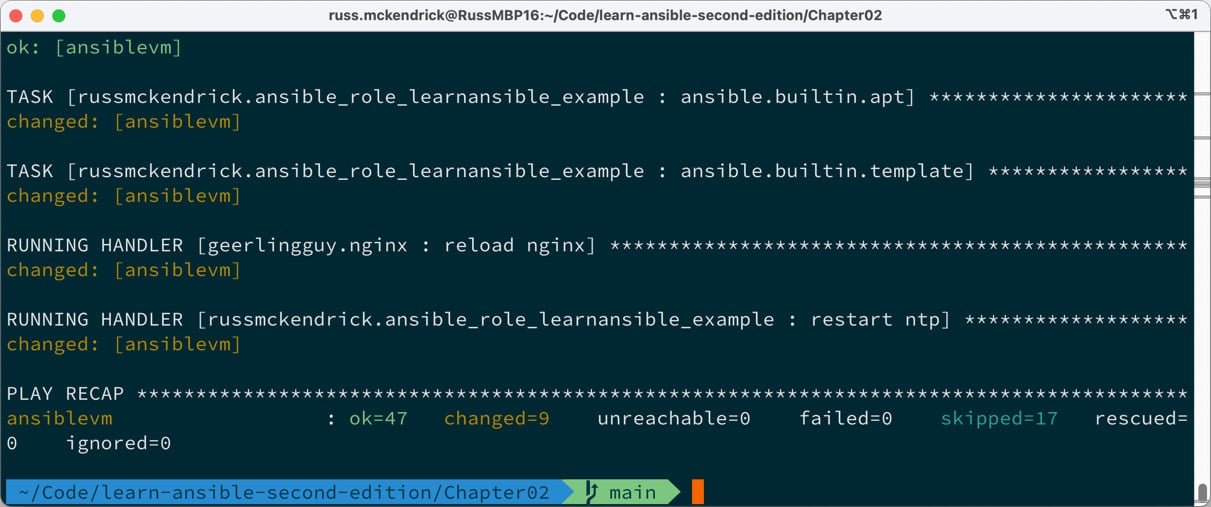


Figure 2.12 – Running the playbook, which uses the roles downloaded from Ansible Galaxy

The two additional roles did a more thorough update of the operating system than our role did, installing the NGINX package and starting the service up; this means that if you put the IP address returned by the multipass info ansiblevm command into your browser, you can see the default NGINX page.

Once again, when ready, you can stop and delete the VM by using the following two commands:

$ multipass stop ansiblevm

$ multipass delete --purge ansiblevm

CopyExplain

So now that you understand what an Ansible Role is, how to publish it to Ansible Galaxy, and how to incorporate our own published and community roles into an Ansible playbook, what else can Ansible Galaxy do?

Ansible collections

At the start of this chapter, we discussed how the Ansible development team decoupled Ansible Modules away from Ansible Core and how this affected the release life cycle.

All of these modules, plugins, and other supporting code are available on Ansible Galaxy; for example, the AWS collection from the Amazon namespace can be found at https://galaxy.ansible.com/amazon/aws; you can install the collection using the following command:

$ ansible-galaxy collection install amazon.aws

CopyExplain

Running this command will download and install the collection to ~/.ansible/collections/ansible\_collections/amazon/aws, as seen in the following terminal output:

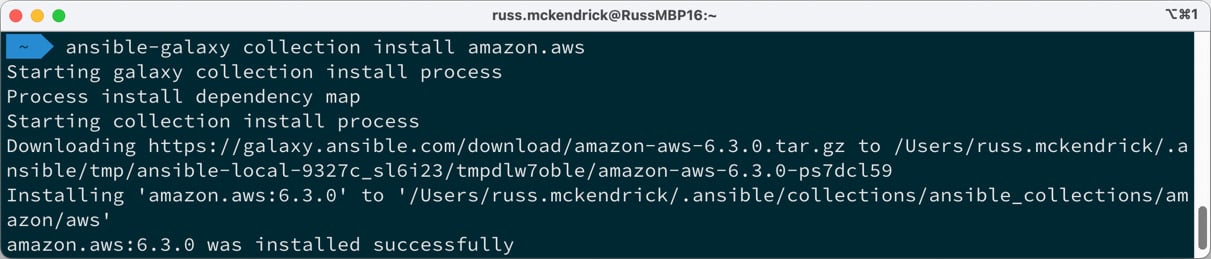


Figure 2.13 – Installing the amazon.aws collection

However, just having the collection installed doesn’t mean that you will be able to use it within your playbooks just yet; for example, the Amazon AWS module requires some additional Python libraries to be installed, and typically, each collection will come with a requirements.txt file that lists the required Python libraries that need to be installed on your system for the collections modules and plugins to work.

To install these libraries, you should use pip to install them:

$ pip install -r ~/.ansible/collections/ansible\_collections/amazon/aws/requirements.txt

CopyExplain

Once installed, you will be able to use the modules and plugins which make up the collection.

Ansible Galaxy commands

Before finishing this chapter on Ansible Galaxy, let’s quickly discuss some other useful commands.

The ansible-galaxy command has some of the basic functionality you would expect, such as the following:

$ ansible-galaxy --version

$ ansible-galaxy --help

CopyExplain

This displays details of the version and basic help options on the command, respectively.

The ansible-galaxy command is split into two parts, which we have already touched upon.

First, there is ansible-galaxy collection; from here, you can add the following commands:

* download: Retrieves collections and their dependencies, such as **tarballs**, which is an archive format for Linux machines for offline installations.
* init: Set up a new collection with the foundational structure.
* build: Construct an Ansible collection artifact suitable for publication to Ansible Galaxy.
* publish: Release a collection artifact to Ansible Galaxy.
* install: Add collection(s) from the specified file(s), URL(s), or directly from Ansible Galaxy.
* list: Display the name and version of every collection in the collections path.
* verify: Contrast the checksums of the installed collection(s) with those on the server; any dependencies are not verified.

Secondly, there is ansible-galaxy role, as I am sure you will have already guessed; these commands are for working with roles:

* init: Set up a new role with the foundational structure of a role
* remove: Erase roles from the specified roles path
* delete: Remove the role from Galaxy. Note that this does not affect or modify the actual GitHub repository
* list: Display the name and version of every role in the roles path
* search: Query the Galaxy database using tags, platforms, authors, and keywords
* import: Incorporate a role into a Galaxy server
* setup: Oversee the connection between Galaxy and the designated source
* info: Obtain detailed information about a particular role
* install: Add role(s) from specified file(s), URL(s), or directly from Ansible Galaxy

For more help on any of these commands, you can append --help to the end. For example, you can run the following:

$ ansible-galaxy role search --help

CopyExplain

This will give you detailed information on how to search Ansible Galaxy; for example, to search for my roles, you would need to run the following:

$ ansible-galaxy role search --author russmckendrick

CopyExplain

This returns a list of all of the roles I have published to Ansible Galaxy, or you could run this:

$ ansible-galaxy role list

CopyExplain

This lists all the roles you have installed in the roles path (which is ~/.ansible/roles/). From there, you could run something like the following:

$ ansible-galaxy role info russmckendrick.ansible\_role\_learnansible\_example

CopyExplain

This helps you obtain information on a role you have installed, which concludes our look at the ansible-galaxy command.

Summary

In this chapter, we have had an in-depth look at Ansible Galaxy, the website, and the command-line tool. We have also discussed the Ansible development and release cycle and understood what an Ansible Role is.

I am sure that you will agree that Ansible Galaxy offers valuable community services in that it allows users to share roles for everyday tasks and provides a way for users to contribute to the Ansible community by publishing their roles.

However, just be careful. Remember to check the code and read through bug trackers before using roles from Ansible Galaxy in production environments; after all, many roles need escalated privileges to execute their tasks successfully.

As mentioned in this chapter, we will be creating our own Ansible Roles throughout the remainder of this title, and there will be additional hints and tips on creating and using roles as our Ansible playbooks get more and more sophisticated.

In our next chapter, we will look at more Ansible commands and tools that ship as part of Ansible Core.