Network Automation Workshop

Introduction to Ansible for network engineers and operators
Housekeeping

- Timing
- Breaks
- Takeaways
What you will learn

- Introduction to Ansible automation
- How Ansible works for network automation
- Understanding Ansible modules and playbooks
- Executing Ansible playbooks to:
  - Make configuration changes
  - Gather information (Ansible facts)
- Using Jinja to template network configurations
- Using Ansible Tower to scale automation to the enterprise
Introduction

Topics Covered:

● What is the Ansible Automation Platform?
● What can it do?
● Why Network Automation?
● How Ansible Network Automation works
Automation happens when one person meets a problem they never want to solve again
Teams are automating...

Lines Of Business
Network
Security
Operations
Developers
Infrastructure
Ad-hoc Automation is happening in silos

- Developers: Ansible used in silo
- Security: DIY scripting automation
- Infrastructure: Open source config management tool
- Network: Proprietary vendor supplied automation

Is organic automation enough?
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...
When automation crosses teams, you need an automation platform.
Red Hat Ansible Automation Platform

Network

Lines of business

Security

Operations

Infrastructure

Developers

Engage

Ansible SaaS: Engage users with an automation focused experience

Scale

Ansible Tower: Operate & control at scale

Create

Ansible Engine: Universal language of automation

Fueled by an open source community
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>+more</td>
<td>Shares</td>
<td>Cisco</td>
<td>Juniper</td>
<td>+more</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>+more</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+more</td>
<td>MikroTik</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Juniper</td>
<td></td>
<td>Devops</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>Storage</td>
<td></td>
<td>OpenSwitch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RHEL</td>
<td>Netapp</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>Red Hat Storage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>Infinidat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+more</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+more</td>
</tr>
</tbody>
</table>

+more

Operating Systems
RHEL
Linux
Windows
+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

Operating Systems
RHEL
Linux
Windows
+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
Red Hat Ansible Tower by the numbers:

- **94%** Reduction in recovery time following a security incident
- **84%** Savings by deploying workloads to generic systems appliances using Ansible Tower
- **67%** Reduction in man hours required for customer deliveries

Financial summary:

- ROI on Ansible Tower: 146%
- Payback on Ansible Tower: <3 MONTHS

**SOURCE:** “The Total Economic Impact™ Of Red Hat Ansible Tower, a June 2018 commissioned study conducted by Forrester Consulting on behalf of Red Hat.”

USE CASE: NETWORK AUTOMATION
71% of networks are still driven manually via CLI

Source: Gartner, Look Beyond Network Vendors for Innovation. January 2018
NOT AS SIMPLE ANYMORE
WHY ANSIBLE? (for networks)

**SIMPLE**
- For operators, not developers
- Download and go
- Existing knowledge reuse

**POWERFUL**
- Connect via Plugins
- Easy platform enablement
- Leverage Linux tools

**AGENTLESS**
- Ideal for network gear
- No agents to exploit or update
- Standards-based SSH
ANSIBLE NETWORK AUTOMATION

65+ Network Platforms

1000+ Network Modules

15* Galaxy Network Roles

ansible.com/for/networks
galaxy.ansible.com/ansible-network

*Roles developed and maintained by Ansible Network Engineering
Common use cases

Backup and Restore
- Schedule backups
- Restore from any timestamp
- Build workflows that rollback

Configuration Compliance
- Check configuration standards
- Track configuration drift
- Enforce configuration policy

Dynamic Documentation
- Build reports
- Grab software versions, MTU, interfaces status
- Audit system services and other common config
How Ansible Network Automation works

Module code is executed locally on the control node.

Module code is copied to the managed node, executed, then removed.
Red Hat Ansible Engine: Universal language of automation
Red Hat Ansible Automation Platform

Engage
Ansible SaaS: Engage users with an automation-focused experience

Scale
Ansible Tower: Operate & control at scale

Create
Simple
Human readable automation

Powerful
Thousands of integrations

Agentless
No agents to exploit or update

Fueled by an open source community
Exercise 1

Topics Covered:

- Understanding Inventory
- An example Ansible Playbook
PLAYBOOKS ARE WRITTEN IN YAML
Tasks are executed sequentially
Invoke Ansible modules
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/
PLUGINS ARE "GEARS IN THE ENGINE"
Code that plugs into the core engine
Adaptability for various uses & platforms

```yaml
{{ some_variable | to_nice_yam1 }}
```
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
Understanding Inventory

rtr1 ansible_host=18.220.156.59
rtr2 ansible_host=18.221.53.11
rtr3 ansible_host=13.59.242.237
rtr4 ansible_host=3.16.82.231
rtr5
rtr6
Understanding Inventory - Groups

There is always a group called "all" by default

[cisco]
rtr1 ansible_host=18.220.156.59 private_ip=172.16.184.164

[arista]
rtr2 ansible_host=18.221.53.11 private_ip=172.17.229.213
rtr4 ansible_host=3.16.82.231 private_ip=172.17.209.186

[juniper]
rtr3 ansible_host=13.59.242.237 private_ip=172.16.39.75

Groups can be nested

[routers:children]
cisco
juniper
arista
Understanding Inventory - Variables

Host variables apply to the host and override group vars

Group variables apply for all devices in that group

[cisco]
  rtr1 ansible_host=52.14.208.176 private_ip=172.16.59.243

[arista]
  rtr2 ansible_host=18.221.195.152 private_ip=172.17.235.51
  rtr4 ansible_host=18.188.124.127 private_ip=172.17.43.134

[juniper]
  rtr3 ansible_host=3.15.11.56 private_ip=172.16.94.233

[cisco:vars]
  ansible_user=ec2-user
  ansible_network_os=ios
  ansible_connection=network_cli
A Sample Ansible Playbook

---

- name: deploy vlans
  hosts: cisco
  gather_facts: no

tasks:
  - name: ensure vlans exist
    nxos_vlan:
      vlan_id: 100
      admin_state: up
    name: WEB

- Playbook is a list of plays.
- Each play is a list of tasks.
- Tasks invoke modules.
- A playbook can contain more than one play.
Exercise 1 - Exploring the lab environment

In this lab you will explore the lab environment and build familiarity with the lab inventory.

Approximate time: 10 mins
Exercise 2

Topics Covered:

- An Ansible Play
- Ansible Modules
- Running an Ansible Playbook
An Ansible Playbook Example

---
- name: snmp ro/rw string configuration
  hosts: cisco
  gather_facts: no

  tasks:
  - name: ensure snmp strings are present
    ios_config:
      lines:
        - snmp-server community ansible-public RO
        - snmp-server community ansible-private RW
Ansible Playbook - Play definition

- The **name** parameter describes the Ansible Play
- Target devices using the **hosts** parameter
- Optionally disable **gather_facts**

---

- `name: snmp ro/rw string configuration`
- `hosts: cisco`
- `gather_facts: no`
Modules

Modules do the actual work in Ansible, they are what gets executed in each playbook task.

- Typically written in Python (but not limited to it)
- Modules can be idempotent
- Modules take user input in the form of parameters

```yaml
tasks:
  - name: ensure snmp strings are present
    ios_config:
      commands:
        - snmp-server community ansible-public RO
        - snmp-server community ansible-private RW
```
Network modules

Ansible modules for network automation typically references the vendor OS followed by the module name.

- *_facts
- *_command
- *_config

More modules depending on platform

- Arista EOS = eos_*
- Cisco IOS/IOS-XE = ios_*
- Cisco NX-OS = nxos_*
- Cisco IOS-XR = iosxr_*
- F5 BIG-IP = bigip_*
- F5 BIG-IQ = bigiq_*
- Juniper Junos = junos_*
- VyOS = vyos_*
Running a playbook

---
- name: snmp ro/rw string configuration
  hosts: cisco
  gather_facts: no

  tasks:
  - name: ensure snmp strings are present
    ios_config:
      commands:
      - snmp-server community ansible-public RO
      - snmp-server community ansible-private RW

[student1@ansible networking-workshop]$ ansible-playbook playbook.yml

PLAY [snmp ro/rw string configuration] ************************************************************************************************************
TASK [ensure that the desired snmp strings are present] ******************************************************************************************
changed: [rtr1]

PLAY RECAP ****************************************************************************************************************************************
rtr1    : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
Displaying output

[student1@ansible networking-workshop]$ ansible-playbook playbook.yml -v
Using /home/student1/.ansible.cfg as config file

PLAY [snmp ro/rw string configuration] *****************************************************************

TASK [ensure that the desired snmp strings are present] *****************************************************************
changed: [rtr1] => changed=true
  ansible_facts:
    discovered_interpreter_python: /usr/bin/python
  banners: {}
  commands:
    - snmp-server community ansible-public RO
    - snmp-server community ansible-private RW
  updates:
    - snmp-server community ansible-public RO
    - snmp-server community ansible-private RW

PLAY RECAP *********************************************************************************************
rtr1           : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

Increase the level of verbosity by adding more "v's" -vvvv
Exercise 2 - Execute your first network automation playbook

In this lab you will use Ansible to update the configuration of routers. This exercise will not have you create an Ansible Playbook; you will use an existing one.

Approximate time: 15 mins
Exercise 3

Topics Covered:

- Ansible Documentation and `ansible-doc`
- Facts for Network Devices
- The debug module
"Ansible for Network Automation" Documentation

Module Documentation

- Documentation is required as part of module submission
- Multiple Examples for every module
- Broken into relevant sections
Module Documentation

Documentation right on the command line

# List out all modules installed
$ ansible-doc -l
...
ios_banner
ios_command
ios_config
...

# Read documentation for installed module
$ ansible-doc ios_command
> IOS_COMMAND

Sends arbitrary commands to an ios node and returns the results read from the
device. This module includes an argument that will cause the module to wait for a
specific condition before returning or timing out if the condition is not met. This
module does not support running commands in configuration mode. Please use
[ios_config] to configure IOS devices.

Options (= is mandatory):
...
Fact modules

Arista EOS -> eos_facts
Cisco IOS -> ios_facts
Juniper Junos -> junos_facts
Fact modules return structured data

rtr1#show version
Cisco IOS XE Software, Version 16.09.02
Cisco IOS Software [Fuji], Virtual XE Software (X86_64_LINUX_IOSD-UNIVERSALK9-M), Version 16.9.2, RELEASE SOFTWARE (fc4)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
Compiled Mon 05-Nov-18 19:26 by mcppre

[student1@ansible ~]$ ansible -m ios_facts rtr1
<<abbreviated output>>

[student1@ansible ~]$ ansible -m ios_facts rtr1
<<abbreviated output>>

    "ansible_net_iostype": "IOS-XE",
    "ansible_net_memfree_mb": 1853921,
    "ansible_net_memtotal_mb": 2180495,
    "ansible_net_model": "CSR1000V",
    "ansible_net_neighbors": {},
    "ansible_net_python_version": "2.7.5",
    "ansible_net_serialnum": "964A1H0D1RM",
    "ansible_net_system": "ios",
    "ansible_net_version": "16.09.02",
Ansible Fact Playbook Example

---

- name: gather information from routers
  hosts: cisco
  gather_facts: no

  tasks:
    - name: gather router facts
      ios_facts:
Running the Ansible Playbook

What did this Ansible Playbook do?
Where are the facts?
How do I use the facts?
Running the Ansible Playbook with verbosity

[student1@ansible networking-workshop]$ ansible-playbook facts.yml -v

PLAY [gather information from routers] ********************************************************************
Using /home/student1/.ansible.cfg as config file

TASK [gather router facts] *********************************************************************
ok: [rtr1] => changed=false
  ansible_net_iostype: IOS-XE
  ansible_net_memtotal_mb: 2180495
  ansible_net_model: CSR1000V
  ansible_net_python_version: 2.7.5
  ansible_net_serialnum: 964A1H0D1RM
  ansible_net_system: ios
  ansible_net_version: 16.09.02
<<abbreviated output>>

PLAY RECAP ************************************************************************************
rtr1 : ok=1  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
Displaying output - The “debug” module

The **debug** module is used like a "print" statement in most programming languages. Variables are accessed using "{{{ }}}" - quoted curly braces

- **name**: display version
  ```yaml
display:
  msg: "The IOS version is: {{ ansible_net_version }}"
```

- **name**: display serial number
  ```yaml
display:
  msg: "The serial number is: {{ ansible_net_serialnum }}"
```
Running the Ansible Playbook with verbosity

[student1@ansible networking-workshop]$ ansible-playbook facts.yml

PLAY [gather information from routers] ******************************************************

TASK [gather router facts] ***************************************************************
  ok: [rtr1]

TASK [display version] ********************************************************************
  ok: [rtr1] =>
    msg: 'The IOS version is: 16.09.02'

TASK [display serial number] *************************************************************
  ok: [rtr1] =>
    msg: The serial number is: 964A1H0D1RM

PLAY RECAP ****************************************************************************
  rtr1    : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
## Build reports with Ansible Facts

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Model Type</th>
<th>Mgmt0 IP Address</th>
<th>Code Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>n9k</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.3</td>
<td>7.0(3)</td>
</tr>
<tr>
<td>n9k2</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.4</td>
<td>7.0(3)</td>
</tr>
<tr>
<td>n9k3</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.5</td>
<td>7.0(3)</td>
</tr>
<tr>
<td>n9k4</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.6</td>
<td>7.0(2)</td>
</tr>
<tr>
<td>n9k5</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.7</td>
<td>7.0(3)</td>
</tr>
<tr>
<td>n9k6</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.8</td>
<td>7.0(3)</td>
</tr>
</tbody>
</table>
Exercise 3 - Ansible Facts

Demonstration use of Ansible facts on network infrastructure.

Approximate time: 15 mins
Exercise 4

Topics Covered:

- Understand group variables
- Understand Jinja2
- cli_config module
Group variables

Group variables are variables that are common between two or more devices. Group variables can be associated with an individual group (e.g. “cisco”) or a nested group (e.g. routers).

Examples include

- NTP servers
- DNS servers
- SNMP information

Basically network information that is common for that group
Inventory versus group_vars directory

Group variables can be stored in a directory called `group_vars` in YAML syntax. In exercise one we covered `host_vars` and `group_vars` with relationship to inventory. What is the difference?

**inventory**

Can be used to set variables to connect and authenticate **to the device**.

Examples include:
- Connection plugins (e.g. `network_cli`)
- Usernames
- Platform types
  - (`ansible_network_os`)

**group_vars**

Can be used to set variables to configure **on the device**.

Examples include:
- VLANs
- Routing configuration
- System services (NTP, DNS, etc)
Examining a group_vars file

At the same directory level as the Ansible Playbook create a folder named `group_vars`. Group variable files can simply be named the group name (in this case `all.yml`)

```
[student1@ansible networking-workshop]$ cat group_vars/all.yml

nodes:
  rtr1:
    Loopback100: "192.168.100.1"
  rtr2:
    Loopback100: "192.168.100.2"
  rtr3:
    Loopback100: "192.168.100.3"
  rtr4:
    Loopback100: "192.168.100.4"
```
Jinja2

- Ansible has native integration with the Jinja2 templating engine
- Render data models into device configurations
- Render device output into dynamic documentation

Jinja2 enables the user to manipulate variables, apply conditional logic and extend programmability for network automation.
Network Automation config modules

```
cli_config (agnostic)
ios_config:
nxos_config:
iosxr_config:
eos_config
*os_config:
```

Variables

Template

---
Jinja2 Templating Example (1/2)

Variables

| ntp_server: 192.168.0.250 |
| name_server: 192.168.0.251 |

Jinja2 Template

```text
! 
ntp server {{ntp_server}}
!
ip name-server {{name_server}}
!
```

Generated Network Configuration

```
rtr1
```
```
! ip name-server 192.168.0.251
! ntp server 192.168.0.250
!
```
```
rtrX
```
```
! ip name-server 192.168.0.251
! ntp server 192.168.0.250
!
```
Jinja2 Templating Example (2/2)

Variables

```
nodes:
    rtr1:
        Loopback100: "192.168.100.1"
    rtr2:
        Loopback100: "192.168.100.2"
    rtr3:
        Loopback100: "192.168.100.3"
    rtr4:
        Loopback100: "192.168.100.4"
```

Jinja2 Template

```
{% for interface,ip in nodes[inventory_hostname].items() %}
interface {{interface}}
    ip address {{ip}} 255.255.255.255
{% endfor %}
```

Generated Network Configuration

```
rtr1
interface Loopback100
    ip address 192.168.100.1
!
rtr2
interface Loopback100
    ip address 192.168.100.2
!
rtrX
interface Loopback100
    ip address X
!```
The cli_config module

Agnostic module for network devices that uses the network_cli connection plugin.

---

- name: configure network devices
  hosts: rtr1,rtr2
  gather_facts: false
  tasks:
    - name: configure device with config
      cli_config:
        config: "{{ lookup('template', 'template.j2') }}"
Exercise 4 - Network Configuration with Jinja Templates

Demonstration templating a network configuration and pushing it a device

Approximate time: 15 mins
Tower Introduction

Topics Covered:

● What is Ansible Tower?

● Job Templates
  ○ Inventory
  ○ Credentials
  ○ Projects
Engage Ansible SaaS: Engage users with an automation focused experience

Red Hat Ansible Automation Platform

Network

Lines of business

Security

Operations

Infrastructure

Developers

Engage

Ansible SaaS: Engage users with an automation focused experience

Scale

Control
Web UI and API

Delegation
Role Based Access Controls

Scale
Scalable Execution Capacity

Create

Ansible Engine: Universal language of automation

Fueled by an open source community
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 Ansible Tower

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

**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.

**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.

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

**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.
FEATURE OVERVIEW:

Job Template
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
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.
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.
Exercise 5 - Explore Red Hat Ansible Tower

Explore and understand the lab environment. Locate and understand:

- Ansible Tower **Inventory**
- Ansible Tower **Credentials**
- Ansible Tower **Projects**

Approximate time: 15 mins
Exercise 6

Topics Covered:

- Building a Job Template
- Executing a Job Template
Expanding on Job Templates

Job Templates can be found and created by clicking the Templates button under the RESOURCES section on the left menu.
Executing an existing Job Template

Job Templates can be launched by clicking the rocketship button for the corresponding Job Template.
Creating a new Job Template (1/2)

New Job Templates can be created by clicking the **plus button** +
Creating a new Job Template (2/2)

This **New Job Template** window is where the inventory, project and credential are assigned. The red asterisk (*) means the field is required.
Exercise 6 - Creating a Tower Job Template

Demonstrate a network backup configuration job template for Red Hat Ansible Tower.

Approximate time: 15 mins
Exercise 7

Topics Covered:

- Understanding Extra Vars
- Building a Tower Survey
- Self-service IT with Tower Surveys
Surveys

Tower surveys allow you to configure how a job runs via a series of questions, making it simple to customize your jobs in a user-friendly way.

An Ansible Tower survey is a simple question-and-answer form that allows users to customize their job runs. Combine that with Tower's role-based access control, and you can build simple, easy self-service for your users.
Creating a Survey (1/2)

Once a Job Template is saved, the **Add Survey Button** will appear.

Click the button to open the Add Survey window.
Creating a Survey (2/2)

The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.
Using a Survey

When launching a job, the user will now be prompted with the Survey. The user can be required to fill out the Survey before the Job Template will execute.
Exercise 7- Creating a Survey

Demonstrate the use of Ansible Tower survey feature

Approximate time: 15 mins
Exercise 8

Topics Covered:

- Understanding Organizations
- Understanding Teams
- Understanding Users
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 with the exception of users.

- 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.
Viewing Users

Clicking on the **Users** button in the left menu will open up the Users window.
Exercise 8 - Understanding RBAC

The objective of this exercise is to understand Role Based Access Controls (RBAC)

Approximate time: 15 mins
Exercise 9

Topics Covered:

- Understanding Workflows
  - Branching
  - Convergence / Joins
  - Conditional Logic
Workflows

Workflows can be found alongside Job Templates by clicking the **Templates** button under the **RESOURCES** section on the left menu.
Adding a new Workflow Template
To add a new **Workflow** click on the green + button
This time select the **Workflow Template**
Creating the Workflow

Fill out the required parameters and click **SAVE**. As soon as the Workflow Template is saved the WORKFLOW VISUALIZER will open.
Workflow Visualizer

The workflow visualizer will start as a blank canvas.
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
Exercise 9 - Creating a Workflow

Demonstrate the use of Ansible Tower workflow

Approximate time: 15 mins
Next Steps

GET STARTED
ansible.com/get-started
ansible.com/tower-trial

JOIN THE COMMUNITY
ansible.com/community

WORKSHOPS & TRAINING
ansible.com/workshops
Red Hat Training

SHARE YOUR STORY
Follow us @Ansible
Friend us on Facebook
Chat with us

- **Slack**
  
  https://ansiblenetwork.slack.com
  

- **IRC**
  
  #ansible-network on freenode
  
  [http://webchat.freenode.net/?channels=ansible-network](http://webchat.freenode.net/?channels=ansible-network)
 Bookmark the Github organization

- Examples, samples and demos
- Run network topologies right on your laptop
Thank you