# Red Hat Enterprise Linux Automation with Ansible

FEEDBACK TRANSLATIONS CERTIFICATE OF ATTENDANCE

# Getting Roles and Modules from Content Collections ☆

## Objectives

After completing this section, you should be able to obtain a set of related roles, supplementary modules, and other content from content collections, and use them in a playbook.

## Discussing Content Collections

*Ansible content collections* are a distribution format for Ansible content. A collection provides a set of related modules, roles, and plug-ins that you can download to your control node and then use in your playbooks.

For example:

- The `redhat.insights` collection groups modules and roles that you can use to register a system with Red Hat Insights for Red Hat Enterprise Linux.
- The `cisco.ios` collection groups modules and plug-ins that manage Cisco IOS network appliances. The Cisco company supports and maintains that collection.
- The `community.crypto` collection provides modules that create SSL/TLS certificates.

Content collections allow updates to the core Ansible code to be separated from updates to modules and plug-ins. This allows vendors and developers to maintain and distribute their collections at their own pace, independently of Ansible releases. You can develop your own collections to provide custom roles and modules to your teams.

Content collections also give you more flexibility. By using collections, you can install only the content you need instead of installing all supported modules. You can also select a specific version of a collection (possibly an earlier or later one) or choose between a version of a collection supported by Red Hat or vendors or one provided by the community.

Ansible 2.9 and later support collections. Later versions of Ansible and Red Hat Ansible Automation Platform will provide further enhancements to collection support and will use them extensively. Understanding how collections work will be important.

The `ansible` RPM package provided with Red Hat Ansible Automation Platform 1.2 and Ansible 2.9 automatically installs all the modules that earlier versions of Ansible did. Future versions of Ansible and Red Hat Ansible Automation Platform will remove most modules from the main RPM package and place them in collections that might be included or that you might have to download.

### Organizing Collections in Namespaces

To make it easier to specify collections and their contents by name, collection names are organized into *namespaces*. Vendors, partners, developers, and content creators can use namespaces to assign unique names to their collections without conflicting with other developers.

The namespace is the first part of a collection name. For example, all the collections that the Ansible community maintains are in the `community` namespace, and have names like `community.crypto`, `community.postgresql`, and `community.rabbitmq`. Collections that Red Hat maintains and supports might use the `redhat` namespace, and have names like `redhat.rhv`, `redhat.satellite`, and `redhat.insights`.

### Selecting Collection Sources

Ansible provides two official sources to download and install collections: Ansible automation hub and Ansible Galaxy.

**Ansible automation hub**
Ansible automation hub hosts Ansible content collections that Red Hat and its partners support for their customers. Red Hat reviews, maintains, updates, and fully supports those collections. For example, the `redhat.rhv`, `redhat.satellite`, `redhat.insights`, and `cisco.ios` collections are available on that platform.

You need a valid Red Hat Ansible Automation Platform subscription to access Ansible automation hub. Use the Ansible automation hub web UI at https://cloud.redhat.com/ansible/automation-hub/ (https://cloud.redhat.com/ansible/automation-hub/) to list and access the collections.

**Ansible Galaxy**
Ansible Galaxy hosts collections that have been submitted by a variety of Ansible developers and users. Ansible Galaxy is a public library with no formal support guarantees, but which allows public access. For example, the `community.crypto`, `community.postgresql`, and `community.rabbitmq` collections are all available from that platform.

Use the Ansible Galaxy web UI at https://galaxy.ansible.com/ (https://galaxy.ansible.com/) to search it for collections.

## Installing Content Collections

Before your playbooks can use content from a collection, you must install that collection on your control node. Use the `ansible-galaxy` command to download collections from a number of possible sources, including Ansible Galaxy.

The following example uses the `ansible-galaxy` command with the `collection` argument to download and then install the `community.crypto` collection on the local system.

```
[user@controlnode ~]$ ansible-galaxy collection install community.crypto
```

The command can also install a collection from a local or a remote tar archive.

```
[user@controlnode ~]$ ansible-galaxy collection install \
> /tmp/community-dns-1.2.0.tar.gz
[user@controlnode ~]$ ansible-galaxy collection install \
> http://www.example.com/redhat-insights-1.0.5.tar.gz
```

The Ansible configuration directive `collections_paths` specifies a colon-separated list of paths on the system where Ansible looks for installed collections. You can set this directive in the `ansible.cfg` configuration file. By default, the `ansible-galaxy` command installs the collections in the first directory that the `collections_paths` directive defines.

The default value for `collections_paths` is `~/.ansible/collections:/usr/share/ansible/collections`. Therefore, the `ansible-galaxy` command installs collections in the `~/.ansible/collections` directory by default.

If you want to install a collection in a different directory, use the `--collections-path` (or `-p`) option.

```
[root@controlnode ~]# ansible-galaxy collection install \
> -p /usr/share/ansible/collections community.postgresql
```

> **IMPORTANT**
>
> When using the `--collections-path` (or `-p`) option, ensure you select a directory listed in the `collections_paths` directive. The `ansible-playbook` command also uses that directive to locate collections. If you do not use a path defined in the `collections_paths` directive, then your playbooks will not find the

collections that you installed.

### Installing Collections with a Requirements File

You can create a `requirements.yml` file to list all the collections that you need to install. By adding a `collections/requirements.yml` file in your Ansible project, your team members can immediately identify the required collections. Also, automation controller detects that file and automatically installs the collections before running your playbooks.

The following `requirements.yml` file lists several collections to install. Notice that you can target a specific collection version and also provide local or remote tar archives.

```
---
collections:
  - name: community.crypto

  - name: ansible.posix
    version: 1.2.0

  - name: /tmp/community-dns-1.2.0.tar.gz

  - name: http://www.example.com/redhat-insights-1.0.5.tar.gz
```

The `ansible-galaxy` command can then use that file to install all those collections. Use the `--requirements-file` (or `-r`) option to provide the `requirements.yml` file to the command.

```
[root@controlnode ~]# ansible-galaxy collection install -r requirements.yml
```

### Configuring Collection Sources

By default, the `ansible-galaxy` command uses Ansible Galaxy at https://galaxy.ansible.com/ (https://galaxy.ansible.com/) to download collections.

For the command to also use Ansible automation hub, add the following directives to the `ansible.cfg` file.

```
...output omitted...
[galaxy]
server_list = automation_hub, galaxy  (1)

[galaxy_server.automation_hub]
url=https://cloud.redhat.com/api/automation-hub/  (2)
auth_url=https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token  (3)
token=eyJh...Jf0o  (4)

[galaxy_server.galaxy]
url=https://galaxy.ansible.com/
```

(1) List all the repositories that the `ansible-galaxy` command must use in order. For each name you define, add a `[galaxy_server.name]` section to provide the connection parameters. Because Ansible automation hub might not provide all the collections that your playbooks need, you can add Ansible Galaxy in the last position as a fallback. This way, if the collection is not available in Ansible automation hub, then the `ansible-galaxy` command uses Ansible Galaxy to retrieve it.

(2) Provide the URL to access the repository.

(3) Provide the URL for authentication.

(4) To access Ansible automation hub, you need an authentication token associated with your account. Use the Ansible automation hub web UI to generate that token. For more details on that process, see the links in the References section.

Instead of a token, you can use the `username` and `password` parameters to provide your customer portal user name and password.

```
...output omitted...
[galaxy_server.automation_hub]
url=https://cloud.redhat.com/api/automation-hub/
username=operator1
password=Sup3r53cR3t
...output omitted...
```

You might not want to expose your credentials in the `ansible.cfg` file because the file could potentially get committed using version control. In that case, remove the authentication parameters from the `ansible.cfg` file and define them as environment variables. You define the environment variables as follows:

```
ANSIBLE_GALAXY_SERVER_<server_id>_<key>=value
```

**`server_id`**
Server identifier in uppercase. The server identifier is the name you use in the `server_list` parameter and in the name of the `[galaxy_server.server_id]` section.

**`key`**
Name of the parameter in uppercase.

The following example provides the `token` parameter as an environment variable:

```
[user@controlnode ~]$ cat ansible.cfg
...output omitted...
[galaxy_server.automation_hub]
url=https://cloud.redhat.com/api/automation-hub/
auth_url=https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token
[user@controlnode ~]$ export \
> ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_TOKEN='eyJh...Jf0o'
[user@controlnode ~]$ ansible-galaxy collection install ansible.posix
```

## Using Collections

After you install a collection, you can use it with ad hoc commands and playbooks. Access the collection documentation from Ansible automation hub or the Ansible Galaxy web UI to retrieve information about the roles and modules it provides. Alternatively, you can inspect the collection directory structure on your system. The collection stores the modules in the `plugins/modules/` directory and the roles in the `roles/` directory.

```
[user@controlnode ~]$ tree \
> ~/.ansible/collections/ansible_collections/redhat/insights/
/home/user/.ansible/collections/ansible_collections/redhat/insights/
...output omitted...
├── plugins
│   ├── action
│   │   └── insights_config.py
│   ├── inventory
│   │   └── insights.py
│   └── modules
│       ├── insights_config.py
│       └── insights_register.py
...output omitted...
├── roles
│   ├── compliance
│   │   ├── meta
│   │   │   └── main.yml
│   │   ├── README.md
│   │   ├── tasks
│   │   │   ├── install.yml
│   │   │   ├── main.yml
│   │   │   └── run.yml
│   │   └── tests
│   │       ├── compliance.yml
│   │       ├── install-only.yml
│   │       └── run-only.yml
│   └── insights_client
...output omitted...
```

To use a module or a role, refer to it with its *fully qualified collection name (FQCN)*. Based on the preceding output, you refer to the `insights_client` role as `redhat.insights.insights_client`.

The following ad hoc command calls the `mail` module from the `community.general` collection.

```
[user@controlnode ~]$ ansible localhost -m community.general.mail \
> -a 'subject="Hello World" to=root'
```

The following playbook invokes the `mysql_user` module from the `community.mysql` collection.

```
---
- name: Create the operator1 user in the test database
  hosts: db.example.com

  tasks:
    - name: Ensure the operator1 database user is defined
      community.mysql.mysql_user:
        name: operator1
        password: Secret0451
        priv: '.:ALL'
        state: present
```

The following playbook uses the `organizations` role from the `redhat.satellite` collection.

```
---
- name: Add the test organizations to Red Hat Satellite
  hosts: localhost

  tasks:
    - name: Ensure the organizations exist
      include_role:
        name: redhat.satellite.organizations
      vars:
        satellite_server_url: https://sat.example.com
        satellite_username: admin
        satellite_password: Sup3r53cr3t
        satellite_organizations:
          - name: test1
            label: tst1
            state: present
            description: Test organization 1
          - name: test2
            label: tst2
            state: present
            description: Test organization 2
```

### Using Ansible Built-in Collections after Ansible 2.9

In future versions of Ansible, the core installation will always include a special collection named `ansible.builtin`. This collection will include a set of common modules, such as `copy`, `template`, `file`, `yum`, `command`, and `service`.

You will always be able to use the short names of these modules in your playbooks. For example, you will still be able to use `file` instead of `ansible.builtin.file`. This will allow many Ansible 2.9 playbooks to work without modification, although you might need to install additional collections for modules that are not included in `ansible.builtin`.

However, Red Hat recommends that you use the FQCN notation to prevent future conflicts with collections that might use the same module names.

The following playbook uses the FQCN notation for the `yum`, `copy`, and `service` modules.

```
---
- name: Install and start Apache HTTPD
  hosts: web

  tasks:
    - name: Ensure the httpd package is present
      ansible.builtin.yum:
        name: httpd
        state: present

    - name: Ensure the index.html file is present
      ansible.builtin.copy:
        src: files/index.html
        dest: /var/www/html/index.html
        owner: root
        group: root
        mode: 0644
        setype: httpd_sys_content_t

    - name: Ensure the httpd service is started
      ansible.builtin.service:
        name: httpd
        state: started
        enabled: true
```

**REFERENCES**

Using collections — Ansible Documentation
(https://docs.ansible.com/ansible/2.9/user_guide/collections_using.html)

Galaxy User Guide — Ansible Documentation (https://docs.ansible.com/ansible/2.9/galaxy/user_guide.html)

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH07S08) → NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH07S10)

Privacy Policy (http://s.bl-1.com/h/cZrgWbQn?url=https://www.redhat.com/en/about/privacy-policy?extIdCarryOver=true&sc_cid=701f2000001D8QoAAK)

Red Hat Training Policies (http://s.bl-1.com/h/cZrb2DXG?url=https://www.redhat.com/en/about/red-hat-training-policies)

Terms of Use (https://www.redhat.com/en/about/terms-use)

All policies and guidelines (https://www.redhat.com/en/about/all-policies-guidelines)

Release Notes (https://learn.redhat.com/t5/Red-Hat-Learning-Subscription/Red-Hat-Learning-Subscription-Release-Notes/ba-p/22952)

Cookie Preferences