Red Hat Ansible Automation Platform 2.4

Red Hat Ansible Automation Platform creator guide

Learn to create automation content with Ansible

Last Updated: 2024-09-18

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible Automation Platform creator guide

Learn to create automation content with Ansible

Legal Notice

Abstract

This guide helps developers learn how to use Ansible to create content for automation.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

If you have a suggestion to improve this documentation, or find an error, you can contact technical support at https://access.redhat.com to open a request.

CHAPTER 1. PREFACE

Using automation execution environments to automate content within the Red Hat Ansible Automation Platform

You can use Execution Environments as reproducible, portable, consistent and shareable container images. They control all of the dependencies of an Ansible Automation Platform job's runtime environment from system dependencies, Python dependencies, Ansible versions, and Ansible content in the form of Collections.

CHAPTER 2. INTRODUCTION TO CONTENT CREATOR WORKFLOWS AND AUTOMATION EXECUTION ENVIRONMENTS

2.1. ABOUT CONTENT WORKFLOWS

Before Red Hat Ansible Automation Platform 2.0, an automation content developer may have needed so many Python virtual environments that they required their own automation in order to manage them. To reduce this level of complexity, Ansible Automation Platform 2.0 is moving away from virtual environments and using containers, referred to as automation execution environments, instead, as they are straightforward to build and manage and are more shareable across teams and orgs.

As automation controller shifts to using automation execution environments, tools like automation content navigator and Ansible Builder ensure that you can take advantage of those automation execution environments locally within your own development system.

Additional resources

- See the Automation Content Navigator Creator Guide for more on using automation content navigator.
- For more information on Ansible Builder, see Creating and Consuming Execution Environments .

2.2. ARCHITECTURE OVERVIEW

The following list shows the arrangements and uses of tools available on Ansible Automation Platform 2.0, along with how they can be utilized:

- automation content navigator only – can be used today in Ansible Automation Platform 1.2
- automation content navigator + downloaded automation execution environments – used directly on laptop/workstation
- automation content navigator + downloaded automation execution environments + automation controller – for pushing/executing locally → remotely
- automation content navigator + automation controller + Ansible Builder + Layered custom EE – provides even more control over utilized content for how to execute automation jobs

CHAPTER 3. UNDERSTANDING ANSIBLE CONCEPTS

As a automation developer, review the following Ansible concepts to create successful Ansible playbooks and automation execution environments before beginning your Ansible development project.

3.1. PREREQUISITES

- Ansible is installed. For information about installing Ansible, see Installing Ansible in the Ansible documentation.

3.2. ABOUT ANSIBLE PLAYBOOKS

Playbooks are files written in YAML that contain specific sets of human-readable instructions, or “plays”, that you send to run on a single target or groups of targets.

Playbooks can be used to manage configurations of and deployments to remote machines, as well as sequence multi-tier rollouts involving rolling updates. Use playbooks to delegate actions to other hosts, interacting with monitoring servers and load balancers along the way. Once written, playbooks can be used repeatedly across your enterprise for automation.

3.3. ABOUT ANSIBLE ROLES

A role is Ansible’s way of bundling automation content in addition to loading related vars, files, tasks, handlers, and other artifacts automatically by utilizing a known file structure. Instead of creating huge playbooks with hundreds of tasks, you can use roles to break the tasks apart into smaller, more discrete units of work.

You can find roles for provisioning infrastructure, deploying applications, and all of the tasks you do every day on Ansible Galaxy. Filter your search by **Type** and select **Role**. Once you find a role that you are interested in, you can download it by using the **ansible-galaxy** command that comes bundled with Ansible:

```
$ ansible-galaxy role install username.rolename
```

3.4. ABOUT CONTENT COLLECTIONS

An Ansible Content Collection is a ready-to-use toolkit for automation. It includes several types of content such as playbooks, roles, modules, and plugins all in one place. The following diagram shows the basic structure of a collection:

```
collection/
├── docs/
├── galaxy.yml
├── meta/
│   └── runtime.yml
├── plugins/
│   ├── modules/
│   │   └── module1.py
│   ├── inventory/
│   ├── lookup/
│   ├── filter/
│   └── .../
├── README.md
```

```
├── roles/
│   ├── role1/
│   ├── role2/
│   └── .../
├── playbooks/
│   ├── files/
│   ├── vars/
│   ├── templates/
│   ├── playbook1.yml
│   └── tasks/
└── tests/
    ├── integration/
    └── unit/
```

In Red Hat Ansible Automation Platform, automation hub serves as the source for Ansible Certified Content Collections.

3.5. ABOUT EXECUTION ENVIRONMENTS

Automation execution environments are consistent and shareable container images that serve as Ansible control nodes. Automation execution environments reduce the challenge of sharing Ansible content that has external dependencies.

Automation execution environments contain:

- Ansible Core
- Ansible Runner
- Ansible Collections
- Python libraries
- System dependencies
- Custom user needs

You can define and create an automation execution environment using Ansible Builder.

Additional resources

- For more information about Ansible Builder, see Creating and Consuming Execution Environments.

CHAPTER 4. TOOLS AND COMPONENTS

Learn more about the Red Hat Ansible Automation Platform tools and components you will use in creating automation execution environments.

4.1. ABOUT ANSIBLE BUILDER

Ansible Builder is a command line tool that automates the process of building automation execution environments by using the metadata defined in various Ansible Collections or by the user.

Before Ansible Builder was developed, Red Hat Ansible Automation Platform users could run into dependency issues and errors when creating custom virtual environments or containers that included all of the required dependencies installed.

Now, with Ansible Builder, you can easily create a customizable automation execution environments definition file that specifies the content you want to be included in your automation execution environments such as collections, third-party Python requirements, and system-level packages. This allows you to fulfill all of the necessary requirements and dependencies to get jobs running.

NOTE

Red Hat currently does not support users who choose to provide their own container images when building automation automation execution environments.

4.2. USES FOR AUTOMATION CONTENT NAVIGATOR

Automation content navigator is a command line, content-creator-focused tool with a text-based user interface. You can use automation content navigator to:

- Launch and watch jobs and playbooks.
- Share stored, completed playbook and job run artifacts in JSON format.
- Browse and introspect automation execution environments.
- Browse your file-based inventory.
- Render Ansible module documentation and extract examples you can use in your playbooks.
- View a detailed command output on the user interface.

4.3. ABOUT AUTOMATION HUB

Automation hub provides a place for Red Hat subscribers to quickly find and use content that is supported by Red Hat and our technology partners to deliver additional reassurance for the most demanding environments.

At a high level, automation hub provides an overview of all partners participating and providing certified, supported content.

From a central view, users can dive deeper into each partner and check out the collections.

Additionally, a searchable overview of all available collections is available.

4.4. ABOUT THE ANSIBLE COMMAND LINE INTERFACE

Using Ansible on the command line is a useful way to run tasks that you do not repeat very often. The recommended way to handle repeated tasks is to write a playbook.

An ad hoc command for Ansible on the command line follows this structure:

```
$ ansible [pattern] -m [module] -a "[module options]"
```

4.5. ADDITIONAL RESOURCES

- For more information on how to use Ansible as a command line tool, refer to Working with command line tools in the Ansible *User Guide*.
- To upload content to automation hub, see Uploading content to automation hub in the Ansible Automation Platform product documentation.

CHAPTER 5. SETTING UP YOUR DEVELOPMENT ENVIRONMENT

You can follow the procedures in this section to set up your development environment to create automation execution environments.

5.1. INSTALLING ANSIBLE BUILDER

Prerequisites

- You have installed the Podman container runtime.
- You have valid subscriptions attached on the host. Doing so allows you to access the subscription-only resources needed to install **ansible-builder**, and ensures that the necessary repository for **ansible-builder** is automatically enabled. See Attaching your Red Hat Ansible Automation Platform subscription for more information.

Procedure

1. In your terminal, run the following command to activate your Ansible Automation Platform repo:

```
# dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms ansible-builder
```

5.2. INSTALLING AUTOMATION CONTENT NAVIGATOR ON RHEL FROM AN RPM

You can install automation content navigator on Red Hat Enterprise Linux (RHEL) from an RPM.

Prerequisites

- You have installed Python 3.10 or later.
- You have installed RHEL 8.6 or later.
- You registered your system with Red Hat Subscription Manager.

NOTE

Ensure that you only install the navigator matching your current Red Hat Ansible Automation Platform environment.

Procedure

1. Attach the Red Hat Ansible Automation Platform SKU:

```
$ subscription-manager attach --pool=<sku-pool-id>
```

2. Install automation content navigator with the following command:
v.2.4 for RHEL 8 for x86_64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms
ansible-navigator
```

v.2.4 for RHEL 9 for x86-64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms
ansible-navigator
```

Verification

- Verify your automation content navigator installation:

```
$ ansible-navigator --help
```

The following example demonstrates a successful installation:

```
$ ansible-navigator --help
usage: ansible-navigator [-h] [--version] [--cdcp COLLECTION_DOC_CACHE_PATH] [--ce CONTAINER_ENGINE] [--dc DISPLAY_COLOR] [--ecmd EDITOR_COMMAND]
                         [--econ EDITOR_CONSOLE] [--ee EXECUTION_ENVIRONMENT] [--eei EXECUTION_ENVIRONMENT_IMAGE]
                         [--eev EXECUTION_ENVIRONMENT_VOLUME_MOUNTS [EXECUTION_ENVIRONMENT_VOLUME_MOUNTS ...]] [--la LOG_APPEND] [--lf LOG_FILE]
                         [--ll LOG_LEVEL] [-m MODE] [--osc4 OSC4] [--penv PASS_ENVIRONMENT_VARIABLE [PASS_ENVIRONMENT_VARIABLE ...]]
                         [--pp PULL_POLICY] [--senv SET_ENVIRONMENT_VARIABLE [SET_ENVIRONMENT_VARIABLE ...]]
                         {subcommand} --help ...

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit

  <... output truncated ...>

Subcommands:
  {subcommand} --help
    collections         Explore available collections
    config              Explore the current ansible configuration
    doc                 Review documentation for a module or plugin
    images              Explore execution environment images
    inventory           Explore an inventory
    replay              Explore a previous run using a playbook artifact
    run                 Run a playbook
    welcome             Start at the welcome page
```

5.3. DOWNLOADING BASE AUTOMATION EXECUTION ENVIRONMENTS

Base images that ship with Ansible Automation Platform 2.0 are hosted on the Red Hat Ecosystem Catalog (registry.redhat.io).

Prerequisites

- You have a valid Red Hat Ansible Automation Platform subscription.

Procedure

1. Log in to registry.redhat.io

```
$ podman login registry.redhat.io
```

2. Pull the base images from the registry

```
$ podman pull registry.redhat.io/aap/<image name>
```

CHAPTER 6. CREATING CONTENT

Use the guidelines in this section of the *Creator Guide* to learn more about the developing the content you will use in Red Hat Ansible Automation Platform.

6.1. CREATING PLAYBOOKS

Playbooks contain one or more plays. A basic play contains the following sections:

- Name: a brief description of the overall function of the playbook, which assists in keeping it readable and organized for all users.
- Hosts: identifies the target(s) for Ansible to run against.
- Become statements: this optional statement can be set to **true**/**yes** to enable privilege escalation using a become plugin (such as **sudo**, **su**, **pfexec**, **doas**, **pbrun**, **dzdo**, **ksu**).
- Tasks: this is the list actions that get executed against each host in the play.

Example playbook

```
- name: Set Up a Project and Job Template
  hosts: host.name.ip
  become: true

  tasks:
    - name: Create a Project
      ansible.controller.project:
        name: Job Template Test Project
        state: present
        scm_type: git
        scm_url: https://github.com/ansible/ansible-tower-samples.git

    - name: Create a Job Template
      ansible.controller.job_template:
        name: my-job-1
        project: Job Template Test Project
        inventory: Demo Inventory
        playbook: hello_world.yml
        job_type: run
        state: present
```

6.2. CREATING COLLECTIONS

You can create your own Collections locally with the Ansible Galaxy CLI tool. You can use the **collection** subcommand to activate the Collection-specific commands.

Prerequisites

- You have Ansible-core version 2.15 or newer installed in your development environment.

Procedure

1. In your terminal, go to where you want your namespace root directory to be. For simplicity, this should be a path in COLLECTIONS_PATH but that is not required.

2. Run the following command, replacing **my_namespace** and **my_collection_name** with your own values:

```
$ ansible-galaxy collection init <my_namespace>.<my_collection_name>
```

NOTE

Make sure you have the proper permissions to upload to a namespace by checking under the **My Content** tab on galaxy.ansible.com or console.redhat.com/ansible/automation-hub

The earlier command will create a directory named from the namespace argument (if one does not already exist) and then create a directory under that with the Collection name. Inside of that directory will be the default or "skeleton" Collection. This is where you can add your roles or plugins and start working on developing your own Collection.

In relation to execution environments, Collection developers can declare requirements for their content by providing the appropriate metadata in Ansible Builder.

Requirements from a Collection can be recognized in these ways:

- A file **meta/execution-environment.yml**, which references the Python or **bindep** requirements files.
- A file named **requirements.txt**, which includes information about the Python dependencies, and is sometimes found at the root level of the Collection.
- A file named **bindep.txt**, which includes system-level dependencies, and is sometimes found at the root level of the Collection.
- If any of these files are in the **build_ignore** of the Collection, Ansible Builder will not pick up on these. The **build_ignore** section filters any files or directories that should not be included in the build artifact.

Collection maintainers can verify that ansible-builder recognizes the requirements they expect by using the **introspect** command:

```
$ ansible-builder introspect --sanitize ~/.ansible/collections/
```

Additional resources

- For more information about creating collections, see Creating collections in the Ansible *Developer Guide*.

6.3. CREATING ROLES

You can create roles by using the Ansible Galaxy CLI tool. You can access Role-specific commands from the **roles** subcommand.

```
ansible-galaxy role init <role_name>
```

Standalone roles outside of Collections are still supported, but create new roles inside of a Collection to take advantage of all the features Ansible Automation Platform has to offer.

Procedure

1. In your terminal, go to the **roles** directory inside a collection.
2. Create a role called **role_name** inside the collection:

```
$ ansible-galaxy role init my_role
```

The collection now includes a role named **my_role** inside the **roles** directory:

```
~/.ansible/collections/ansible_collections/<my_namespace>/<my_collection_name>
...
└── roles/
    └── my_role/
        ├── .travis.yml
        ├── README.md
        ├── defaults/
        │   └── main.yml
        ├── files/
        ├── handlers/
        │   └── main.yml
        ├── meta/
        │   └── main.yml
        ├── tasks/
        │   └── main.yml
        ├── templates/
        ├── tests/
        │   ├── inventory
        │   └── test.yml
        └── vars/
            └── main.yml
```

3. A custom role skeleton directory can be supplied by using the **--role-skeleton** argument. This allows organizations to create standardized templates for new roles to suit their needs.

```
ansible-galaxy role init my_role --role-skeleton ~/role_skeleton
```

This will create a role named **my_role** by copying the contents of **~/role_skeleton** into **my_role**. The contents of **role_skeleton** can be any files or folders that are valid inside a role directory.

Additional resources

- For more information about creating roles, see Creating roles in the Ansible Galaxy documentation.

6.4. CREATING AUTOMATION EXECUTION ENVIRONMENTS

An automation execution environments definition file will specify

- An Ansible version

- A Python version (defaults to system Python)
- A set of required Python libraries
- Zero or more Content Collections (optional)
- Python dependencies for those specific Collections

The concept of specifying a set of Collections for an environment is to resolve and install their dependencies. The Collections themselves are not required to be installed on the machine that you are generating the automation execution environments on.

An automation execution environments is built from this definition, and results in a container image. Please read the Ansible Builder documentation to learn the steps involved in creating these images.

CHAPTER 7. MIGRATING EXISTING CONTENT

Use the following sections learn how to use the **awx-manage** command to assist with additional steps in the migration process once you have upgraded to Red Hat Ansible Automation Platform 2.0 and automation controller 4.0. Additionally, learn more about migrating between versions of Ansible.

7.1. MIGRATING VIRTUAL ENVS TO AUTOMATION EXECUTION ENVIRONMENTS

Use the following sections to assist with additional steps in the migration process once you have upgraded to Red Hat Ansible Automation Platform 2.0 and automation controller 4.0.

7.1.1. Listing custom virtual environments

You can list the virtual environments on your automation controller instance using the **awx-manage** command.

Procedure

1. SSH into your automation controller instance and run:

```
$ awx-manage list_custom_venvs
```

A list of discovered virtual environments will appear.

```
# Discovered virtual environments:
/var/lib/awx/venv/testing
/var/lib/venv/new_env

To export the contents of a virtual environment, re-run while supplying the path as an argument:
awx-manage export_custom_venv /path/to/venv
```

7.1.2. Viewing objects associated with a custom virtual environment

View the organizations, jobs, and inventory sources associated with a custom virtual environment using the **awx-manage** command.

Procedure

1. SSH into your automation controller instance and run:

```
$ awx-manage custom_venv_associations /path/to/venv
```

A list of associated objects will appear.

```
inventory_sources:
- id: 15
  name: celery
job_templates:
- id: 9
  name: Demo Job Template @ 2:40:47 PM
- id: 13
```

```
  name: elephant
organizations
- id: 3
  name: alternating_bongo_meow
- id: 1
  name: Default
projects: []
```

7.1.3. Selecting the custom virtual environment to export

Select the custom virtual environment you want to export by using **awx-manage export_custom_venv** command.

Procedure

1. SSH into your automation controller instance and run:

```
$ awx-manage export_custom_venv /path/to/venv
```

The output from this command will show a **pip freeze** of what is in the specified virtual environment. This information can be copied into a **requirements.txt** file for Ansible Builder to use for creating a new automation execution environments image.

```
numpy==1.20.2
pandas==1.2.4
python-dateutil==2.8.1
pytz==2021.1
six==1.16.0

To list all available custom virtual environments run:
awx-manage list_custom_venvs
```

NOTE

Pass the **-q** flag when running **awx-manage list_custom_venvs** to reduce output.

7.2. MIGRATING BETWEEN ANSIBLE CORE VERSIONS

Migrating between versions of Ansible Core requires you to update your playbooks, plugins and other parts of your Ansible infrastructure to ensure they work with the latest version. This process requires that changes are validated against the updates made to each successive version of Ansible Core. If you intend to migrate from earlier versions of Ansible to Ansible-core 2.15, you first need to verify that you meet the requirements of the Ansible version that follows your version, and from there make successive updates to 2.15.

7.2.1. Ansible Porting Guides

The Ansible *Porting Guide* is a series of documents that provide information on the behavioral changes between consecutive Ansible versions. Refer to the guides when migrating from version of Ansible to a newer version.

7.2.2. Additional resources

- Refer to the Ansible 2.9 for behavior changes between Ansible 2.8 and Ansible 2.9.
- Refer to the Ansible 2.10 for behavior changes between Ansible 2.9 and Ansible 2.10.

CHAPTER 8. EXECUTING YOUR CONTENT WITH AUTOMATION CONTENT NAVIGATOR

Now that you have your automation execution environments built, you can use automation content navigator to validate that the content will run in the same manner as the automation controller will run it.

8.1. RUNNING ANSIBLE PLAYBOOKS WITH AUTOMATION CONTENT NAVIGATOR

As a content creator, you can execute your Ansible playbooks with automation content navigator and interactively delve into the results of each play and task to verify or troubleshoot the playbook. You can also execute your Ansible playbooks inside an execution environment and without an execution environment to compare and troubleshoot any problems.

8.1.1. Executing a playbook from automation content navigator

You can run Ansible playbooks with the automation content navigator text-based user interface to follow the execution of the tasks and delve into the results of each task.

Prerequisites

- A playbook.
- A valid inventory file if not using **localhost** or an inventory plugin.

Procedure

1. Start automation content navigator

```
$ ansible-navigator
```

2. Run the playbook.

```
$ :run
```

3. Optional: type **ansible-navigator run simple-playbook.yml -i inventory.yml** to run the playbook.

4. Verify or add the inventory and any other command line parameters.

```
INVENTORY OR PLAYBOOK NOT FOUND, PLEASE CONFIRM THE FOLLOWING
──────────────────────────────────────────────────────────────────────────────
──────────────────────

  Path to playbook: /home/ansible-navigator_demo/simple_playbook.yml
  Inventory source: /home/ansible-navigator-demo/inventory.yml
Additional command line parameters: Please provide a value (optional)
──────────────────────────────────────────────────────────────────────────────
──────────────────────

                                  Submit Cancel
```

5. Tab to **Submit** and hit Enter. You should see the tasks executing.

```
  PLAY NAME     OK  CHANGED    UNREACHABLE      FAILED    SKIPPED    IGNORED    IN PROGRESS      TASK COUNT         PROGRESS
0|all            6        0              0           6          0          0              0              12         COMPLETE
```

6. Type the number next to a play to step into the play results, or type **:<number>** for numbers above 9.

```
   RESULT   HOST      NUMBER   CHANGED   TASK                        TASK ACTION                        DURATION
 3 OK       node-0         3     False   Gathering Facts             gather_facts                             1s
 4 OK       node-1         4     False   Gathering Facts             gather_facts                             1s
 5 OK       node-2         5     False   Gathering Facts             gather_facts                             1s
 6 FAILED   main-0         6     False   Gather the package facts    ansible.builtin.package_facts            1s
 7 FAILED   infra-0        7     False   Gather the package facts    ansible.builtin.package_facts            1s
 8 FAILED   lb-0           8     False   Gather the package facts    ansible.builtin.package_facts            1s
 9 FAILED   node-0         9     False   Gather the package facts    ansible.builtin.package_facts            1s
10 FAILED   node-1        10     False   Gather the package facts    ansible.builtin.package_facts            1s
11 FAILED   node-2        11     False   Gather the package facts    ansible.builtin.package_facts            0s
```

Notice failed tasks show up in red if you have colors enabled for automation content navigator.

7. Type the number next to a task to review the task results, or type **:<number>** for numbers above 9.

```
PLAY [all:6] ***************************************************************************************************************
TASK [Gather the package facts] ********************************************************************************************
FAILED: [main-0] Could not detect a supported package manager from the following list: ['apt', 'apk', 'rpm', 'portage', 'pkg'
 0 ---
 1 duration: 1.339719
 2 end: '2021-06-10T18:52:32.968770'
 3 event_loop: null
 4 host: main-0
 5 ignore_errors: null
 6 play: all
```

8. Optional: type **:doc** bring up the documentation for the module or plugin used in the task to aid in troubleshooting.

```
ANSIBLE.BUILTIN.PACKAGE_FACTS (MODULE)
 0 ---
 1 doc:
 2   author:
 3   - Matthew Jones (@matburt)
 4   - Brian Coca (@bcoca)
 5   - Adam Miller (@maxamillion)
 6   collection: ansible.builtin
 7   description:
 8   - Return information about installed packages as facts.
<... output omitted ...>
11   module: package_facts
12   notes:
13   - Supports C(check_mode).
14   options:
15     manager:
16       choices:
17       - auto
18       - rpm
19       - apt
20       - portage
21       - pkg
22       - pacman

<... output truncated ...>
```

Additional resources

- ansible-playbook
- Ansible playbooks

8.1.2. Reviewing playbook results with an automation content navigator artifact file

Automation content navigator saves the results of the playbook run in a JSON artifact file. You can use this file to share the playbook results with someone else, save it for security or compliance reasons, or review and troubleshoot later. You only need the artifact file to review the playbook run. You do not need access to the playbook itself or inventory access.

Prerequisites

- A automation content navigator artifact JSON file from a playbook run.

Procedure

- Start automation content navigator with the artifact file.

```
$ ansible-navigator replay simple_playbook_artifact.json
```

1. Review the playbook results that match when the playbook ran.

```
   PLAY NAME     OK  CHANGED   UNREACHABLE     FAILED   SKIPPED   IGNORED   IN PROGRESS     TASK COUNT        PROGRESS
0|all            12        0             0          0        25         0             0             37        COMPLETE
```

You can now type the number next to the plays and tasks to step into each to review the results, as you would after executing the playbook.

Additional resources

- ansible-playbook
- Ansible playbooks

CHAPTER 9. CONCLUSION

You should now be able to customize an automation execution environments for your particular automation needs, as well as share and use them via a container registry.