
Red Hat
Ansible Automation
Platform
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
Red Hat
Training

Housekeeping

- Timing
- Breaks
- Takeaways
- Materials: 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?

Red Hat
Ansible Automation
Platform

Red Hat

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

Red Hat

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...**

Red Hat

Ansible automates technologies you use

Time to automate is measured in minutes

Cloud

AWS
Azure
Digital Ocean
Google
OpenStack
Rackspace
+more

Operating Systems

RHEL
Linux
Windows
+more

Virt & Container

Docker
VMware
RHV
OpenStack
OpenShift
+more

Storage

Netapp
Red Hat Storage
Infinidat
+more

Windows

ACLs
Files
Packages
IIS
Regedits
Shares
Services
Configs
Users
Domains
+more

Network

A10
Arista
Aruba
Cumulus
Bigswitch
Cisco
Dell
Extreme
F5
Lenovo
MikroTik
Juniper
OpenSwitch
+more

Security

Checkpoint
Cisco
CyberArk
F5
Fortinet
Juniper
IBM
Palo Alto
Snort
+more

Monitoring

Dynatrace
Datadog
LogicMonitor
New Relic
Sensu
+more

Devops

Jira
GitHub
Vagrant
Jenkins
Slack
+more

Section 1
Ansible Engine

Red Hat
Ansible Automation
Platform

Past

Red Hat

Courses on Ansible

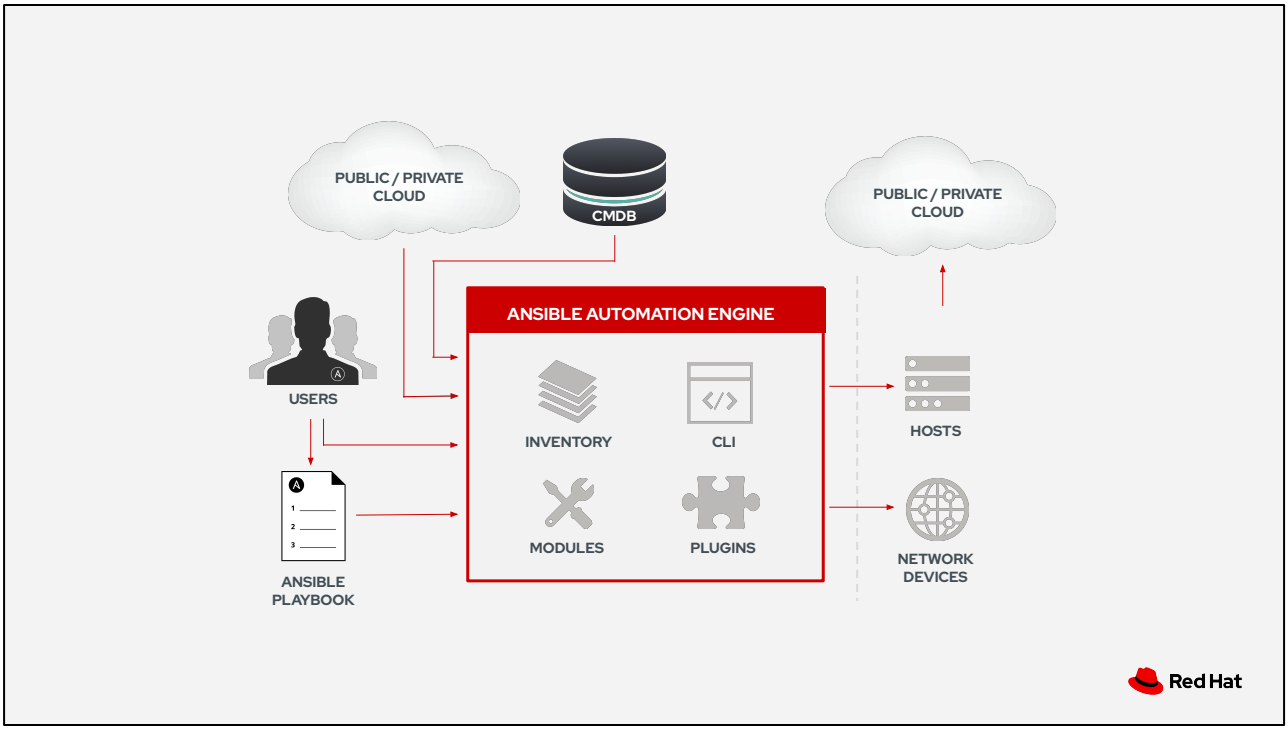
DO407
DO409
DO410

Section 1.1

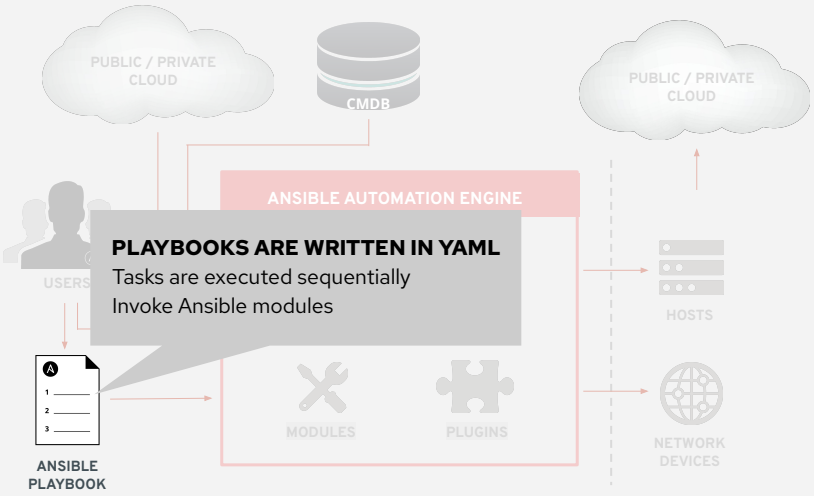
Topics Covered:

- Understanding the Ansible Infrastructure
- Ansible Tower (Enterprise Solutions)

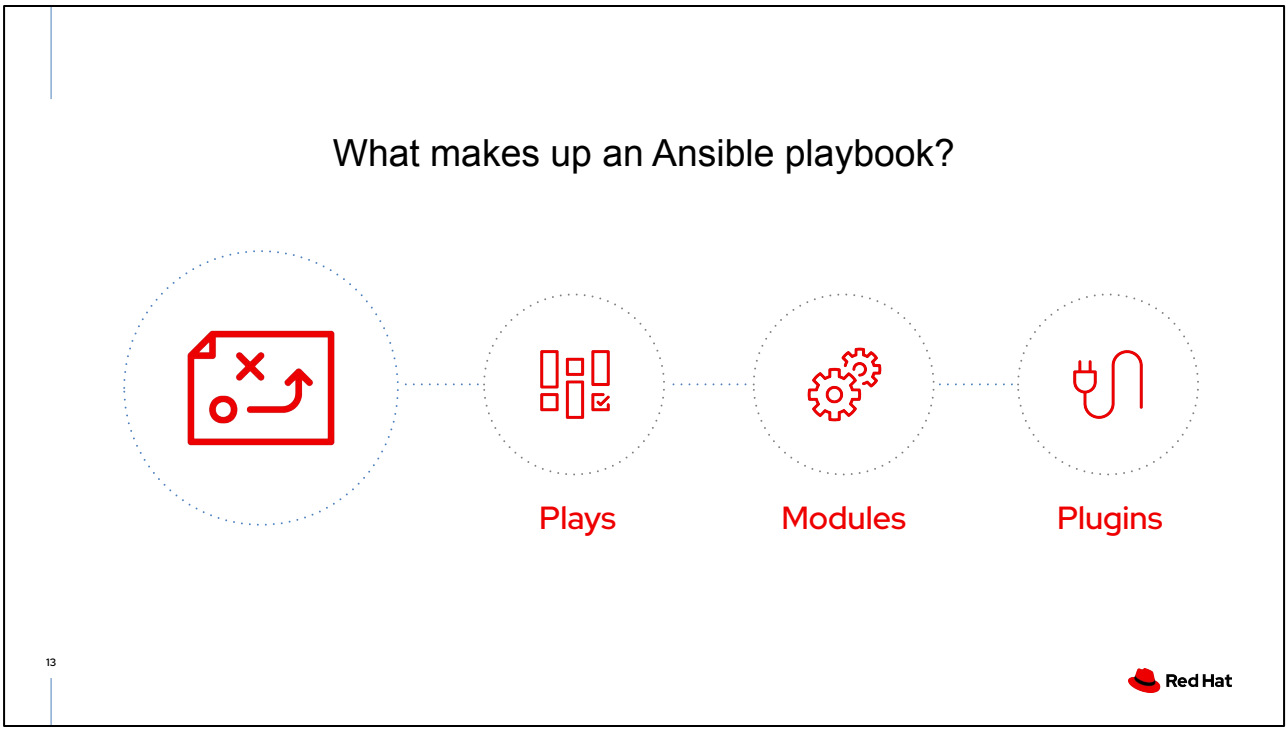
Red Hat
Ansible Automation
Platform



- Ansible.CFG / Inventory
- Modules / Plugins
- Playbooks / Ad-Hoc Commands
- Control Node / Managed Node

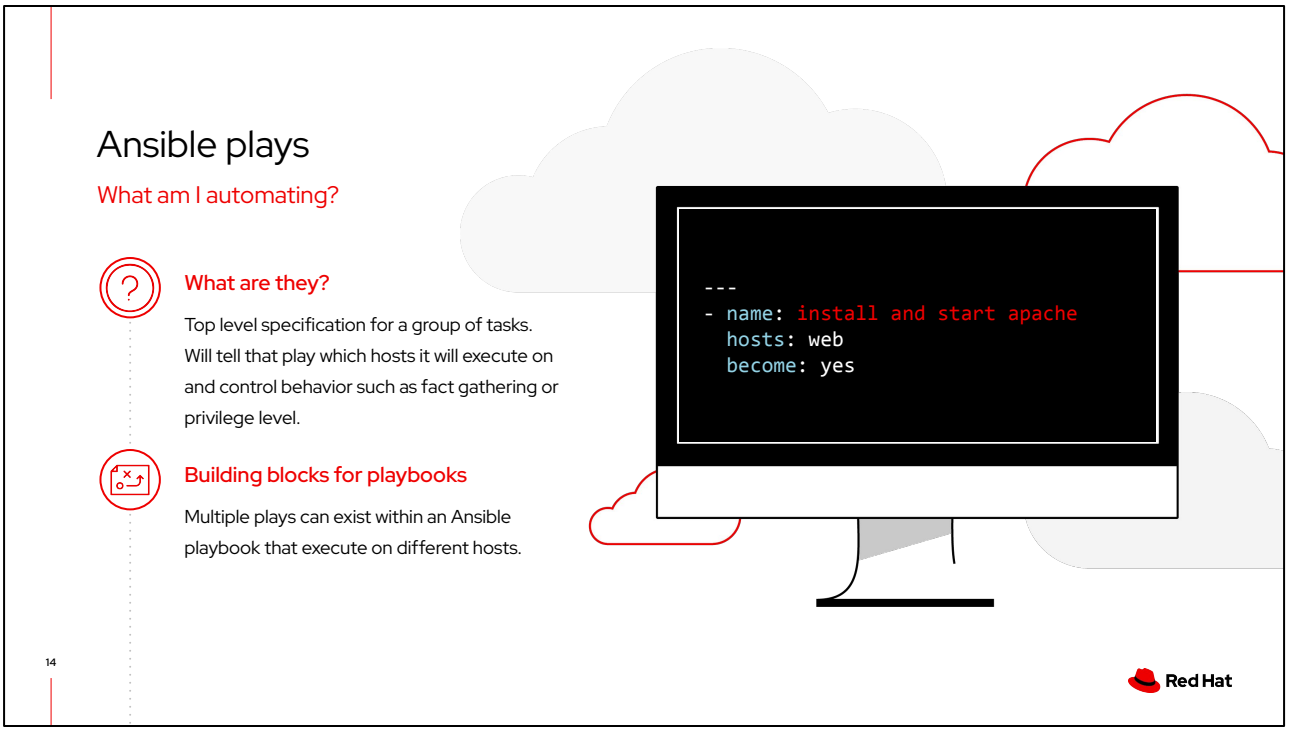

PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
PLAYBOOKS ARE WRITTEN IN YAML
Tasks are executed sequentially
Invoke Ansible modules
HOSTS
MODULES
PLUGINS
NETWORK DEVICES
ANSIBLE PLAYBOOK

Red Hat



So what does a playbook consist of? There are three main pieces a playbook consists of: plays, modules and plugins. We will look at all components one by one in the next slides. There will be more questions down the line though: How to make playbooks reusable - and how to package and version control all mentioned components? This will be answered as well, stay tuned!

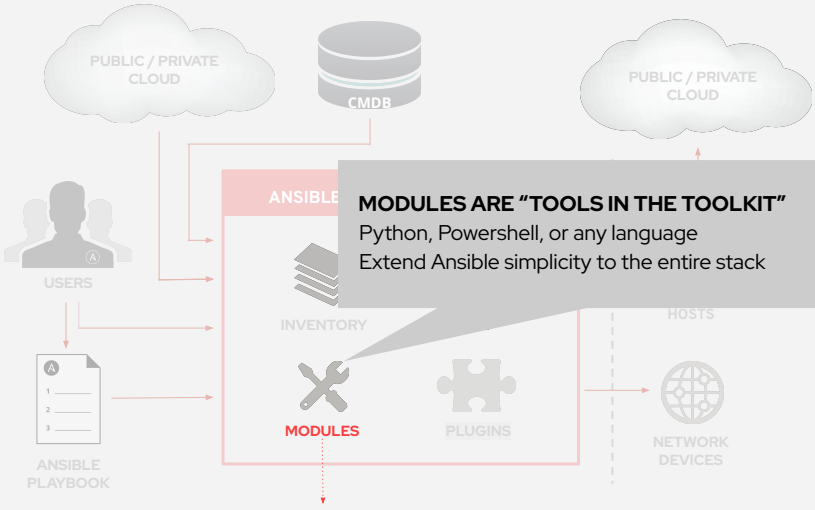
Ending question: So where do we start, what is a play?



A playbook is composed of one or more ‘plays’ in an ordered list. The terms ‘playbook’ and ‘play’ are sports analogies. Each play executes part of the overall goal of the playbook. The play usually defines which machines are affected by the keyword “hosts”. Also connection details are defined, here that the connection should use a privilege escalation via “become”. In the beginning it is not uncommon that a playbook only has one play.
Each play runs one or more tasks. Each task calls an Ansible module.

Ending question: What are the modules?

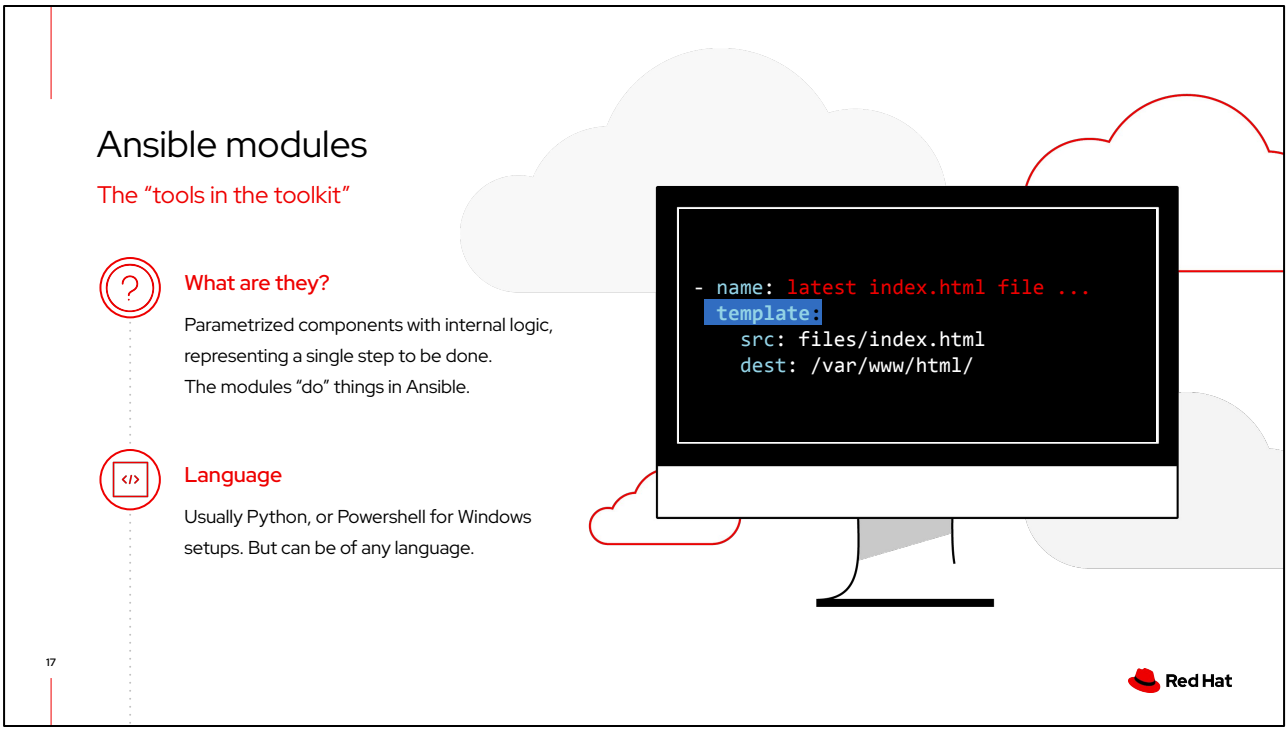
```
---
- name: install and start apache
  hosts: web
  become: yes

  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/
```

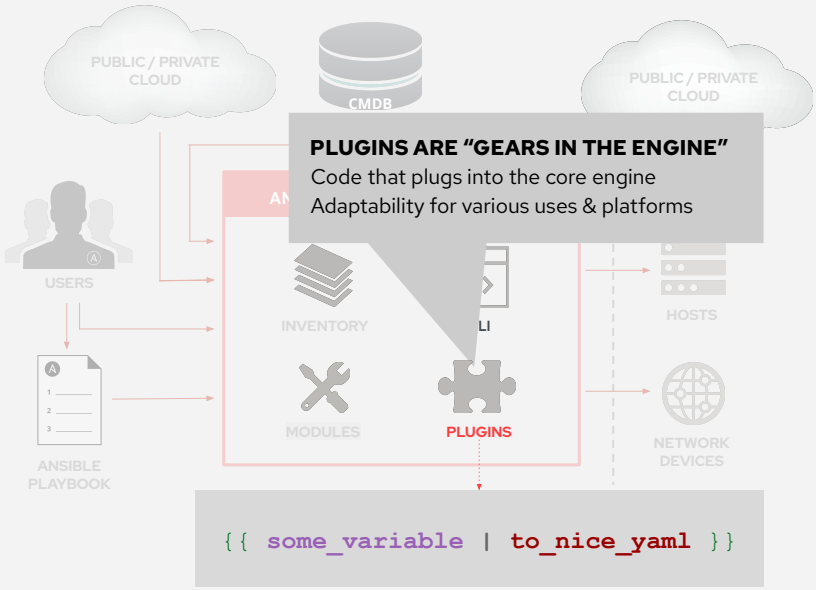
Red Hat



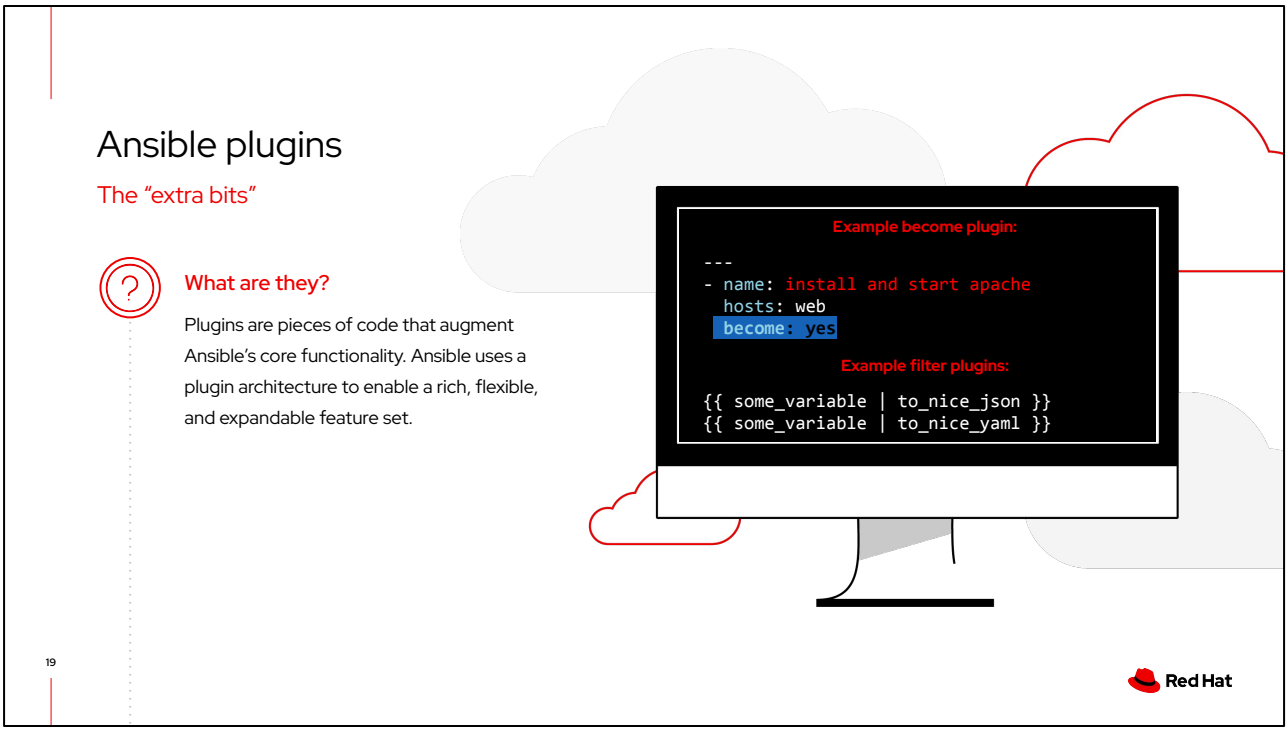
What are modules? What they are made of?
They are parametrized components with internal logic, representing a single step to be done. The modules “do” things in Ansible.
Usually they are written in Python, or Powershell for Windows setups. But they can be of any language.

Value: Flexibility, just like lego building blocks, to write the automation step by step they need.
Example: If the need to automate certain tasks in VMWare, they just check out the modules they need, write them underneath, add the parameters, and are done.

Ending question: but are modules all that we have? Are there any other ways in which we can change the automation?

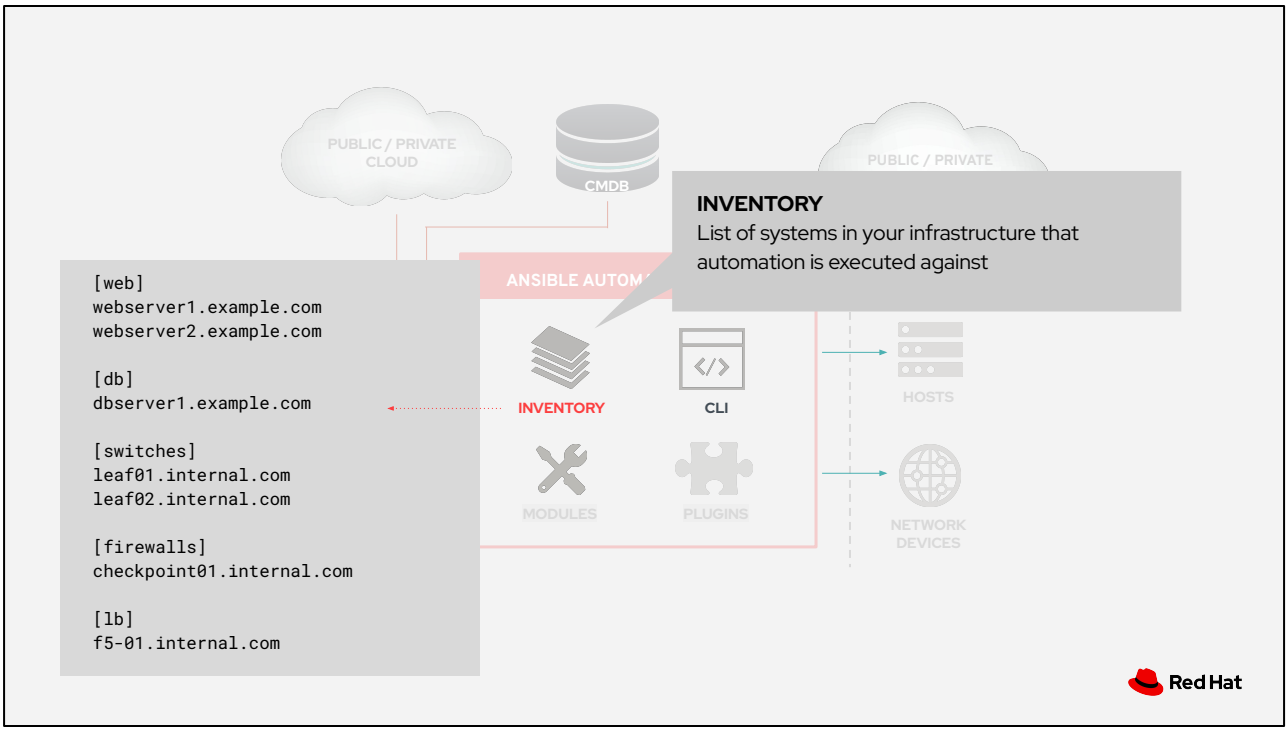

PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
PLUGINS ARE "GEARS IN THE ENGINE"
Code that plugs into the core engine
Adaptability for various uses & platforms
USERS
INVENTORY
HOSTS
MODULES
PLUGINS
NETWORK DEVICES
ANSIBLE PLAYBOOK
{{ some_variable | to_nice_yaml }}

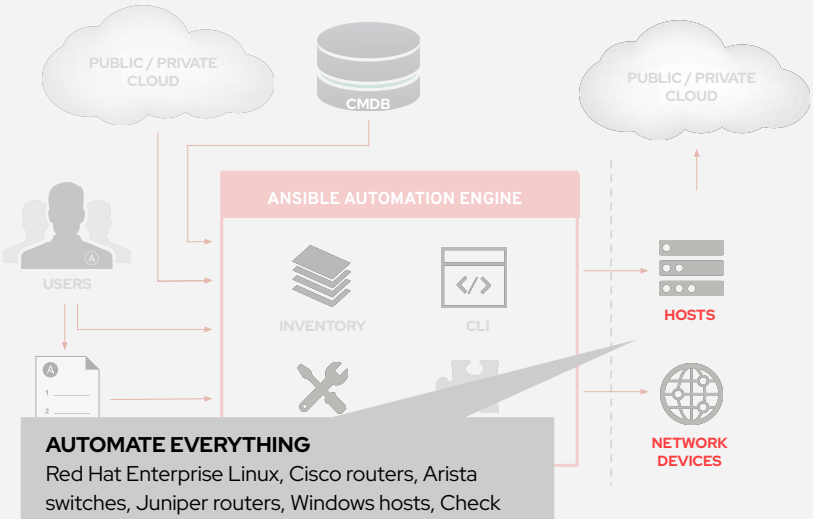
Red Hat



There are plugins. 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. Ansible ships with a number of handy plugins, and you can easily write your own. Filter plugins are a good example of plugins. They can transform or format whatever was piped into the filter
Value: Extending the automation, providing interfaces to bring in additional or custom functions.
Example: If there is a custom authorization system on place, a become plugin might help to implement support for that in Ansible.

Ending question: But how do we avoid that every new user of Ansible starts to write a new playbook for something we already solved? If each team has an Apache server, how do we avoid that they all write “their” playbook to automate Apache?


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE
INVENTORY
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
ANSIBLE AUTOM
INVENTORY
CLI
HOSTS
MODULES
PLUGINS
NETWORK DEVICES
Red Hat



AUTOMATE EVERYTHING

Red Hat Enterprise Linux, Cisco routers, Arista switches, Juniper routers, Windows hosts, Check Point firewalls, NetApp storage, F5 load balancers and more

Red Hat

LINUX AUTOMATION

150+

Linux Modules

AUTOMATE EVERYTHING LINUX

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

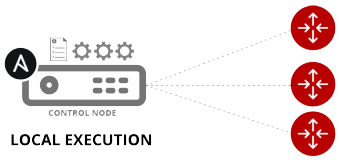
ONLY REQUIREMENTS:
Python 2 (2.6 or later)
***or* Python 3 (3.5 or later)**

ansible.com/get-started

Red Hat

How Ansible Automation works

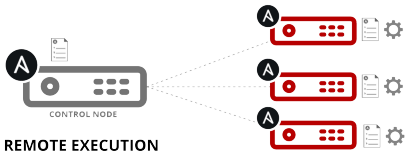
Module code is executed locally on the control node



LOCAL EXECUTION

NETWORKING DEVICES

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

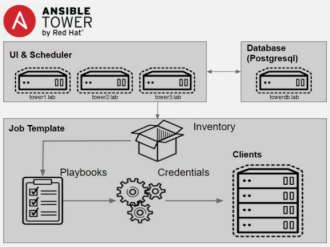


REMOTE EXECUTION

LINUX/WINDOWS HOSTS

Red Hat

Ansible Tower



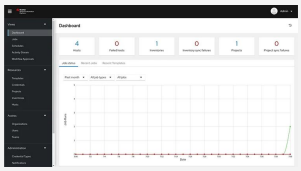
Integrated

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

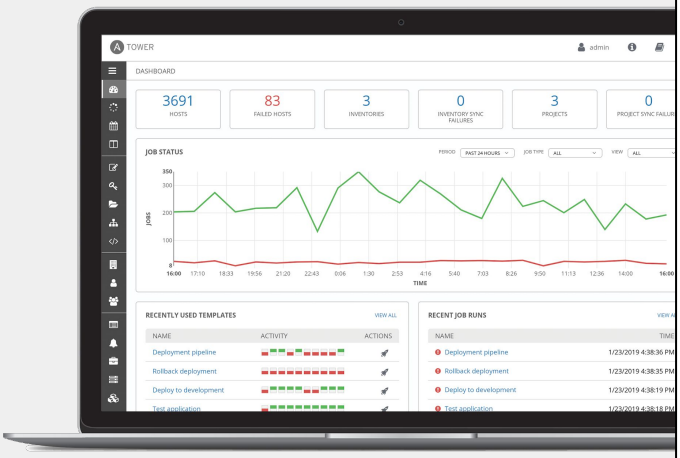
More Efficient & More Secure

Red Hat

What is Ansible Tower?

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

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

Integrate with enterprise authentication like TACACS+, RADIUS, Azure AD. Setup token authentication with OAuth 2. Setup notifications with PagerDuty, Slack and Twilio.

Centralized logging

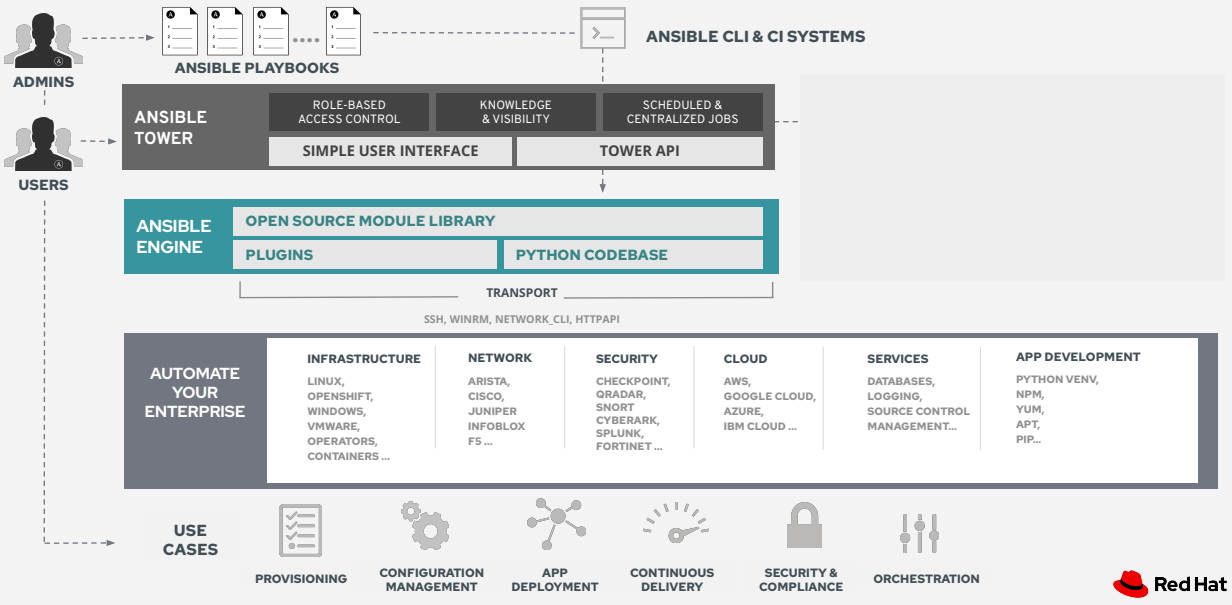
All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Ansible Tower's API.

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.

Red Hat

Ansible Automation Engine



Section 1.2

Topics Covered:

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

Red Hat
Ansible Automation
Platform

Inventory

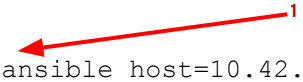
- Ansible works against multiple systems in an inventory
- Inventory is usually file based
- Can have multiple groups
- Can have variables for each group or even host

Understanding Inventory - (Simple Inventory)

```
# Static inventory example:
[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

```
[app1srv]
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
```



- Inventory can be written in short format and expanded using **[x:y]** syntax

```
[web]
Node-1 ansible_host=10.42.0.31
Node-2 ansible_host=10.42.0.32

...

Node-30 ansible_host=10.42.0.60
```

Red Hat

Understanding Inventory - Variables

```
[app1srv]
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
```

Red Hat

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):

- ANSIBLE_CONFIG (environment variable if set)
- ansible.cfg (in the current directory)
- ~/.ansible.cfg (in the home directory)
- /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 `ping`

```
# 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

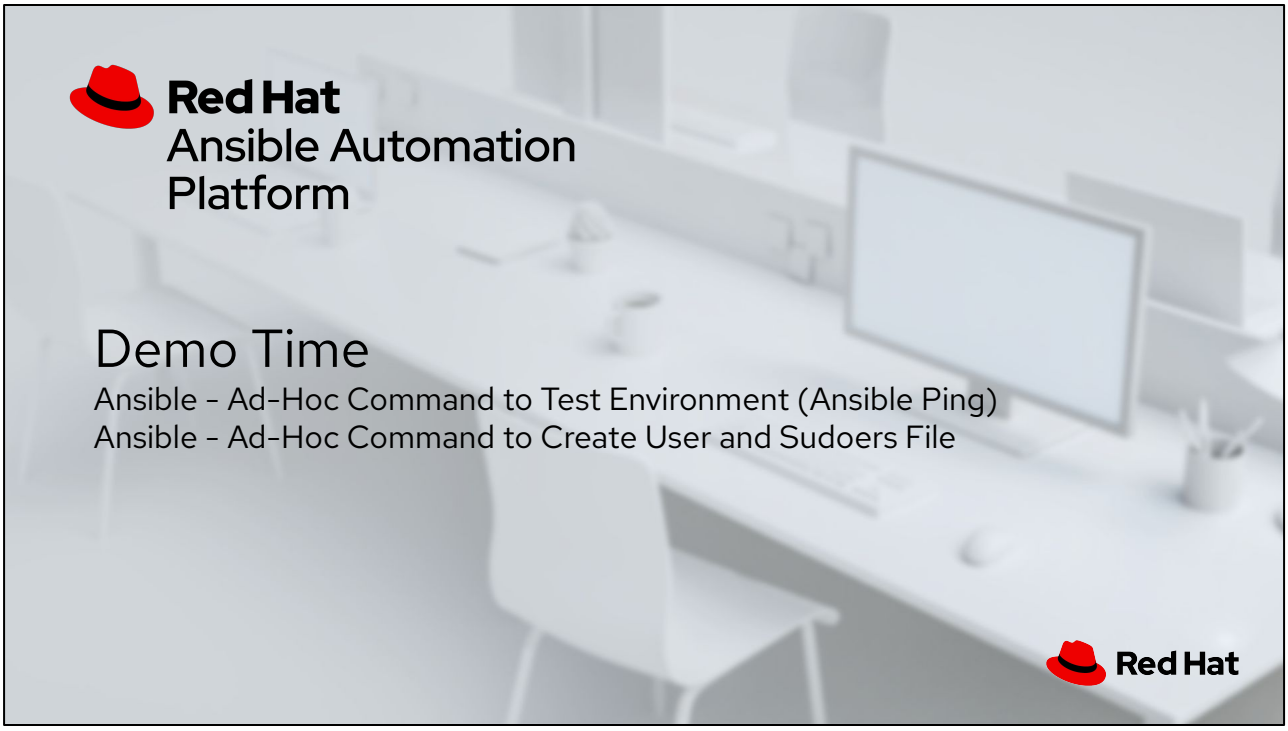
`-b, --become`
Run ad-hoc command with elevated rights such as sudo, the default method

`-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
```


Red Hat
Ansible Automation
Platform
Demo Time
Ansible - Ad-Hoc Command to Test Environment (Ansible Ping)
Ansible - Ad-Hoc Command to Create User and Sudoers File
Red Hat

Section 1.3

Topics Covered:

- Playbooks basics
- Running a playbook

Red Hat
Ansible Automation
Platform

Red Hat

An Ansible Playbook

A play

```
---
- 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

A task

```
---
- 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

module

```
---
- 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
```

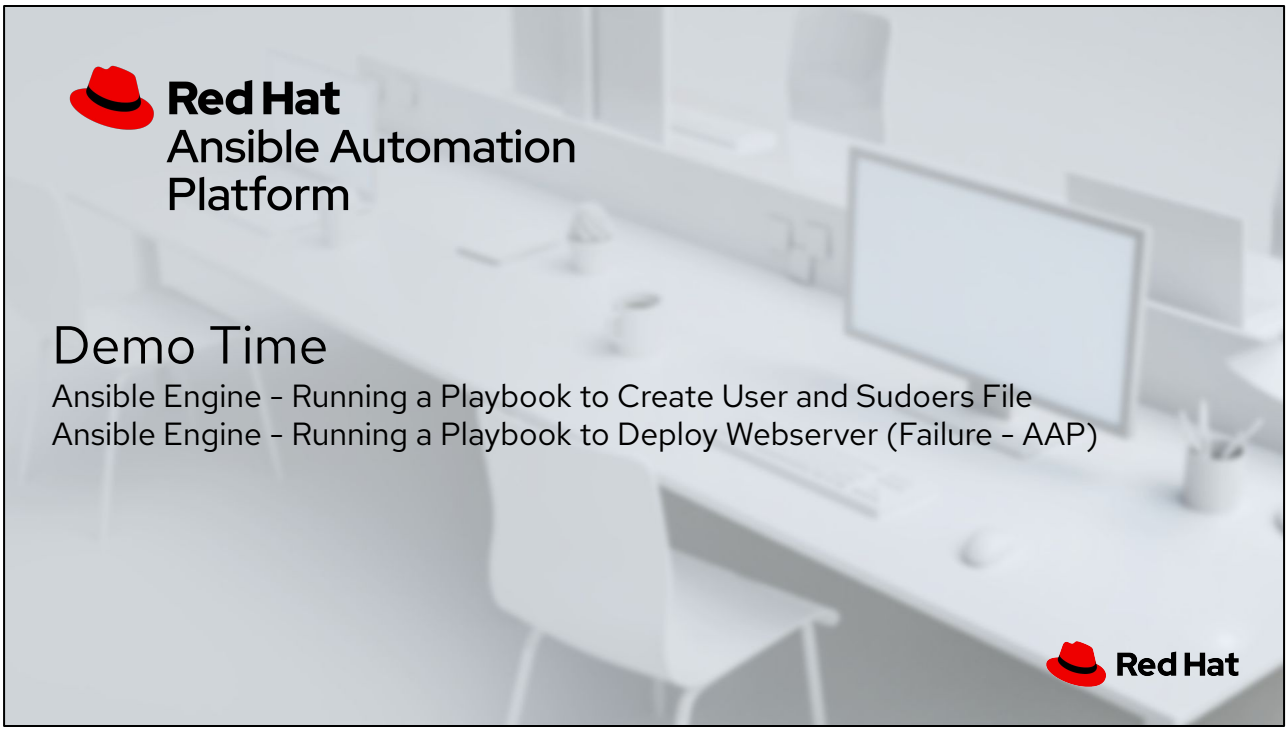
Running an Ansible Playbook:

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
```



Demonstrate Failure from past playbooks. Explain why this failure occurred.

If time permits - share demo on a system with Ansible version < 2.9

Section 1.4

Topics Covered:

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

Red Hat
Ansible Automation
Platform

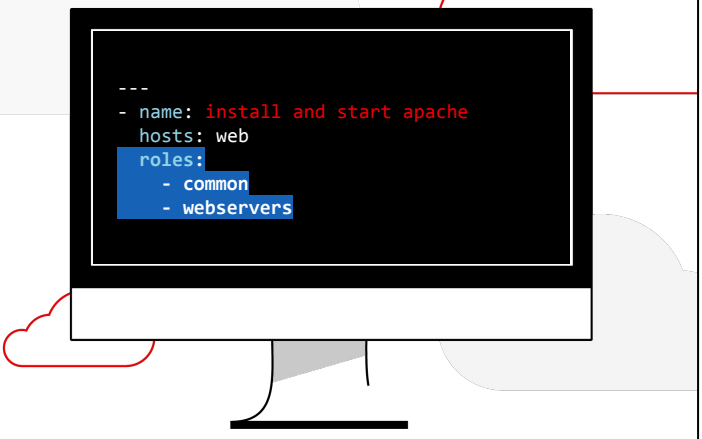
Red Hat

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.



Red Hat

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

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

Playbook using a Role

```
---
- name: Install Packages
  hosts: web
  become: yes
  roles:
    -  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

Role structure

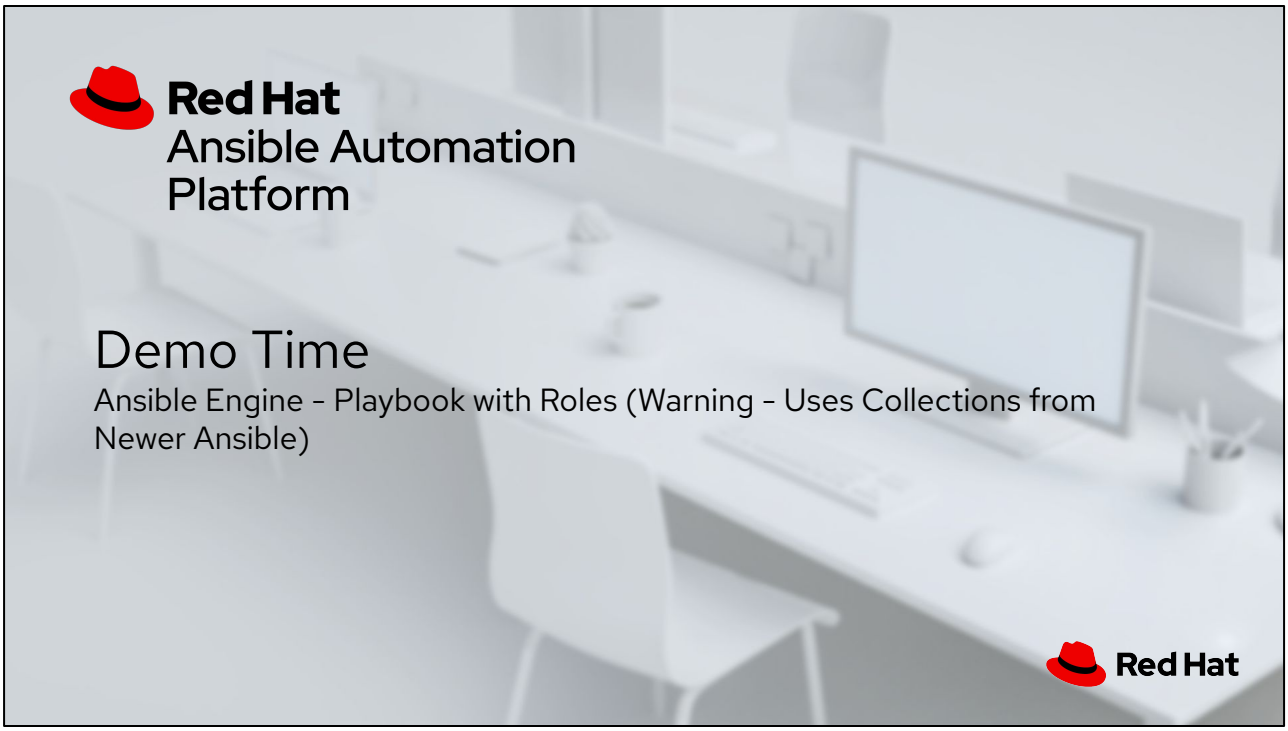
- **Defaults**: default variables with lowest precedence (e.g. port)
- **Files**: contains files that are deployed
- **Handlers**: contains all handlers
- **Meta**: role metadata including dependencies to other roles.

  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)

```
role_name/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
```


Ansible Galaxy
Sharing Content
Community
Roles, and more



Use my Gitlab/Ansible Galaxy Roles:

Github URLs:
https://github.com/tmichett/deploy_packages
https://github.com/tmichett/manage_firewall
https://github.com/tmichett/manage_services

Ansible Galaxy URLs:
https://galaxy.ansible.com/tmichett/deploy_packages
https://galaxy.ansible.com/tmichett/manage_firewall
https://galaxy.ansible.com/tmichett/manage_services

Explain why collections are needed here ….

Section 2
Ansible Automation Platform 1.2

Red Hat
Ansible Automation
Platform

Present

Red Hat

Courses on Ansible

RH294
DO447

Section 2.1

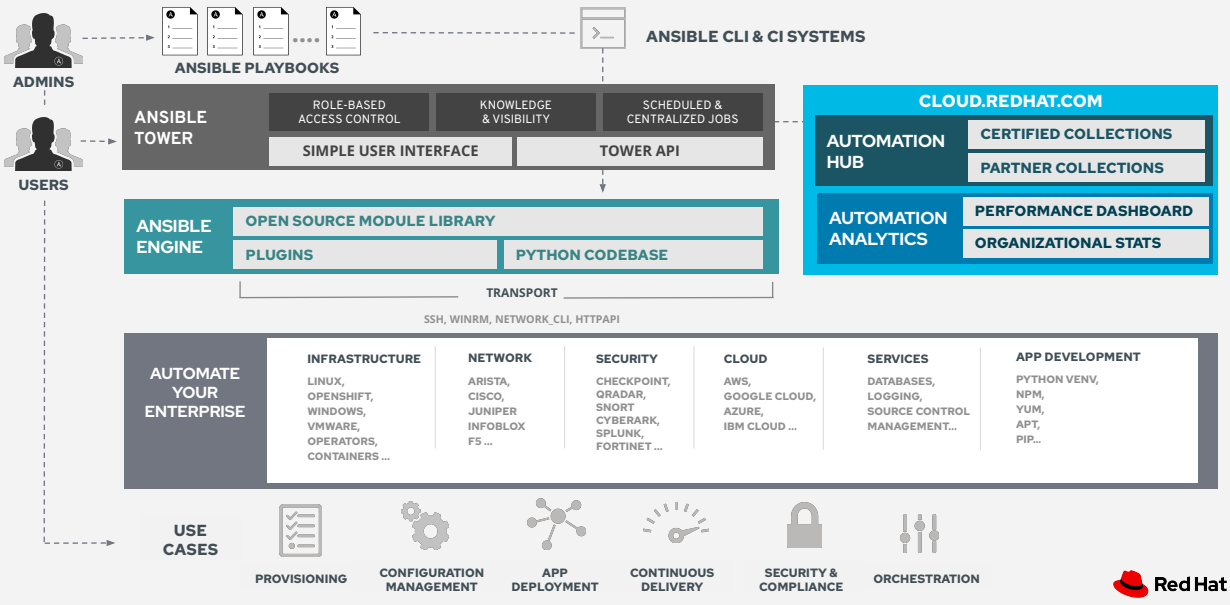
Topics Covered:

- Ansible Automation Hub
- Ansible Collections

Red Hat
Ansible Automation
Platform

Red Hat

Ansible Automation Platform



Ansible Automation Hub

Trusted source

Customer controlled

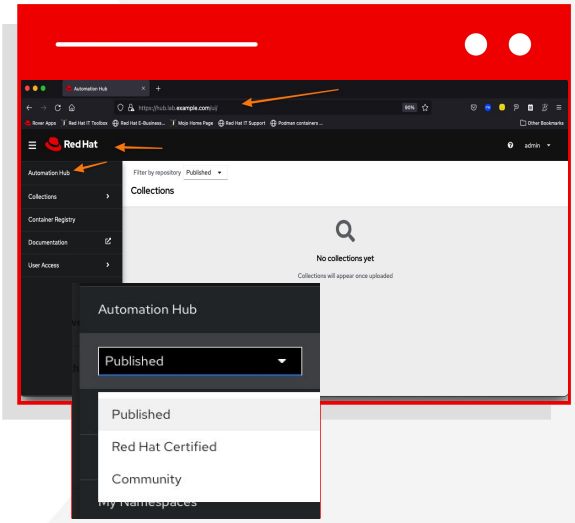
Deploying either on-prem or to a cloud, customers can run their own private instances of Automation Hub integrated into Red Hat Ansible Automation Platform.

Private content

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

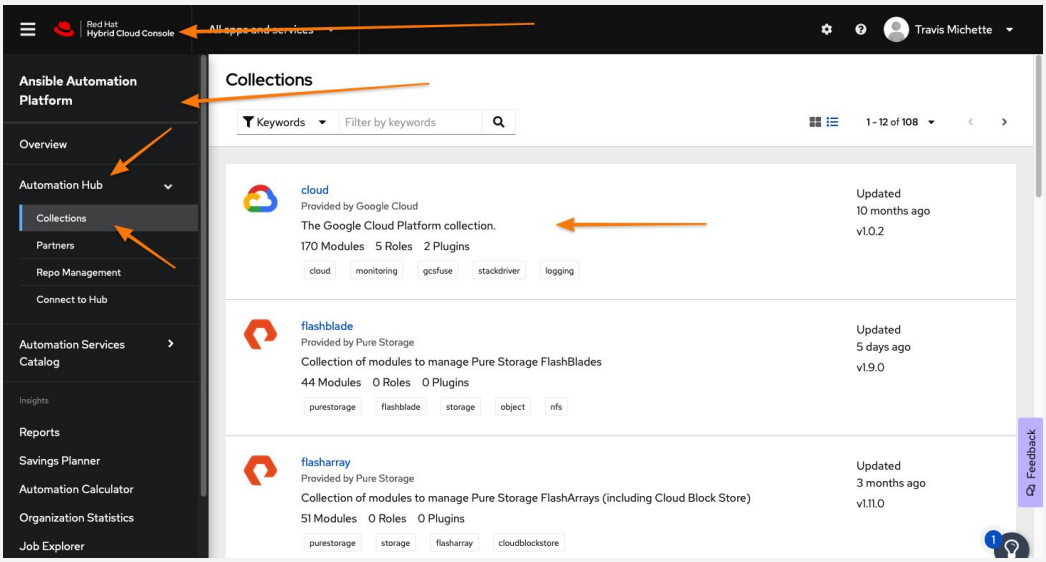
Customizable Content Catalog

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



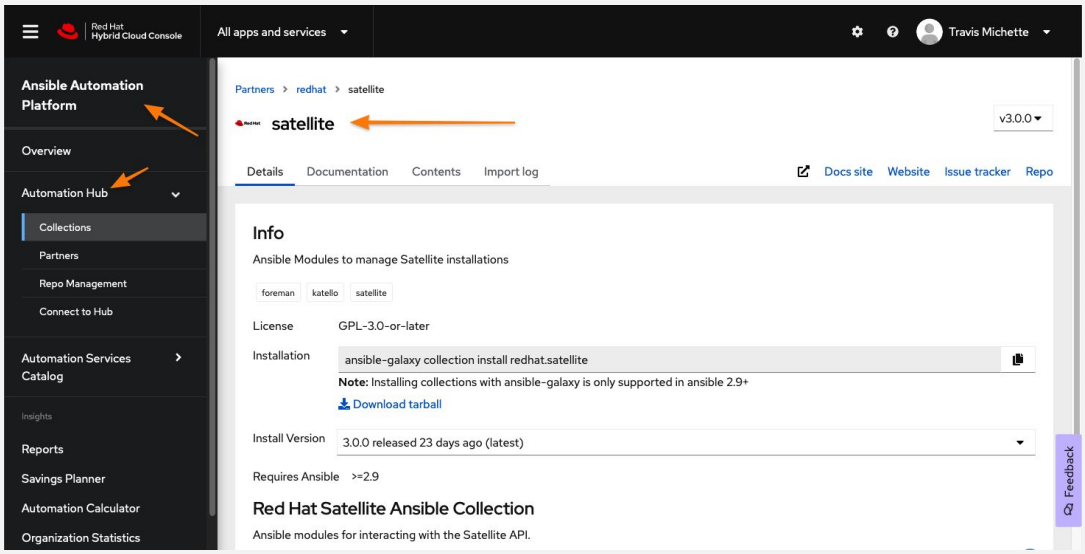
Red Hat

Ansible Automation Hub



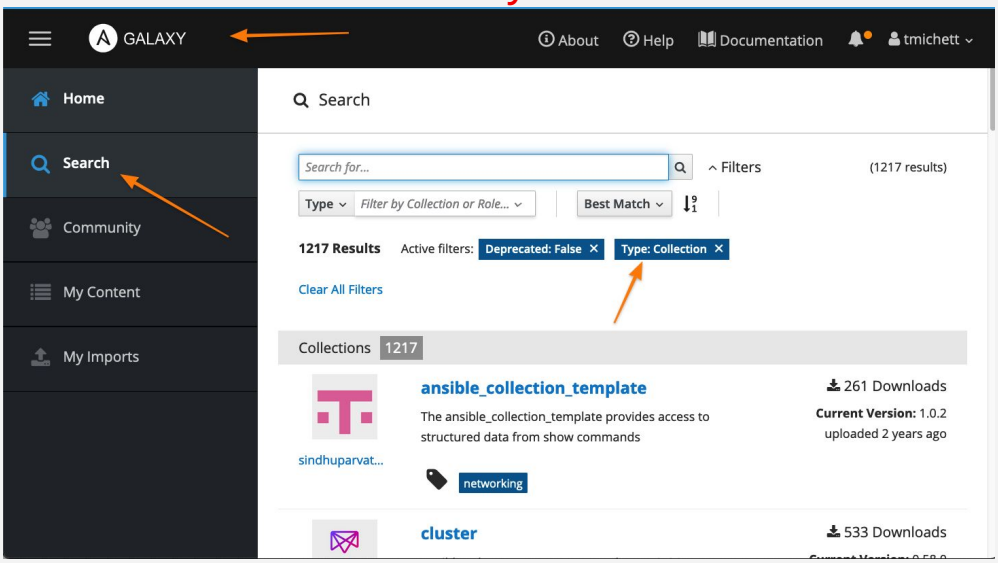
Red Hat

Ansible Automation Hub Collections



Red Hat

Ansible Galaxy Collections



Red Hat

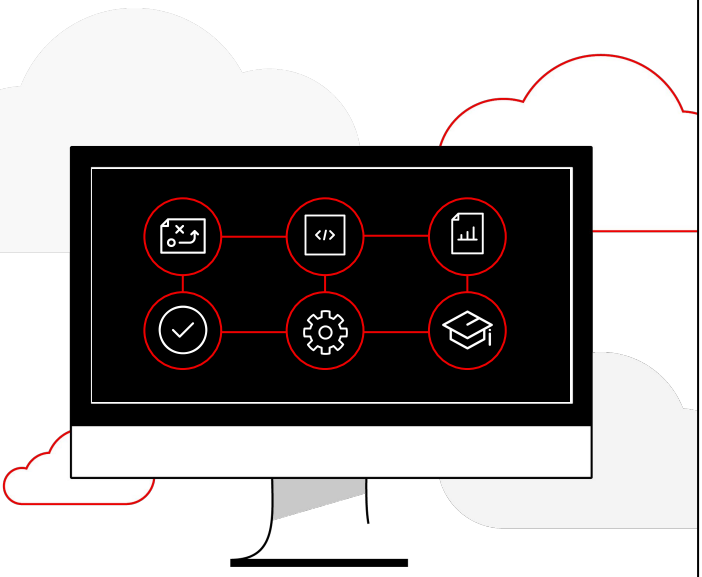
Collections

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?

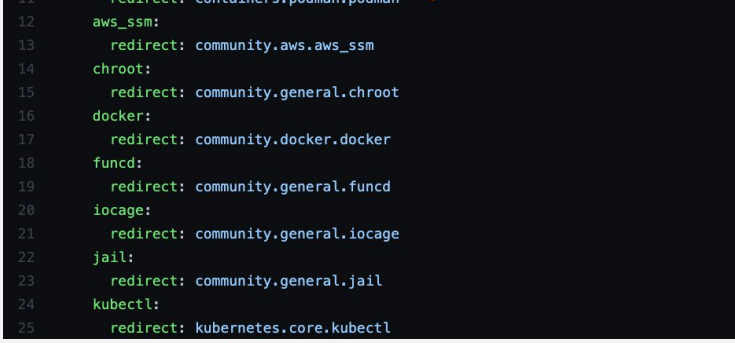
- Ansible 2.9 introduced the concept of collections and provided mapping for Ansible modules that were moved into a collection namespace.
- Collections allowed development of Ansible core components to be separated from module and plug-in development.
- Upstream Ansible unbundled modules from Ansible 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/**

Red Hat

One of the problems collections help solve is separation of development with Ansible core components vs. automation modules. Now Ansible modules and plugins can be grouped together by categories, maintained by vendors, or community developers without conflicting with other developers or the main Ansible project.

Collections and Changes to Ansible Modules

```
# 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
```



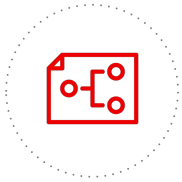
Red Hat

https://github.com/ansible/ansible/blob/devel/lib/ansible/config/ansible_builtin_runtime.yml

Show the mapping from Ansible Modules and their collections

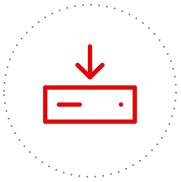
Accessing collections

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

Red Hat

Collections are brought into the entire structure via three ways: Requirements files next to your playbooks describe what collections should be present for the automation-content at hand. Automation controller can read those, or searches to Private Automation hub automatically for needed collections and pull them automatically.
There is also a CLI for downloading collections. We will look at the specifics of Automation controller and the CLI later.

Value: Consuming collections is done automatically in the background, there is no need to track or worry where they are and how uploaded what when where. Example: Collection is added for mycorp frontend: it can be automatically consumed via the configured paths.

Ending question: This is great! But what is if there is a new version of a collection? Or of Ansible? Or one of the dependencies? How do I keep certain versions of the stack here and other versions of the stack there? How do I test the stack?

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.

1

2

```
[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"
```

Red Hat

Show the mapping from Ansible Modules and their collections

Ansible Runtime Mapping
https://github.com/ansible/ansible/blob/devel/lib/ansible/config/ansible_builtin_runtime.yml

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
```

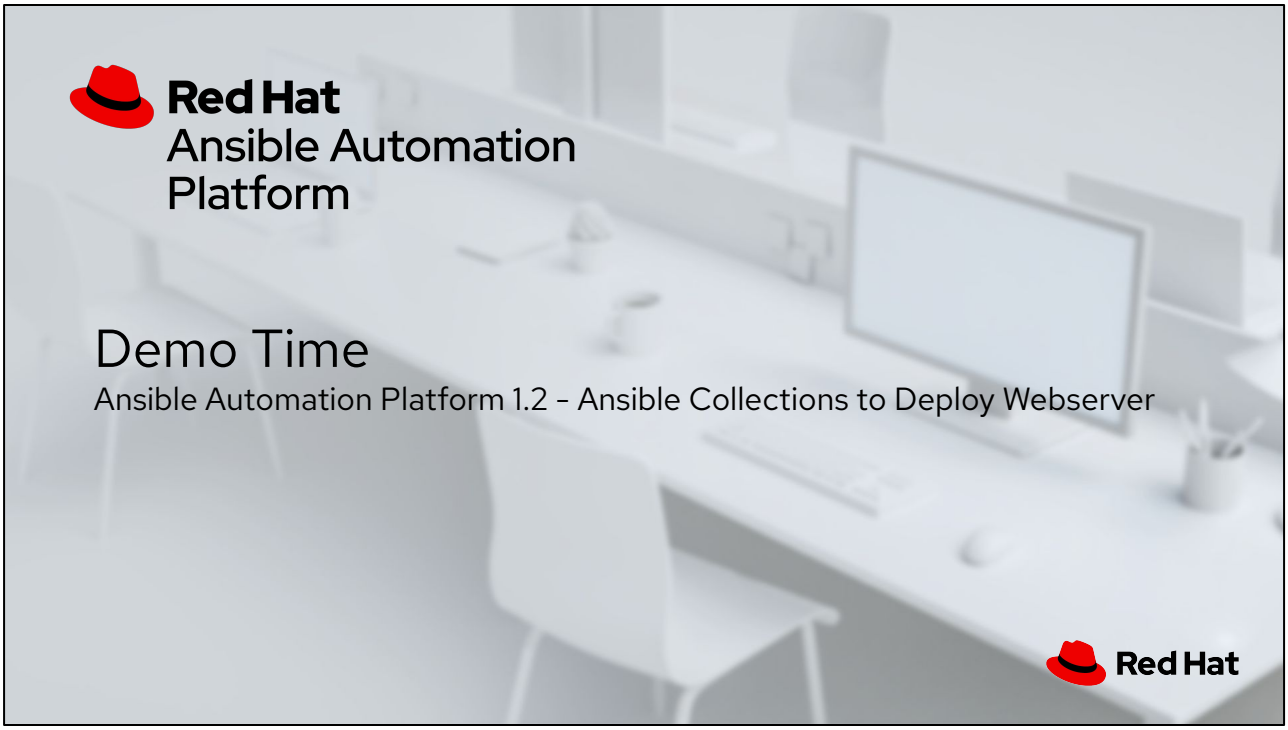
```
    - name: HTTP Service is Open on Firewall
      ansible.posix.firewalld: (1)
        service: http
        state: enabled
        permanent: true
        immediate: yes

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

- **firewalld** 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.

Red Hat

```
    - name: HTTP Service is Open on Firewall
        ansible.posix.firewalld: (1)
          service: http
          state: enabled
          permanent: true
          immediate: yes
```

(1) Preferred way to write task with collection modules using the FQCN. It would be possible to define and declare the collection at the beginning of the playbook so you could still just reference **firewalld** as a module, but that is generally considered not best practices.


Red Hat
Ansible Automation
Platform
Demo Time
Ansible Automation Platform 1.2 - Ansible Collections to Deploy Webserver
Red Hat

Break Time

Red Hat
Ansible Automation
Platform

Red Hat

Section 3
Ansible Automation Platform 2.x

Red Hat
Ansible Automation
Platform

Future

Red Hat

Courses on Ansible

DO374
DO467 (Coming Soon …)

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

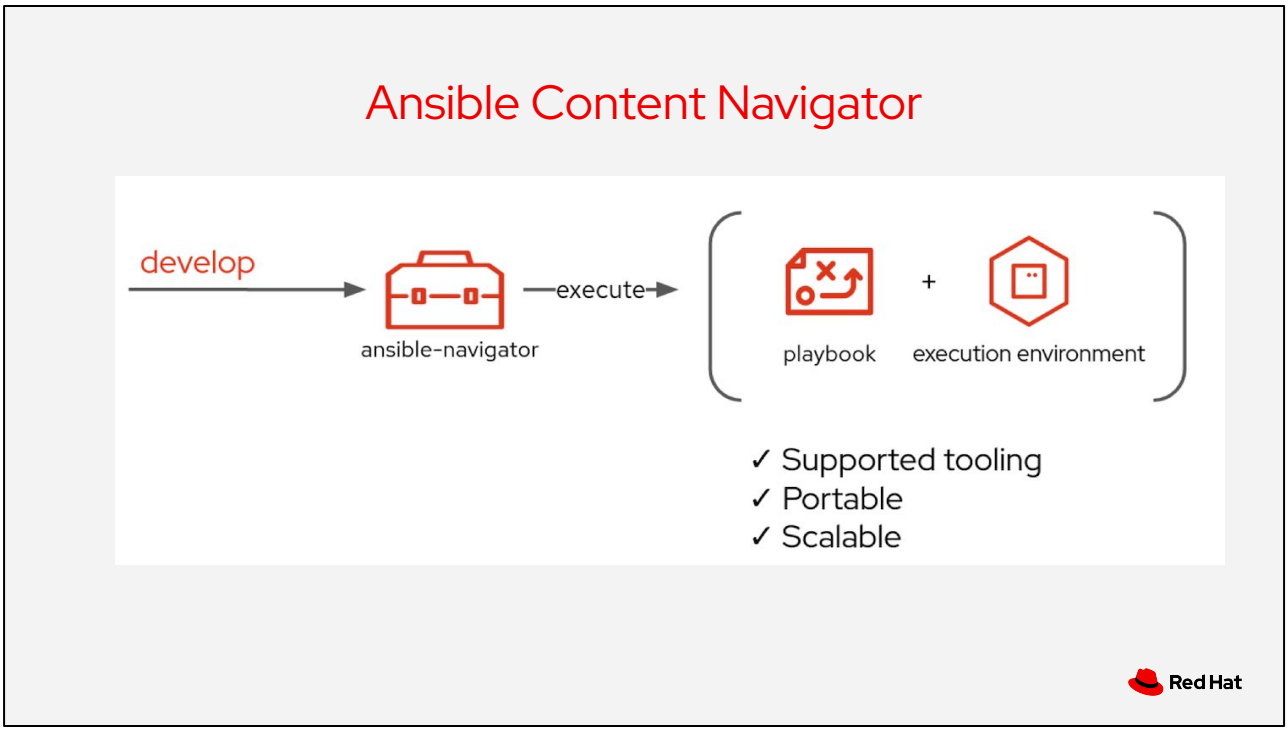
Section 3.1

Topics Covered:

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

Red Hat
Ansible Automation
Platform

Red Hat



Ansible Automation Platform 2 provides Ansible Core 2.11 as well as several Red Hat Certified Content Collections providing a similar experience to Ansible 2.9. A benefit of execution environments is that they can be used to run older versions of Ansible. Some of the demos in this webinar will use an Ansible EE based on Ansible 2.9 so that collection mapping is done automatically and older playbooks don't need to be touched.

The **ansible-navigator** tool replaces and extends the functionality of the **ansible-playbook**, **ansible-inventory**, **ansible-config** commands and more.

Ansible Content Navigator

Ansible Command	Automation Content Navigator Subcommands
ansible-config	ansible-navigator config
ansible-doc	ansible-navigator doc
ansible-inventory	ansible-navigator inventory
ansible-playbook	ansible-navigator run

```
# 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
```

Red Hat

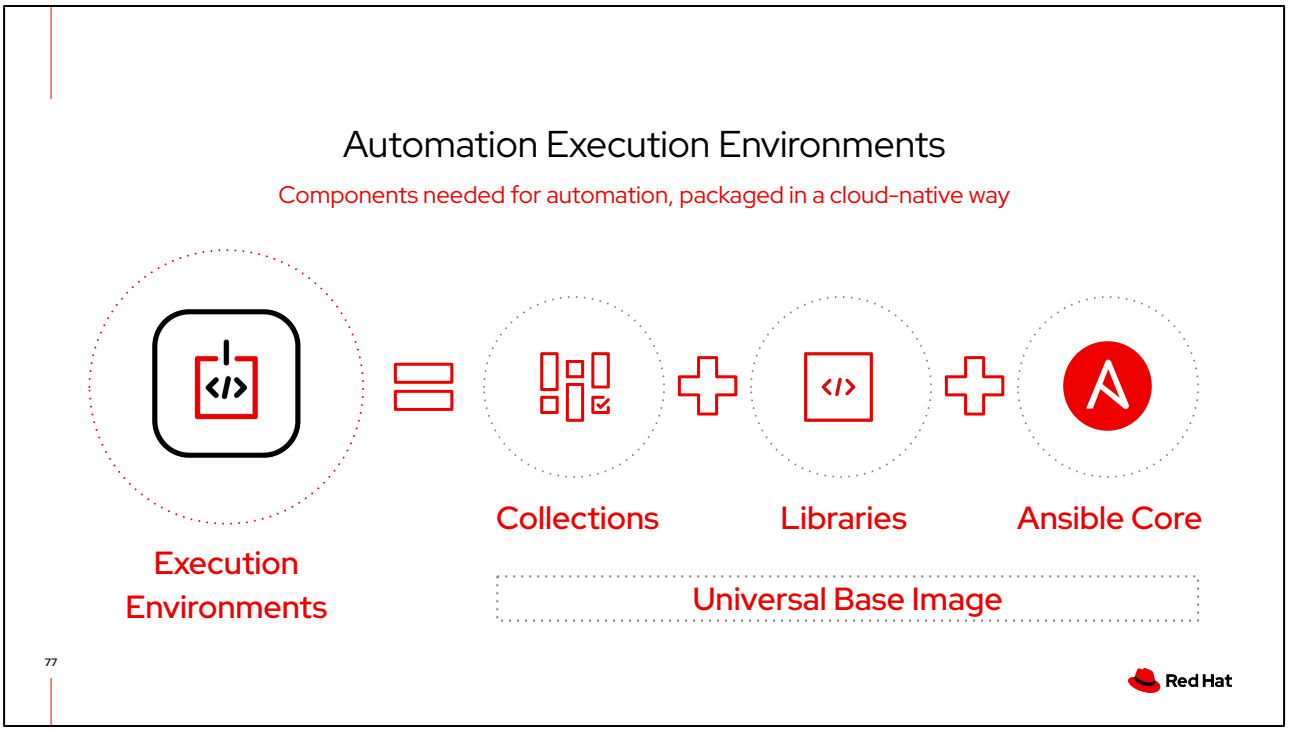
ansible-navigator.yml

```
---
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)**

Red Hat

Ansible Content Navigator is works very similar to **ansible** and **ansible-playbook** commands as it relies on a configuration file. The **ansible-navigator.yml** file utilizes the **ansible.cfg** file and further provides customizations for configurations on developing, testing, and running playbooks in your Ansible projec



The components can be kept together in a typical way most of people are familiar with today: a standardized container! Meet the Automation execution environments: containers that only package the components needed for automation, packaged in a cloud-native way.

Value: Standardized, cloud native package containing all the components of the automation, including the dependencies and the core Ansible version. This all based on the tested and trusted RHEL Universal Base Image.
Example: AWS EE, containing the aws collection, the necessary AWS Python libraries and the Ansible version the team has already tested.

Ending question: Now we have the format to package and ship it in ways most people are familiar with. How do we actually built those?

Ansible Execution Environments

```
  EE-29-RHEL8:LATEST (PRIMARY)      DESCRIPTION
0 | Image information               Information collected from image inspection
1 | General information             OS and python version information
2 | Ansible version and collections  Information about ansible and ansible collections
3 | Python packages                  Information about python and python packages
4 | Operating system packages        Information about operating system packages
5 | Everything                       All image information
```

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

Red Hat

NOTICE: The EE of Ansible 2.9.2 doesn't show collections installed. This is for backwards compatibility of existing and older playbooks.

Ansible Execution Environments - SSH Keys

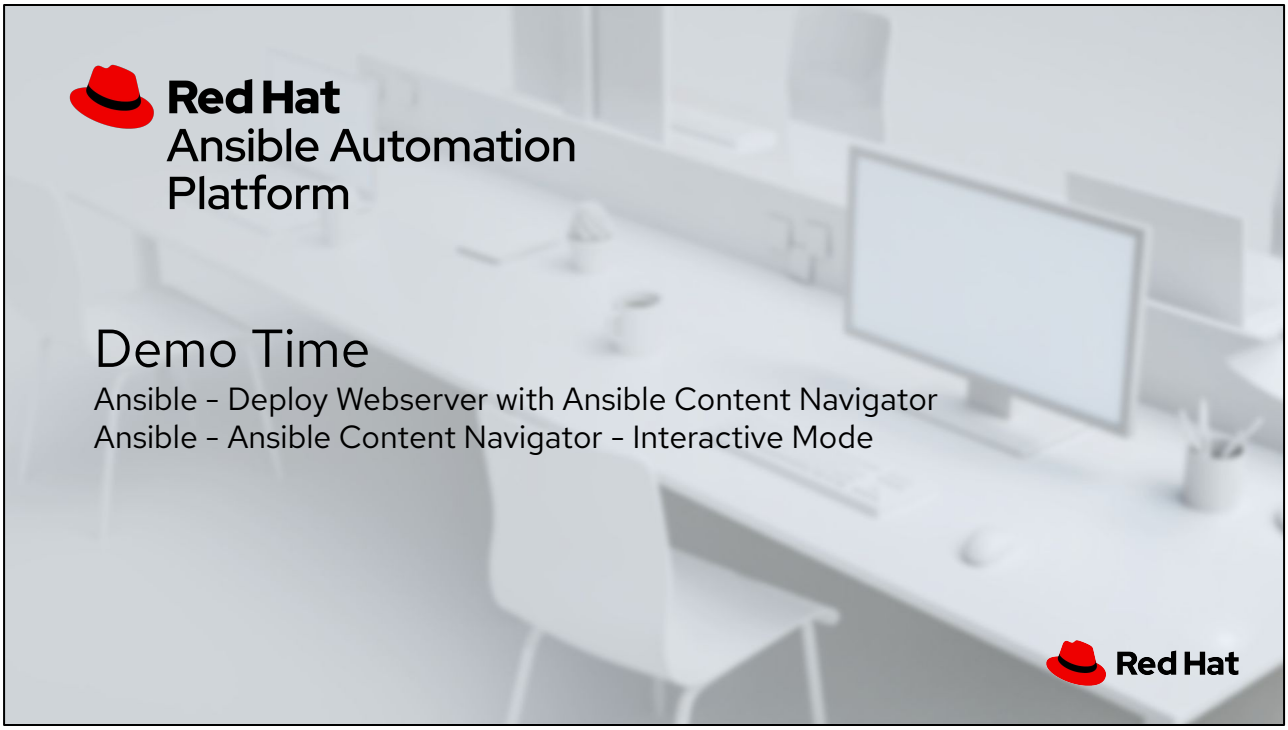
- **Execution Environment** - Leverages containers to run Ansible Playbooks
  - **Contains**
    - Ansible Core
    - Ansible Collections
    - Python Environment
  - **Requires**
    - Configuration Files
    - Inventory
    - SSH
      - SSH Keys must be provided through the SSH-Agent service

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

```
[student@workstation ~]$ ssh-add ~/.ssh/lab_rsa
```

Red Hat

SSH Agent is used to allow the SSH keys to pass through to the Ansible Execution Environment (EE) which is a container with all the Ansible and Python components required to run the Ansible Automation tasks.

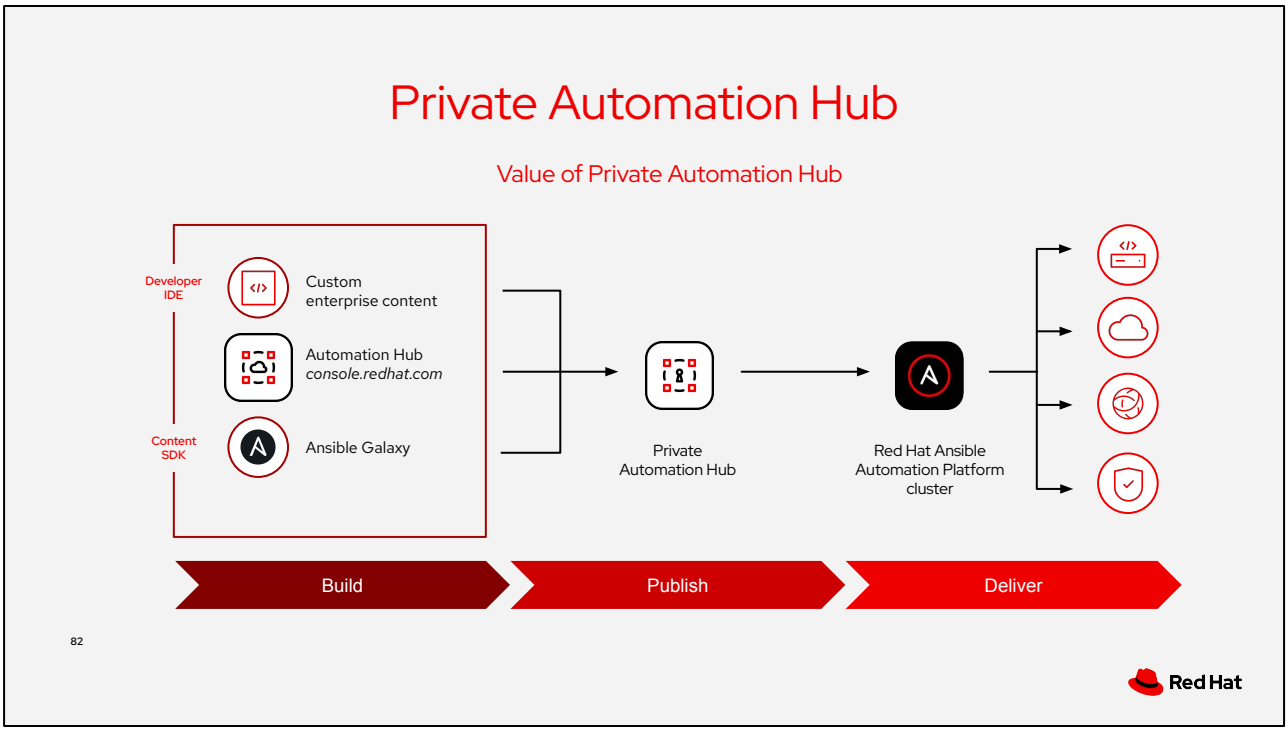

Red Hat
Ansible Automation
Platform
Demo Time
Ansible - Deploy Webserver with Ansible Content Navigator
Ansible - Ansible Content Navigator - Interactive Mode
Red Hat

Section 3.2

Topics Covered:

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

Red Hat
Ansible Automation
Platform



Private automation hub provides an on-site location for publishing collections and Ansible Execution Environments. This provides a centralized location for Automation Controller and for developers using Ansible Navigator to obtain EEs and content collections.

https://docs.ansible.com/ansible/latest/reference_appendices/automationhub.html

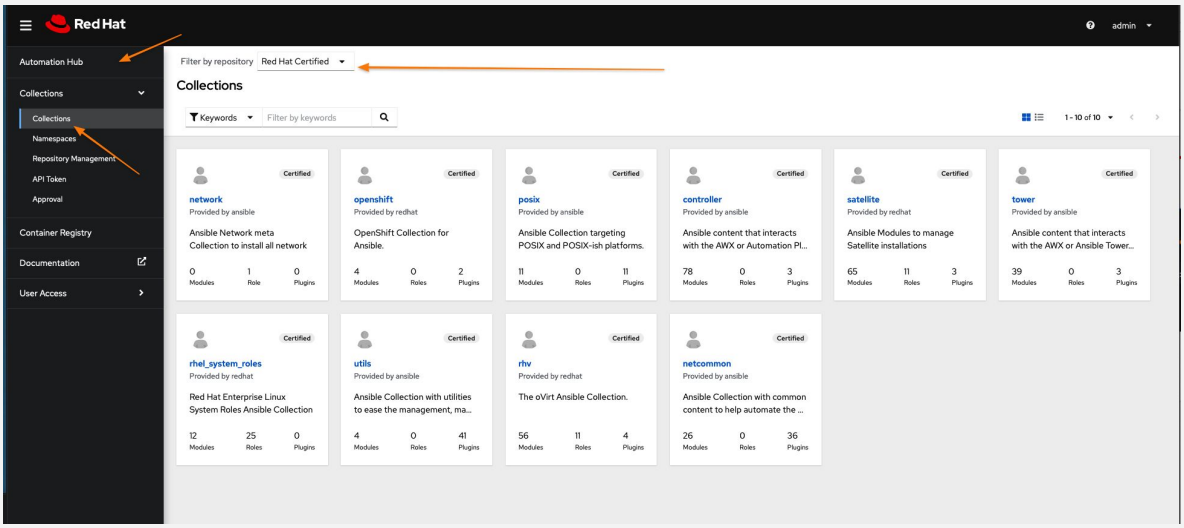
Content is provided on the left, either privately for example hosted on your Github repository, or from Red Hat and partners via Automation Hub, or via the community repository Ansible Galaxy.
This is imported into Private Automation Hub, and provided from there towards different components of the platform like Automation controller: there the content itself is accessed and downloaded, and Automation controller uses it to automate the target systems on the right.

value: clear workflow, enabling CI/CD pipelines and versioned views of content
example: Private Automation Hub can store multiple versions of the Check Point collection, for the different Check Point versions used in house.

Next question: How do the components like Automation controller actually pull

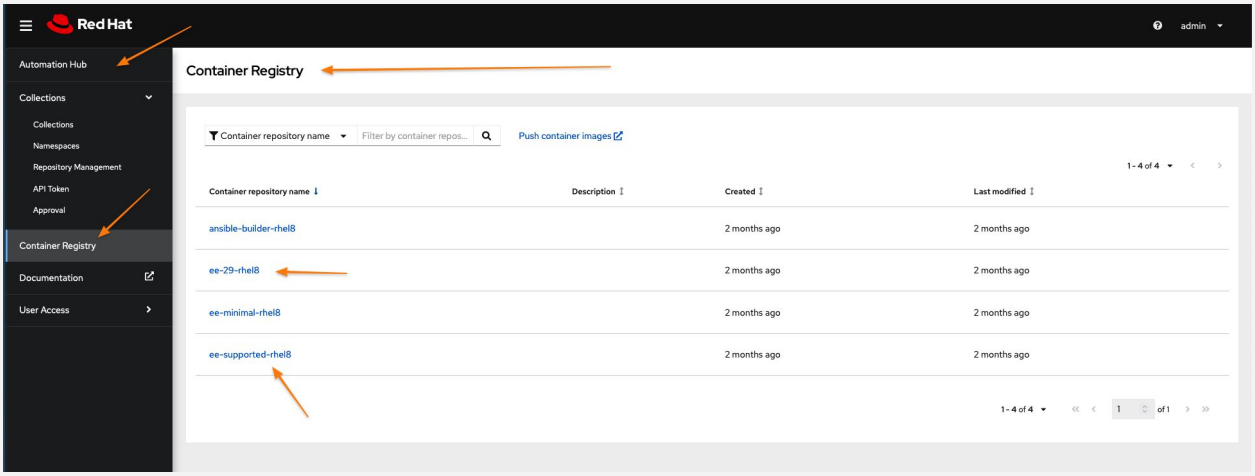
the collections code, in technical terms? How does it really work?

Automation Hub - Collections



Red Hat

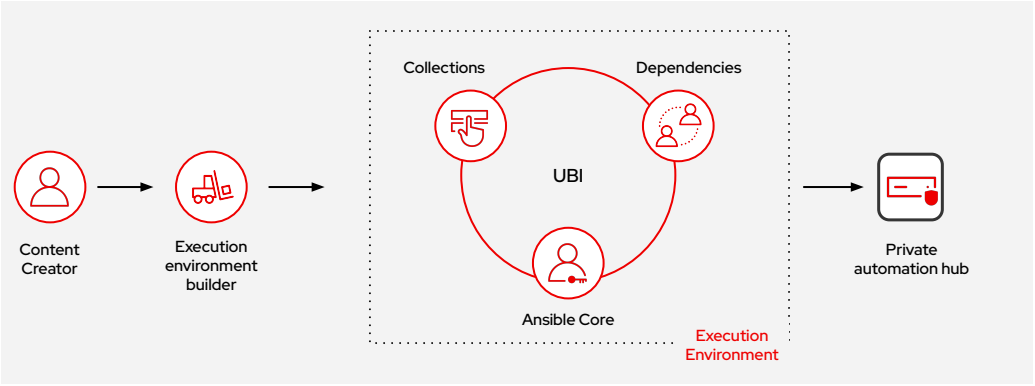
Automation Hub - Execution Environments



Red Hat

Build, create, publish

Development cycle of an automation execution environment



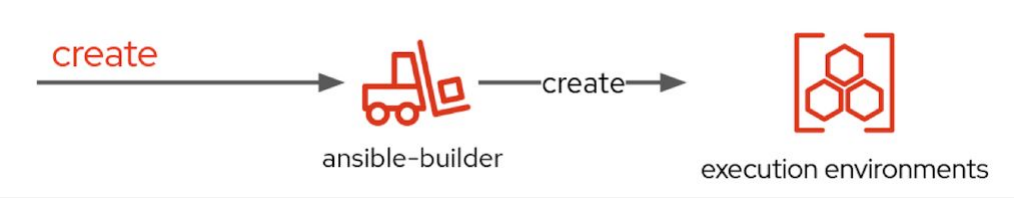
Red Hat

Building EEs is rather simple: we provide the tool execution environment builder, the CLI tool ansible-builder for this, which knows how to bring the pieces together in just the right way and create the EE. They can be published on Private Automation hub or in any container registry.

Value: Clear and simple tooling to let the creators focus on what they are best at: the automation content.
Example: The creators take the Check Point collection and dependencies and Ansible version, and just run ansible-builder. They don't have to worry about Dockerfiles.

Ending question: And how do we run automation content with EEs? ansible-playbook was never written with containers in mind!

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
```

Red Hat

Ansible Builder Blog:
https://www.ansible.com/blog/introduction-to-ansible-builder

Ansible Builder Images
https://blog.networktocode.com/post/ansible-builder-runner-ee/

https://quay.io/repository/ansible/ansible-runner?tag=latest&tab=tags

Ansible Execution Environments - Building/Customizing

execution-environment.yml **(1)**

```
---
version: 1
Build_arg_defaults:
  EE_BASE_IMAGE: 'hub.lab.example.com/ee-minimal-rhel8:2.0' (1a)
  EE_BUILDER_IMAGE: 'hub.lab.example.com/ansible-builder-rhel8:2.0' (1b)
dependencies:
  galaxy: requirements.yml (1c)(2)
  python: requirements.txt (1d)(3)
  system: bindep.txt (1e)(4)
```

requirements.yml **(1c)(2)**

```
---
collections:
  - name: /build/exercise.motd.tar.gz (2a)
    type: file
```

requirements.txt **(3)**

```
# Python dependencies
funmotd (3a)
```

bindep.txt **(4)**

```
# System-level dependencies
hostname (4a)
```

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
```

Red Hat

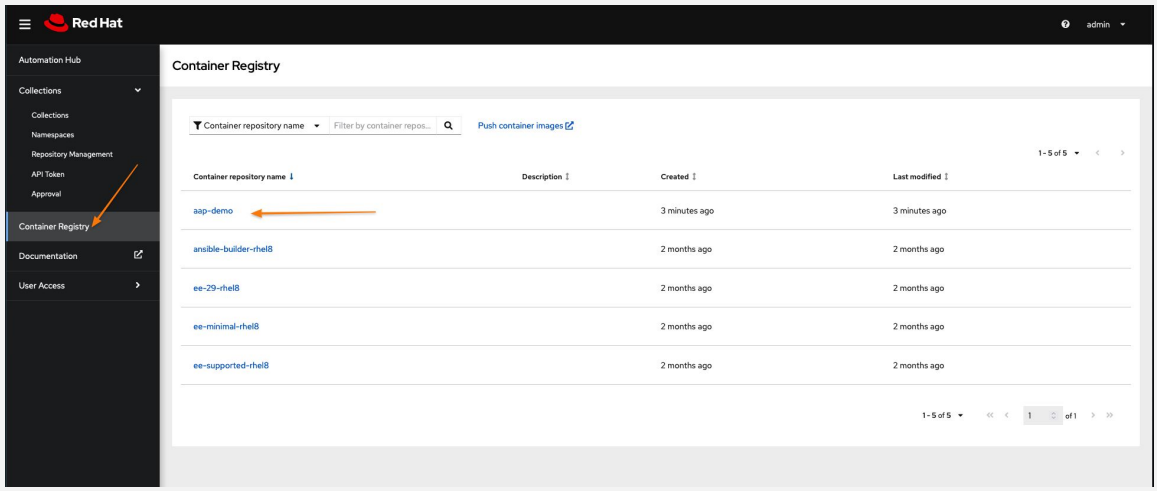
Ansible Execution Environments - Publishing

```
# 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
```

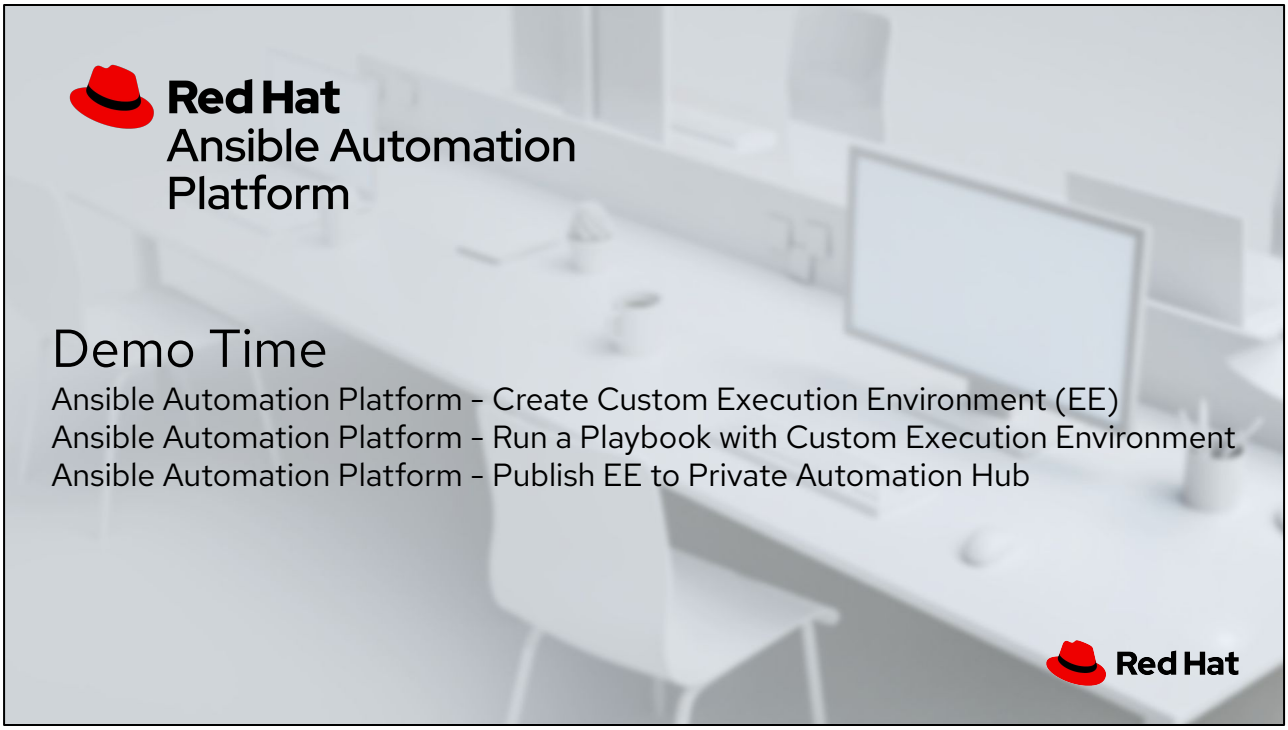
```
# Using podman to Push Image to Private Automation Hub
[user@ansible] $ podman push hub.lab.example.com/aap-demo:latest
```

```
# 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.



Red Hat



NOTE: Demos might need changed based on time !!!!

Section 3.3

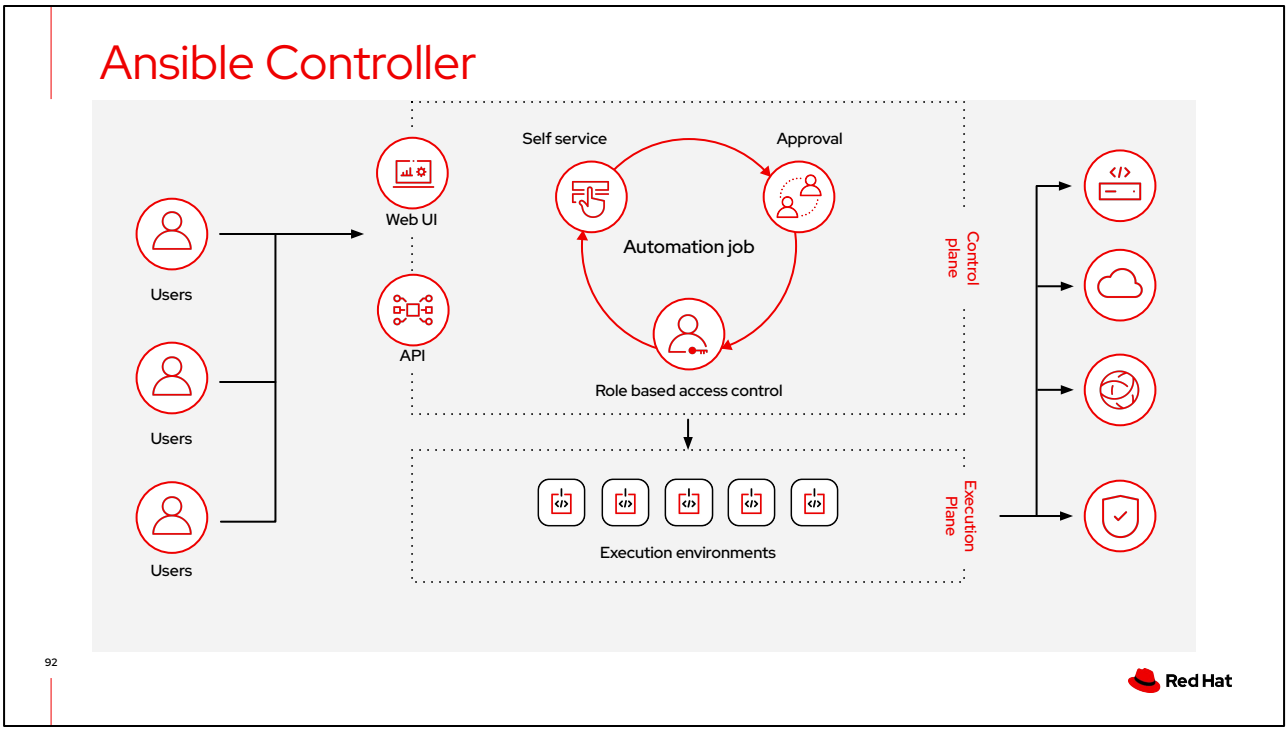
Topics Covered:

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

Red Hat
Ansible Automation
Platform

Red Hat

Emphasize this is the new replacement for Ansible Tower. Leverages EEs.

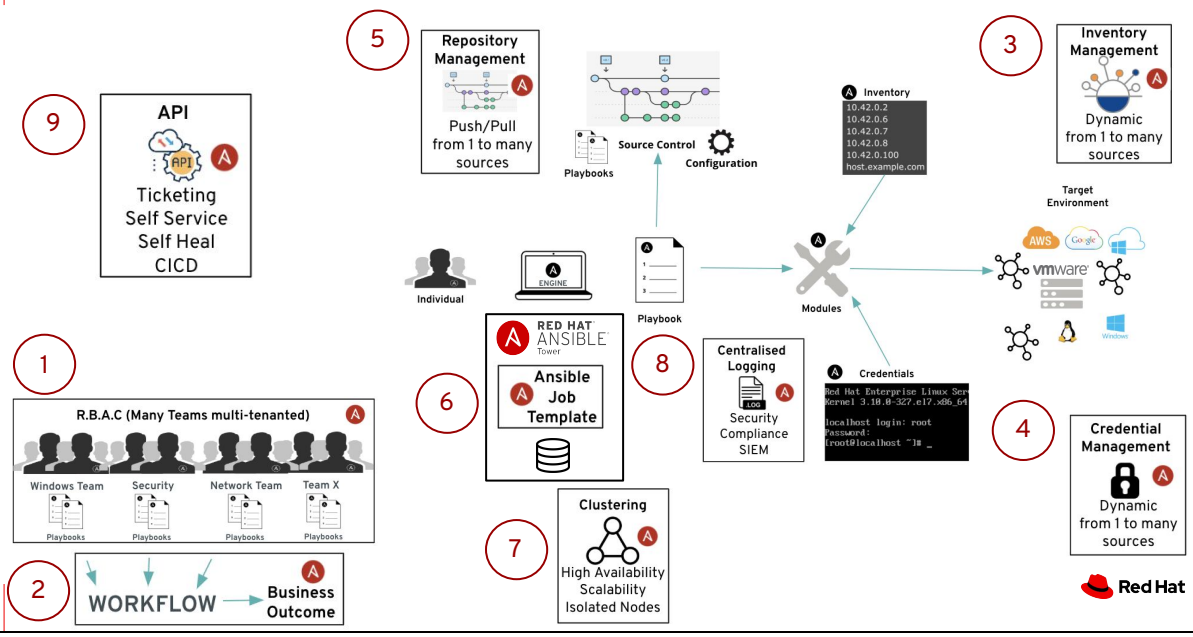


How do the bits related to each other, and how does the communication flow works?

In this image we see the overall architecture of the different components and how they work with each other, from a consumer perspective. The consumer on the left side accesses the interface of their choice - web UI or API - and triggers pre-created automation content via a self-service portal. This process is governed by enterprise RBAC support, and approval processes make sure that no one requests more automation than they should be granted.

In the end, the execution of the content will be triggered, and that will automate the pieces the user needs to be.

How Ansible Works - Ansible Controller



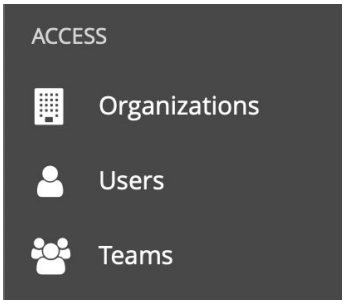
Role Based Access Control (RBAC)

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.

Red Hat

User Management

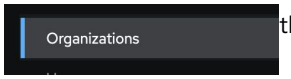
- An **Organization** is a logical collection of users, teams, projects, inventories and more. All entities belong to an organization.
- A **User** is an account to access Ansible Tower and its services given the permissions granted to it.
- **Teams** provide a means to implement role-based access control schemes and delegate responsibilities across organizations.



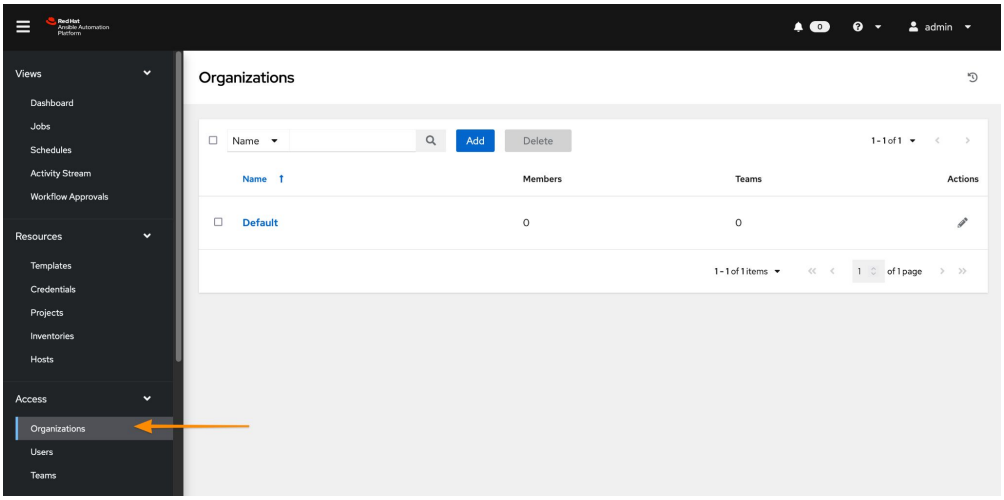
Red Hat

Viewing Organizations

Clicking on the **Organizations** button



the left menu will open up the Organizations window

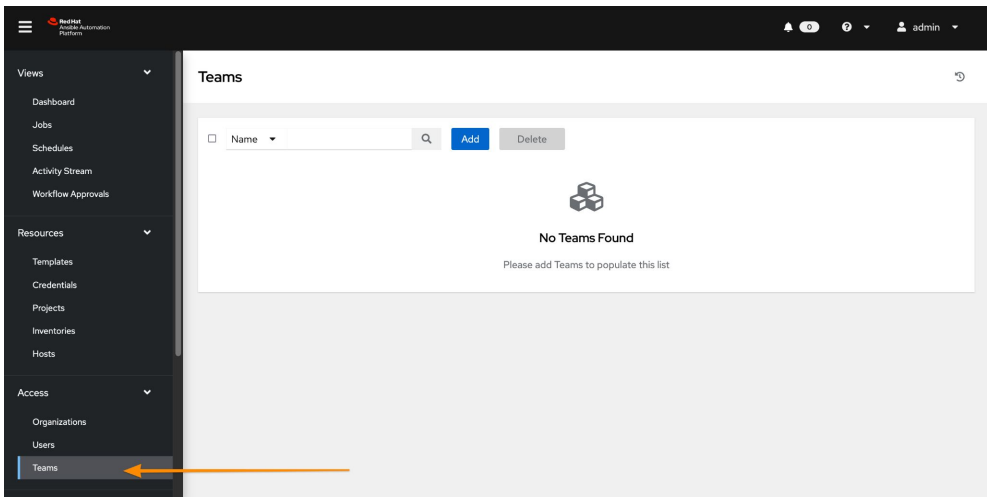


Red Hat

Viewing Teams

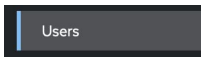
Clicking on the **Teams** button will open up the Teams window

in the left menu

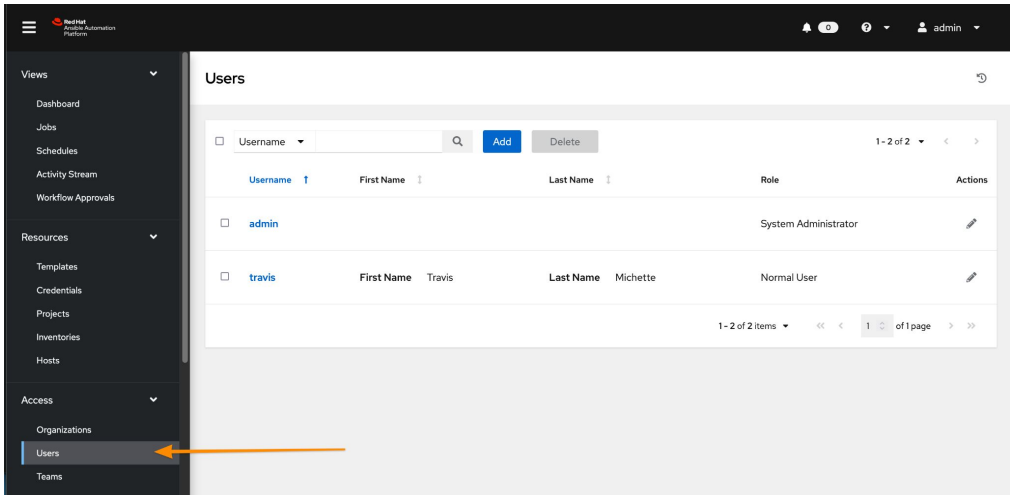


Red Hat

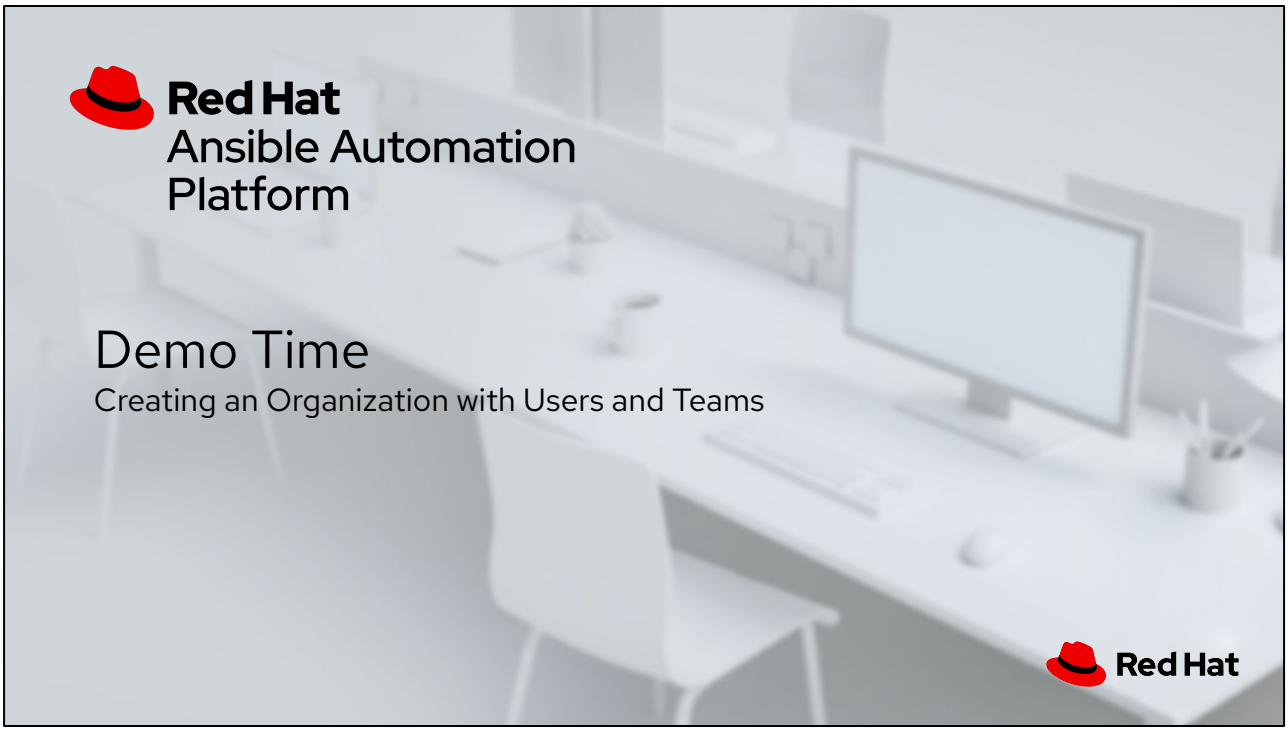
Viewing Users

Clicking on the **Users** button 

in the left menu will open up the Users window



Red Hat



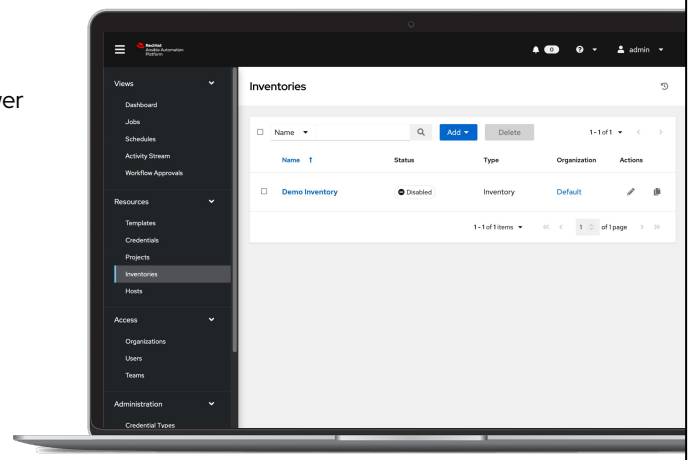
Demo completed

https://github.com/tmichett/AAP_Webinar/blob/main/Workshop_Guide/Chapters/CH3/Section3.adoc

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

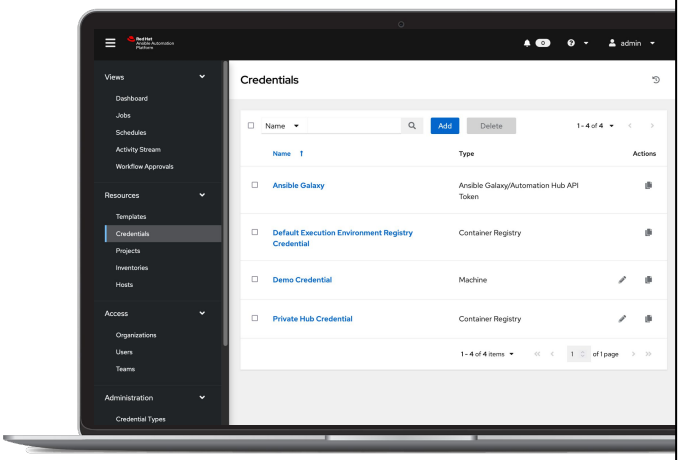


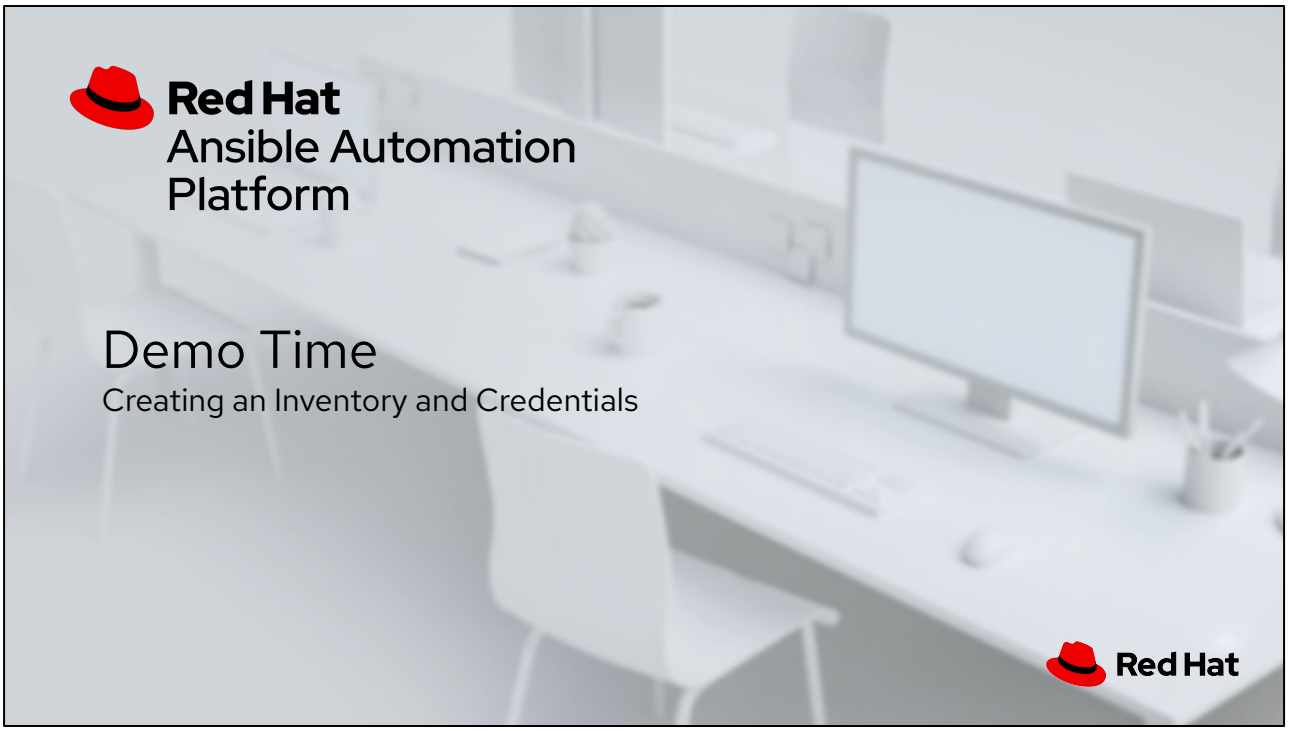
Credentials

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.



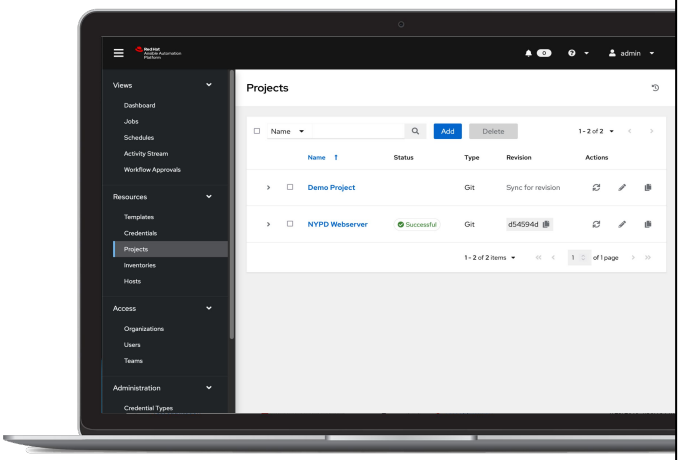


Create SCM Credential ahead of time from SSH Key
Demo Completed

Project

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.



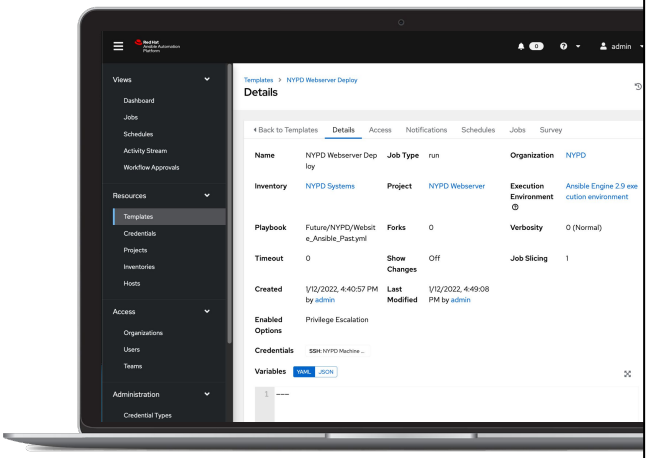
Job Templates

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



Show example playbook
Attach “Become” Credentials to the job template

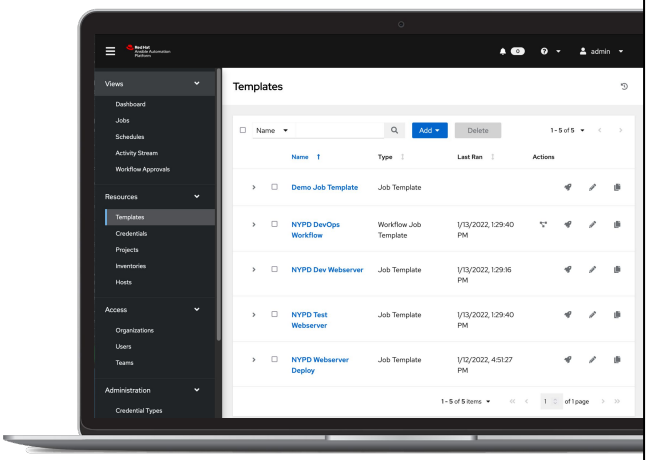
Workflows

Recall that everything in Ansible Tower revolves around the concept of a Job Template. J**ob 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:

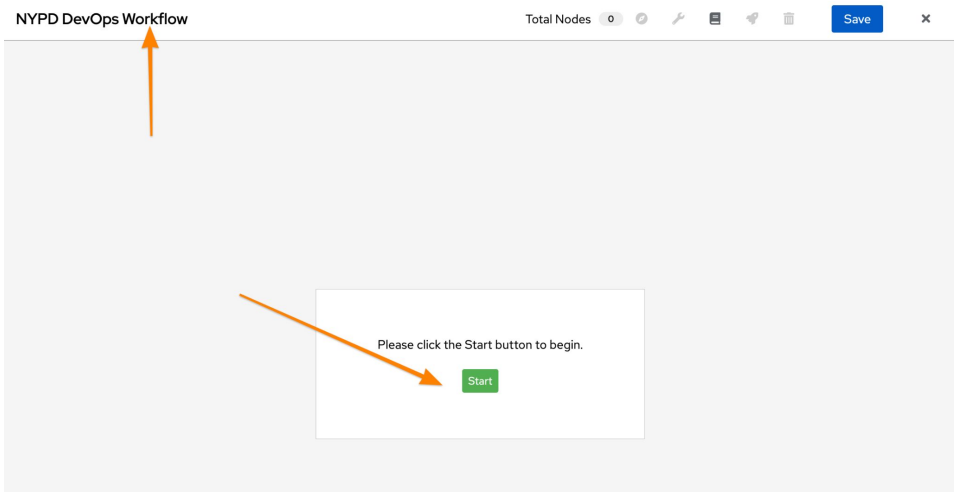
- An **Inventory** to run the job against
- A **Credential** to login to devices.
- A **Project** which contains Ansible Playbooks
- Existing **Job Templates** to execute



Red Hat

Workflow Visualizer

The workflow visualizer will start as a blank canvas.

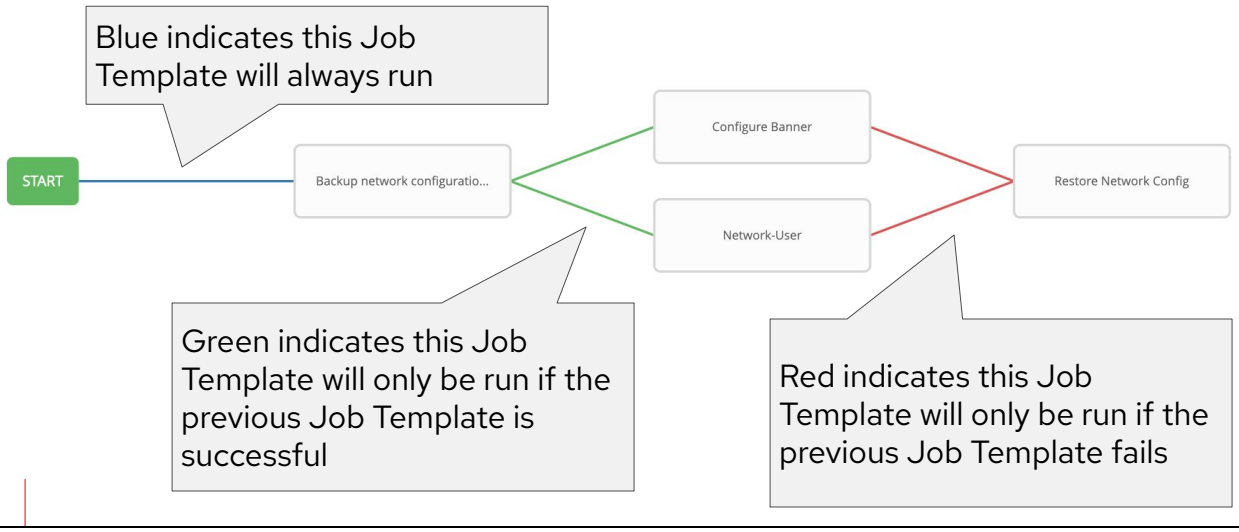


Red Hat

Once a workflow has been created, the **Workflow Visualizer** opens so that the workflow can be created and defined.

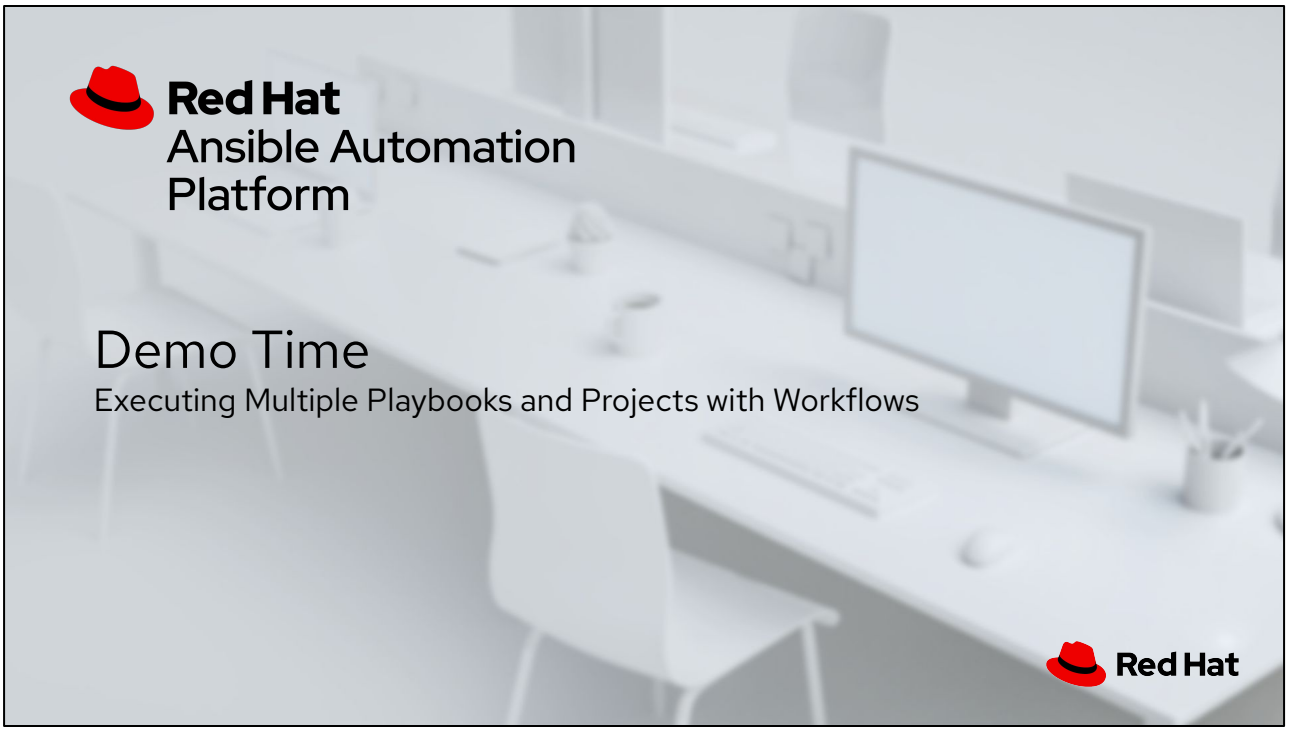
Visualizing a Workflow

Workflows can branch out, or converge in.



Workflow Jobs

Always - Always runs the workflow item
Success - Runs workflow item on success
Failure - Runs workflow item on failure



Demo Completed

Section 4
Ansible Automation

Training

Red Hat
Ansible Automation
Platform

Red Hat

Training at Red Hat

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

IDC

Source: IDC Business Value White Paper, sponsored by Red Hat. Doc. #US46999720, December 2020
(*per employee trained numbers based on average of 130 staff members trained per year discussed during IDC interviews)

Red Hat

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."

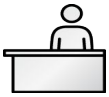
IDC

Source: IDC Business Value White Paper, sponsored by Red Hat. Doc. #US46999720, December 2020
(*per employee trained numbers based on average of 130 staff members trained per year discussed during IDC interviews)

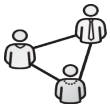
Red Hat

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

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

Red Hat

Red Hat Learning Subscription Evolution

Introducing a Premium subscription tier

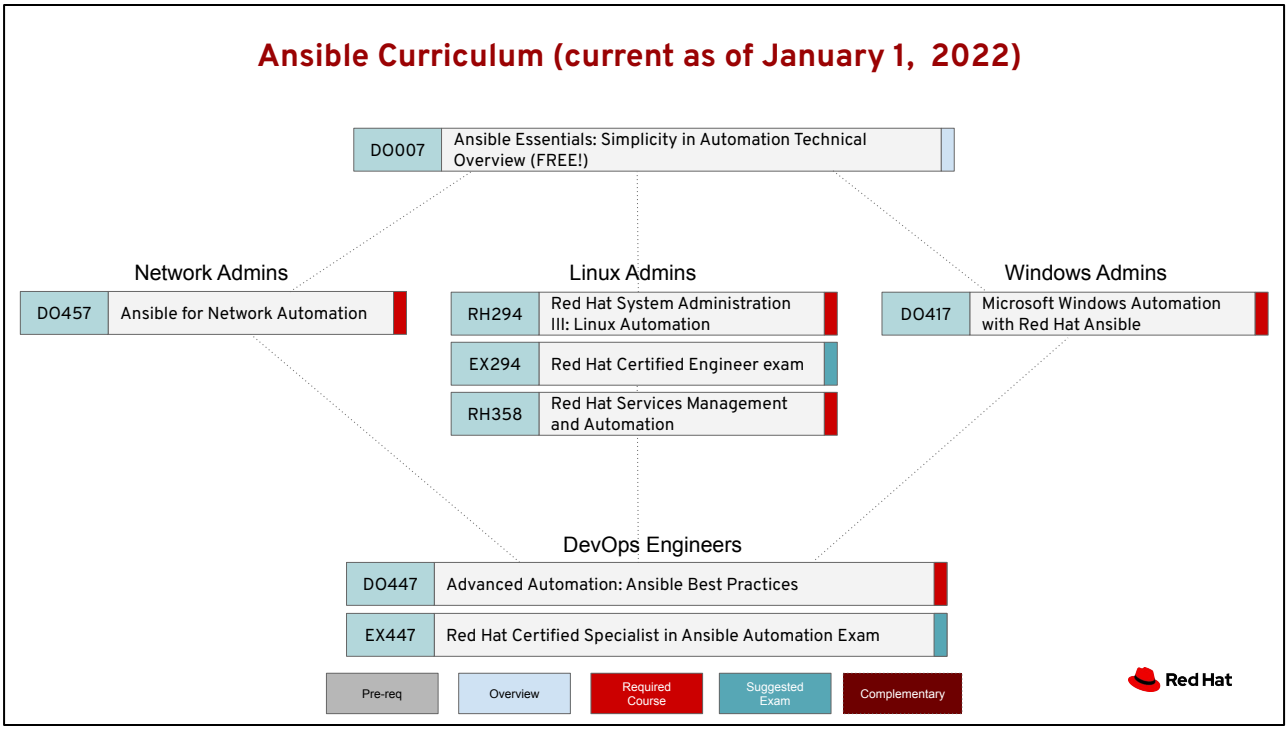
STANDARD

+

MODULARIZED VIRTUAL TRAINING

=

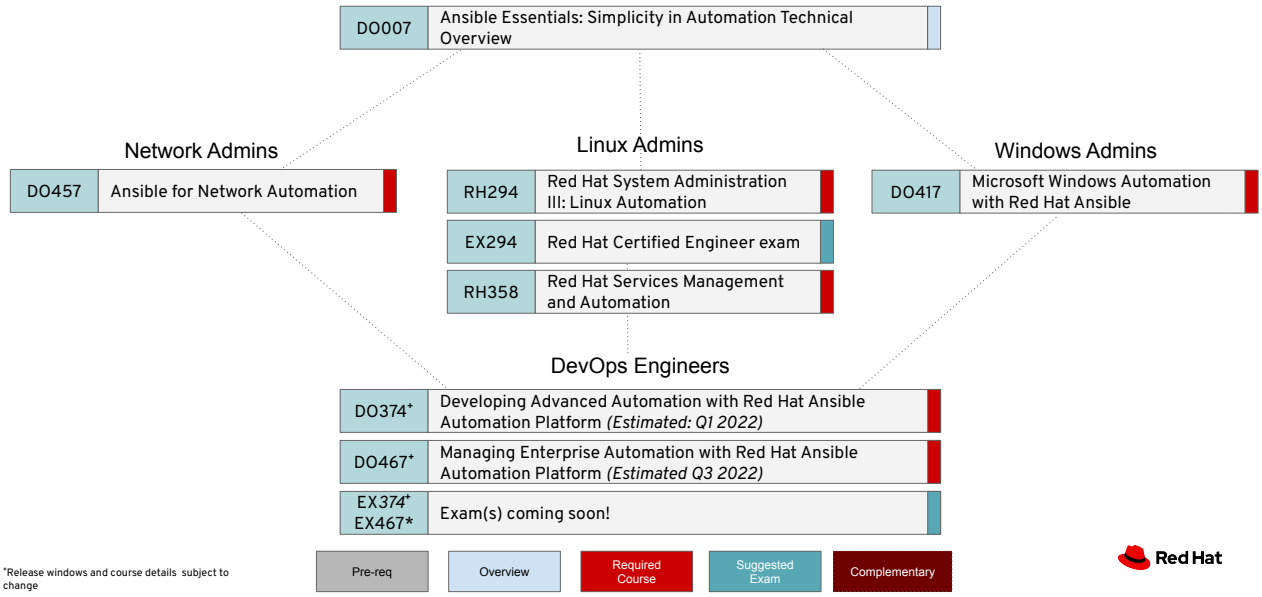
PREMIUM



Needs Overhaul and path for the new planned courses

DO374 and DO467 (DevOPS Engineers Future)

Ansible Curriculum (Future as of Q2/Q3Y22)



*Release windows and course details subject to change

Red Hat

Thank you


linkedin.com/company/red-hat


youtube.com/AnsibleAutomation


facebook.com/ansibleautomation


twitter.com/ansible


github.com/ansible

Red Hat