



Red Hat Ansible Automation Platform

Network Automation Workshop

Introduction to Ansible for network engineers and operators



Red Hat

Housekeeping

- Timing
- Breaks
- Takeaways

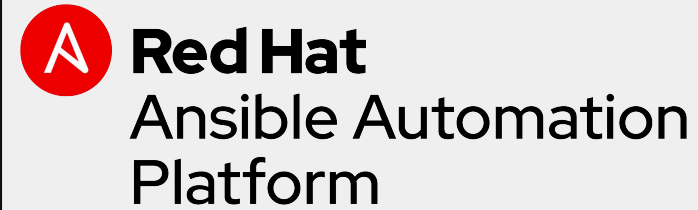
What you will learn

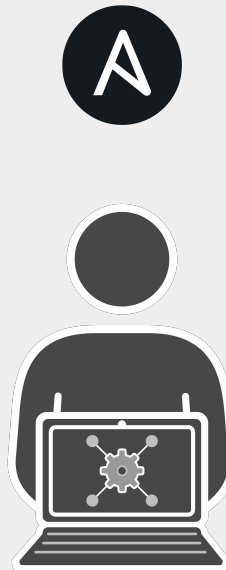
- Introduction to Ansible automation
- How Ansible works for network automation
- Understanding Ansible modules and playbooks
- Executing Ansible playbooks to:
 - Make configuration changes
 - Gather information (Ansible facts)
- Using Jinja to template network configurations
- Using Ansible Tower to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

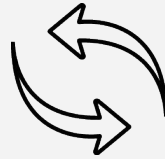
Is organic
automation enough?

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

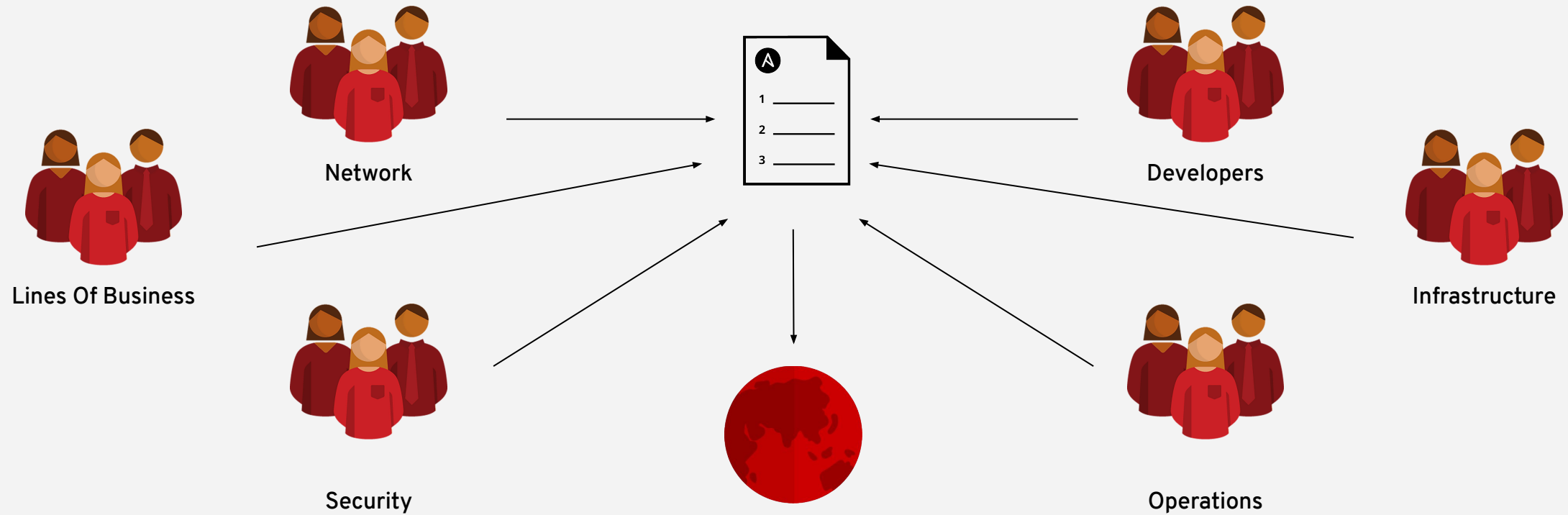
Infrastructure

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

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

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:

146%

ROI on Ansible Tower

< 3 MONTHS

Payback on Ansible Tower



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)

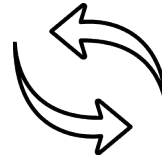


SIMPLE

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

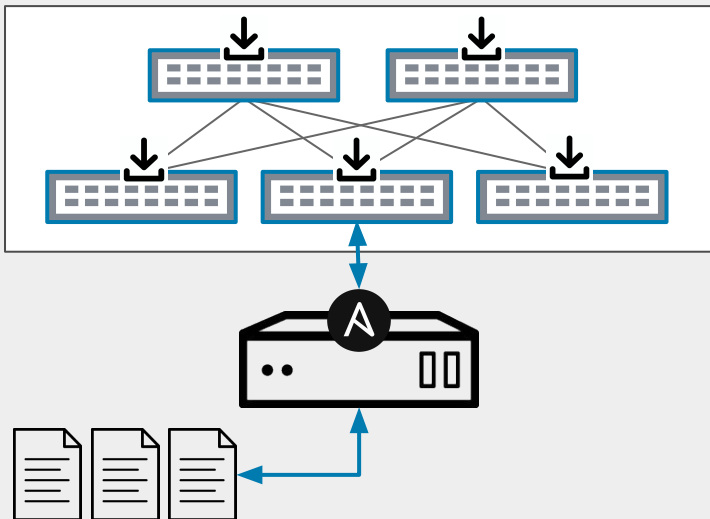
15*

Galaxy
Network Roles

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

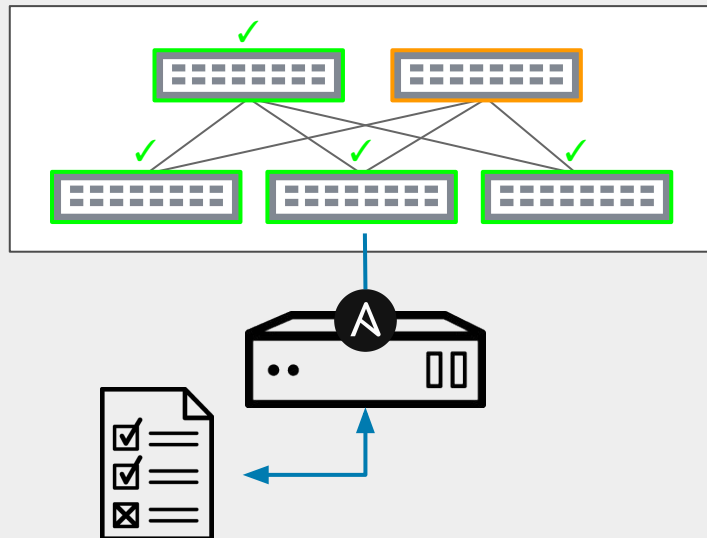
**Roles developed and maintained by Ansible Network Engineering*

Common use cases



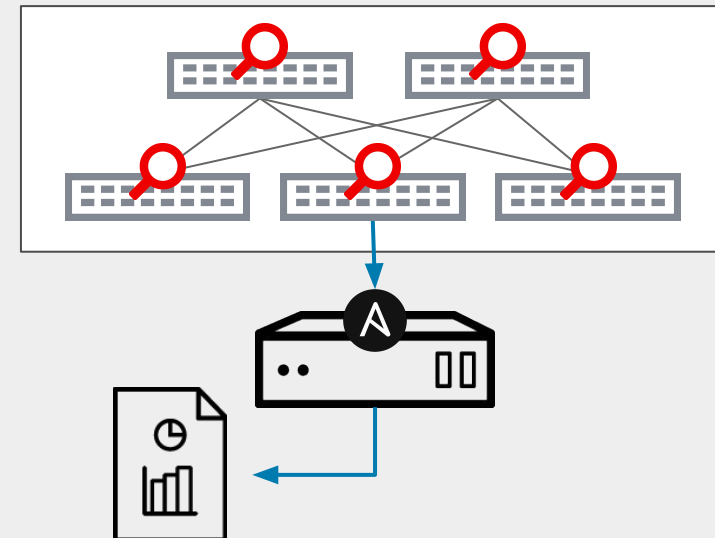
Backup and Restore

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



Configuration Compliance

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

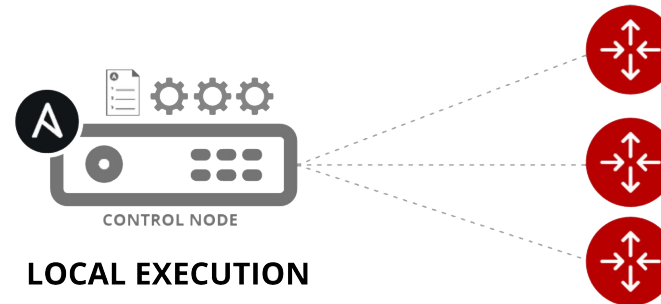


Dynamic Documentation

- Build reports
- Grab software versions, MTU, interfaces status
- Audit system services and other common config

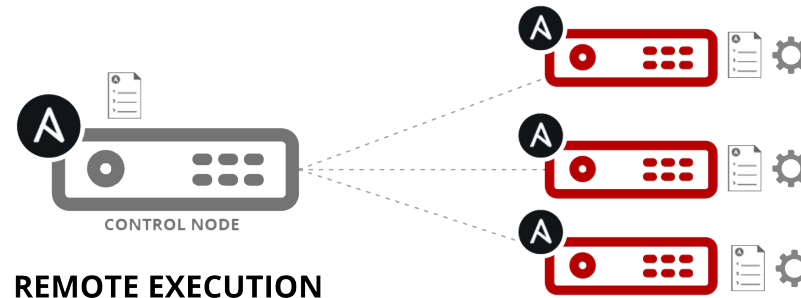
How Ansible Network Automation works

Module code is executed locally on the control node



**NETWORKING
DEVICES**

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



**LINUX/WINDOWS
HOSTS**



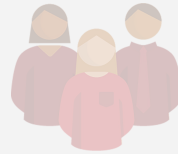
Red Hat Ansible Automation Platform

Red Hat Ansible Engine:

Universal language of automation



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

Simple

Human readable automation

Powerful

Thousands of integrations

Agentless

No agents to exploit or update

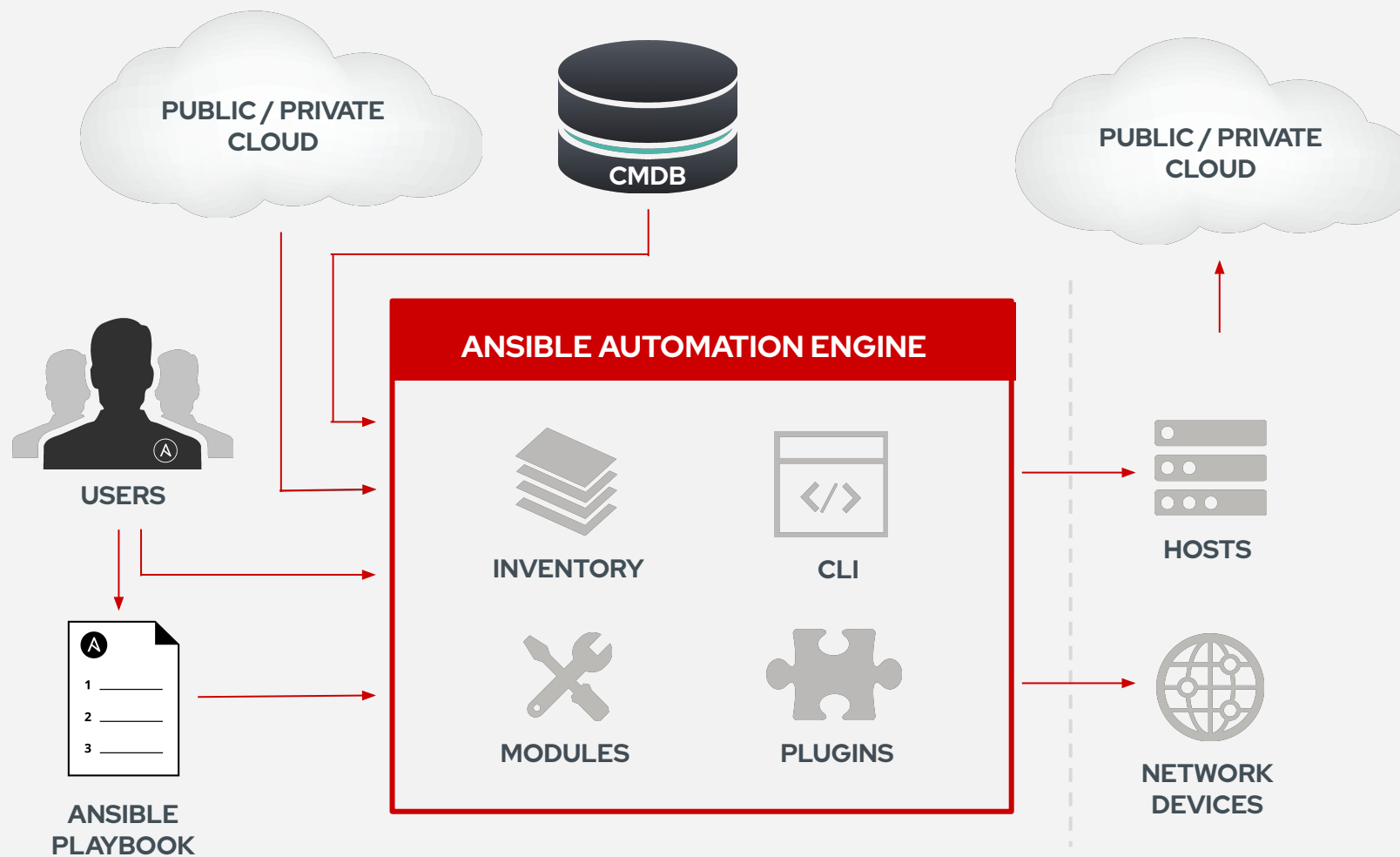
Fueled by an open source community

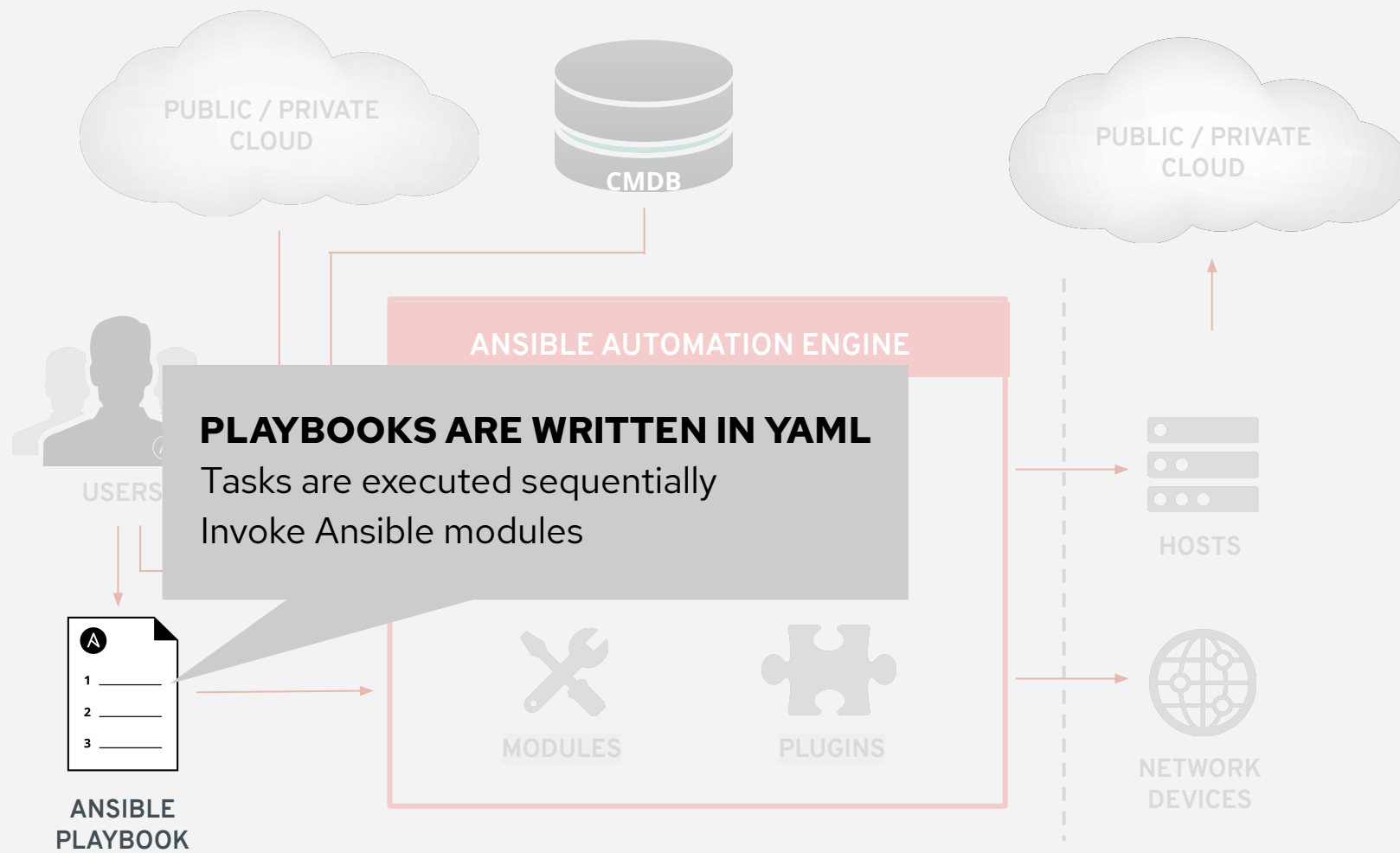
Exercise 1

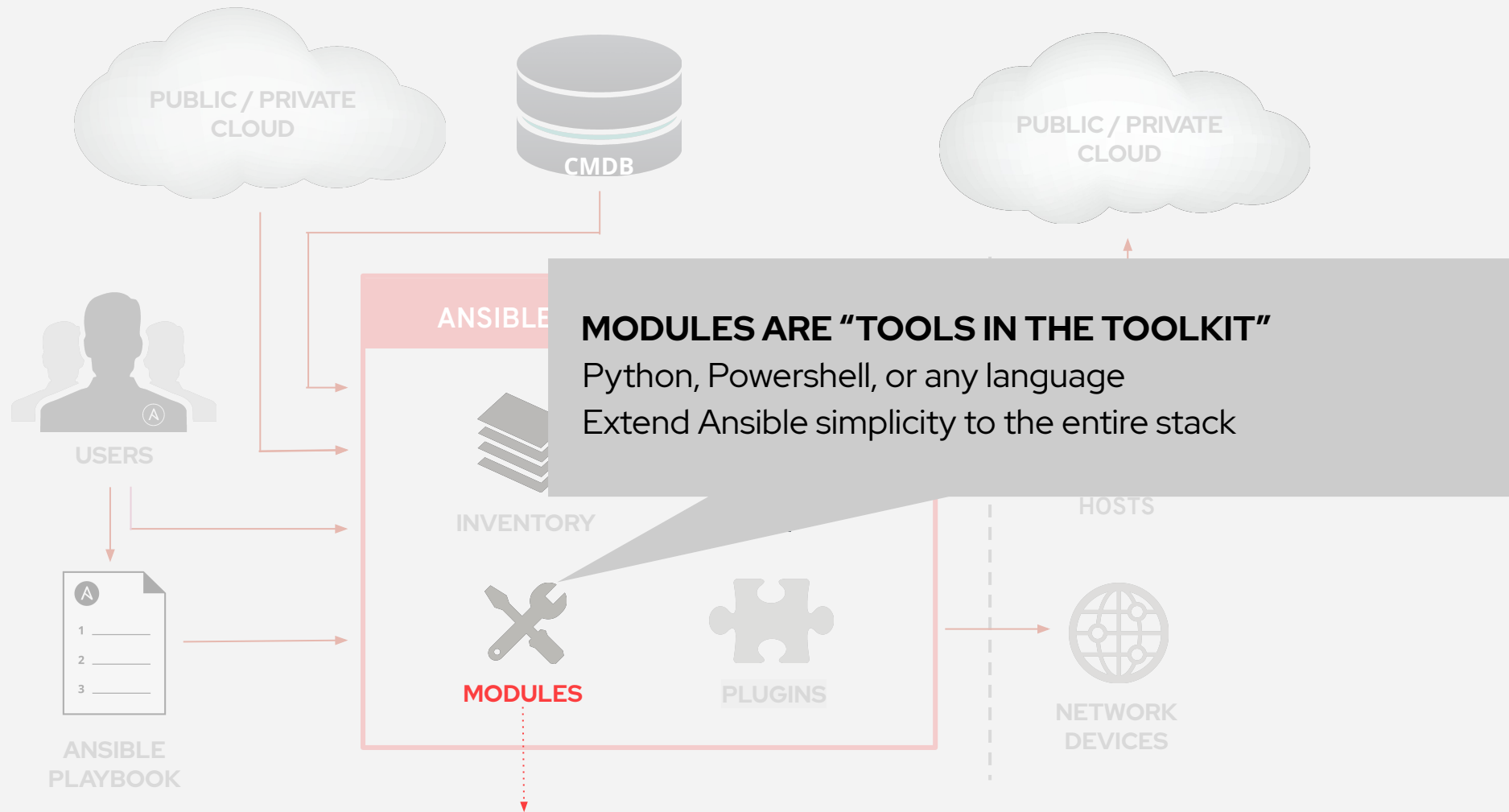
Topics Covered:

- Understanding Inventory
- An example Ansible Playbook

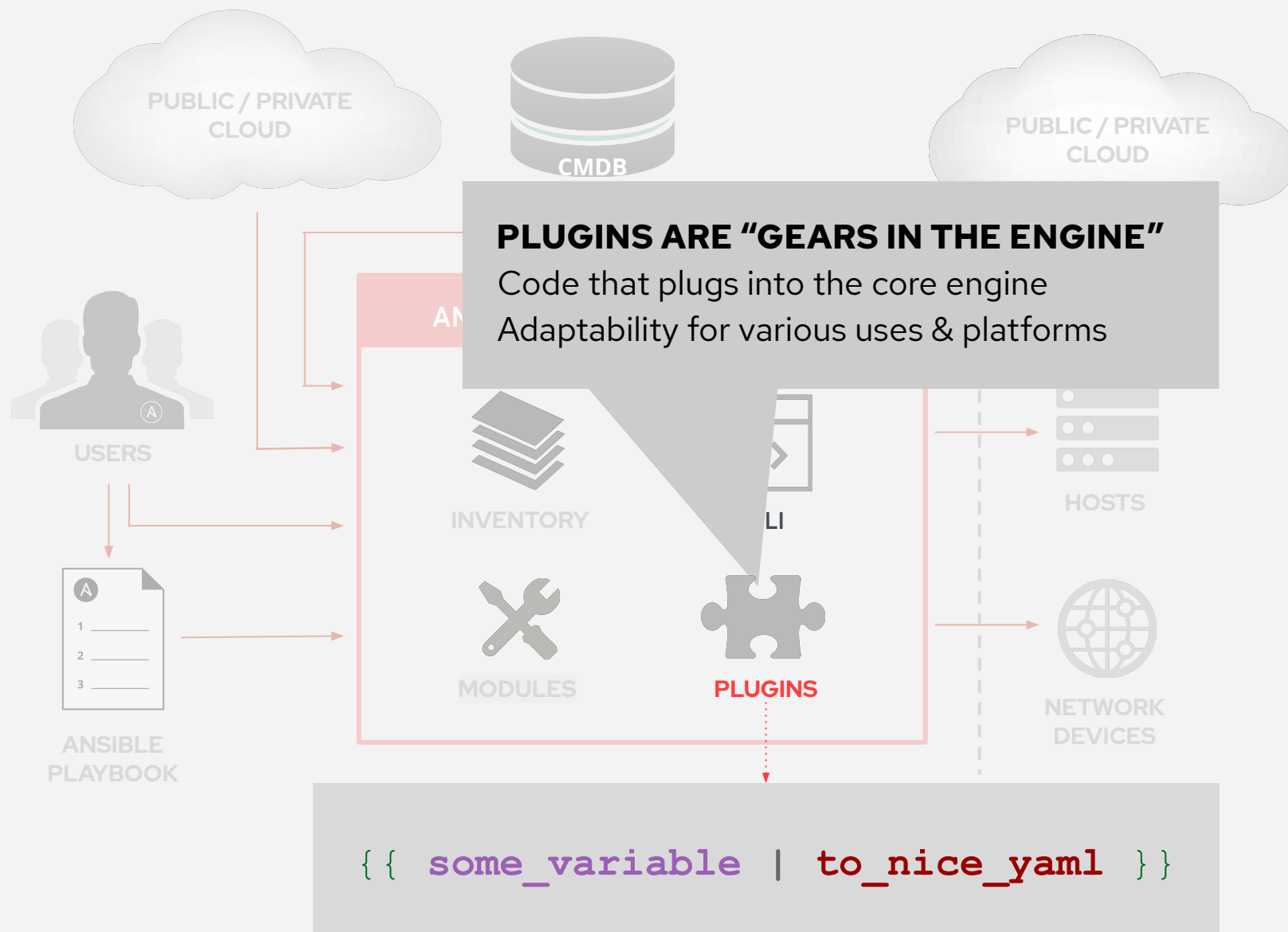


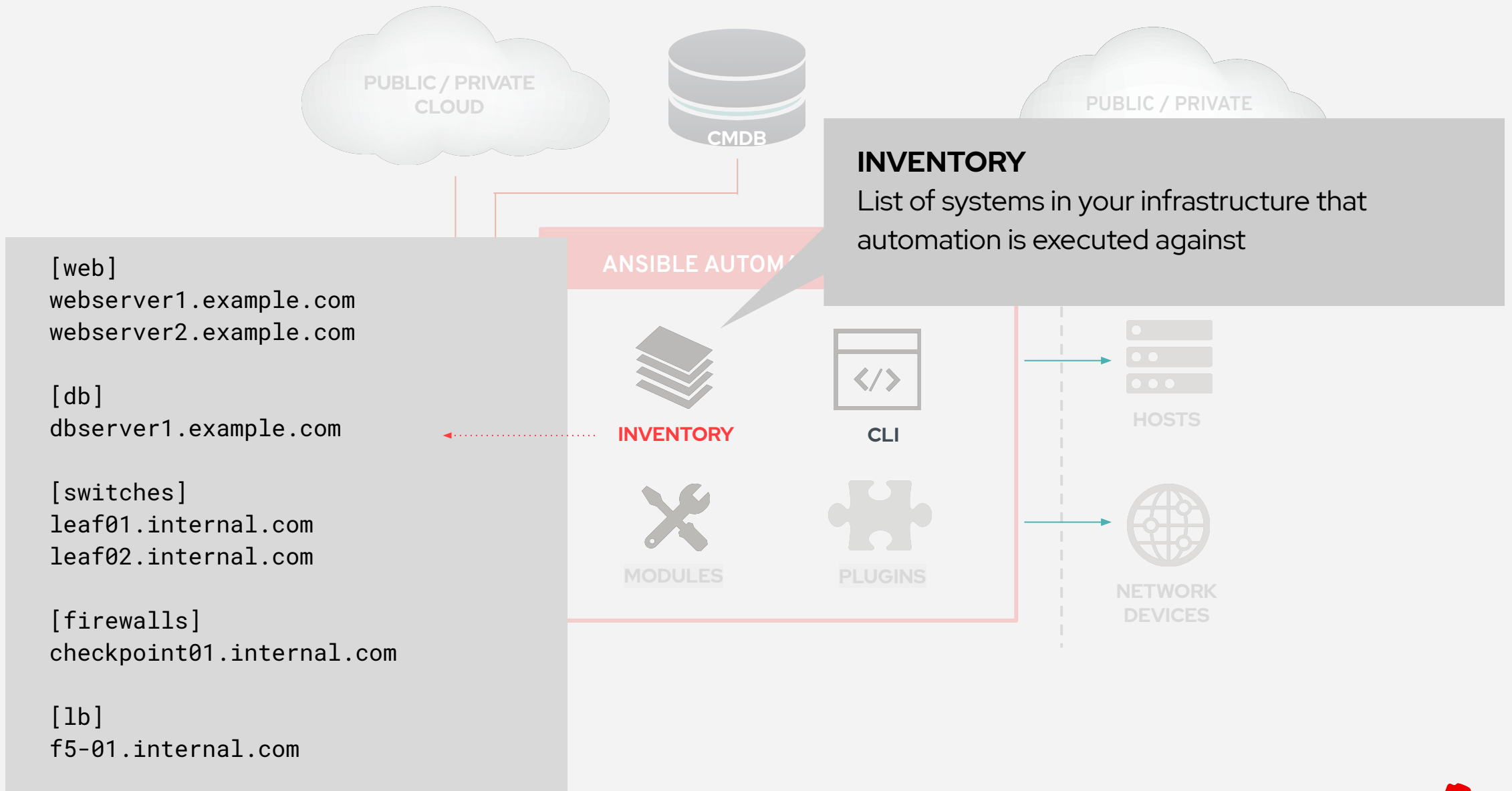






```
- name: latest index.html file is present
  template:
    src: files/index.html
    dest: /var/www/html/
```





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

```
[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
```

Group variables apply for all devices in that group

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.



Red Hat Ansible Automation Platform

Exercise 1 - Exploring the lab environment

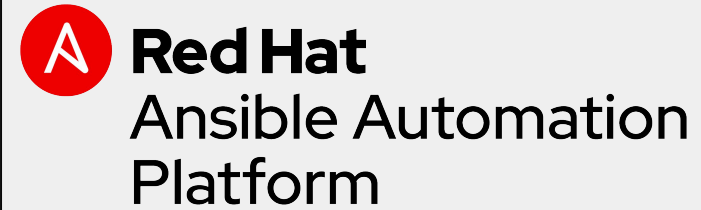
In this lab you will explore the lab environment and build familiarity with the lab inventory.

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

```
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



Red Hat Ansible Automation Platform

Exercise 2 - Execute your first network automation playbook

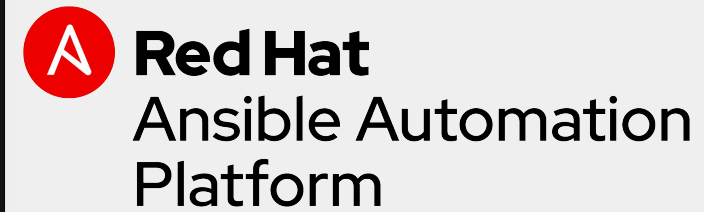
In this lab you will use Ansible to update the configuration of routers. This exercise will not have you create an Ansible Playbook; you will use an existing one.

Approximate time: 15 mins

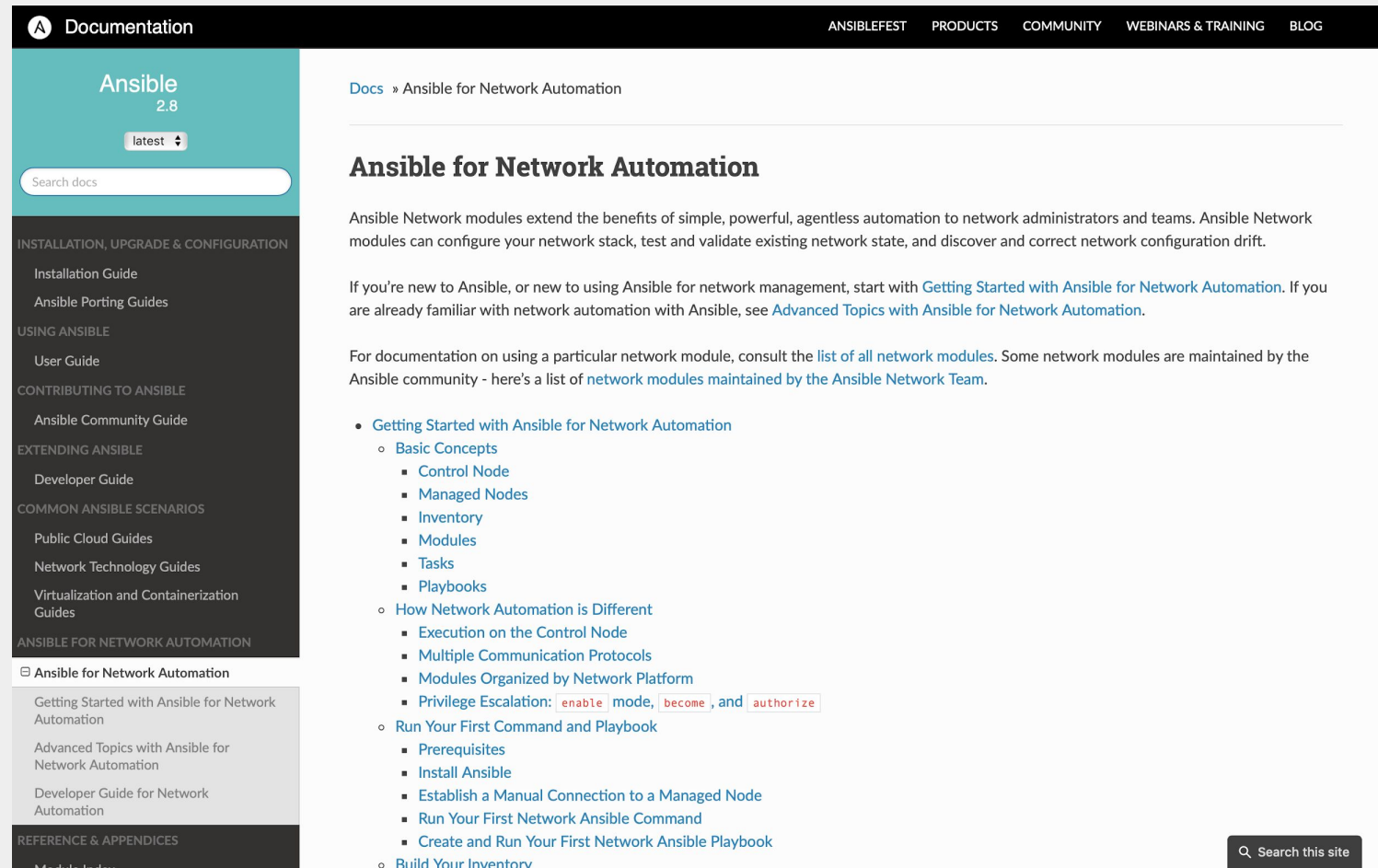
Exercise 3

Topics Covered:

- Ansible Documentation and *ansible-doc*
- Facts for Network Devices
- The debug module



"Ansible for Network Automation" Documentation



The screenshot shows the Ansible documentation website. The top navigation bar includes links for ANSIBLEFEST, PRODUCTS, COMMUNITY, WEBINARS & TRAINING, and BLOG. The left sidebar features the Ansible logo, version 2.8, a search bar, and a list of categories: INSTALLATION, UPGRADE & CONFIGURATION; USING ANSIBLE; CONTRIBUTING TO ANSIBLE; EXTENDING ANSIBLE; COMMON ANSIBLE SCENARIOS; and ANSIBLE FOR NETWORK AUTOMATION. The 'ANSIBLE FOR NETWORK AUTOMATION' section is expanded, showing links to Getting Started, Advanced Topics, and Developer Guide. The main content area displays the 'Ansible for Network Automation' page, which includes an introduction, a list of links for getting started, and a search bar at the bottom right.

Documentation

Ansible 2.8

latest

Search docs

INSTALLATION, UPGRADE & CONFIGURATION

Installation Guide

Ansible Porting Guides

USING ANSIBLE

User Guide

CONTRIBUTING TO ANSIBLE

Ansible Community Guide

EXTENDING ANSIBLE

Developer Guide

COMMON ANSIBLE SCENARIOS

Public Cloud Guides

Network Technology Guides

Virtualization and Containerization Guides

ANSIBLE FOR NETWORK AUTOMATION

Ansible for Network Automation

Getting Started with Ansible for Network Automation

Advanced Topics with Ansible for Network Automation

Developer Guide for Network Automation

REFERENCE & APPENDICES

Module Index

Docs » Ansible for Network Automation

Ansible for Network Automation

Ansible Network modules extend the benefits of simple, powerful, agentless automation to network administrators and teams. Ansible Network modules can configure your network stack, test and validate existing network state, and discover and correct network configuration drift.

If you're new to Ansible, or new to using Ansible for network management, start with [Getting Started with Ansible for Network Automation](#). If you are already familiar with network automation with Ansible, see [Advanced Topics with Ansible for Network Automation](#).

For documentation on using a particular network module, consult the [list of all network modules](#). Some network modules are maintained by the Ansible community - here's a list of [network modules maintained by the Ansible Network Team](#).

- Getting Started with Ansible for Network Automation
 - Basic Concepts
 - Control Node
 - Managed Nodes
 - Inventory
 - Modules
 - Tasks
 - Playbooks
 - How Network Automation is Different
 - Execution on the Control Node
 - Multiple Communication Protocols
 - Modules Organized by Network Platform
 - Privilege Escalation: `enable`, `mode`, `become`, and `authorize`
 - Run Your First Command and Playbook
 - Prerequisites
 - Install Ansible
 - Establish a Manual Connection to a Managed Node
 - Run Your First Network Ansible Command
 - Create and Run Your First Network Ansible Playbook
 - Build Your Inventory

Search this site

<http://bit.ly/AnsibleNetwork>

Module Documentation

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

[Docs](#) » [Module Index](#)

Module Index

- [All Modules](#)
- [Cloud Modules](#)
- [Clustering Modules](#)
- [Commands Modules](#)
- [Crypto Modules](#)
- [Database Modules](#)
- [Files Modules](#)
- [Identity Modules](#)
- [Inventory Modules](#)
- [Messaging Modules](#)
- [Monitoring Modules](#)
- [Network Modules](#)
- [Notification Modules](#)
- [Packaging Modules](#)
- [Remote Management Modules](#)
- [Source Control Modules](#)
- [Storage Modules](#)
- [System Modules](#)
- [Utilities Modules](#)
- [Web Infrastructure Modules](#)
- [Windows Modules](#)

service - Manage services.

- [Synopsis](#)
- [Options](#)
- [Examples](#)
 - [Status](#)
 - [Support](#)

Synopsis

- Controls services on remote hosts. Supported init systems include BSD init, OpenRC, SysV, Solaris SMF, systemd, upstart.

Options

parameter	required	default	choices	comments
arguments	no			Additional arguments provided on the command line aliases: args
enabled	no		• yes • no	Whether the service should start on boot. At least one of state and enabled are required.
name	yes			Name of the service.
pattern	no			If the service does not respond to the status command, name a substring to look for as would be found in the output of the ps command as a stand-in for a status result. If the string is found, the service will be assumed to be running.
runlevel	no	default		For OpenRC init scripts (ex: Gentoo) only. The runlevel that this service belongs to.
sleep (added in 1.3)	no			If the service is being <code>restarted</code> then sleep this many seconds between the stop and start command. This helps to workaround badly behaving init scripts that exit immediately after signalling a process to stop.
state	no		• started • stopped • restarted • reloaded	<code>started / stopped</code> are idempotent actions that will not run commands unless necessary. <code>restarted</code> will always bounce the service. <code>reloaded</code> will always reload. At least one of state and enabled are required. Note that reloaded will start the service if it is not already started, even if your chosen init system wouldn't normally.
use (added in 2.2)	no	auto		The service module actually uses system specific modules, normally through auto detection, this setting can force a specific module. Normally it uses the value of the <code>'ansible_service_mgr'</code> fact and falls back to the old <code>'service'</code> module when none matching is found.

<https://docs.ansible.com/>

Module Documentation

Documentation right on the command line

```
# List out all modules installed
$ ansible-doc -l
...
ios_banner           Manage multiline banners on Cisco IOS devices
ios_command          Run commands on remote devices running Cisco IOS
ios_config           Manage Cisco IOS configuration sections
...

# 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



eos_facts



ios_facts



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 mcpre
..
.
<rest of output removed for brevity>
```

```
[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

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

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

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

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

- 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**
debug:
msg: "The IOS version is: {{ ansible_net_version }}"
- **name:** **display serial number**
debug:
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

Hostname	Model Type	Mgmt0 IP Address	Code Version
n9k	Nexus9000 9000v Chassis	192.168.2.3	7.0(3)I7(1)
n9k2	Nexus9000 9000v Chassis	192.168.2.4	7.0(3)I7(1)
n9k3	Nexus9000 9000v Chassis	192.168.2.5	7.0(3)I7(1)
n9k4	Nexus9000 9000v Chassis	192.168.2.6	7.0(2)I7(1)
n9k5	Nexus9000 9000v Chassis	192.168.2.7	7.0(3)I7(1)
n9k6	Nexus9000 9000v Chassis	192.168.2.8	7.0(3)I7(1)



Red Hat Ansible Automation Platform

Exercise 3 - Ansible Facts

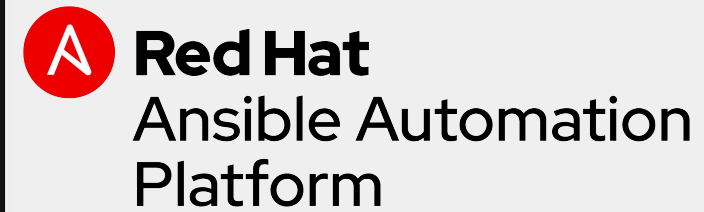
Demonstration use of Ansible facts on network infrastructure.

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:

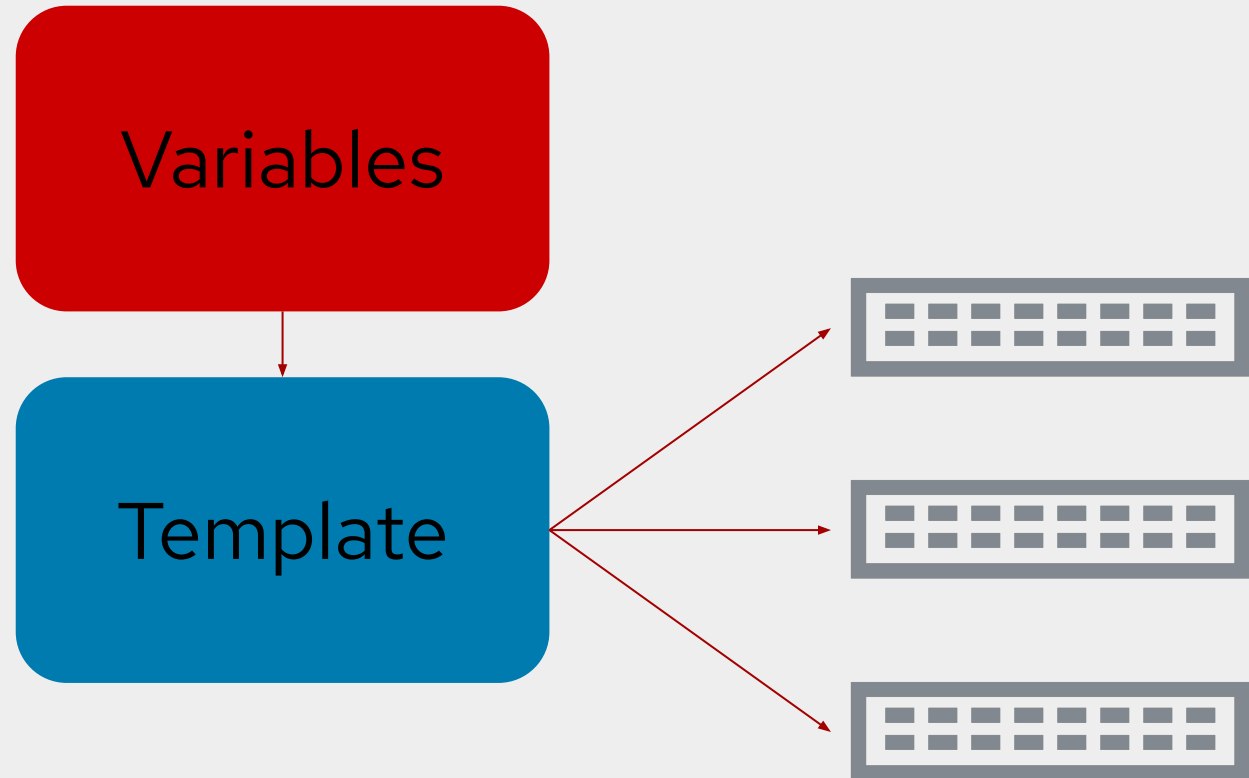
iosxr_config:

eos_config

.

.

*os_config:



Jinja2 Templating Example (1/2)

Variables

```
ntp_server: 192.168.0.250
name_server: 192.168.0.251
```

Jinja2 Template

```
!
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') }}"
```



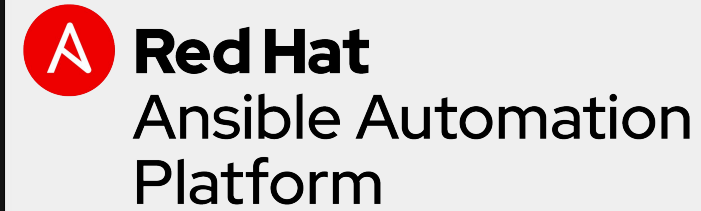
Red Hat Ansible Automation Platform

Exercise 4 - Network Configuration with Jinja Templates

Demonstration templating a network configuration and pushing it a device

Approximate time: 15 mins

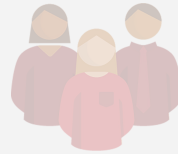
Tower Introduction



Topics Covered:

- What is Ansible Tower?
- Job Templates
 - Inventory
 - Credentials
 - Projects

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

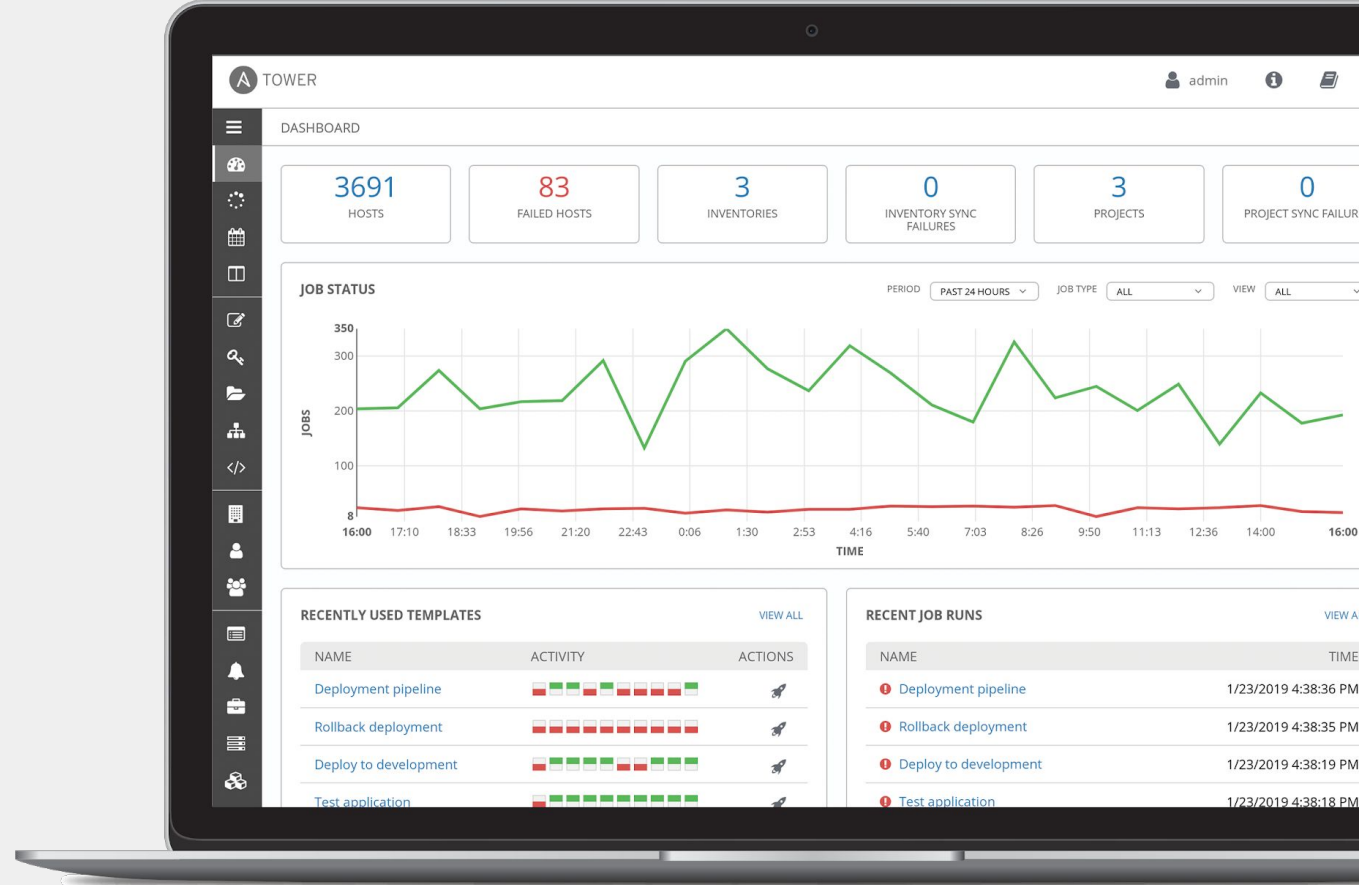
Ansible Engine: Universal language of automation

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

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.

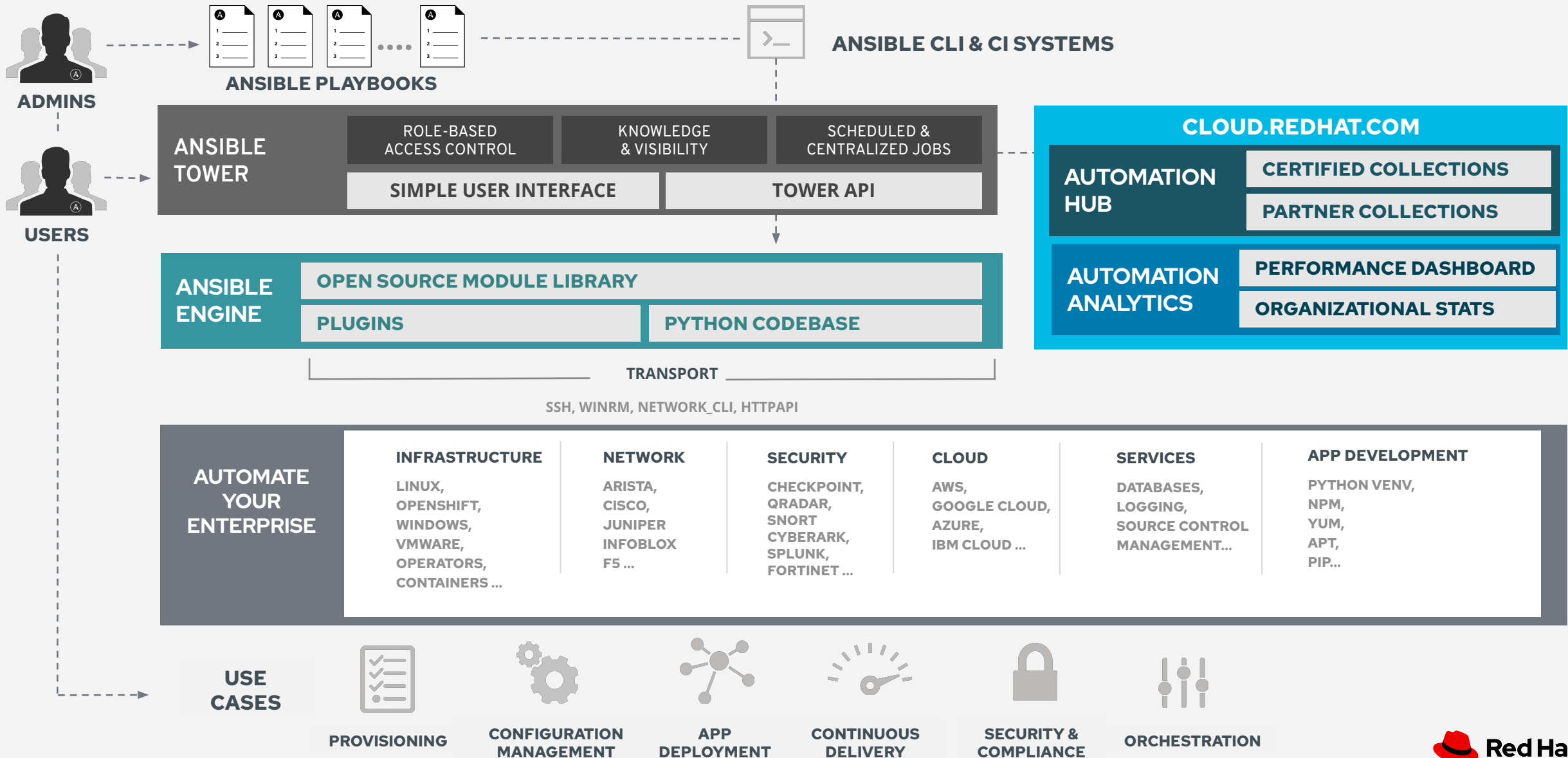
Centralized logging

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

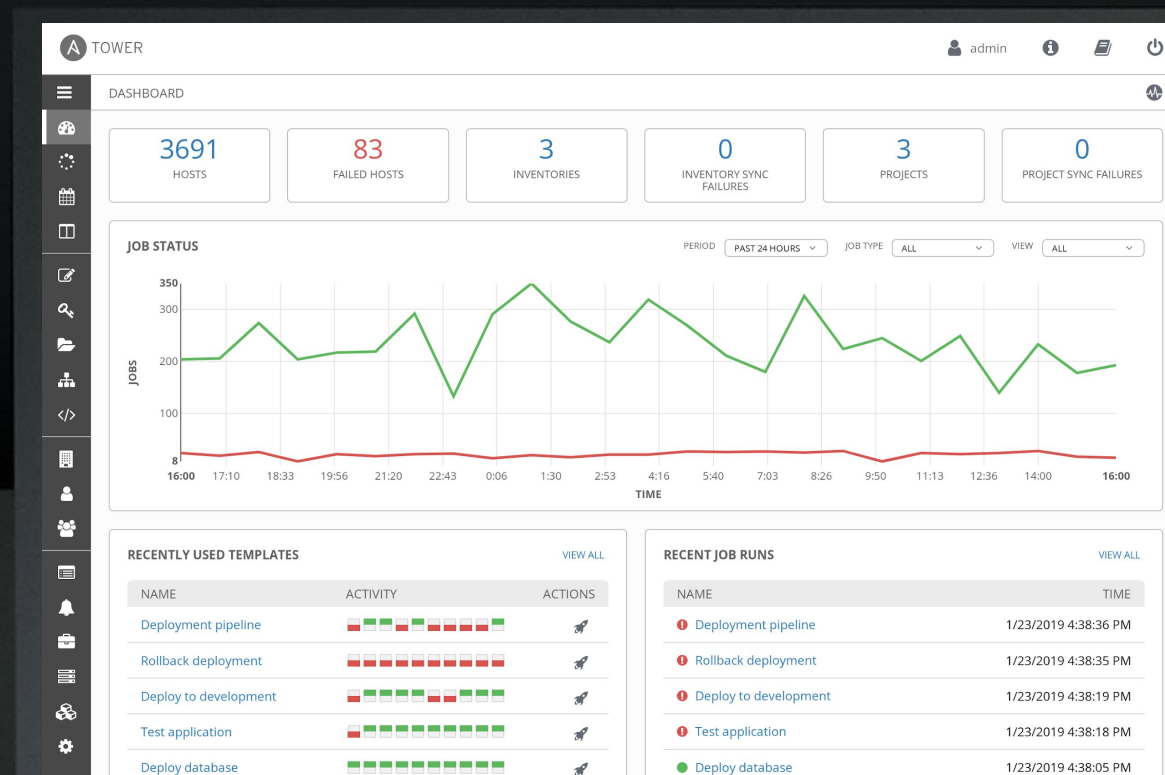
Ansible Tower's multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.

Ansible Automation Platform





Job Template



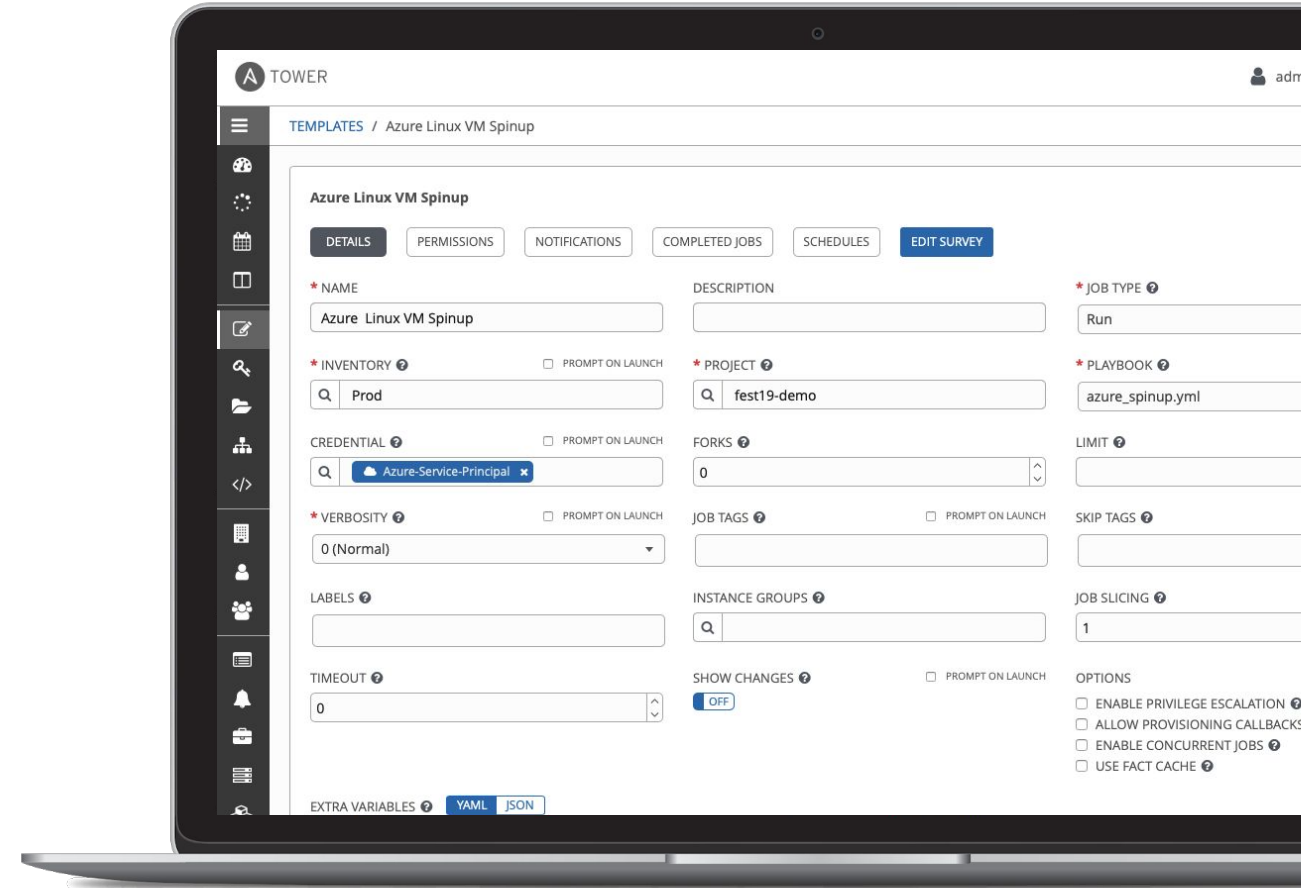
Job Templates

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:

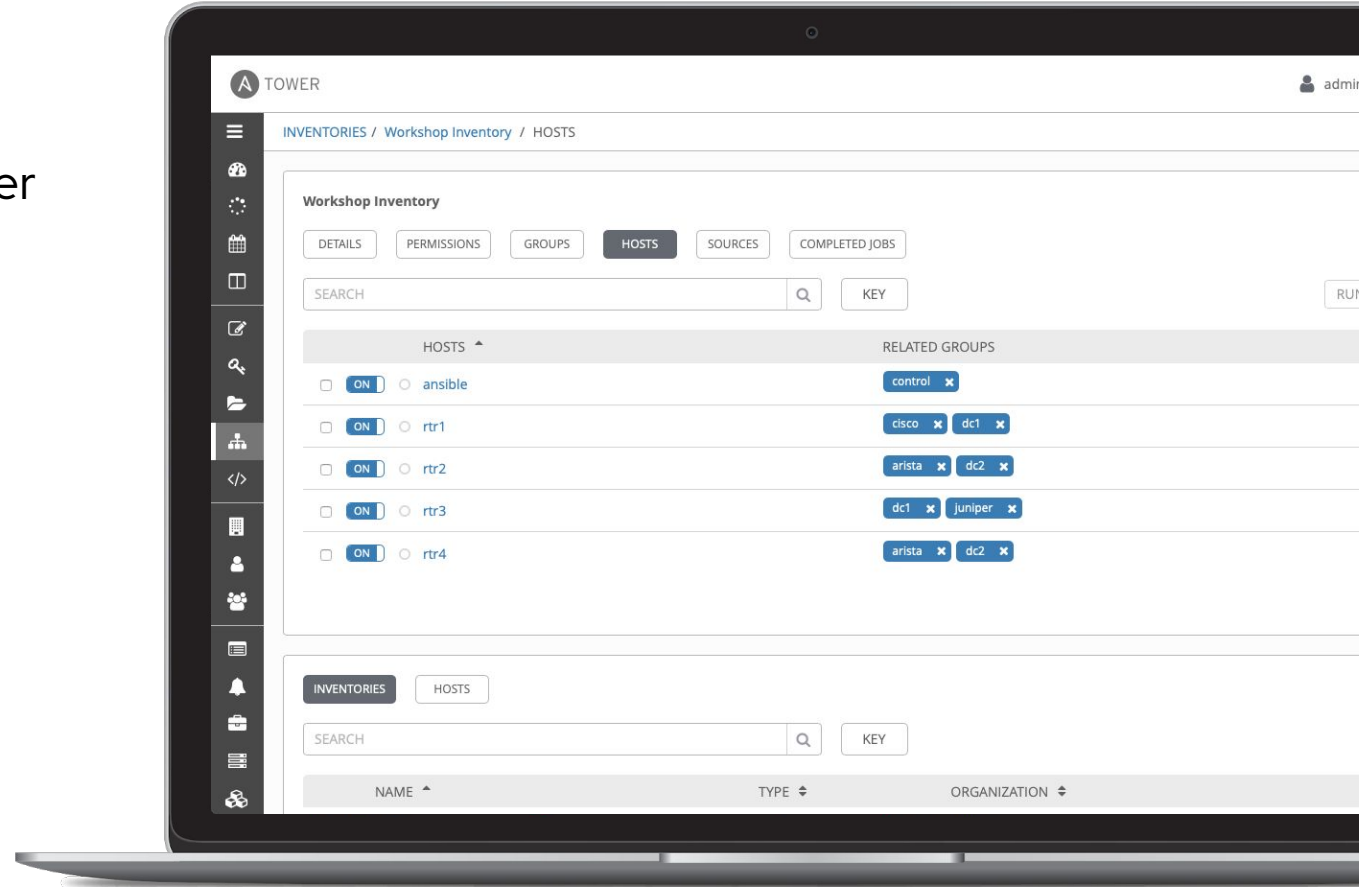
- An **Inventory** to run the job against
- A **Credential** to login to devices.
- A **Project** which contains Ansible Playbooks



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

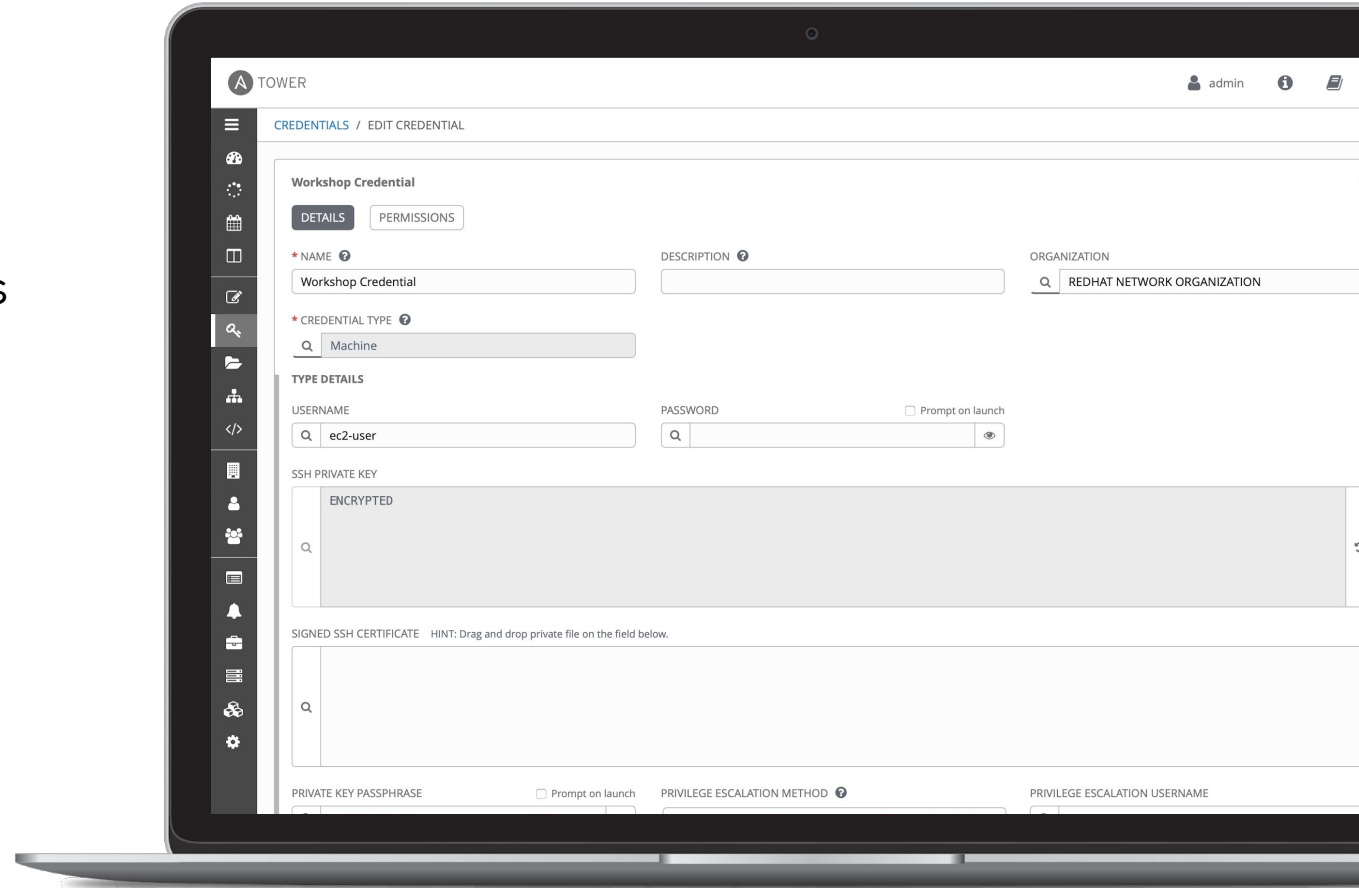


Credentials

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

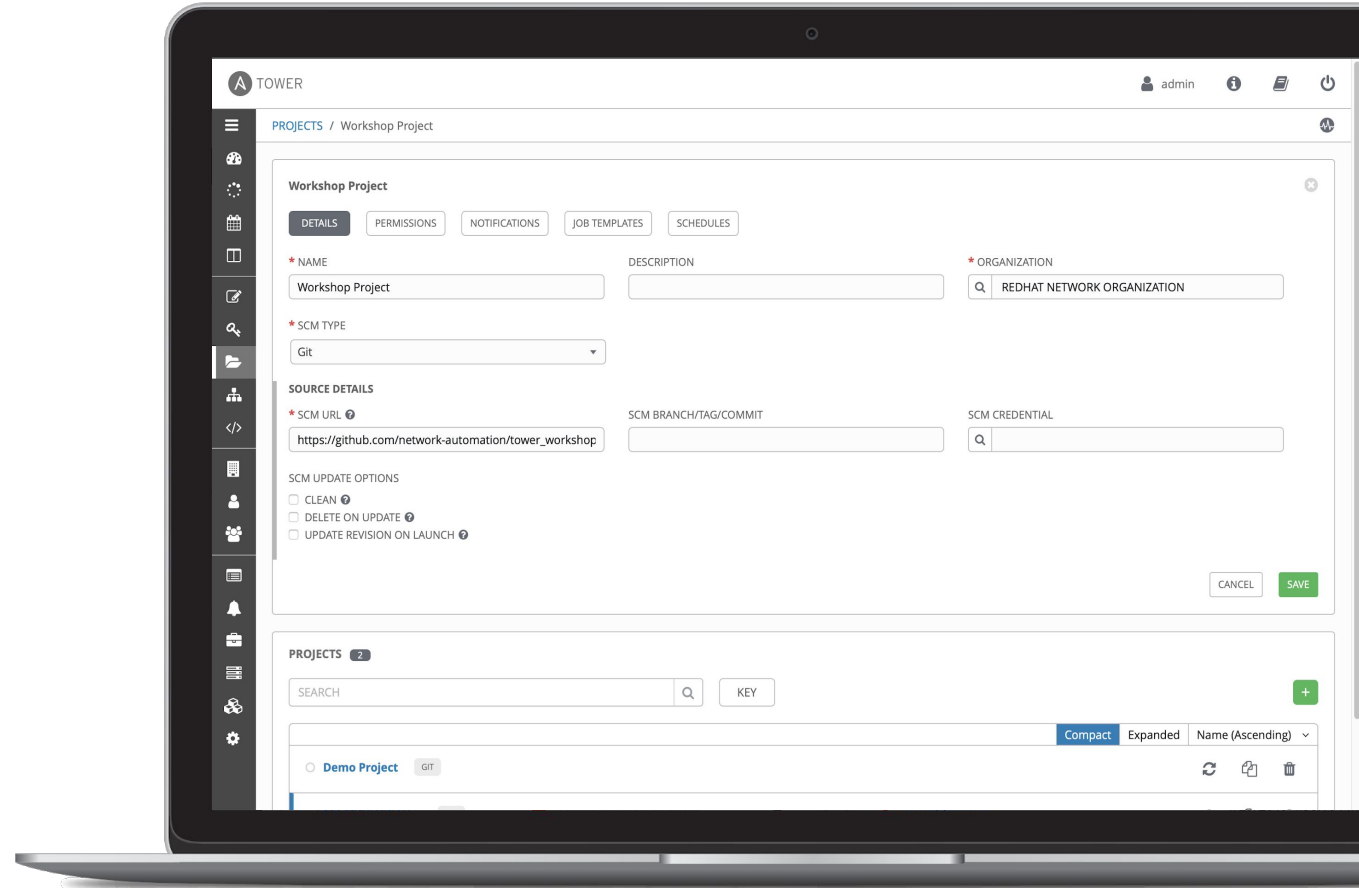
Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.



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.





Red Hat Ansible Automation

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.

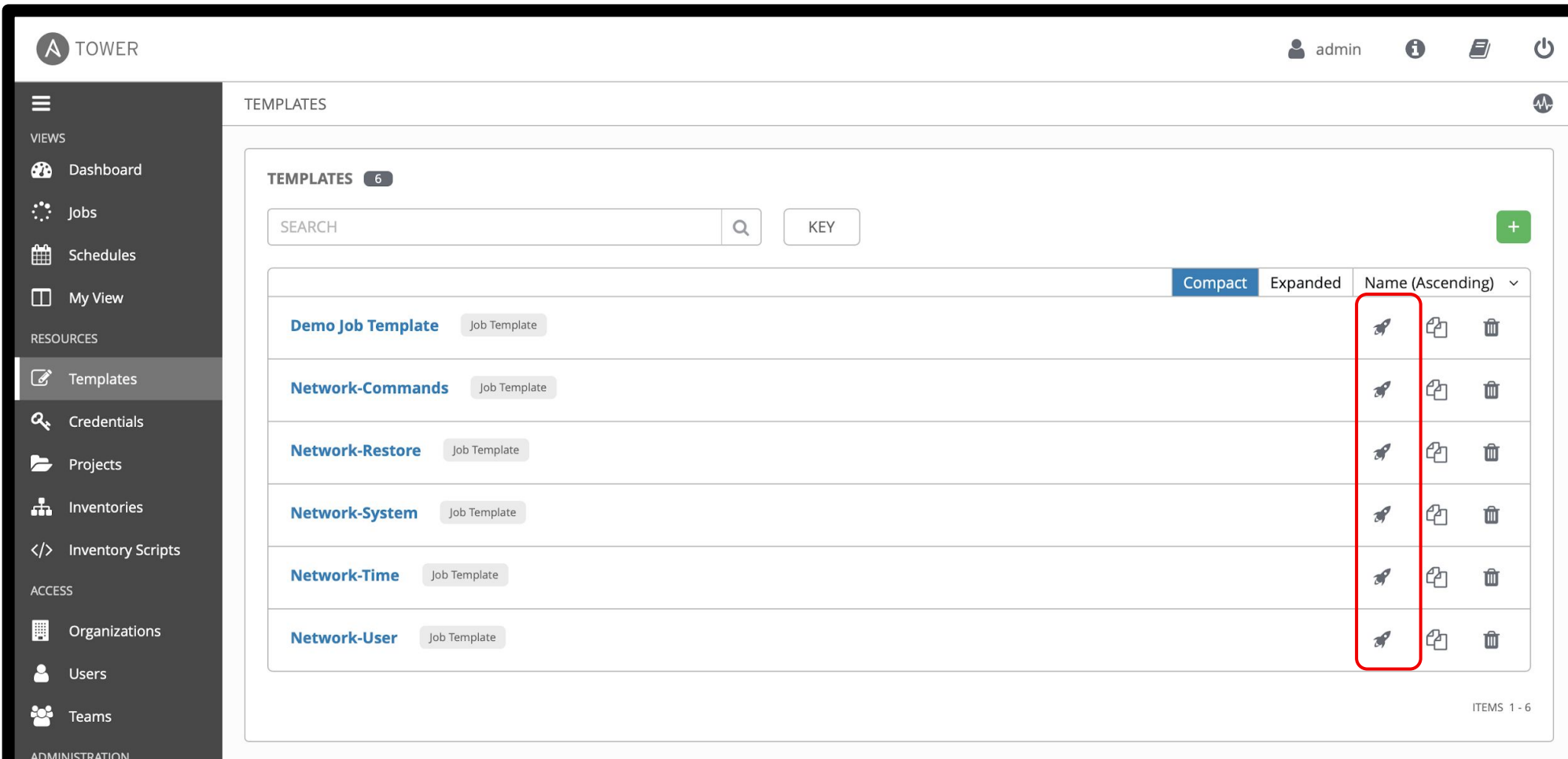


The screenshot displays the Tower web interface. On the left is a dark sidebar menu with sections: VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), ACCESS (Organizations, Users, Teams), and ADMINISTRATION. The 'Templates' item under RESOURCES is highlighted. The main content area is titled 'TEMPLATES' and shows a list of 6 templates. At the top of the list are a search bar and a 'KEY' button. The list is in 'Compact' view, showing columns for the template name, a 'Job Template' tag, and action icons (rocket, copy, delete). The templates listed are Demo Job Template, Network-Commands, Network-Restore, Network-System, Network-Time, and Network-User. A green '+' button is in the top right of the list area. The bottom right corner of the list area shows 'ITEMS 1 - 6'.

TEMPLATES 6			
SEARCH		Q	KEY
		Compact	Expanded
		Name (Ascending) v	
Demo Job Template	Job Template	🚀	📄
Network-Commands	Job Template	🚀	📄
Network-Restore	Job Template	🚀	📄
Network-System	Job Template	🚀	📄
Network-Time	Job Template	🚀	📄
Network-User	Job Template	🚀	📄

Executing an existing Job Template

Job Templates can be launched by clicking the **rocketship button**  for the corresponding Job Template



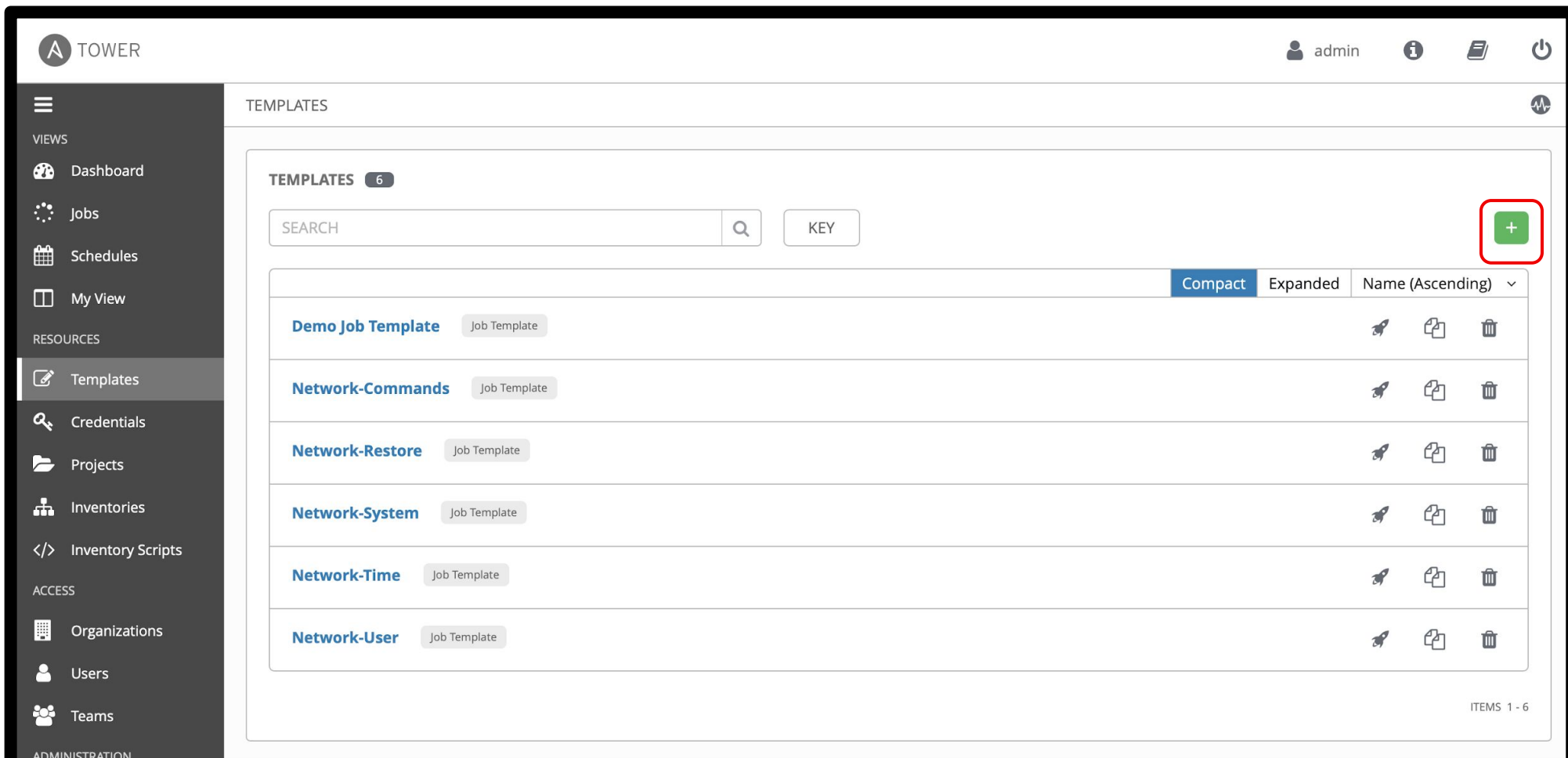
The screenshot shows the Tower web interface. The left sidebar contains navigation links for VIEWS (Dashboard, Jobs, Schedules, My View) and RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts). The main content area is titled 'TEMPLATES' and shows a list of 6 job templates. The 'Demo Job Template' is highlighted. The rocketship icon in the 'Demo Job Template' row is circled in red.

TEMPLATES 6		SEARCH	KEY	+
		Compact	Expanded	Name (Ascending) v
Demo Job Template	Job Template			  
Network-Commands	Job Template			  
Network-Restore	Job Template			  
Network-System	Job Template			  
Network-Time	Job Template			  
Network-User	Job Template			  

ITEMS 1 - 6

Creating a new Job Template (1/2)

New Job Templates can be created by clicking the **plus button**



The screenshot shows the Tower web interface. The left sidebar contains navigation links for VIEWS (Dashboard, Jobs, Schedules, My View) and RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts). The main content area is titled 'TEMPLATES' and shows a list of 6 job templates. A green plus button is highlighted in the top right corner of the templates list. The list includes templates like 'Demo Job Template', 'Network-Commands', 'Network-Restore', 'Network-System', 'Network-Time', and 'Network-User'. Each template has a 'Job Template' tag and icons for launch, copy, and delete. The interface also shows a search bar, a 'KEY' button, and a 'Compact' view toggle.

TEMPLATES 6		SEARCH	Q	KEY		
				Compact	Expanded	Name (Ascending) v
Demo Job Template	Job Template					Launch Copy Delete
Network-Commands	Job Template					Launch Copy Delete
Network-Restore	Job Template					Launch Copy Delete
Network-System	Job Template					Launch Copy Delete
Network-Time	Job Template					Launch Copy Delete
Network-User	Job Template					Launch Copy Delete

ITEMS 1 - 6

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 .

NEW JOB TEMPLATE

DETAILS | PERMISSIONS | COMPLETED JOBS | SCHEDULES | ADD SURVEY

* NAME	DESCRIPTION	* JOB TYPE ? ☐ PROMPT ON LAUNCH
		Run
* INVENTORY ? ☐ PROMPT ON LAUNCH	* PROJECT ?	* PLAYBOOK ?
		Choose a playbook
CREDENTIAL ? ☐ PROMPT ON LAUNCH	FORKS ?	LIMIT ? ☐ PROMPT ON LAUNCH
	0	
* VERBOSITY ? ☐ PROMPT ON LAUNCH	JOB TAGS ? ☐ PROMPT ON LAUNCH	SKIP TAGS ? ☐ PROMPT ON LAUNCH
0 (Normal)		
LABELS ?	INSTANCE GROUPS ?	JOB SLICING ?
		1
TIMEOUT ?	SHOW CHANGES ? ☐ PROMPT ON LAUNCH	OPTIONS
0	OFF	☐ ENABLE PRIVILEGE ESCALATION ?
		☐ ALLOW PROVISIONING CALLBACKS ?



Red Hat Ansible Automation Platform

Exercise 6 - Creating a Tower Job Template

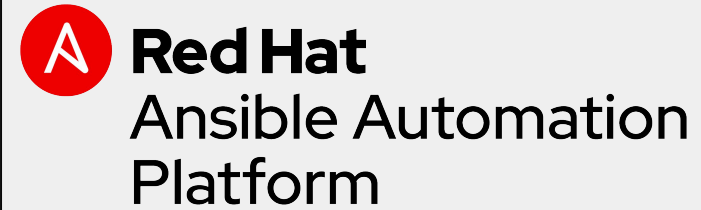
Demonstrate a network backup configuration job template for Red Hat Ansible Tower.

Approximate time: 15 mins

Exercise 7

Topics Covered:

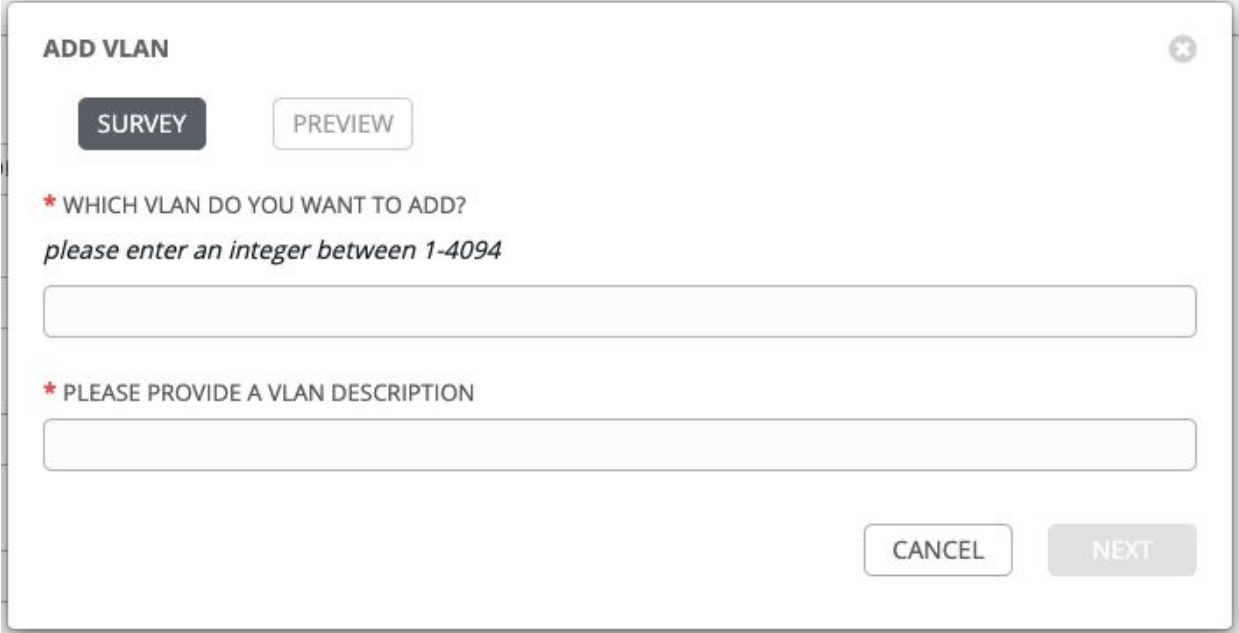
- Understanding Extra Vars
- Building a Tower Survey
- Self-service IT with Tower Surveys



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.



The screenshot shows a web interface for adding a VLAN. At the top, there's a title 'ADD VLAN' and a close button. Below the title are two buttons: 'SURVEY' (highlighted) and 'PREVIEW'. The form contains two required questions, each with a red asterisk. The first question is 'WHICH VLAN DO YOU WANT TO ADD?' with a subtext 'please enter an integer between 1-4094' and a text input field. The second question is 'PLEASE PROVIDE A VLAN DESCRIPTION' with a text input field. At the bottom right, there are 'CANCEL' and 'NEXT' buttons.

ADD VLAN

SURVEY PREVIEW

* WHICH VLAN DO YOU WANT TO ADD?
please enter an integer between 1-4094

* PLEASE PROVIDE A VLAN DESCRIPTION

CANCEL NEXT

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.

ADD SURVEY

The screenshot displays the Tower web interface. On the left is a dark sidebar with navigation links: TOWER, VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), and ACCESS (Organizations). The main content area shows the 'Configure Banner' window for a job template. The window has tabs for DETAILS, PERMISSIONS, NOTIFICATIONS, COMPLETED JOBS, SCHEDULES, and EDIT SURVEY (which is highlighted with a red box). The 'EDIT SURVEY' tab contains several configuration fields: NAME (Configure Banner), DESCRIPTION, JOB TYPE (Run), INVENTORY (Workshop Inventory), PROJECT (Workshop Project), PLAYBOOK (network_banner.yml), CREDENTIAL (Workshop Credential), FORKS (0), LIMIT, VERBOSITY (0 (Normal)), JOB TAGS, SKIP TAGS, LABELS, INSTANCE GROUPS, and JOB SLICING. Each field has a search icon and a 'PROMPT ON LAUNCH' checkbox. The top right of the interface shows the user 'admin' and various system icons.

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.

The screenshot displays the 'CONFIGURE BANNER | SURVEY' window with a toggle switch set to 'ON'. The window is divided into two main sections: 'EDIT SURVEY PROMPT' on the left and 'PREVIEW' on the right.

EDIT SURVEY PROMPT

- * PROMPT**: A text input field containing 'Please enter the banner text'.
- DESCRIPTION**: A text input field containing 'Please type into the text field the desired banner'.
- * ANSWER VARIABLE NAME**: A text input field containing 'net_banner'.
- * ANSWER TYPE**: A dropdown menu currently set to 'Textarea'.
- MINIMUM LENGTH**: A numeric input field set to '0'.
- MAXIMUM LENGTH**: A numeric input field set to '4096'.
- DEFAULT ANSWER**: An empty text input field.
- REQUIRED**: A checkbox that is checked.

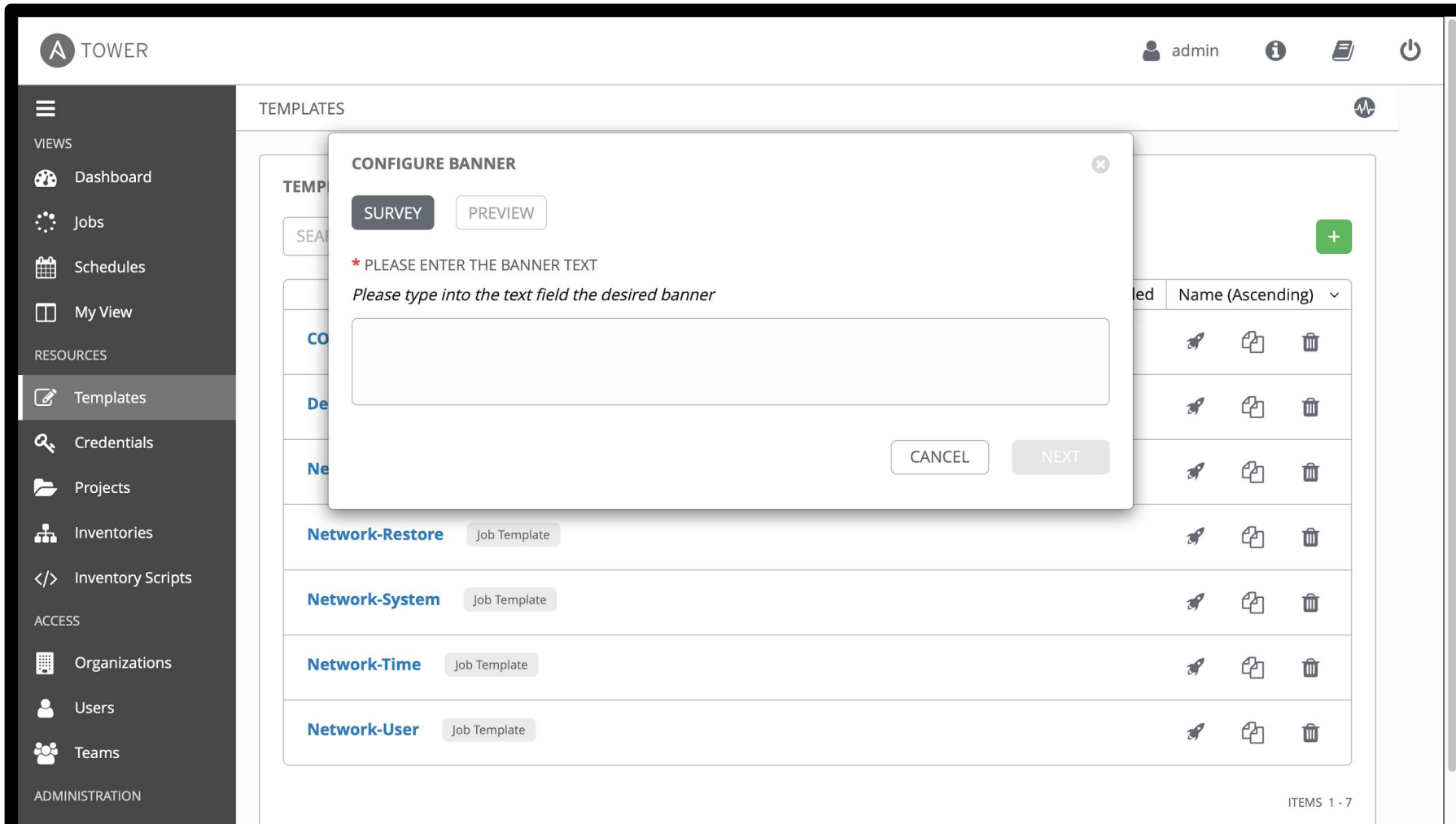
PREVIEW

- * PLEASE ENTER THE BANNER TEXT**: A heading for the preview section.
- Please type into the text field the desired banner*: A prompt text displayed above a large, empty text area.
- Below the text area are three vertical dots, a blue edit icon, and a trash icon.

At the bottom of the window, there are buttons for 'CLEAR', 'UPDATE', 'CANCEL', and 'OK'.

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.



The screenshot displays the Tower web interface. On the left is a dark sidebar with navigation links: VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), ACCESS (Organizations, Users, Teams), and ADMINISTRATION. The main content area is titled 'TEMPLATES' and shows a list of job templates. A modal dialog box titled 'CONFIGURE BANNER' is open, featuring 'SURVEY' and 'PREVIEW' tabs. The 'SURVEY' tab is active, showing a text input field with the placeholder text 'Please type into the text field the desired banner'. Below the input field are 'CANCEL' and 'NEXT' buttons. The background list of templates includes 'Network-Restore', 'Network-System', 'Network-Time', and 'Network-User', each with a 'Job Template' label and action icons (run, copy, delete). The top right of the interface shows the user 'admin' and system status icons. The bottom right corner of the interface indicates 'ITEMS 1 - 7'.

Template Name	Type	Actions
Network-Restore	Job Template	Run, Copy, Delete
Network-System	Job Template	Run, Copy, Delete
Network-Time	Job Template	Run, Copy, Delete
Network-User	Job Template	Run, Copy, Delete



Red Hat Ansible Automation Platform

Exercise 7- Creating a Survey

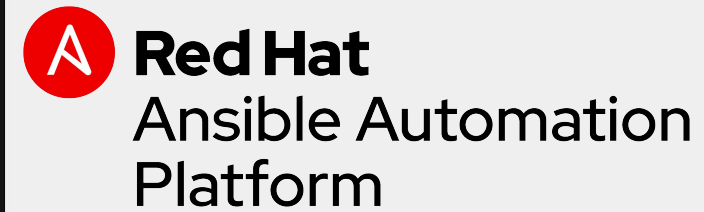
Demonstrate the use of Ansible Tower survey feature

Approximate time: 15 mins

Exercise 8

Topics Covered:

- Understanding Organizations
- Understanding Teams
- Understanding Users



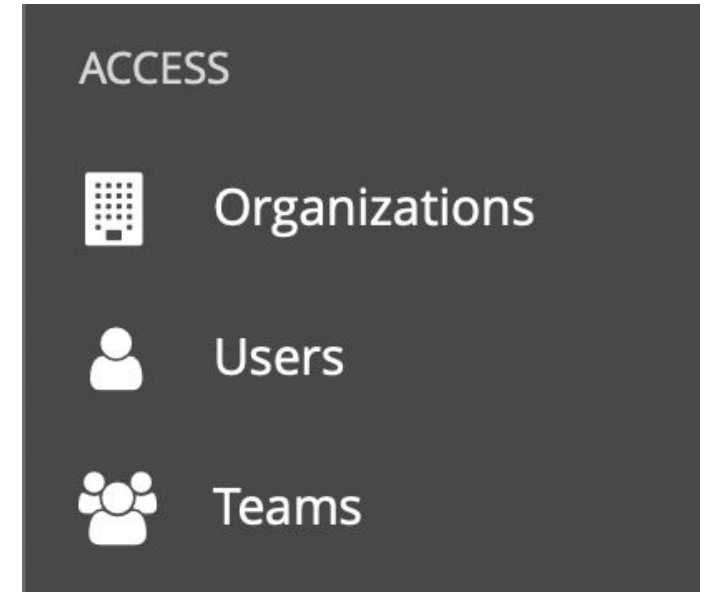
Role Based Access Control (RBAC)

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.



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 will open up the Organizations window



Organizations

in the left menu

TOWER admin

ORGANIZATIONS

ORGANIZATIONS 3

SEARCH KEY +

Default

- 0 USERS
- 1 INVENTORIES
- 1 JOB TEMPLATES
- 0 TEAMS
- 1 PROJECTS
- 0 ADMINS

REDHAT COMPUTE ORGANIZATION

- 0 USERS
- 0 INVENTORIES
- 0 JOB TEMPLATES
- 2 TEAMS
- 0 PROJECTS
- 0 ADMINS

REDHAT NETWORK ORGANIZATION

- 2 USERS
- 1 INVENTORIES
- 6 JOB TEMPLATES
- 2 TEAMS
- 1 PROJECTS
- 1 ADMINS

ITEMS 1 - 3

Viewing Teams

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



in the left menu

The screenshot shows the Tower web interface. The top header includes the "TOWER" logo, a user profile for "admin", and icons for information, a document, and a power button. The left sidebar contains a menu with sections: VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), ACCESS (Organizations, Users, Teams), and ADMINISTRATION. The "Teams" menu item is highlighted. The main content area is titled "TEAMS" and features a sub-header "TEAMS 4". Below this is a search bar with a "KEY" input field and a green "+" button. A table lists the teams:

NAME ^	ORGANIZATION ^	ACTIONS
Compute T1	REDHAT COMPUTE ORGANIZATION	
Compute T2	REDHAT COMPUTE ORGANIZATION	
Netadmin	REDHAT NETWORK ORGANIZATION	
Netops	REDHAT NETWORK ORGANIZATION	

ITEMS 1 - 4

Viewing Users

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



in the left menu

The screenshot shows the Tower web interface. The left sidebar contains a menu with sections: VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), and ACCESS (Organizations, Users, Teams). The "Users" button is highlighted. The main content area is titled "USERS" and shows a list of 8 users. At the top of the list are search and key filters, and a green "+" button. The table has columns for USERNAME, FIRST NAME, LAST NAME, and ACTIONS. The users listed are admin, bbelcher, gbelcher, lbelcher, libelcher, network-admin, network-operator, and tbelcher. Each user row has edit and delete icons in the ACTIONS column.

USERNAME	FIRST NAME	LAST NAME	ACTIONS
admin			
bbelcher	Bob	Belcher	
gbelcher	Gene	Belcher	
lbelcher	Louise	Belcher	
libelcher	Linda	Belcher	
network-admin	Larry	Niven	
network-operator	Issac	Assimov	
tbelcher	Tina	Belcher	

ITEMS 1 - 8



Exercise 8 - Understanding RBAC

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

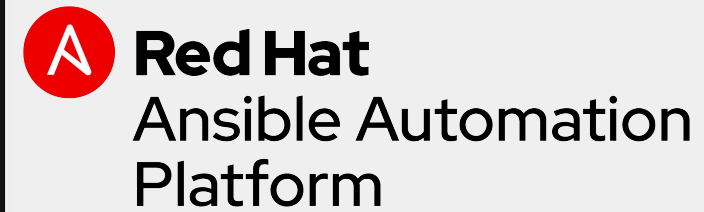
Approximate time: 15 mins



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.



The screenshot displays the Red Hat Tower web interface. The left sidebar contains a navigation menu with sections: VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), ACCESS (Organizations, Users, Teams), and ADMINISTRATION. The 'Templates' button under the RESOURCES section is highlighted. The main content area is titled 'TEMPLATES' and shows a list of 6 templates. The list includes a search bar, a 'KEY' button, and a '+ ' button. The templates are displayed in a table with columns for Name, Action, and a dropdown menu. The templates listed are: Demo Job Template, Network-Commands, Network-Restore, Network-System, Network-Time, and Network-User. Each template has a 'Job Template' label and icons for launch, copy, and delete. The bottom right corner of the interface shows 'ITEMS 1 - 6'.

TEMPLATES 6	SEARCH	KEY	+
	Compact	Expanded	Name (Ascending) v
Demo Job Template	Job Template		
Network-Commands	Job Template		
Network-Restore	Job Template		
Network-System	Job Template		
Network-Time	Job Template		
Network-User	Job Template		

ITEMS 1 - 6

Adding a new Workflow Template

To add a new **Workflow** click on the green + button



This time select the **Workflow Template**

The screenshot shows the Tower web interface. The left sidebar contains navigation links: TOWER, VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), and ACCESS (Organizations, Users). The main content area is titled 'TEMPLATES' and shows a list of templates. A search bar and a 'KEY' input field are at the top. A table lists templates: 'Backup network configurations', 'Configure Banner', 'Demo Job Template', 'Network-Commands', 'Network-Restore', and 'Network-System'. Each row has a 'Job Template' label and a progress bar. A red box highlights the green plus button and the dropdown menu options 'Job Template' and 'Workflow Template'.

Template Name	Type	Progress	Actions
Backup network configurations	Job Template	100%	Run, Copy, Delete
Configure Banner	Job Template	0%	Run, Copy, Delete
Demo Job Template	Job Template	0%	Run, Copy, Delete
Network-Commands	Job Template	0%	Run, Copy, Delete
Network-Restore	Job Template	0%	Run, Copy, Delete
Network-System	Job Template	0%	Run, Copy, Delete

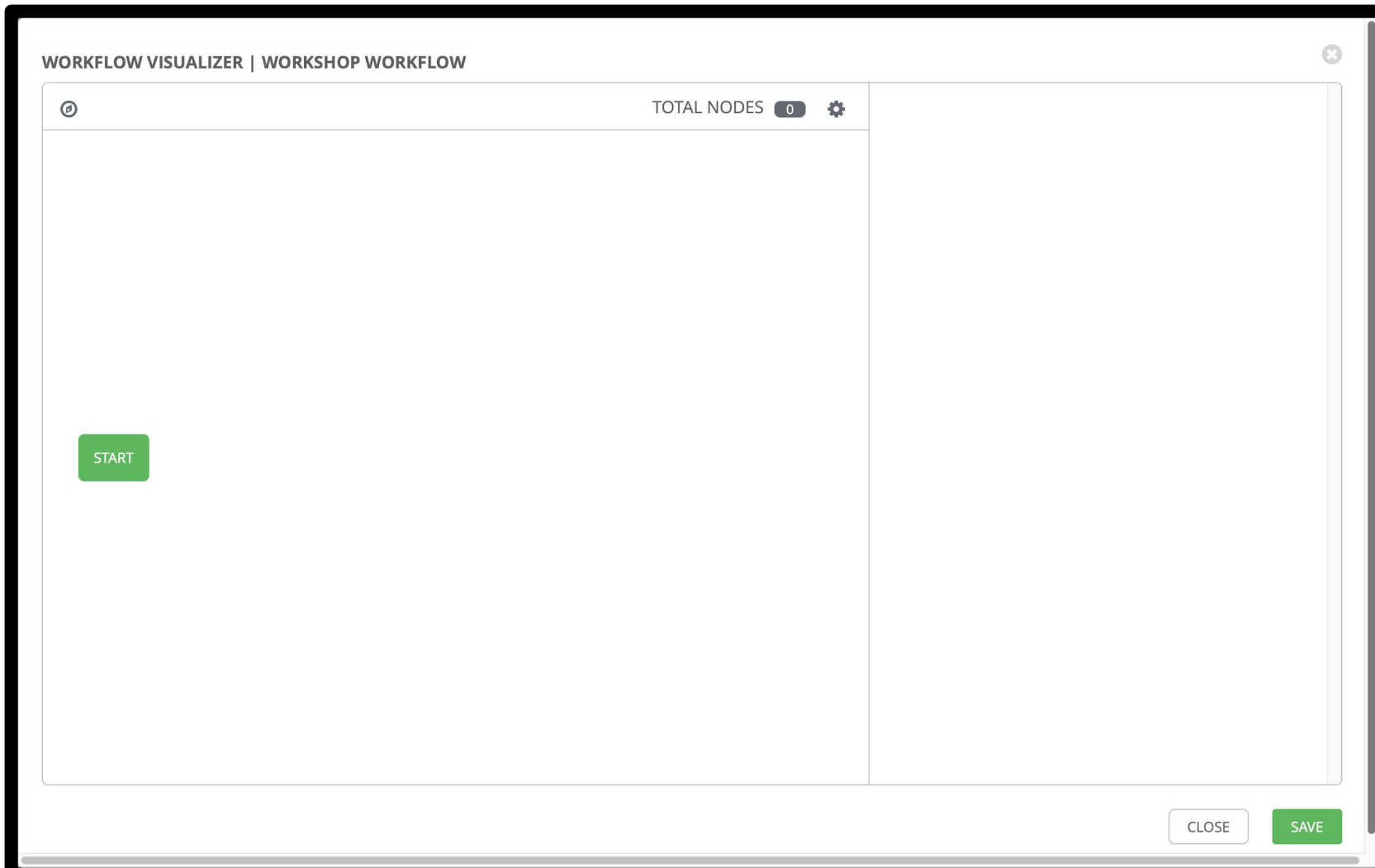
Creating the Workflow

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

The screenshot displays the Tower web interface for configuring a 'WORKSHOP WORKFLOW' template. The left sidebar contains navigation menus for 'VIEWS' (Dashboard, Jobs, Schedules, My View), 'RESOURCES' (Templates, Credentials, Projects, Inventories, Inventory Scripts), and 'ACCESS' (Organizations). The main content area is titled 'TEMPLATES / WORKSHOP WORKFLOW' and features a tabbed interface with 'DETAILS', 'PERMISSIONS', 'NOTIFICATIONS', 'COMPLETED JOBS', 'SCHEDULES', and 'ADD SURVEY'. The 'DETAILS' tab is active, showing a form with the following fields: 'NAME' (set to 'WORKSHOP WORKFLOW'), 'DESCRIPTION' (empty), 'ORGANIZATION' (set to 'Default'), 'INVENTORY' (set to 'Workshop Inventory'), 'PROMPT ON LAUNCH' (checkbox), 'LABELS' (empty), 'EXTRA VARIABLES' (with 'YAML' and 'JSON' tabs), and 'ENABLE CONCURRENT JOBS' (checkbox). The 'WORKFLOW VISUALIZER' button is highlighted with a red box. The bottom of the page shows a code editor with a single line of text: '1 ---'.

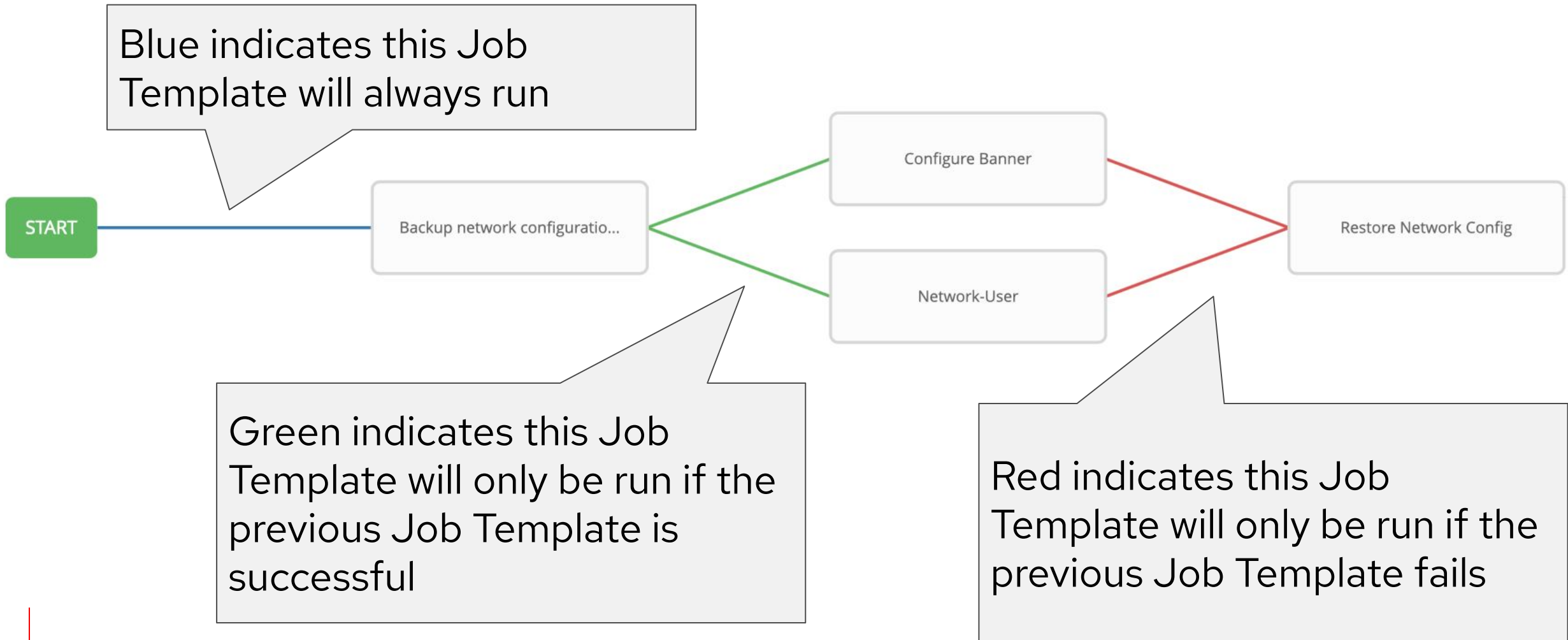
Workflow Visualizer

The workflow visualizer will start as a blank canvas.



Visualizing a Workflow

Workflows can branch out, or converge in.





Red Hat Ansible Automation Platform

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

WORKSHOPS & TRAINING

ansible.com/workshops

[Red Hat Training](#)

JOIN THE COMMUNITY

ansible.com/community

SHARE YOUR STORY

[Follow us @Ansible](#)

[Friend us on Facebook](#)

Chat with us

- **Slack**

<https://ansiblenetwork.slack.com>

Join by clicking here <http://bit.ly/ansibleslack>

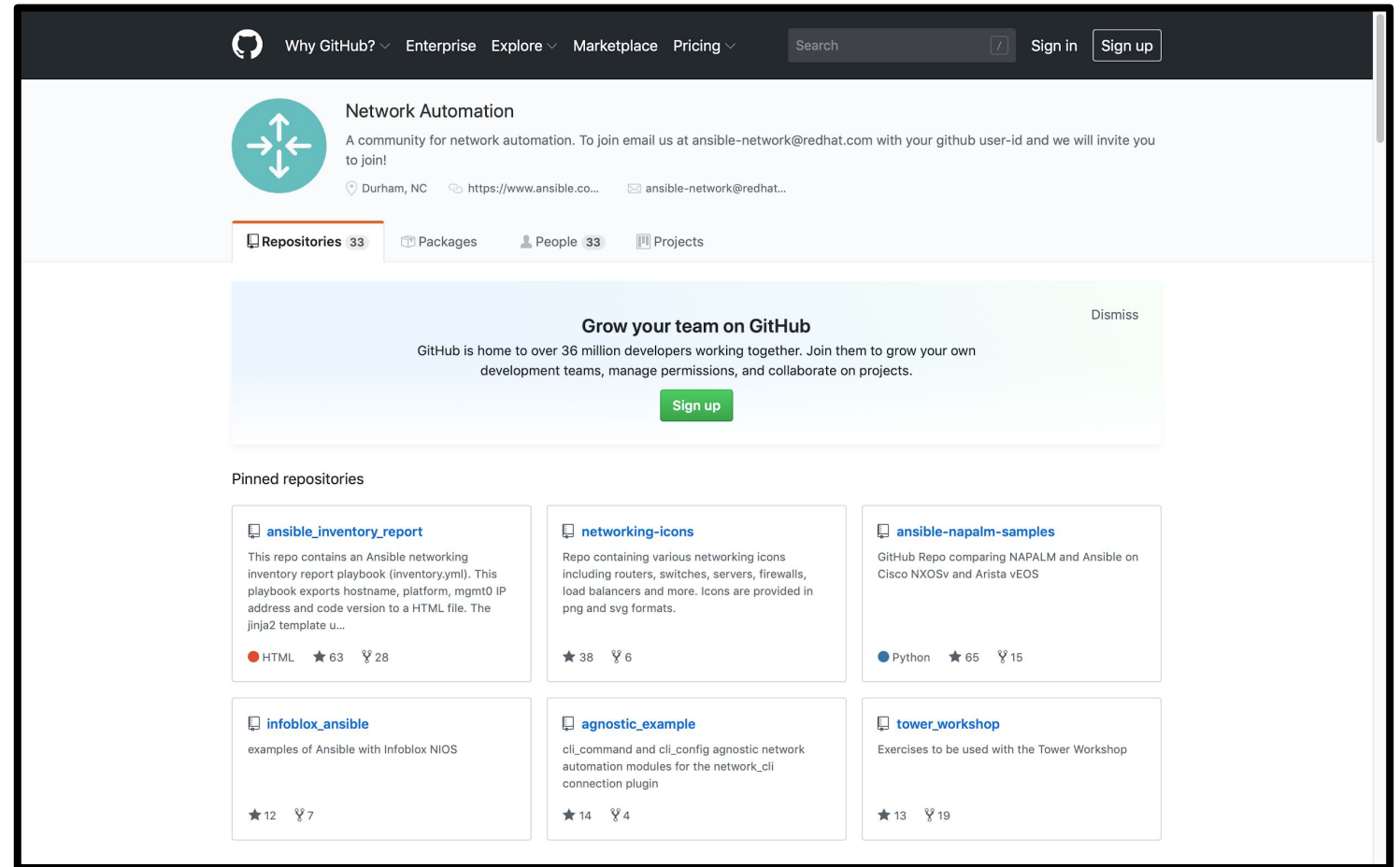
- **IRC**

#ansible-network on freenode

<http://webchat.freenode.net/?channels=ansible-network>

Bookmark the Github organization

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



Thank you



linkedin.com/company/red-hat



youtube.com/AnsibleAutomation



facebook.com/ansibleautomation



twitter.com/ansible



github.com/ansible