Automation with Ansible: Past, Present & Future

Red Hat Virtual Training

New York Virtual Training:
Automation with Ansible: Past, Present & Future

Thursday, January 27, 2022
1:00pm ET; 10:00am PT

For more information, contact Carahsoft or our reseller partners:
redhat@carahsoft.com | 877-RHAT-GOV
New York Virtual Training: Automation with Ansible: Past, Present & Future

Thursday, January 27, 2022
1:00pm ET; 10:00am PT
Connor Murray
Red Hat Account Manager
703-230-7583
Connor.Murray@carahsoft.com
About Carahsoft

Carahsoft is The Trusted Public Sector IT Solutions Provider®, supporting Federal, State and Local Government and Education and Healthcare organizations with IT products, services and training through our partners and contracts.
Red Hat Ansible
*Simple, powerful, agentless IT automation*

Red Hat Ansible Automation Platform provides a foundation for building and operating IT automation, at scale.
Ansible Automation Workshop
Ansible: Past, Present, and Future
An overview of Ansible from Ansible Engine to Ansible Automation Platform 2.x

Travis Michette
Principal Instructor
Housekeeping

- Timing
- Breaks
- Takeaways
- Materials: [https://red.ht/aap2x](https://red.ht/aap2x)
  - RHLS Subscribers - DO374EA
What you will learn

- Introduction to Ansible Automation
- How it works
- Understanding modules, tasks & playbooks
- How to execute Ansible commands & Playbooks
- Evolution of Ansible
  - Ansible Playbooks and Ad-Hoc Commands
  - Ansible Roles
  - Ansible Collections
  - Ansible Execution Environments
- Ansible Content Navigator, Ansible Automation Hub, and Ansible Controller (High-Level Overview)
Agenda

- Introduction
- Ansible Engine (Past)
- Ansible Automation Platform 1.x (Present)
- Break (10 min)
- Ansible Automation Platform 2x (Future)
- Ansible Automation Training
Introduction

Topics Covered:

● What is the Ansible Automation Platform?
● What can it do?
Why Ansible?

Simple
- Human readable automation
- No special coding skills needed
- Tasks executed in order
- Usable by every team
- Get productive quickly

Powerful
- App deployment
- Configuration management
- Workflow orchestration
- Network automation
- Orchestrate the app lifecycle

Agentless
- Agentless architecture
- Uses OpenSSH & WinRM
- No agents to exploit or update
- Get started immediately
- More efficient & more secure
What can I do using Ansible?

Automate the deployment and management of your entire IT footprint.

Do this...
- Orchestration
- Configuration Management
- Application Deployment
- Provisioning
- Continuous Delivery
- Security and Compliance

On these...
- Firewalls
- Load Balancers
- Applications
- Containers
- Clouds
- Servers
- Infrastructure
- Storage
- Network Devices
- And more...
Ansible automates technologies you use

Time to automate is measured in minutes

<table>
<thead>
<tr>
<th>Cloud</th>
<th>Virt &amp; Container</th>
<th>Windows</th>
<th>Network</th>
<th>Security</th>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>Docker</td>
<td>ACLs</td>
<td>A10</td>
<td>Checkpoint</td>
<td>Dynatrace</td>
</tr>
<tr>
<td>Azure</td>
<td>VMware</td>
<td>Files</td>
<td>Arista</td>
<td>Cisco</td>
<td>Datadog</td>
</tr>
<tr>
<td>Digital Ocean</td>
<td>RHV</td>
<td>Packages</td>
<td>Aruba</td>
<td>CyberArk</td>
<td>LogicMonitor</td>
</tr>
<tr>
<td>Google</td>
<td>OpenStack</td>
<td>IIS</td>
<td>Cumulus</td>
<td>F5</td>
<td>New Relic</td>
</tr>
<tr>
<td>OpenStack</td>
<td>OpenShift</td>
<td>Regedits</td>
<td>Bigswitch</td>
<td>Fortinet</td>
<td>Sensu</td>
</tr>
<tr>
<td>Rackspace</td>
<td></td>
<td>Shares</td>
<td>Cisco</td>
<td>Juniper</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Services</td>
<td>Dell</td>
<td>IBM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Configs</td>
<td>Extreme</td>
<td>Palo Alto</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Users</td>
<td>F5</td>
<td>Snort</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Domains</td>
<td>Lenovo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+more</td>
<td>MikroTik</td>
<td></td>
<td>+more</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Juniper</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OpenSwitch</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td>Storage</td>
<td></td>
<td>F5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Systems</td>
<td>Netapp</td>
<td></td>
<td>Lenovo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Red Hat Storage</td>
<td></td>
<td>MikroTik</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Infinidat</td>
<td></td>
<td>Juniper</td>
<td></td>
<td>+more</td>
</tr>
<tr>
<td></td>
<td>+more</td>
<td></td>
<td>OpenSwitch</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+more</td>
<td></td>
<td>+more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating</td>
<td></td>
<td></td>
<td>+more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Systems</td>
<td></td>
<td></td>
<td>+more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RHEL</td>
<td></td>
<td></td>
<td>+more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td></td>
<td></td>
<td>+more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td></td>
<td></td>
<td>+more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+more</td>
<td></td>
<td></td>
<td>+more</td>
<td></td>
<td>+more</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>+more</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Devops

Jira
GitHub
Vagrant
Jenkins
Slack
Section 1
Ansible Engine
Section 1.1

Topics Covered:

● Understanding the Ansible Infrastructure
● Ansible Tower (Enterprise Solutions)
PLAYBOOKS ARE WRITTEN IN YAML

Tasks are executed sequentially
Invoke Ansible modules
What makes up an Ansible playbook?

- Plays
- Modules
- Plugins
Ansible plays

What am I automating?

What are they?
Top level specification for a group of tasks. Will tell that play which hosts it will execute on and control behavior such as fact gathering or privilege level.

Building blocks for playbooks
Multiple plays can exist within an Ansible playbook that execute on different hosts.

---
- name: install and start apache
  hosts: web
  become: yes
- name: install and start apache
  hosts: web
  become: yes

  tasks:
  - name: httpd package is present
    yum:
      name: httpd
      state: latest

  - name: latest index.html file is present
    template:
      src: files/index.html
      dest: /var/www/html/

  - name: httpd is started
    service:
      name: httpd
      state: started
MODULES ARE “TOOLS IN THE TOOLKIT”

Python, Powershell, or any language
Extend Ansible simplicity to the entire stack

- name: latest index.html file is present
  template:
    src: files/index.html
    dest: /var/www/html/
Ansible modules
The “tools in the toolkit”

What are they?
Parametrized components with internal logic, representing a single step to be done. The modules “do” things in Ansible.

Language
Usually Python, or Powershell for Windows setups. But can be of any language.

```yaml
- name: latest index.html file ...
  template:
    src: files/index.html
    dest: /var/www/html/
```
PLUGINS ARE “GEARS IN THE ENGINE”
Code that plugs into the core engine
Adaptability for various uses & platforms

{{ some_variable | to_nice_yaml }}
Ansible plugins

The “extra bits”

What are they?

Plugins are pieces of code that augment Ansible’s core functionality. Ansible uses a plugin architecture to enable a rich, flexible, and expandable feature set.

Example become plugin:

```yaml
- name: install and start apache
  hosts: web
  become: yes
```

Example filter plugins:

```yaml
{{ some_variable | to_nice_json }}
{{ some_variable | to_nice_yaml }}
```
List of systems in your infrastructure that automation is executed against:

[web]
webserver1.example.com
webserver2.example.com

[db]
dbserver1.example.com

[switches]
leaf01.internal.com
leaf02.internal.com

[firewalls]
checkpoint01.internal.com

[lb]
f5-01.internal.com
Automate Everything
Red Hat Enterprise Linux, Cisco routers, Arista switches, Juniper routers, Windows hosts, Check Point firewalls, NetApp storage, F5 load balancers and more
LINUX AUTOMATION

150+ Linux Modules

AUTOMATE EVERYTHING LINUX
Red Hat Enterprise Linux, BSD, Debian, Ubuntu and many more!

ONLY REQUIREMENTS:
Python 2 (2.6 or later)
or Python 3 (3.5 or later)

ansible.com/get-started
How Ansible Automation works

Module code is executed locally on the control node

Module code is copied to the managed node, executed, then removed
Ansible Tower

Integrated
- Manage Projects and Jobs
- No CLI Administration skills needed
- Automated
- Single Management Point

Simple
- Environment Overview
- Configuration management
- Workflow orchestration
- Logging and System Management
- Manage Access & Files

Streamlined
- Web Interface / WebUI
- REST API
- Plugins
- User Management
- Users / Roles / Credentials
- More Efficient & More Secure
Ansible Tower is a UI and RESTful API allowing you to scale IT automation, manage complex deployments and speed productivity.

- Role-based access control
- Deploy entire applications with push-button deployment access
- All automations are centrally logged
- Powerful workflows match your IT processes
Red Hat Ansible Tower

**Push button**
An intuitive user interface experience makes it easy for novice users to execute playbooks you allow them access to.

**RESTful API**
With an API first mentality every feature and function of Tower can be API driven. Allow seamless integration with other tools like ServiceNow and Infoblox.

**RBAC**
Allow restricting playbook access to authorized users. One team can use playbooks in check mode (read-only) while others have full administrative abilities.

**Enterprise integrations**

**Centralized logging**
All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Ansible Tower’s API.

**Workflows**
Ansible Tower’s multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.
Ansible Automation Engine

- **Ansible Tower**
  - Role-Based Access Control
  - Knowledge & Visibility
  - Scheduled & Centralized Jobs
  - Simple User Interface
  - Tower API

- **Ansible Engine**
  - Open Source Module Library
  - Plugins
  - Python Codebase

- **Transport**
  - SSH, WinRM, Network CLI, HTTP API

- **Automate Your Enterprise**
  - Infrastructure
    - Linux, OpenShift, Windows, VMware, Operators, Containers...
  - Network
    - Arista, Cisco, Juniper, InfoBlox, F5...
  - Security
    - Checkpoint, QRadar, Snort, Cyberark, Splunk, Fortinet...
  - Cloud
    - AWS, Google Cloud, Azure, IBM Cloud...
  - Services
    - Databases, Logging, Source Control Management...
  - App Development
    - Python VENV, NPM, YUM, APT, PIP...

- **Use Cases**
  - Provisioning
  - Configuration Management
  - App Deployment
  - Continuous Delivery
  - Security & Compliance
  - Orchestration
Section 1.2

Topics Covered:

- Ansible inventories
- Main Ansible config file
- Modules and ad-hoc commands
Inventory

- Ansible works against multiple systems in an inventory
- Inventory is usually file based
- Can have multiple groups
- Can have variables for each group or even host
# Static inventory example:

```bash
[myservers]
10.42.0.2
10.42.0.6
10.42.0.7
10.42.0.8
10.42.0.100
host.example.com
```
Understanding Inventory - Hosts

- inventory can be written in short format and expanded using \([x:y]\) syntax

```
[appsrv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[web]
node-[1:30] ansible_host=10.42.0.[31:60]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=kev
ansible_ssh_private_key_file=/home/kev/.ssh/id_rsa
```
Understanding Inventory - Variables

[applsrv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[web]
node-[1:30] ansible_host=10.42.0.[31:60]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=ender
ansible_ssh_private_key_file=/home/ender/.ssh/id_rsa
Understanding Inventory - Groups

[nashville]
bnaapp01
bnaapp02

[atlanta]
atlapp03
atlapp04

[south:children]
atlanta
nashville
hsvapp05
Configuration File

- Basic configuration for Ansible
- Can be in multiple locations, with different precedence
- Here: `.ansible.cfg` in the home directory
- Configures where to find the inventory
Ansible Configuration

Configuration files will be searched for in the following order (Highest Precedence to Lowest):

1. ANSIBLE_CONFIG (environment variable if set)
2. ansible.cfg (in the current directory)
3. ~/.ansible.cfg (in the home directory)
4. /etc/ansible/ansible.cfg (installed as Ansible default)
The Ansible Configuration File: `ansible.cfg`

```
[user@ansible] $ cat ansible.cfg

[defaults]
inventory = inventory
remote_user = devops
```
First Ad-Hoc Command: ping

- Single Ansible command to perform a task quickly directly on command line
- Most basic operation that can be performed
- Utilizes a single Ansible Module with options and arguments
- Here: an example Ansible ping - not to be confused with ICMP

```
$ ansible all -m ping
```
Ad-Hoc Commands

# Check connections (submarine ping, not ICMP)
[user@ansible] $ ansible all -m ping

web1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
The Ansible Command

Some basics to keep you from getting stuck
--help (Display some basic and extensive options)

[user@ansible ~]$ **ansible --help**
Usage: ansible <host-pattern> [options]

Define and run a single task 'playbook' against a set of hosts

Options:
- `-a MODULE_ARGS`, `--args=MODULE_ARGS`
  module arguments
- `--ask-vault-pass`
  ask for vault password
- `-B SECONDS`, `--background=SECONDS`
<<<snippet, output removed for brevity>>>
Ad-Hoc Commands

Here are some common options you might use:

- **-m MODULE_NAME, --module-name=MODULE_NAME**
  Module name to execute the ad-hoc command

- **-a MODULE_ARGS, --args=MODULE_ARGS**
  Module arguments for the ad-hoc command

- **-b, --become**
  Run ad-hoc command with elevated rights such as sudo, the default method

- **-e EXTRA_VARS, --extra-vars=EXTRA_VARS**
  Set additional variables as key=value or YAML/JSON
Ad-Hoc Commands

Here are some common options you might use:

# Check connections to all (submarine ping, not ICMP)
[user@ansible] $ ansible all -m ping

# Run a command on all the hosts in the web group
[user@ansible] $ ansible web -m command -a "uptime"

# Collect and display known facts for server "web1"
[user@ansible] $ ansible web1 -m setup
Demo Time
Ansible - Ad-Hoc Command to Test Environment (Ansible Ping)
Ansible - Ad-Hoc Command to Create User and Sudoers File
Section 1.3

Topics Covered:

- Playbooks basics
- Running a playbook
---

- name: install and start apache
  hosts: web
  become: yes

tasks:
  - name: httpd package is present
    yum:
      name: httpd
      state: latest

  - name: latest index.html file is present
    template:
      src: files/index.html
      dest: /var/www/html/

  - name: httpd is started
    service:
      name: httpd
      state: started

---
An Ansible Playbook

---
- name: install and start apache
  hosts: web
  become: yes

tasks:
  - name: httpd package is present
    yum:
      name: httpd
      state: latest

  - name: latest index.html file is present
    template:
      src: files/index.html
      dest: /var/www/html/

  - name: httpd is started
    service:
      name: httpd
      state: started
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
  - name: httpd package is present
    yum:
      name: httpd
      state: latest

  - name: latest index.html file is present
    template:
      src: files/index.html
      dest: /var/www/html/

  - name: httpd is started
    service:
      name: httpd
      state: started
Running an Ansible Playbook:

The most important colors of Ansible

- A task executed as expected, no change was made.
- A task executed as expected, making a change
- A task failed to execute successfully
Running an Ansible Playbook

```
[user@ansible] $ ansible-playbook apache.yml
PLAY [webservers] ************************************************************************************************
TASK [Gathering Facts] ******************************************************************************************
  ok: [web2]
  ok: [web1]
  ok: [web3]
TASK [Ensure httpd package is present] ****************************************************************************
  changed: [web2]
  changed: [web1]
  changed: [web3]
TASK [Ensure latest index.html file is present] ********************************************************************
  changed: [web2]
  changed: [web1]
  changed: [web3]
TASK [Restart httpd] ***********************************************
  changed: [web2]
  changed: [web1]
  changed: [web3]
PLAY RECAP ******************************************************************************************************
  web2   : ok=1  changed=3  unreachable=0  failed=0
  web1   : ok=1  changed=3  unreachable=0  failed=0
  web3   : ok=1  changed=3  unreachable=0  failed=0
```
Demo Time
Ansible Engine - Running a Playbook to Create User and Sudoers File
Ansible Engine - Running a Playbook to Deploy Webserver (Failure - AAP)
Section 1.4

Topics Covered:

- What are roles?
- What is the structure of a Role?
- Ansible Galaxy
Ansible roles

Reusable automation actions

What are they?

Group your tasks and variables of your automation in a reusable structure. Write roles once, and share them with others who have similar challenges in front of them.

```yaml
---
- name: install and start apache
  hosts: web
  roles:
    - common
    - webservers
```
Ansible Roles

- Ansible roles provide a way to reuse Ansible code generically and more effectively and have the following benefits:
  - Groups content for easy sharing of code with others
  - Make large projects manageable
  - Developed in parallel by different parties
  - Written generically and can be placed in version control
- Ansible roles can be shared via SCM or publicly through Ansible Galaxy

Installing and Using a Role

```bash
$ ansible-galaxy install tmichett.deploy_packages
```

Creating an Ansible Role (beyond scope)

- Use the `ansible-galaxy init <RoleName>` command to create a Role
- Empty directories or unused directories can be deleted to clean up the Role
- Populate the various Role structures
  - Must have the following components (at minimum):
    - README.md
    - meta/main.yaml
    - tasks/main.yaml

Playbook using a Role

```yaml
---
- name: Install Packages
  hosts: web
  become: yes
roles:
  - tmichett.deploy_packages
```
Role structure

- **Defaults**: default variables with lowest precedence (e.g. port)
- **Files**: contains files that are deployed
- **Handlers**: contains all handlers
- **Meta**: role metadata including dependencies to other roles.
  
  *TIP: Used to construct some of the Ansible Galaxy automated documentation*
- **README**: contains the README for the role and used for Galaxy README
- **Tasks**: plays or tasks
  
  *TIP: It’s common to include tasks in main.yml with “when” (e.g. OS == xyz)*
- **Templates**: templates to deploy
- **Tests**: place for playbook tests
- **Vars**: variables (e.g. override port)
Ansible Galaxy

Sharing Content  Community  Roles, and more
Demo Time
Ansible Engine - Playbook with Roles (Warning - Uses Collections from Newer Ansible)
Section 2
Ansible Automation Platform 1.2

*Present*
Section 2.1

Topics Covered:

- Ansible Automation Hub
- Ansible Collections
Trusted source

Customer controlled
Deploying either on-prem or to a cloud, customers can run their own private instances of Automation Hub integrated into Red Hat Ansible Automation Platform.

Private content
Manage the lifecycle and internal distribution of in-house Ansible content within.

Customizable Content Catalog
Via sync from community (Galaxy) and supported (Automation Hub) sources, customers can supply internal users with approved content in one controlled location.
Ansible Automation Hub

- Collections
  - cloud
    - Provided by Google Cloud
    - The Google Cloud Platform collection.
    - 170 Modules 5 Roles 2 Plugins
  - flashblade
    - Provided by Pure Storage
    - Collection of modules to manage Pure Storage FlashBlades
    - 44 Modules 0 Roles 0 Plugins
  - flasharray
    - Provided by Pure Storage
    - Collection of modules to manage Pure Storage FlashArrays (including Cloud Block Store)
    - 51 Modules 0 Roles 0 Plugins
Ansible Automation Hub Collections

Info
Ansible Modules to manage Satellite installations

- foreman
- katello
- satellite

License
GPL-3.0-or-later

Installation
ansible-galaxy collection install redhat.satellite

Note: Installing collections with ansible-galaxy is only supported in ansible 2.9+

Download tarball

Install Version
3.0.0 released 23 days ago (latest)

Requires Ansible
>=2.9

Red Hat Satellite Ansible Collection
Ansible modules for interacting with the Satellite API.
Ansible Galaxy Collections
Collections

Simplified and consistent content delivery

What are they?

Collections are a data structure containing automation content:

- Modules
- Playbooks
- Roles
- Plugins
- Docs
- Tests
Ansible Collections - Why?

- Ansible 2.9 introduced the concept of collections and provided mapping for Ansible modules that were moved into a collection namespace.
- Collections allowed development of Ansible core components to be separated from module and plug-in development.
- Upstream Ansible unbundled modules from Ansible core code beginning with Ansible Base 2.10/2.11.
- Never versions of Ansible require collections to be installed in order for modules to be available for Ansible.
- Ansible 2.9 provides a mapping to the Fully Qualified Collection Name (FQCN)
  - https://github.com/ansible/ansible/blob/devel/lib/ansible/config/ansible_builtin_runtime.yml
- Playbooks should be developed using the FQCNs when referring to modules in tasks.
  - AAP requires older playbooks to be refactored to a degree to conform to new modules and component names.
- Collections must be installed for modules to be available for Ansible playbooks
  - `ansible-galaxy collection install -r collections/requirements.yml -p collections/`
Collections and Changes to Ansible Modules

---

```yaml
# GNU General Public License v3.0+ (see COPYING or https://www.gnu.org/licenses/gpl-3.0.txt)
plugin_routing:
  connection:
    # test entries
    redirected_local:
      redirect: ansible.builtin.local
    buildah:
      redirect: containers.podman.buildah
    podman:
      redirect: containers.podman.podman
    aws_ssm:
      redirect: community.aws.aws_ssm
    chroot:
      redirect: community.general.chroot
    docker:
      redirect: community.docker.docker
    funcd:
      redirect: community.general.funcd
    iocage:
      redirect: community.general.iocage
    jail:
      redirect: community.general.jail
    kubectl:
      redirect: kubernetes.core.kubectl
```

---

https://github.com/ansible/ansible/blob/devel/lib/ansible/config/ansible_builtin_runtime.yml
Accessing collections
How to get them

Requirements file
Requirements file defines the required collections for a playbook

Pull via controller
Automation controller pulls the collections from Automation Hub automatically

Command line
CLI access is also possible via ansible-galaxy command
Collections and Playbooks

Older Playbooks

- **podman_container** was a module that was able to be leveraged by the short module name in Ansible < 2.9.
- Ansible versions > 2.9 require that the FQCN be specified to that tasks can reference modules.
- It is possible to define collections at the top of a playbook similar to roles (1).
  - This enables short **module** names to be used versus using the FQCN (2).
- Not recommended as best practice.

```
[user@ansible] $ cat playbook.yml
---
- name: Deploy HTTPD Server Demo
  hosts: localhost
  vars_files:
  - vars/registry_login.yml
  collections:
  - containers.podman
  tasks:
    ## Start and Run the HTTPD Container
    - name: Start the Webserver Container
      podman_container:
        name: Website_Demo
        image: quay.io/redhattraining/httpd-parent:2.4
        state: started
        restart: yes
        ports:
        - "7080:80"
        volume:
        - "/Webhosting:/var/www/html:Z"
```
Ansible Playbook with Collections

---

- name: Playbook to Fully Setup and Configure a Webserver
  hosts: servera
  tasks:
    - name: Install Packages for Webserver
      yum:
        name:
          - httpd
          - firewalld
        state: latest

    - name: Create Content for Webserver
      copy:
        content: "I'm an awesome webserver\n"
        dest: /var/www/html/index.html

    - name: Firewall is Enabled
      systemd:
        name: firewalld
        state: started
        enabled: true

- name: HTTP Service is Open on Firewall
  ansible.posix.firewalld:
    service: http
    state: enabled
    permanent: true
    immediate: yes

- name: httpd is started
  systemd:
    name: httpd
    state: started
    enabled: true

- name: Firewall was a module that was able to be leveraged by the short module name in Ansible <= 2.9, moved to the Posix collection using FQCN (1) above.
Demo Time
Ansible Automation Platform 1.2 - Ansible Collections to Deploy Webserver
Break Time
Section 3
Ansible Automation Platform 2.x

Future
New in Ansible Automation Platform 2.X

What changes?

- **Updated Private Automation Hub**
  Hosting of private content, container registry

- **Automation controller**
  Replaced *Ansible Tower*

- **Automation execution environments**
  Replaced *Ansible Engine*

- ansible-builder and ansible-navigator
  New tools for enterprise automation developers
Section 3.1

Topics Covered:

- Introduction to AAP 2.x Components
  - Ansible Content Navigator
  - Ansible Execution Environments
Ansible Content Navigator

develop → ansible-navigator → execute

- playbook
- execution environment

- Supported tooling
- Portable
- Scalable
## Ansible Content Navigator

<table>
<thead>
<tr>
<th>Ansible Command</th>
<th>Automation Content Navigator Subcommands</th>
</tr>
</thead>
<tbody>
<tr>
<td>ansible-config</td>
<td>ansible-navigator config</td>
</tr>
<tr>
<td>ansible-doc</td>
<td>ansible-navigator doc</td>
</tr>
<tr>
<td>ansible-inventory</td>
<td>ansible-navigator inventory</td>
</tr>
<tr>
<td>ansible-playbook</td>
<td>ansible-navigator run</td>
</tr>
</tbody>
</table>

### # Running Navigator Interactively

```
[user@ansible] $ ansible-navigator run Playbook.yml -m interactive
```

### # Running Navigator Non-Interactively (Similar to `ansible-playbook` output)

```
[user@ansible] $ ansible-navigator run Playbook.yml -m stdout
```
ansible-navigator.yml

- ansible-navigator:
  execution-environment: (1)
    enabled: true
  environment-variables:
    set:
      ANSIBLE_CONFIG: ansible.cfg (2)
  image: hub.lab.example.com/ee-29-rhel8:latest (3)
  logging:
    level: critical
    mode: stdout (4)

- Execution Environment - Configures Ansible Navigator to use an Execution Environment (EE). (1)
- Specifies where Ansible Navigator and the Ansible EE will receive Ansible configuration settings. (2)
  - Provides ansible.cfg file for the container runtime environment
- Specifies Ansible EE to use for Ansible Navigator. (3)
  - Defines container image and registry to be used for Ansible Navigator
- Specified Mode, in this case, we are using STDOUT so that the output will look like it does with the ansible-playbook command. (4)
Automation Execution Environments

Components needed for automation, packaged in a cloud-native way

Execution Environments

Collections

Libraries

Ansible Core

Universal Base Image
## Ansible Execution Environments

<table>
<thead>
<tr>
<th>EE-29-RHEL8:LATEST (PRIMARY)</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Image information</td>
</tr>
<tr>
<td>1</td>
<td>General information</td>
</tr>
<tr>
<td>2</td>
<td>Ansible version and collections</td>
</tr>
<tr>
<td>3</td>
<td>Python packages</td>
</tr>
<tr>
<td>4</td>
<td>Operating system packages</td>
</tr>
<tr>
<td>5</td>
<td>Everything</td>
</tr>
</tbody>
</table>

### EE-29-RHEL8:LATEST (PRIMARY) (OS AND PYTHON VERSION INFORMATION)

```yaml
0 ---
1 friendly: |
2   details: |
3     Red Hat Enterprise Linux release 8.5 (Ootpa)
4 os: |
```

### EE-29-RHEL8:LATEST (PRIMARY) (INFORMATION ABOUT ANSIBLE AND ANSIBLE COLLECTIONS)

```yaml
0 ---
1 ansible: |
2   collections: |
3     details: {} |
4     errors: |
5       - |
6         usage: ansible-galaxy collection [-h] COLLECTION_ACTION ...
7     ansible-galaxy collection: error: argument COLLECTION_ACTION: invalid choice: |
8   version: |
9     details: .9.2
```
Ansible Execution Environments - SSH Keys

- **Execution Environment** - Leverages containers to run Ansible Playbooks
  - **Contains**
    - Ansible Core
    - Ansible Collections
    - Python Environment
  - **Requires**
    - Configuration Files
    - Inventory
    - SSH
      - SSH Keys must be provided through the SSH-Agent service

```bash
[student@workstation ~]$ eval $(ssh-agent)
```

```bash
[student@workstation ~]$ ssh-add ~/.ssh/lab_rsa
```
Demo Time
Ansible - Deploy Webserver with Ansible Content Navigator
Ansible - Ansible Content Navigator - Interactive Mode
Section 3.2

Topics Covered:

- Introduction to AAP 2.x - Ansible Automation Hub
  - Private Automation Hub
  - Custom Execution Environments
Private Automation Hub

Value of Private Automation Hub

Developer IDE

- Custom enterprise content
- Automation Hub
  console.redhat.com

Content SDK

- Ansible Galaxy

Private Automation Hub

Build

Publish

Deliver

Red Hat Ansible Automation Platform cluster
Automation Hub - Collections
## Automation Hub - Execution Environments

### Container Registry

<table>
<thead>
<tr>
<th>Container repository name</th>
<th>Description</th>
<th>Created</th>
<th>Last modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>ansible-builder-rhel8</td>
<td></td>
<td>2 months ago</td>
<td>2 months ago</td>
</tr>
<tr>
<td>ee-29-rhel8</td>
<td></td>
<td>2 months ago</td>
<td>2 months ago</td>
</tr>
<tr>
<td>ee-minimal-rhel8</td>
<td></td>
<td>2 months ago</td>
<td>2 months ago</td>
</tr>
<tr>
<td>ee-supported-rhel8</td>
<td></td>
<td>2 months ago</td>
<td>2 months ago</td>
</tr>
</tbody>
</table>
Build, create, publish

Development cycle of an automation execution environment

- Content Creator
- Execution environment builder
- Collections
- Dependencies
- UBI
-Ansible Core
- Execution Environment
- Private automation hub
Ansible Execution Environments - Building/Customizing

# Running `ansible-builder` to Create Structure
[user@ansible] $ ansible-builder create

# Running `ansible-builder` to Build Execution Environment
[user@ansible] $ ansible-builder build -t ee-motd-minimal-demo:1.0
Ansible Execution Environments - Building/Customizing

1. **execution-environment.yml** - Defines parameters and definitions for building the execution environment (EE) including the base image, and builder image along with all Ansible dependencies.
   a. Defines base container image to be used for creating the EE
   b. Defines the builder image to be used for the EE
   c. Points to file containing the Collections and Roles to be installed and included in the EE
   d. Points to file containing the required Python components to be installed and included in the EE
   e. Points to file containing the system applications to be installed in the EE

2. **requirements.yml** - Defines the collections and roles to be used as part of the Ansible Execution Environment.
   a. Listing of collections to be installed in the EE

3. **requirements.txt** - Defines the python dependencies and requirements needed by the Ansible Execution Environment and the included Ansible Collections.
   a. List of Python tools to be installed in the EE

4. **bindep.txt** - Defines the system packages needed in the Ansible Execution Environment to run effectively and support the installed Ansible Collections and Python modules.
   a. List of system packages needed installed in the EE

**IMPORTANT**
Remember that Ansible Execution Environments are based on containers and container images. The `ansible-builder` command will build and create a new container image based on the `execution-environment.yml` file specifications.

# Building an Execution Environment with `ansible-builder`

[student@workstation EE]$ ansible-builder build -t ee_aap_demo:latest
Ansible Execution Environments - Publishing

# Using `podman` to Tag Image for Upload to Private Automation Hub
[user@ansible] $ podman tag localhost/aap-demo:latest hub.lab.example.com/aap-demo:latest

# Using `podman` to Push Image to Private Automation Hub
[user@ansible] $ podman push hub.lab.example.com/aap-demo:latest

# Using `ansible-navigator` to test image from Private Automation Hub
[user@ansible] $ ansible-navigator run --pp always --eei hub.lab.example.com/aap-demo:latest -m stdout Custom_EE_Playbook.yml -b
Ansible Execution Environments - Publishing Cont.
Demo Time

Ansible Automation Platform - Create Custom Execution Environment (EE)
Ansible Automation Platform - Run a Playbook with Custom Execution Environment
Ansible Automation Platform - Publish EE to Private Automation Hub
Section 3.3

Topics Covered:

- Introduction to AAP 2.x - Ansible Controller
  - Organizations, Teams, and RBAC
  - Inventories and Credentials
  - Projects and Job Templates
  - Workflows
How Ansible Works - Ansible Controller

1. R.B.A.C (Many Teams multi-tenanted)
2. WORKFLOW
3. Inventory Management
4. Credential Management
5. Repository Management
6. Ansible Job Template
7. Clustering
8. Centralised Logging
9. API

Ticketing
Self Service
Self Heal
CICD

How Ansible Works - Ansible Controller

WORKFLOW
Business Outcome

Repository Management
Push/Pull from 1 to many sources

Inventory Management
Dynamic from 1 to many sources

Target Environment

Modules

Individual

Ansible
Engine

Playbooks

Centralised Logging
Security Compliance
SIEM

Credentials

Clustering
High Availability
Scalability
Isolated Nodes

Ansible

Playbooks

Source Control
Configuration

Red Hat
Ansible
Tower

Ansible Job Template

Windows Team
Security
Network Team
Team X

Playbooks
Playbooks
Playbooks
Playbooks

Red Hat Enterprise Linux Server
Kernel 3.10.0-327.17.1.el7.x86_64

root@localhost:~ # _
Role-Based Access Controls (RBAC) are built into Ansible Tower and allow administrators to delegate access to inventories, organizations, and more. These controls allow Ansible Tower to help you increase security and streamline management of your Ansible automation.
User Management

- An **Organization** is a logical collection of users, teams, projects, inventories and more. All entities belong to an organization.

- A **User** is an account to access Ansible Tower and its services given the permissions granted to it.

- **Teams** provide a means to implement role-based access control schemes and delegate responsibilities across organizations.
Viewing Organizations

Clicking on the **Organizations** button in the left menu will open up the Organizations window.
Viewing Teams

Clicking on the **Teams** button in the left menu will open up the Teams window.

No Teams Found

Please add Teams to populate this list.
Viewing Users

Clicking on the **Users** button in the left menu will open up the Users window
Demo Time
Creating an Organization with Users and Teams
Inventory

Inventory is a collection of hosts (nodes) with associated data and groupings that Ansible Tower can connect to and manage.

- Hosts (nodes)
- Groups
- Inventory-specific data (variables)
- Static or dynamic sources
Credentials

Credentials are utilized by Ansible Tower for authentication with various external resources:

- Connecting to remote machines to run jobs
- Syncing with inventory sources
- Importing project content from version control systems
- Connecting to and managing network devices

Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.
Demo Time
Creating an Inventory and Credentials
A project is a logical collection of Ansible Playbooks, represented in Ansible Tower.

You can manage Ansible Playbooks and playbook directories by placing them in a source code management system supported by Ansible Tower, including Git, Subversion, and Mercurial.
Everything in Ansible Tower revolves around the concept of a **Job Template**. Job Templates allow Ansible Playbooks to be controlled, delegated and scaled for an organization.

Job templates also encourage the reuse of Ansible Playbook content and collaboration between teams.

A **Job Template** requires:
- An **Inventory** to run the job against
- A **Credential** to login to devices.
- A **Project** which contains Ansible Playbooks
Demo Time
Creating a Project and Job Template
Recall that everything in Ansible Tower revolves around the concept of a Job Template. **Job Workflows** allow multiple Job Templates to be controlled, delegated and scaled for an organization.

Job workflows allow building Ansible pipelines to execute multiple job templates and other functions depending on if the running Job Template succeeds or fails.

A **Job Workflow** requires:

- An **Inventory** to run the job against
- A **Credential** to login to devices.
- A **Project** which contains Ansible Playbooks
- Existing **Job Templates** to execute
Workflow Visualizer

The workflow visualizer will start as a blank canvas.

Please click the Start button to begin.
Visualizing a Workflow

Workflows can branch out, or converge in.

Blue indicates this Job Template will always run

Green indicates this Job Template will only be run if the previous Job Template is successful

Red indicates this Job Template will only be run if the previous Job Template fails
Demo Time
Executing Multiple Playbooks and Projects with Workflows
Section 4
Ansible Automation

Training
Training at Red Hat
Customer return on investment from training

365% 3-year ROI

IDC conducted a study to explore how Red Hat® training courses impacted the skills, performance, and productivity levels of customers. They found that training for impacted IT professionals and developers consistently increases both individual capability and the ultimate business value of the supported technology.

Other key findings include:

- 44% higher DevOps team productivity
- 34% more efficient IT infrastructure teams
- 59% faster to deploy new IT resources
- 76% faster to full productivity, new hires already trained


(*per employee trained numbers based on average of 130 staff members trained per year discussed during IDC interviews)
Improve productivity with training in Ansible automation

Scale people, processes, and infrastructure

Red Hat Ansible Automation Platform

A powerful foundation to build and operate automation across organizations. Prepare your teams with the right skills to make the most out of new technology investments.

59% faster deployment of new IT resources

“Red Hat Training shows our DevOps team how to automate a repeatable task. They can write one playbook to execute a set of tasks that would have taken hours or days of time.”

“With Red Hat Training it doesn’t matter which engineer is engaged on a project. They are all using Ansible for automating tasks, allowing them collectively to be five times as productive ... This was not possible previously. As a result, they’ve definitely picked up the pace of productivity.”


(*per employee trained numbers based on average of 130 staff members trained per year discussed during IDC interviews)
WAYS TO TRAIN

Onsite Training
Private On-site training and exams delivered at your location or at one of our training centers

Classroom Training
Training and test in a professional classroom environment led by Red Hat Certified Instructors

Virtual Training
Live instructor-led online training with the same high-quality, hands-on labs you'd find in our classrooms

Online Learning
90 days of access to course content and up to 80 hours of hands on labs – all available online, at your pace, and your schedule.
RED HAT LEARNING SUBSCRIPTION

A prescriptive, reliable, learning solution for rapid skills transformation on Red Hat technologies

Simple, flexible, on-demand training

- 24x7 access globally, available offline
- Self-paced, unlimited access to Red Hat courses
- Access to content currently in development
- Updated content pushed as early releases
- Content spanning the entire Red Hat product portfolio
- Early access to completed chapters of courses
Red Hat Learning Subscription Evolution

Introducing a Premium subscription tier

Red Hat Learning Subscription + Modularized Virtual Training = Red Hat Learning Subscription PREMIUM
Ansible Curriculum (Future as of Q2/Q3Y22)

Network Admins
- DO457: Ansible for Network Automation

Linux Admins
- RH294: Red Hat System Administration III: Linux Automation
- EX294: Red Hat Certified Engineer exam
- RH358: Red Hat Services Management and Automation

Windows Admins
- DO417: Microsoft Windows Automation with Red Hat Ansible

DevOps Engineers
- DO374*: Developing Advanced Automation with Red Hat Ansible Automation Platform (Estimated: Q1 2022)
- DO467*: Managing Enterprise Automation with Red Hat Ansible Automation Platform (Estimated Q3 2022)
- EX374*: EX467*: Exam(s) coming soon!

*Release windows and course details subject to change
Thank you for attending!

*Please contact us with questions.*

Connor Murray
703-230-7583
877-RHAT-GOV (Team)
Connor.Murray@carahsoft.com
https://carah.io/redhatsled
Thank you for viewing this Red Hat virtual training! Carahsoft is the Master GSA and SLSA Dealer and Distributor for Red Hat Enterprise Open Source solutions available via GSA, SLSA, ITES-SW2, The Quilt and other contract vehicles.

To learn how to take the next step toward acquiring Red Hat’s solutions, please check out the following resources and information:

For additional resources: carah.io/redhatsled

For additional Red Hat solutions: carah.io/RedHatPortfolio

For upcoming events: carah.io/RedHatEvents

For additional Open Source solutions: carah.io/OpenSourceSolutions

To purchase, check out the contract vehicles available for procurement: carah.io/RedHatContracts

To set up a meeting: redhat@carahsoft.com or 877-RHAT-GOV

For more information, contact Carahsoft or our reseller partners: redhat@carahsoft.com | 877-RHAT-GOV