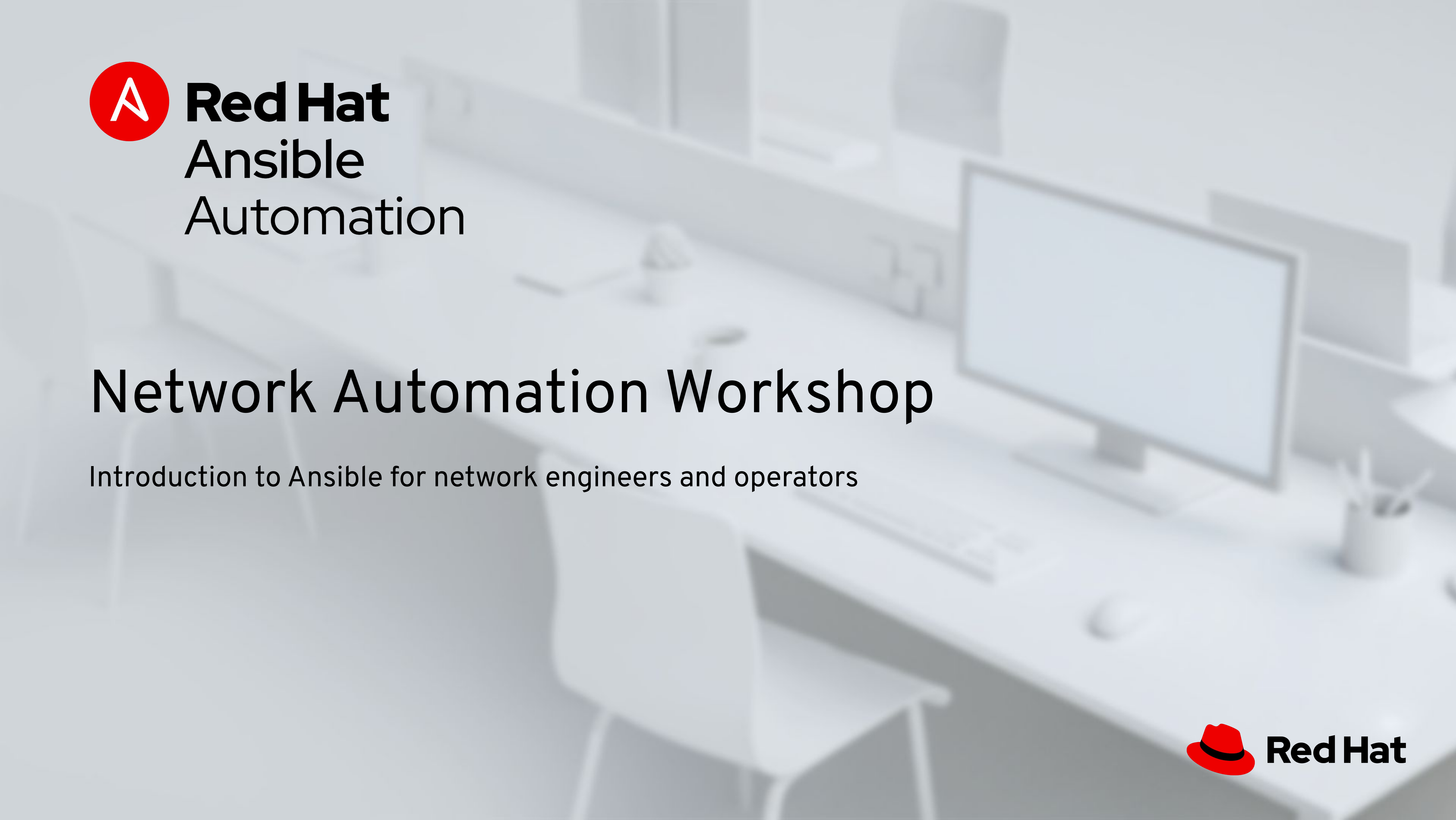


Red Hat
Ansible
Automation

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

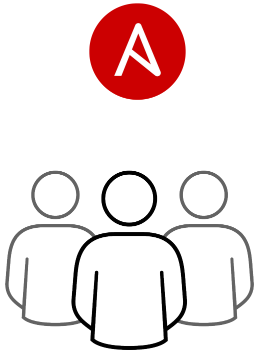
Section 1

Topics Covered:

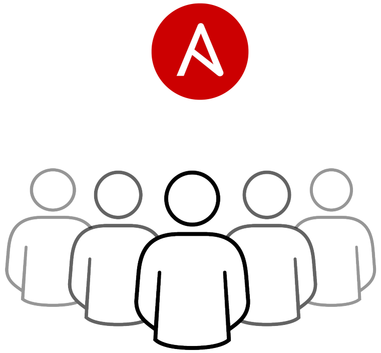
- Why Network Automation?
- How Ansible Network Automation works
- Understanding Inventory
- An example Ansible Playbook

Red Hat
Ansible
Automation

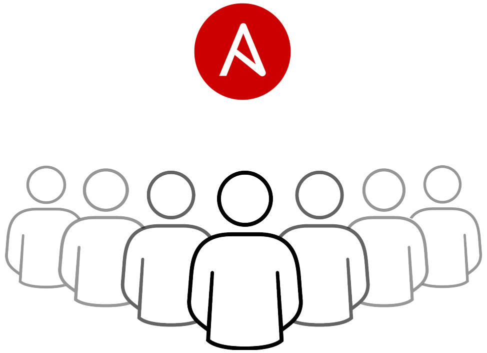
Automation happens when one person meets a problem they never want to solve again



ACCELERATE



INTEGRATE



COLLABORATE

71%

of networks are still driven manually via CLI

Source: Gartner, *Look Beyond Network Vendors for Innovation.* January 2018

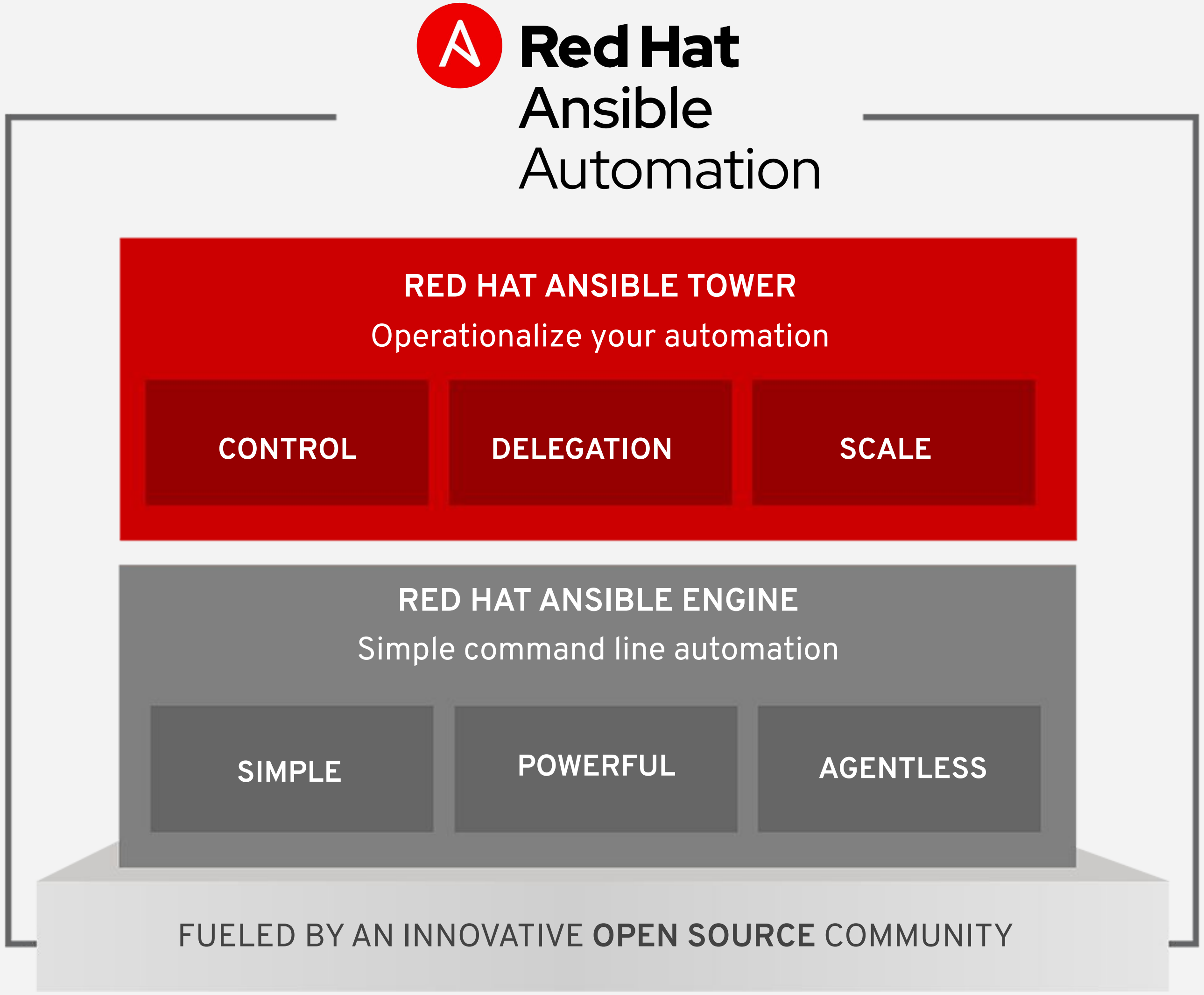
NOT AS SIMPLE ANYMORE

What is Ansible Automation?

Ansible Automation is the enterprise **framework** for automating across IT operations.

Ansible Engine runs Ansible Playbooks, the automation **language** that can perfectly describe an IT application infrastructure.

Ansible Tower allows you **scale** IT automation, manage complex deployments and speed productivity.



Red Hat Ansible Automation

RED HAT ANSIBLE TOWER

Operationalize your automation

CONTROL

DELEGATION

SCALE

RED HAT ANSIBLE ENGINE

Simple command line automation

SIMPLE

POWERFUL

AGENTLESS

FUELED BY AN INNOVATIVE **OPEN SOURCE** COMMUNITY

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*

What can I do using Ansible?

Automate the deployment and management of your entire IT footprint.

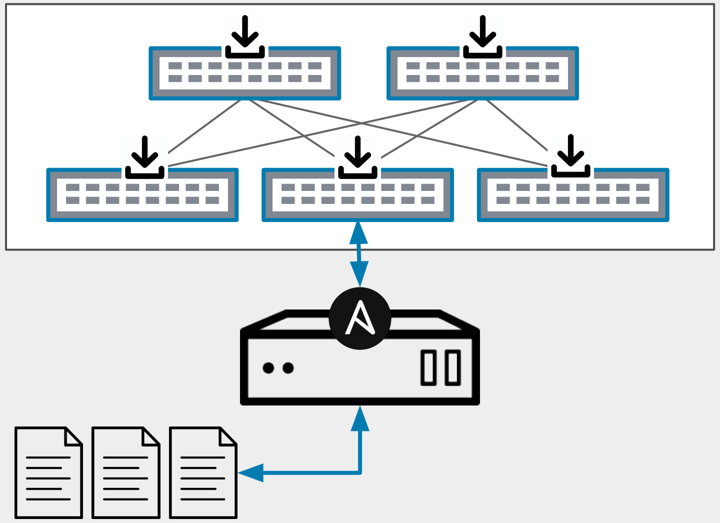
Do this...

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

On these...

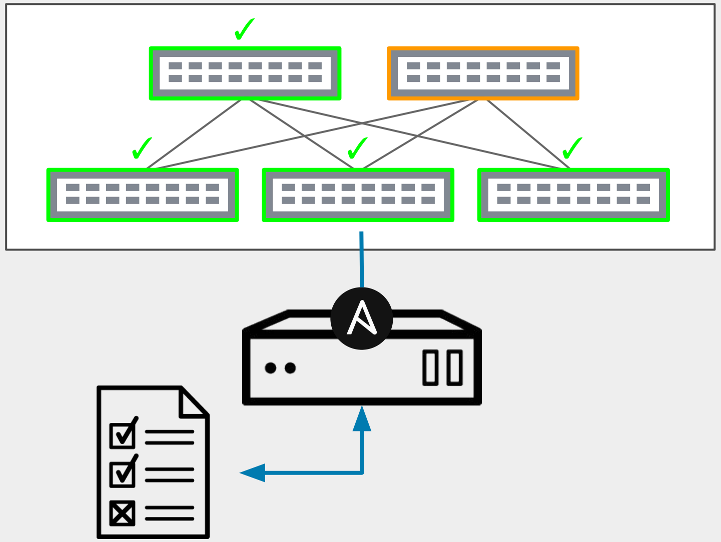
- Firewalls
- Load Balancers
- Applications
- Containers
- Clouds
- Servers
- Infrastructure
- Storage
- Network Devices
- And more...

Common use cases



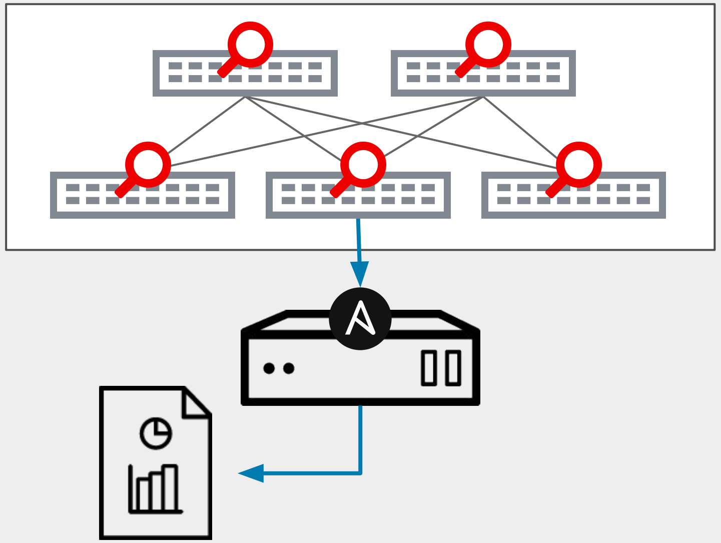
Backup and Restore

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



Configuration Compliance

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

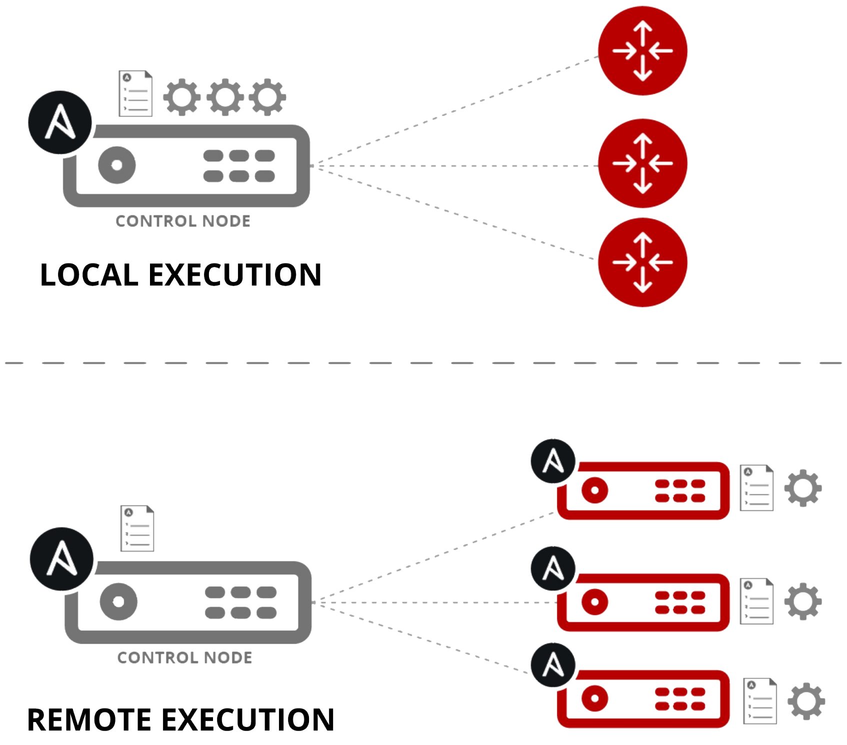


Dynamic Documentation

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

How Ansible Network Automation works

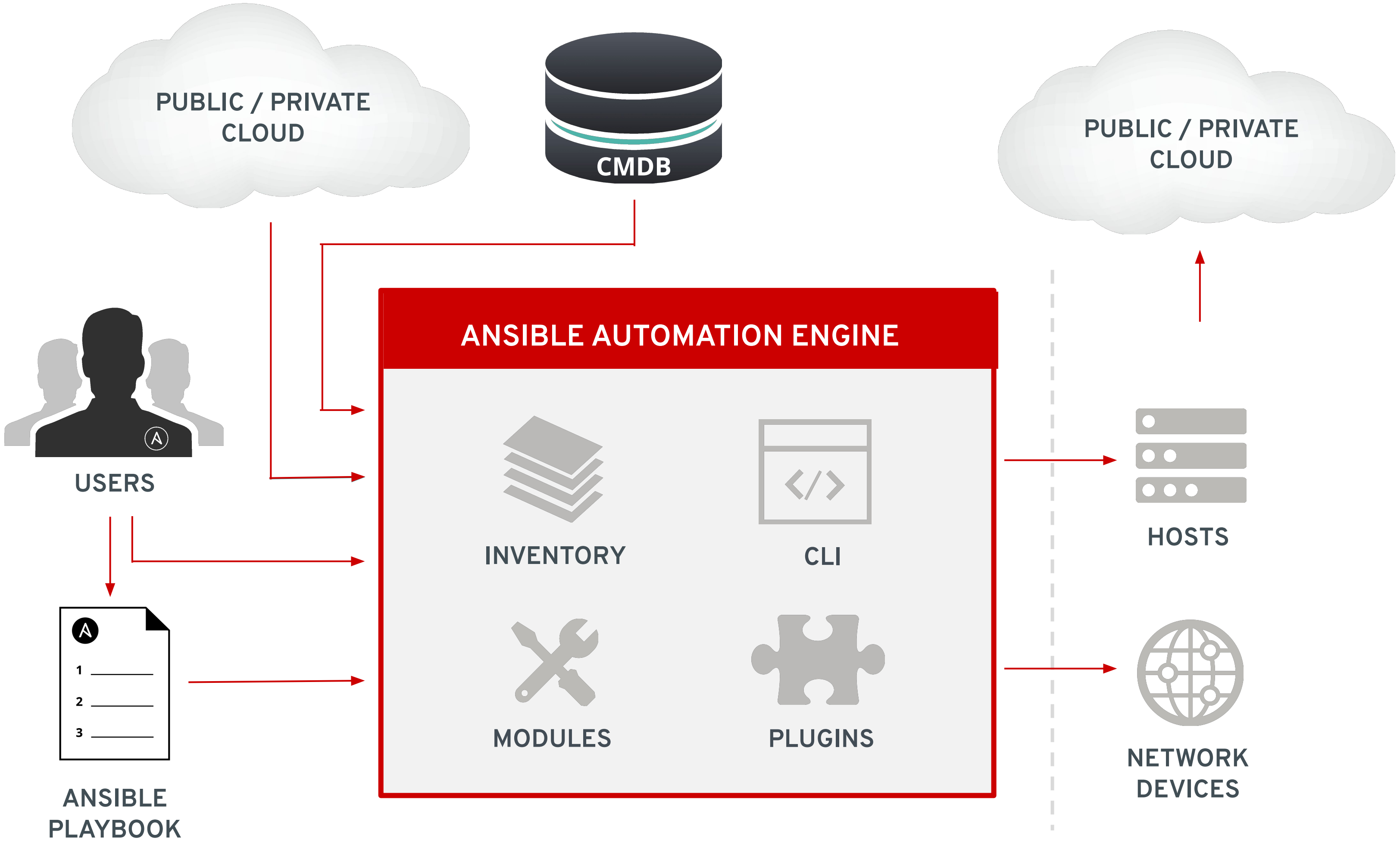
Module code is executed locally on the control node



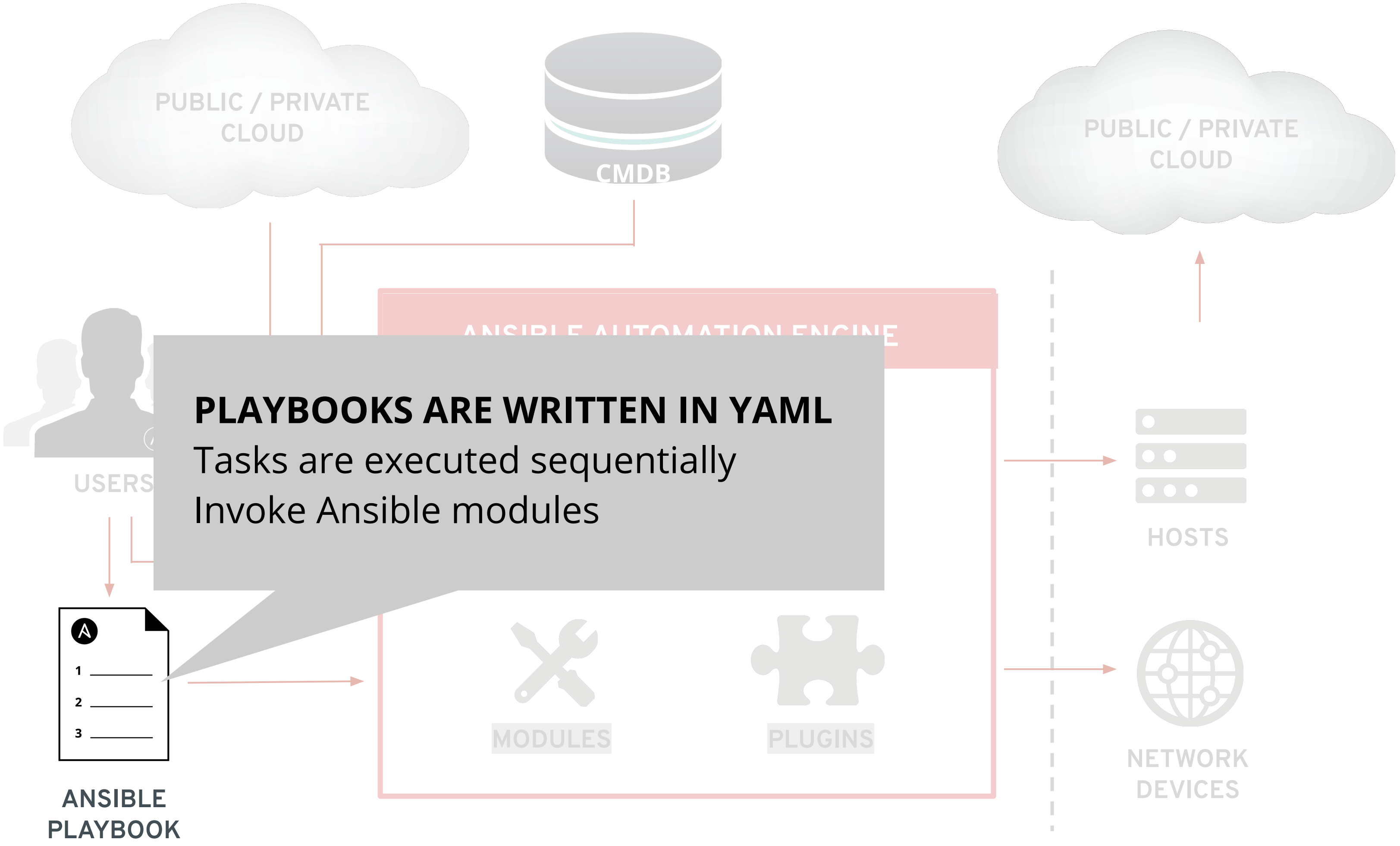
NETWORKING DEVICES

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

LINUX/WINDOWS HOSTS


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
INVENTORY
CLI
HOSTS
MODULES
PLUGINS
NETWORK DEVICES
ANSIBLE PLAYBOOK
1
2
3

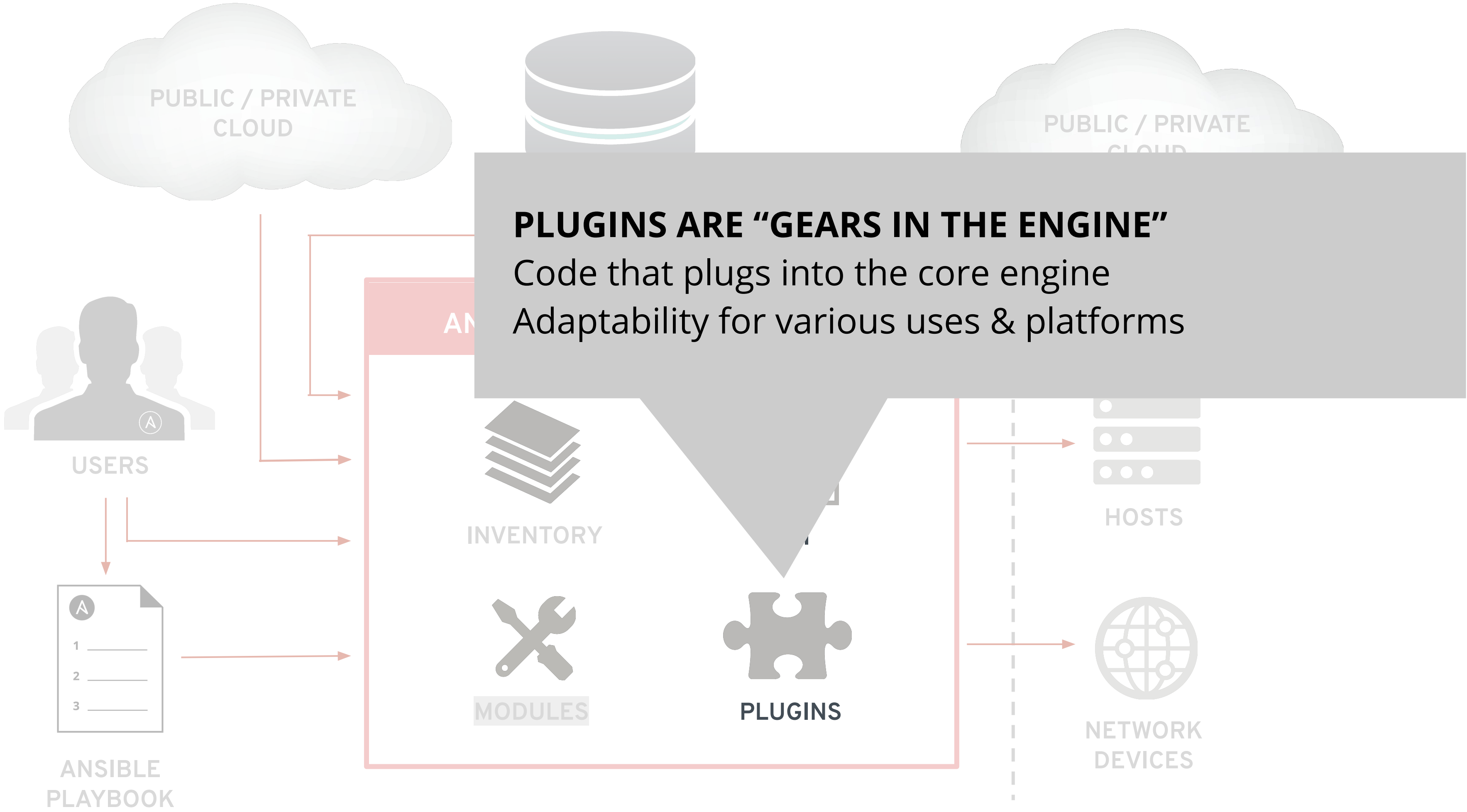
Red Hat



PLAYBOOKS ARE WRITTEN IN YAML

Tasks are executed sequentially

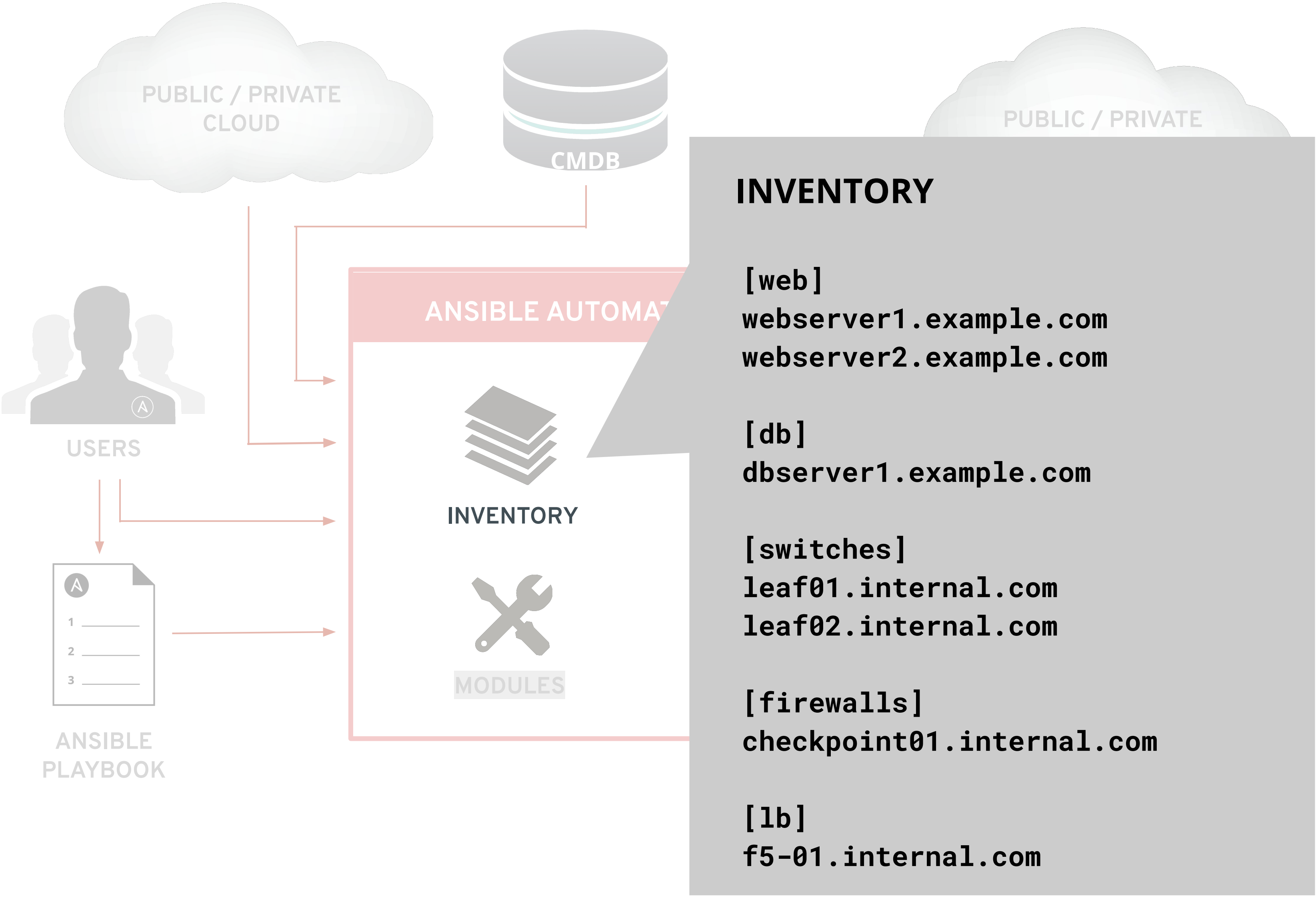
Invoke Ansible modules



PLUGINS ARE “GEARS IN THE ENGINE”

Code that plugs into the core engine

Adaptability for various uses & platforms



INVENTORY

```
[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

```
[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

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

Red Hat

Section 2

Topics Covered:

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

Red Hat
Ansible
Automation

An Ansible Playbook Example

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

  tasks:
    - name: ensure that the desired snmp strings are present
      ios_config:
        commands:
          - 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
- Disable **gather_facts** for network devices

```
---
- 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 that the desired 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_*

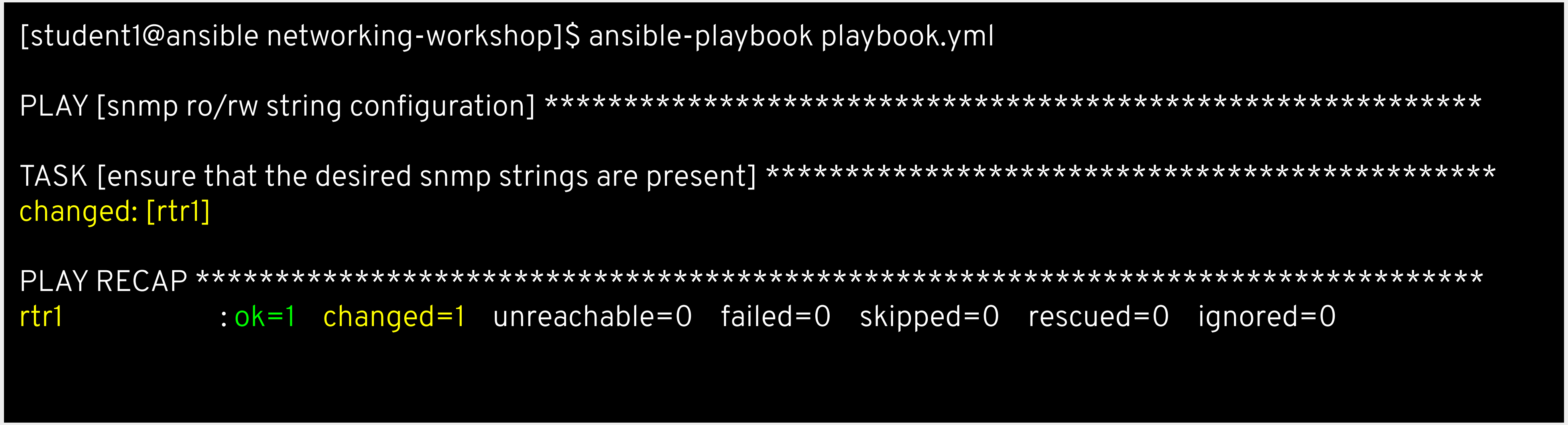
F5 BIG-IQ = bigiq_*

Juniper Junos = junos_*

VyOS = vyos_*

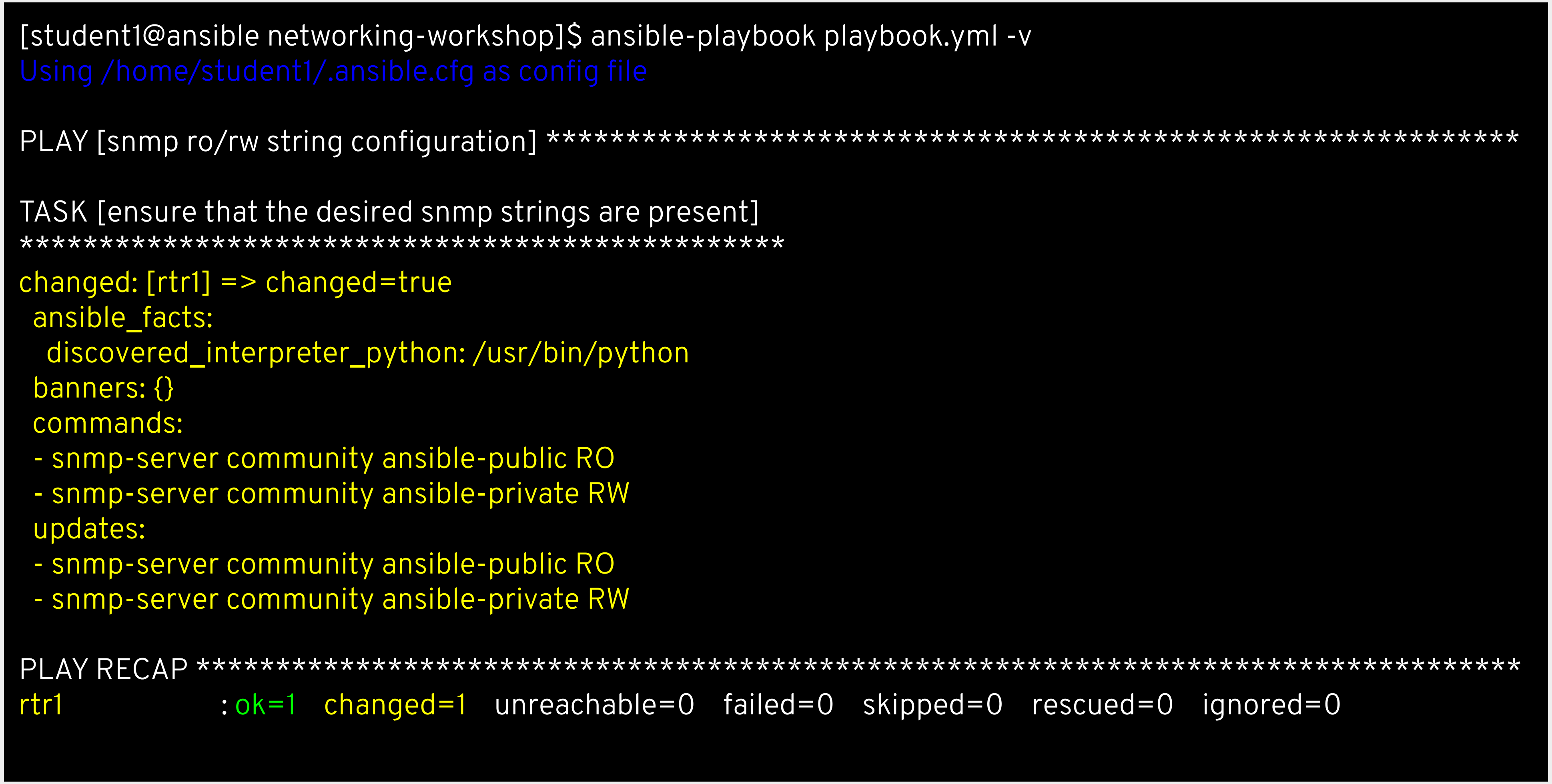
Running a playbook

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

 tasks:
   - name: ensure that the desired 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

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

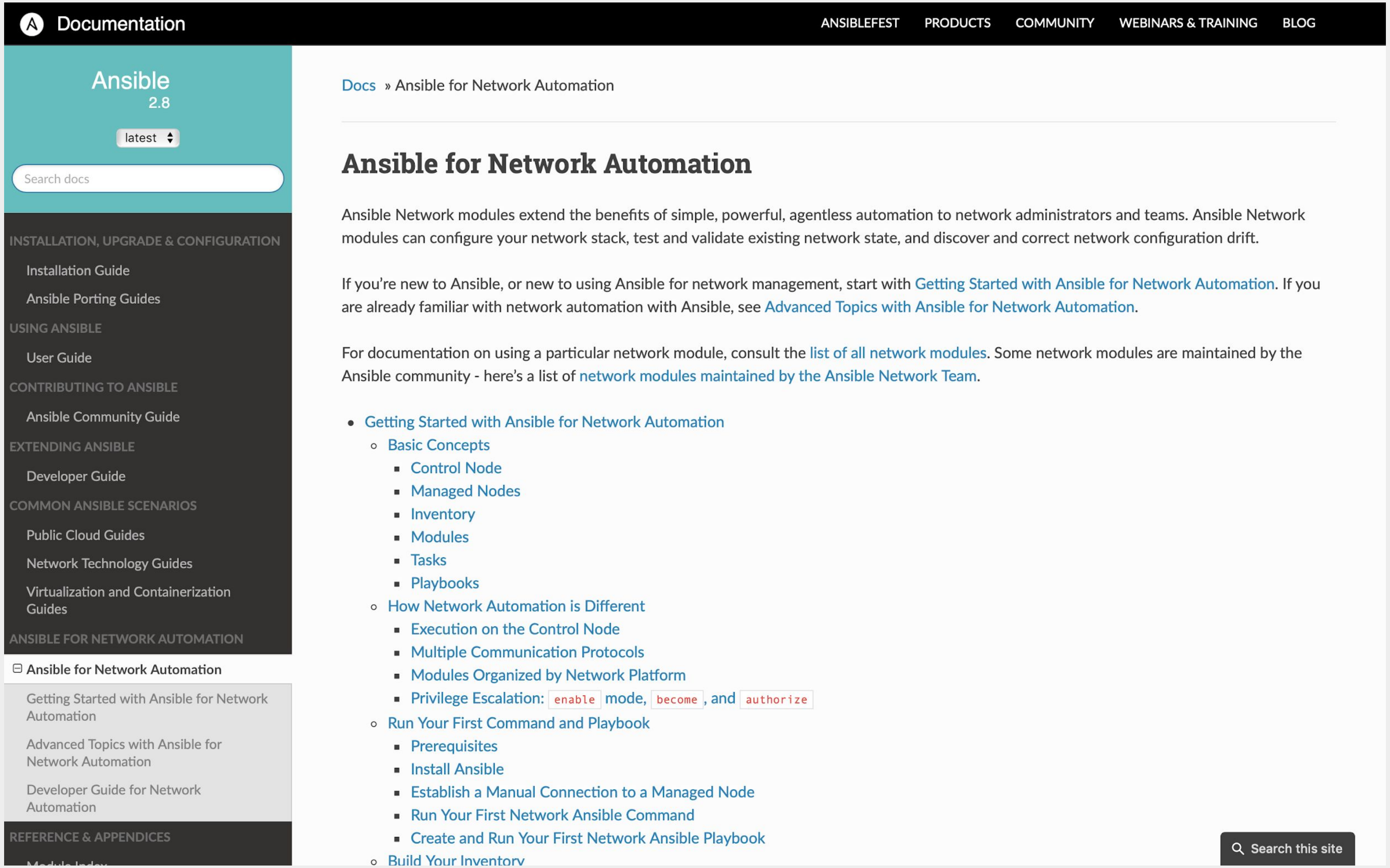
Section 3

Topics Covered:

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

Red Hat Ansible Automation

“Ansible for Network Automation” Documentation



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.

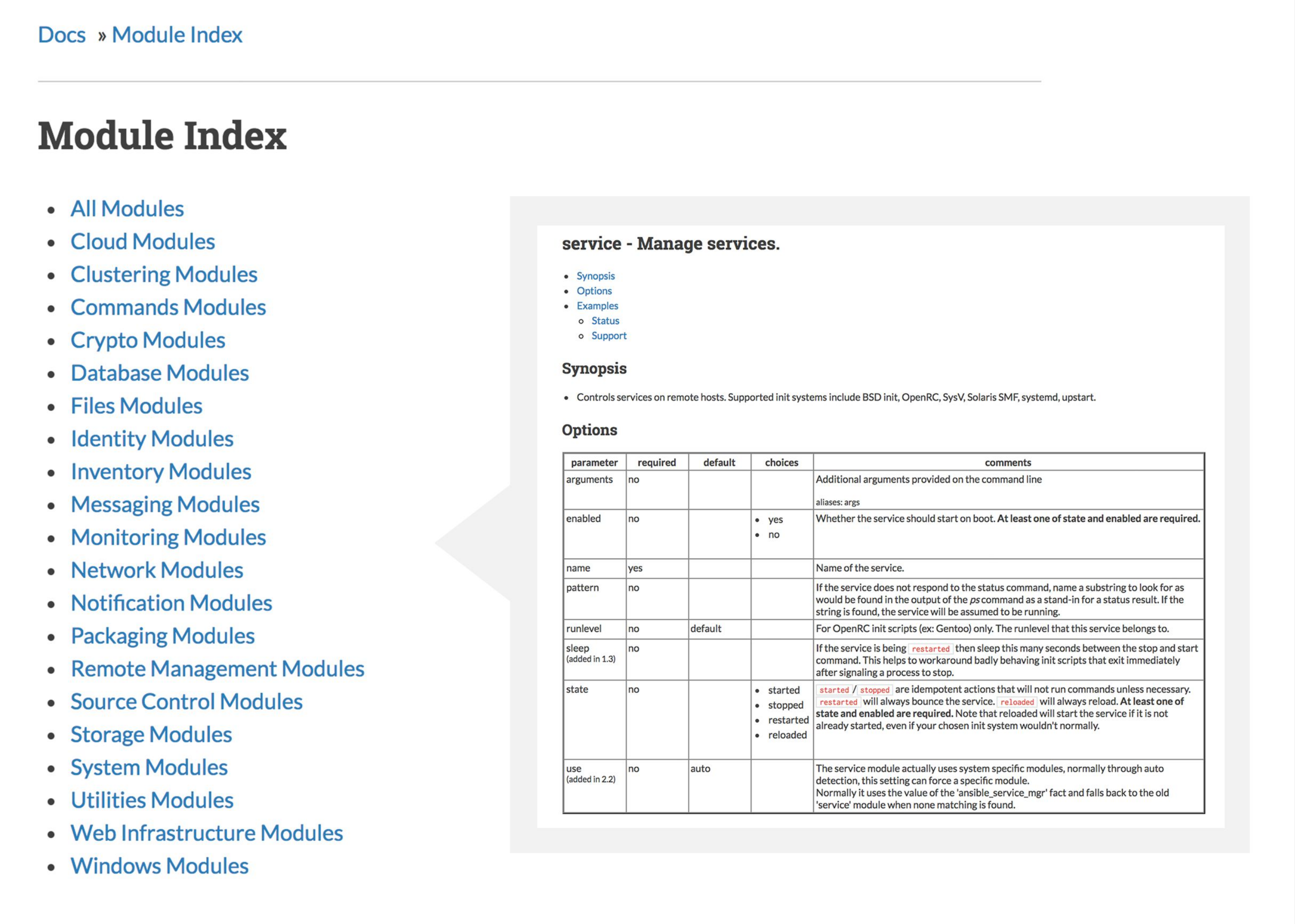
For documentation on using a particular network module, consult the list of all network modules. Some network modules are maintained by the Ansible community - here’s a list of network modules maintained by the Ansible Network Team.

- 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

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.

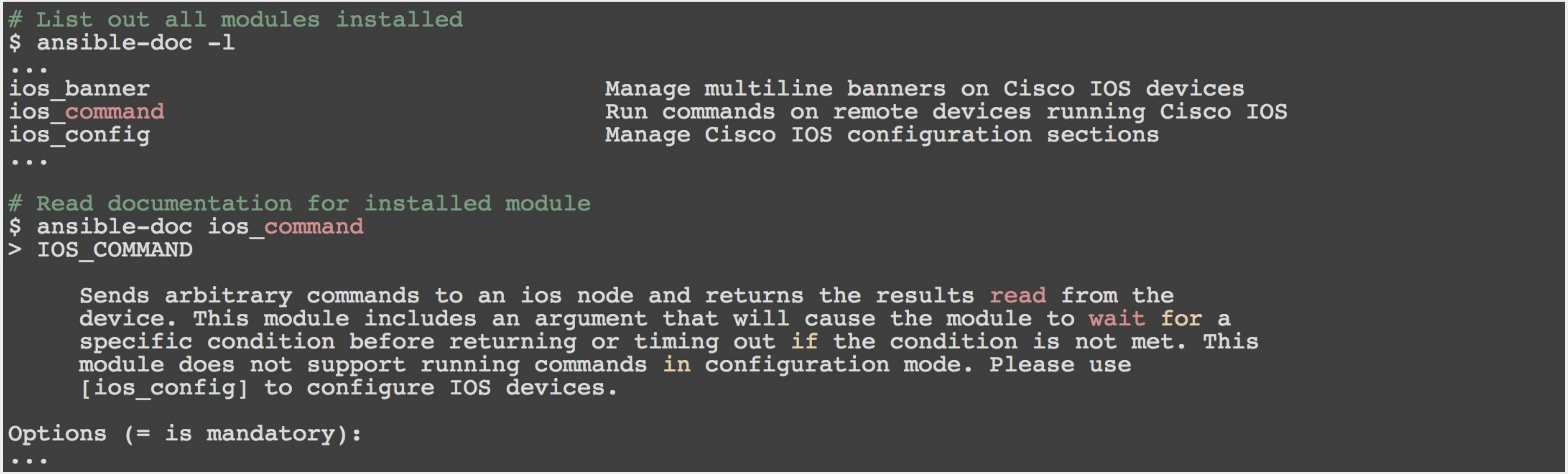
Options

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 `restarted` then sleep this many seconds between the stop and start command. This helps to workaround badly behaving init scripts that exit immediately after signaling a process to stop.
state	no		started stopped restarted reloaded	`started`/`stopped` are idempotent actions that will not run commands unless necessary. `restarted` will always bounce the service. `reloaded` 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 'ansible_service_mgr' fact and falls back to the old 'service' module when none matching is found.

https://docs.ansible.com/

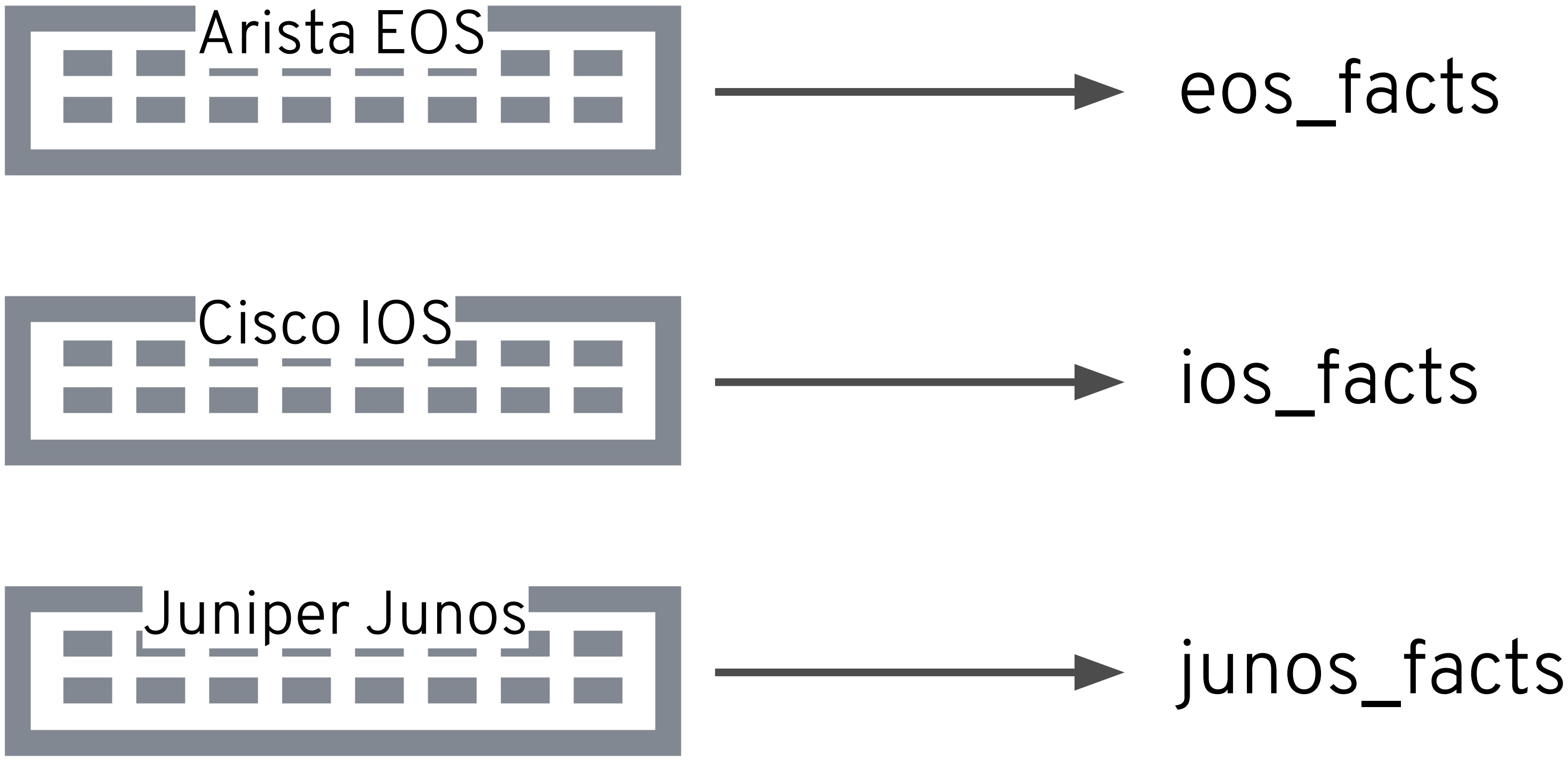
Module Documentation

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



Arista EOS → eos_facts

Cisco IOS → ios_facts

Juniper Junos → junos_facts

Fact modules return structured data

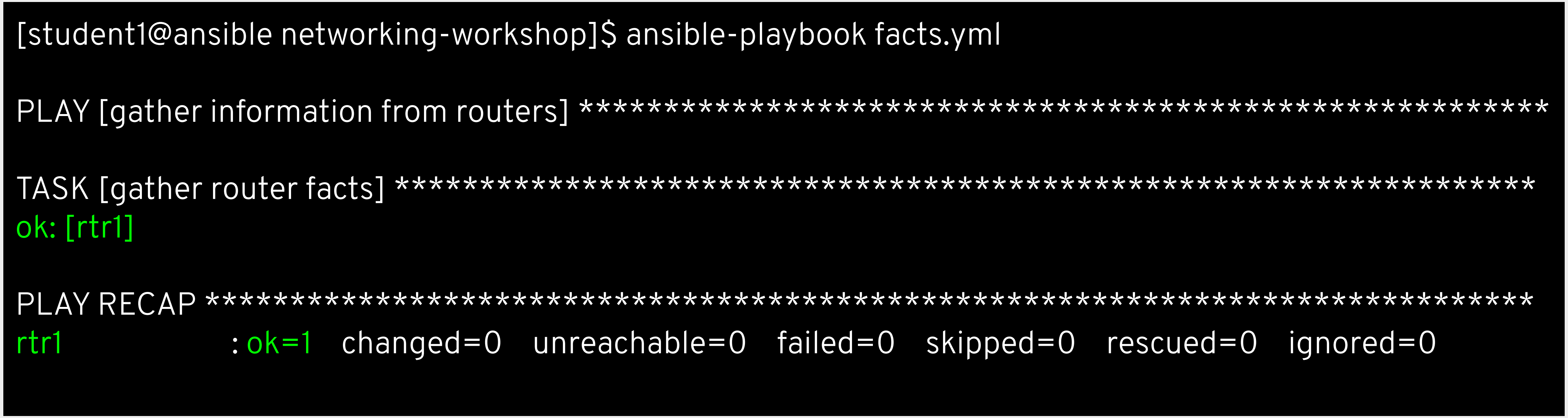
```
rtr1#show version
Cisco IOS XE Software, Version 16.09.02
Cisco IOS Software [Fuji], Virtual XE Software (X86_64_LINUX_IOSD-UNIVERSALK9-M), Version 16.9.2, RELEASE SOFTWARE (fc4)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
Compiled Mon 05-Nov-18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

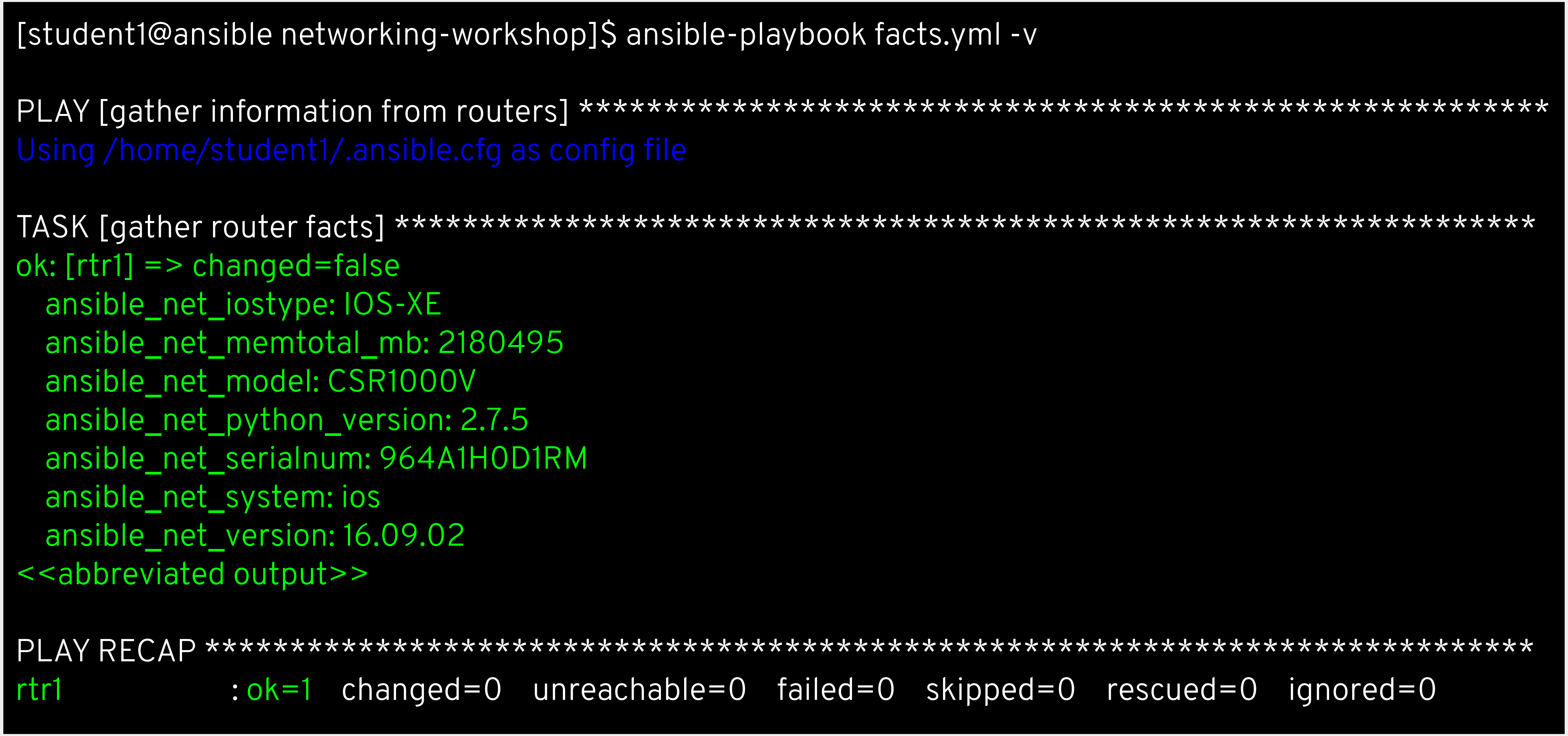
Running the Ansible Playbook



```
[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



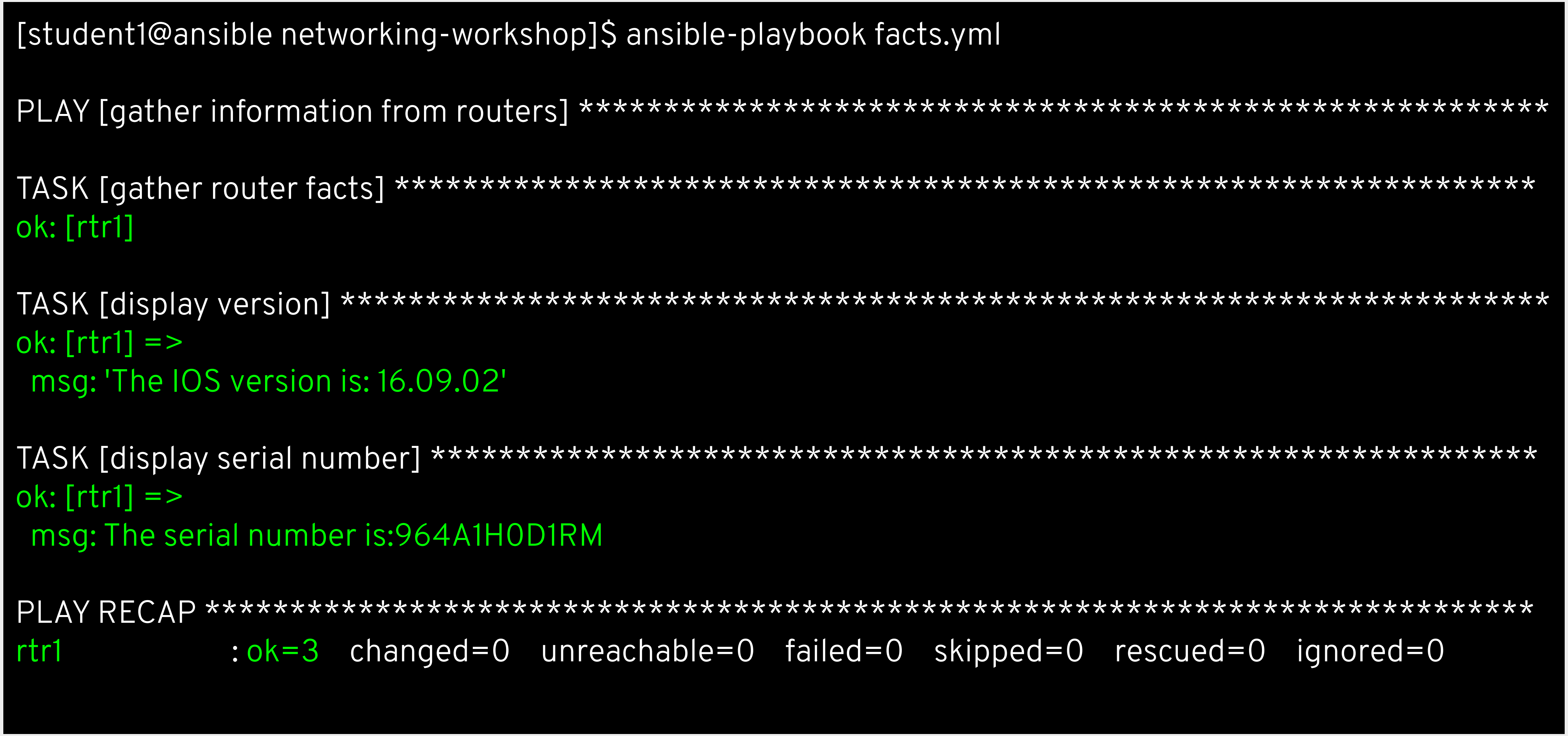
```
[student1@ansible networking-workshop]$ ansible-playbook facts.yml -v

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

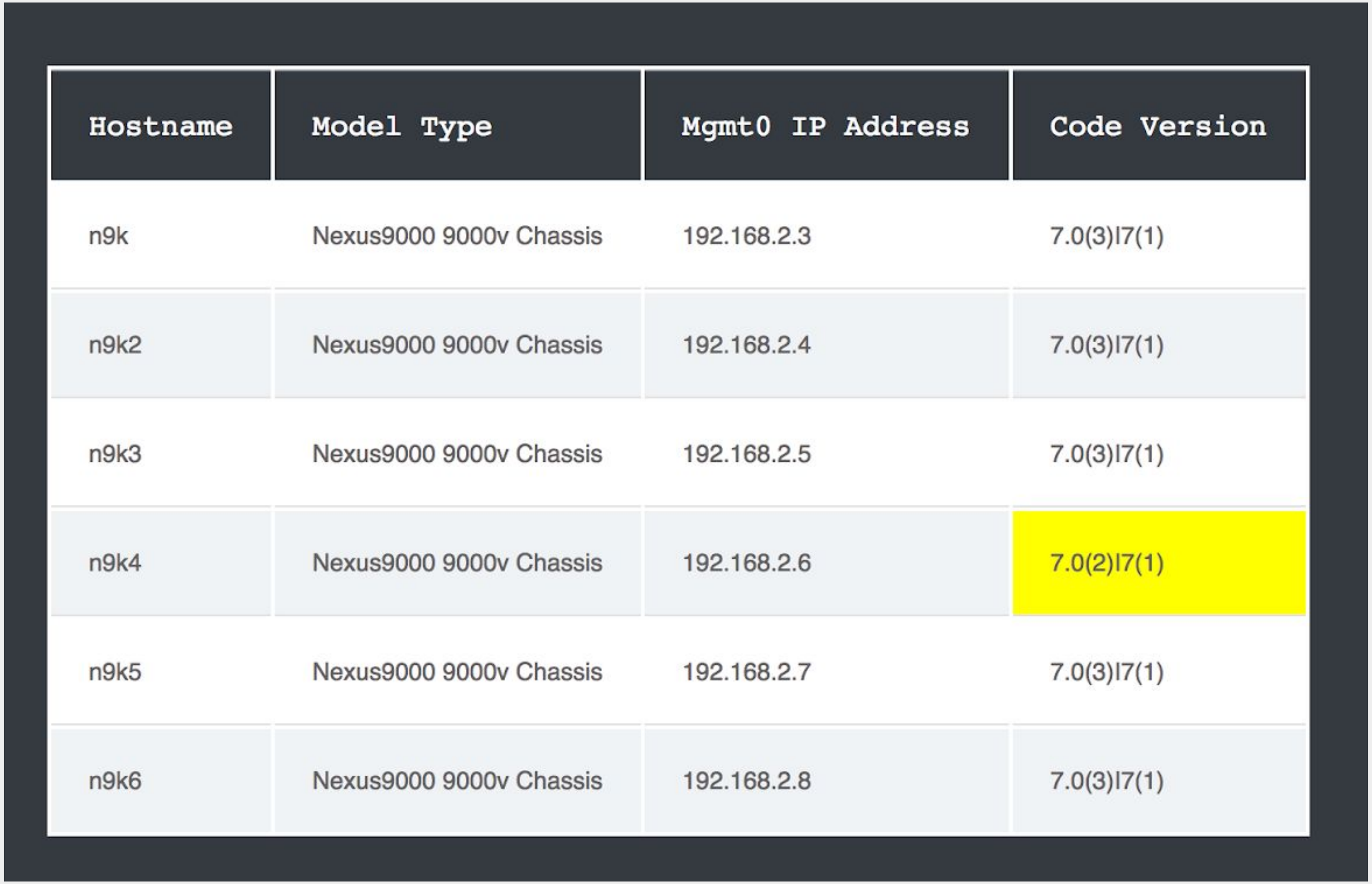
```
- name: display version
  debug:
    msg: "The IOS version is: {{ ansible_net_version }}"

- name: display serial number
  debug:
    msg: "The serial number is:{{ ansible_net_serialnum }}"
```

Running the Ansible Playbook with verbosity

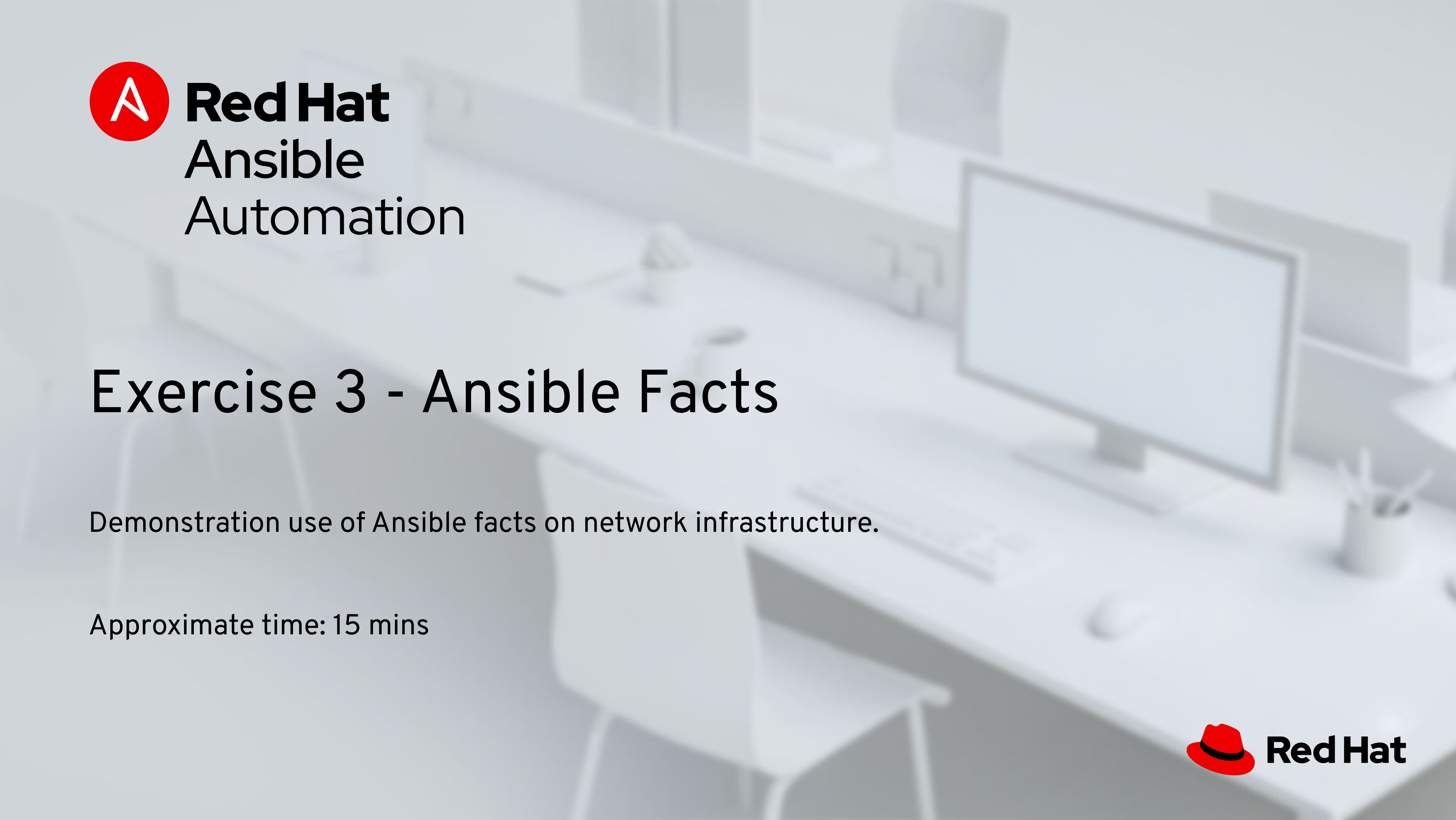


```
[student1@ansible networking-workshop]$ ansible-playbook facts.yml

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

Exercise 3 - Ansible Facts

Demonstration use of Ansible facts on network infrastructure.

Approximate time: 15 mins

Section 4

Topics Covered:

- Understand group variables
- Understand Jinja2
- cli_config module

Red Hat
Ansible
Automation

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 section 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:

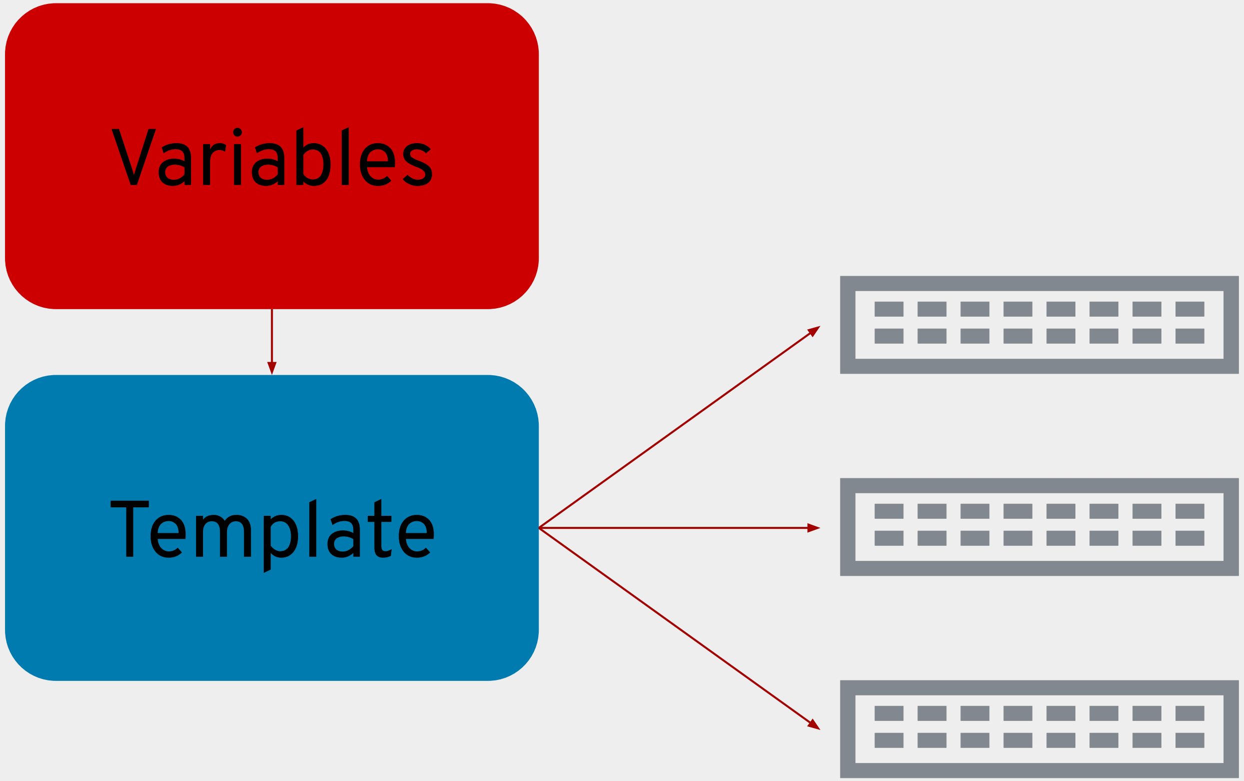
iosxr_config:

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') }}"
```

Exercise 4 - Network Configuration with Jinja Templates

Demonstration templating a network configuration and pushing it a device

Approximate time: 15 mins

Section 5

Topics Covered:

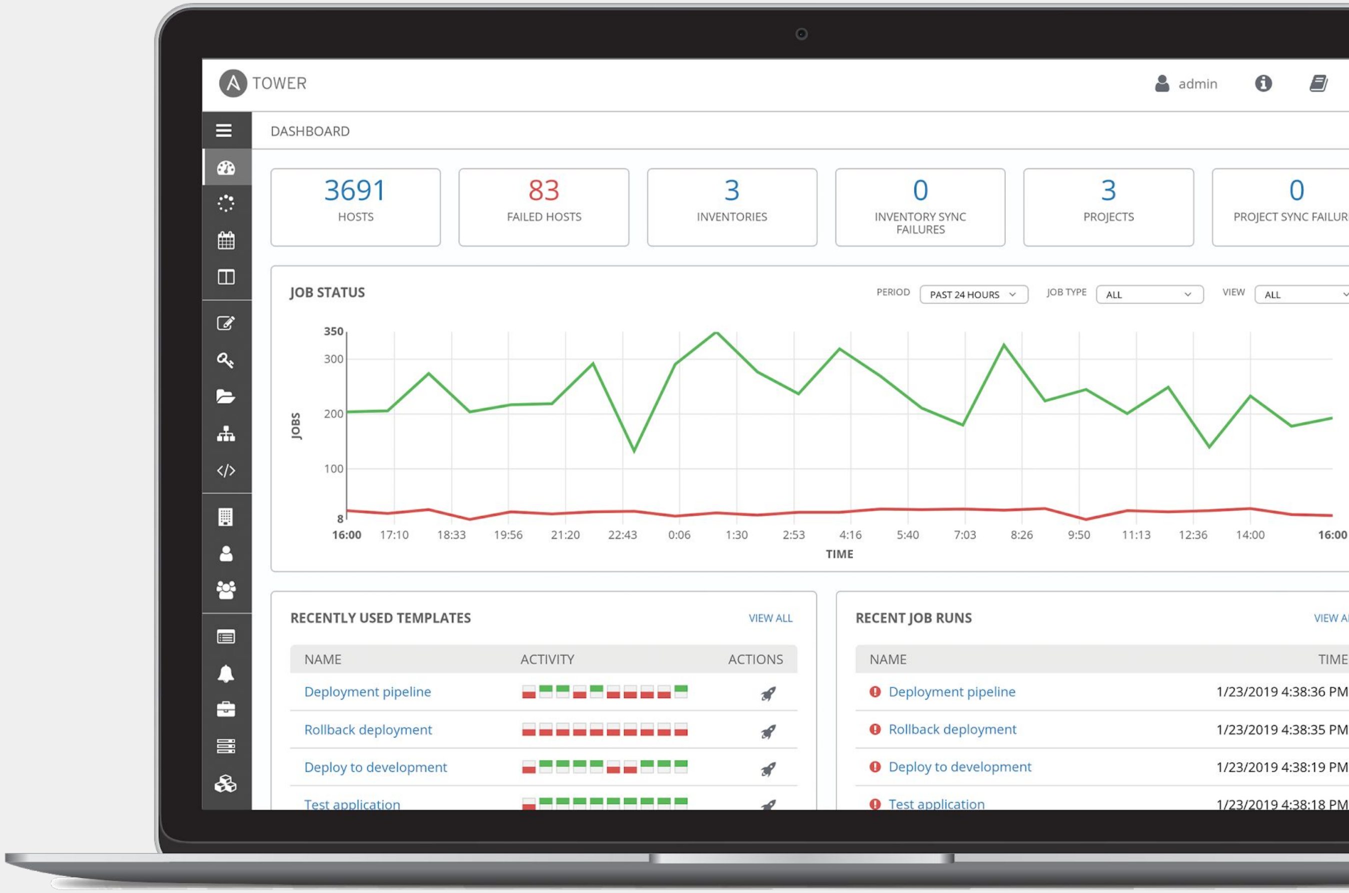
- What is Ansible Tower?
- Job Templates
  - Inventory
  - Credentials
  - Projects

Red Hat
Ansible
Tower

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



3691	83	3	0	3	0
HOSTS	FAILED HOSTS	INVENTORIES	INVENTORY SYNC FAILURES	PROJECTS	PROJECT SYNC FAILUR

RECENTLY USED TEMPLATES

NAME	ACTIVITY	ACTIONS
Deployment pipeline	🟥🟩🟩🟥🟩🟥🟥🟥🟥🟩	🚀
Rollback deployment	🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥	🚀
Deploy to development	🟥🟩🟩🟩🟩🟥🟥🟩🟩🟩	🚀
Test application	🟥🟩🟩🟩🟩🟩🟩🟩🟩🟩	🚀

RECENT JOB RUNS

NAME	TIME
Deployment pipeline	1/23/2019 4:38:36 PM
Rollback deployment	1/23/2019 4:38:35 PM
Deploy to development	1/23/2019 4:38:19 PM
Test application	1/23/2019 4:38:18 PM

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.

Workflows

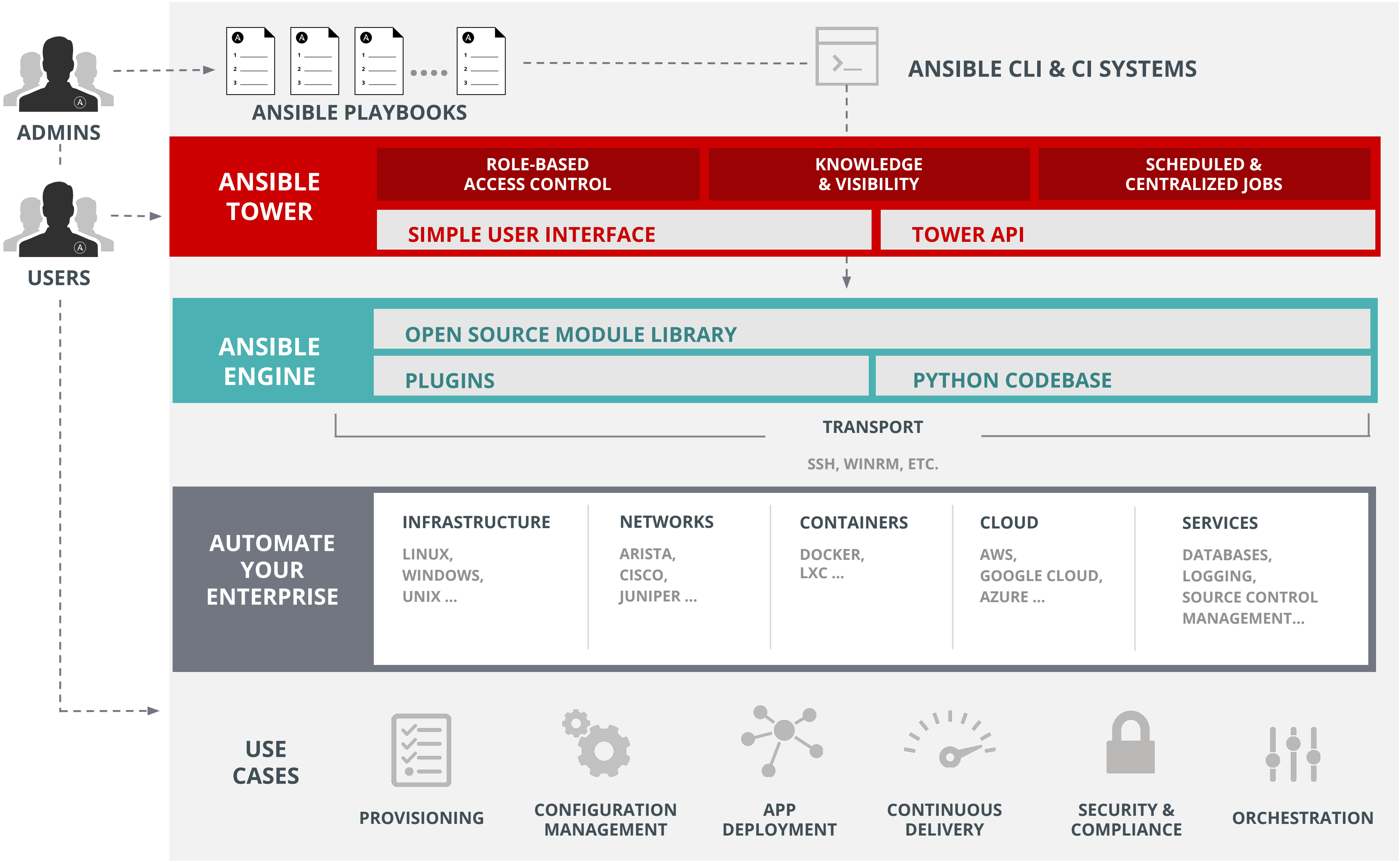
Ansible Tower's multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.

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.



ADMINS

USERS

ANSIBLE PLAYBOOKS

ANSIBLE CLI & CI SYSTEMS

ANSIBLE TOWER

ROLE-BASED ACCESS CONTROL

KNOWLEDGE & VISIBILITY

SCHEDULED & CENTRALIZED JOBS

SIMPLE USER INTERFACE

TOWER API

ANSIBLE ENGINE

OPEN SOURCE MODULE LIBRARY

PLUGINS

PYTHON CODEBASE

TRANSPORT

SSH, WINRM, ETC.

AUTOMATE YOUR ENTERPRISE

INFRASTRUCTURE
LINUX,
WINDOWS,
UNIX ...

NETWORKS
ARISTA,
CISCO,
JUNIPER ...

CONTAINERS
DOCKER,
LXC ...

CLOUD
AWS,
GOOGLE CLOUD,
AZURE ...

SERVICES
DATABASES,
LOGGING,
SOURCE CONTROL
MANAGEMENT...

USE CASES

PROVISIONING

CONFIGURATION MANAGEMENT

APP DEPLOYMENT

CONTINUOUS DELIVERY

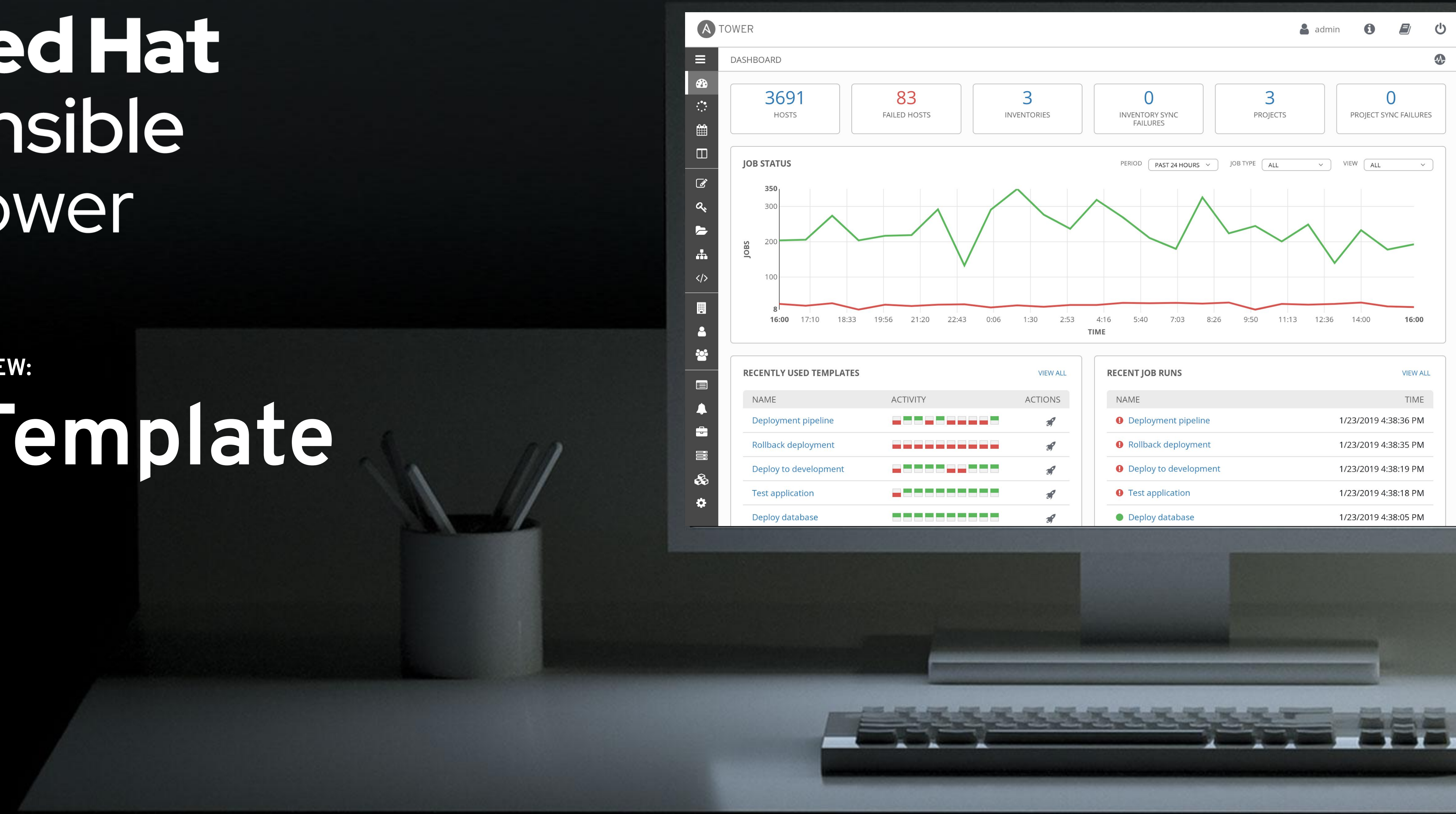
SECURITY & COMPLIANCE

ORCHESTRATION

Red Hat Ansible Tower

FEATURE OVERVIEW:

Job Template



3691	83	3	0	3	0
HOSTS	FAILED HOSTS	INVENTORIES	INVENTORY SYNC FAILURES	PROJECTS	PROJECT SYNC FAILURES

JOB STATUS

PERIOD: PAST 24 HOURS | JOB TYPE: ALL | VIEW: ALL

RECENTLY USED TEMPLATES — VIEW ALL

NAME	ACTIVITY	ACTIONS
Deployment pipeline	🟥🟩🟩🟥🟩🟥🟥🟥🟥🟩	
Rollback deployment	🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥	
Deploy to development	🟥🟩🟩🟩🟩🟥🟥🟩🟩🟩	
Test application	🟥🟩🟩🟩🟩🟩🟩🟩🟩🟩	
Deploy database	🟩🟩🟩🟩🟩🟩🟩🟩🟩🟩	

RECENT JOB RUNS — VIEW ALL

NAME	TIME
Deployment pipeline	1/23/2019 4:38:36 PM
Rollback deployment	1/23/2019 4:38:35 PM
Deploy to development	1/23/2019 4:38:19 PM
Test application	1/23/2019 4:38:18 PM
Deploy database	1/23/2019 4:38:05 PM

Red Hat

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.

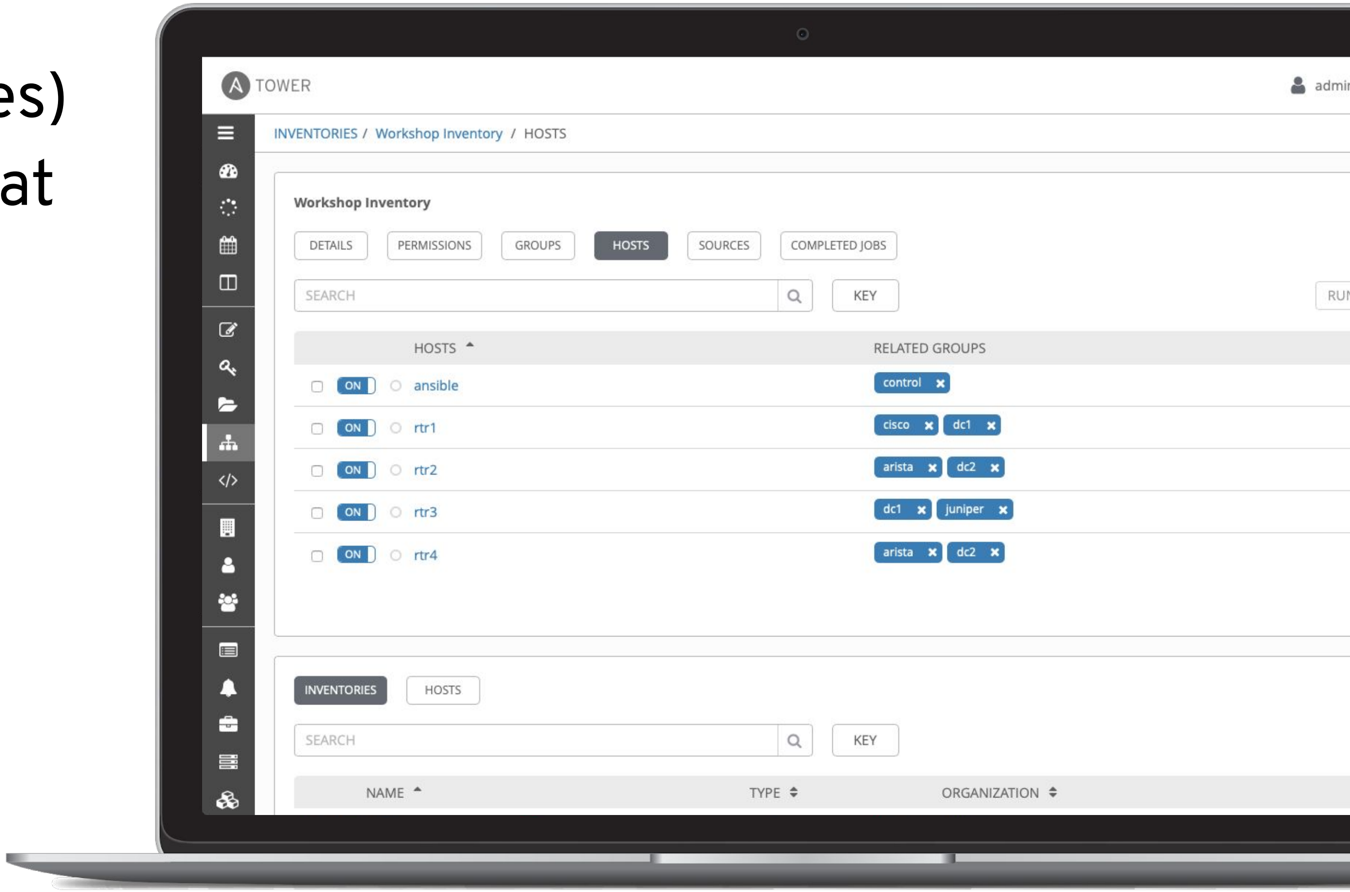
A **Job Template** requires:

- 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



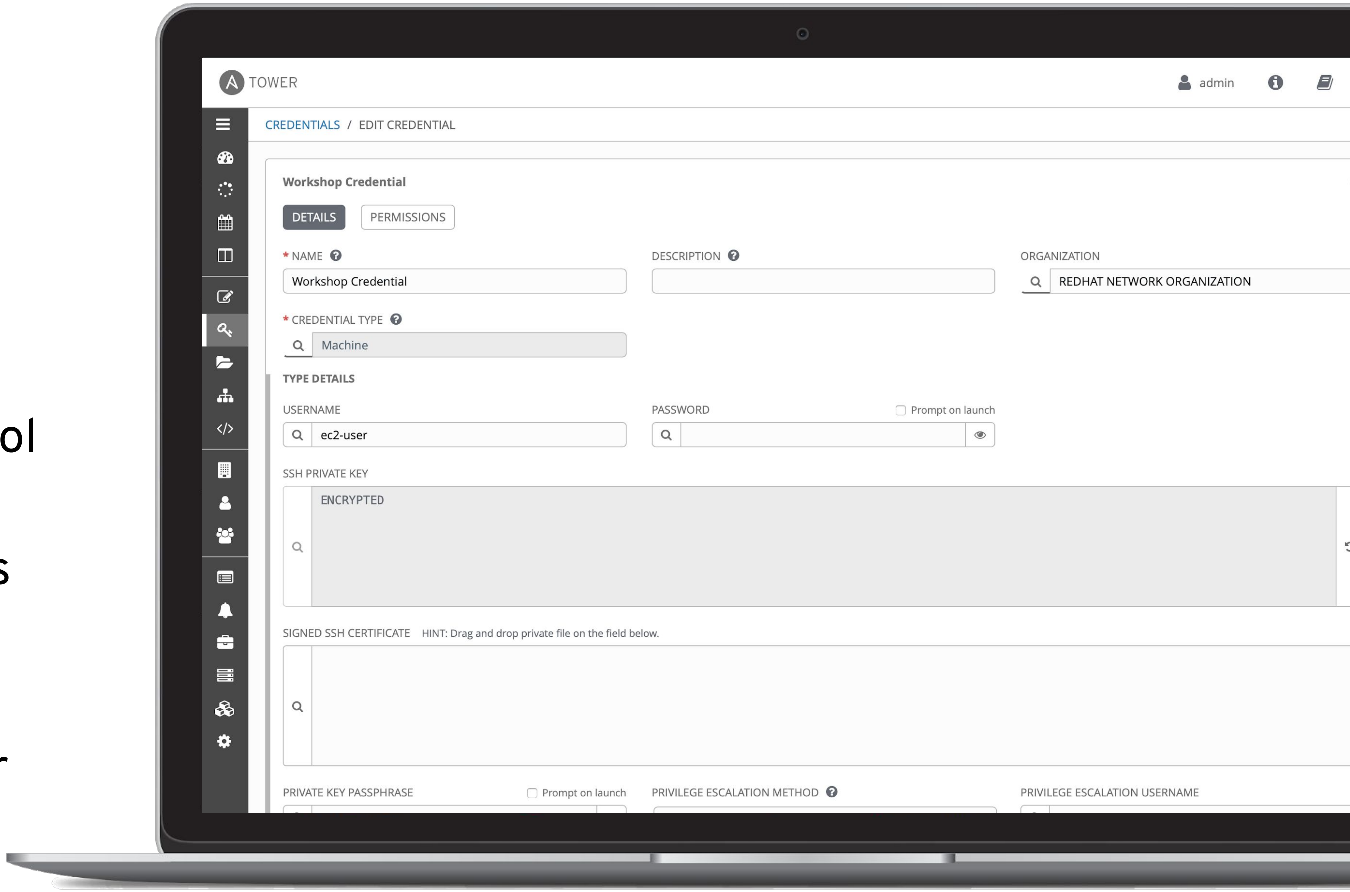
HOSTS	RELATED GROUPS
ON ansible	control
ON rtr1	cisco dc1
ON rtr2	arista dc2
ON rtr3	dc1 juniper
ON rtr4	arista dc2

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.



CREDENTIALS / EDIT CREDENTIAL

Workshop Credential

* NAME: Workshop Credential

DESCRIPTION:

ORGANIZATION: REDHAT NETWORK ORGANIZATION

* CREDENTIAL TYPE: Machine

TYPE DETAILS

USERNAME: ec2-user

PASSWORD: □ Prompt on launch

SSH PRIVATE KEY: ENCRYPTED

SIGNED SSH CERTIFICATE HINT: Drag and drop private file on the field below.

PRIVATE KEY PASSPHRASE □ Prompt on launch

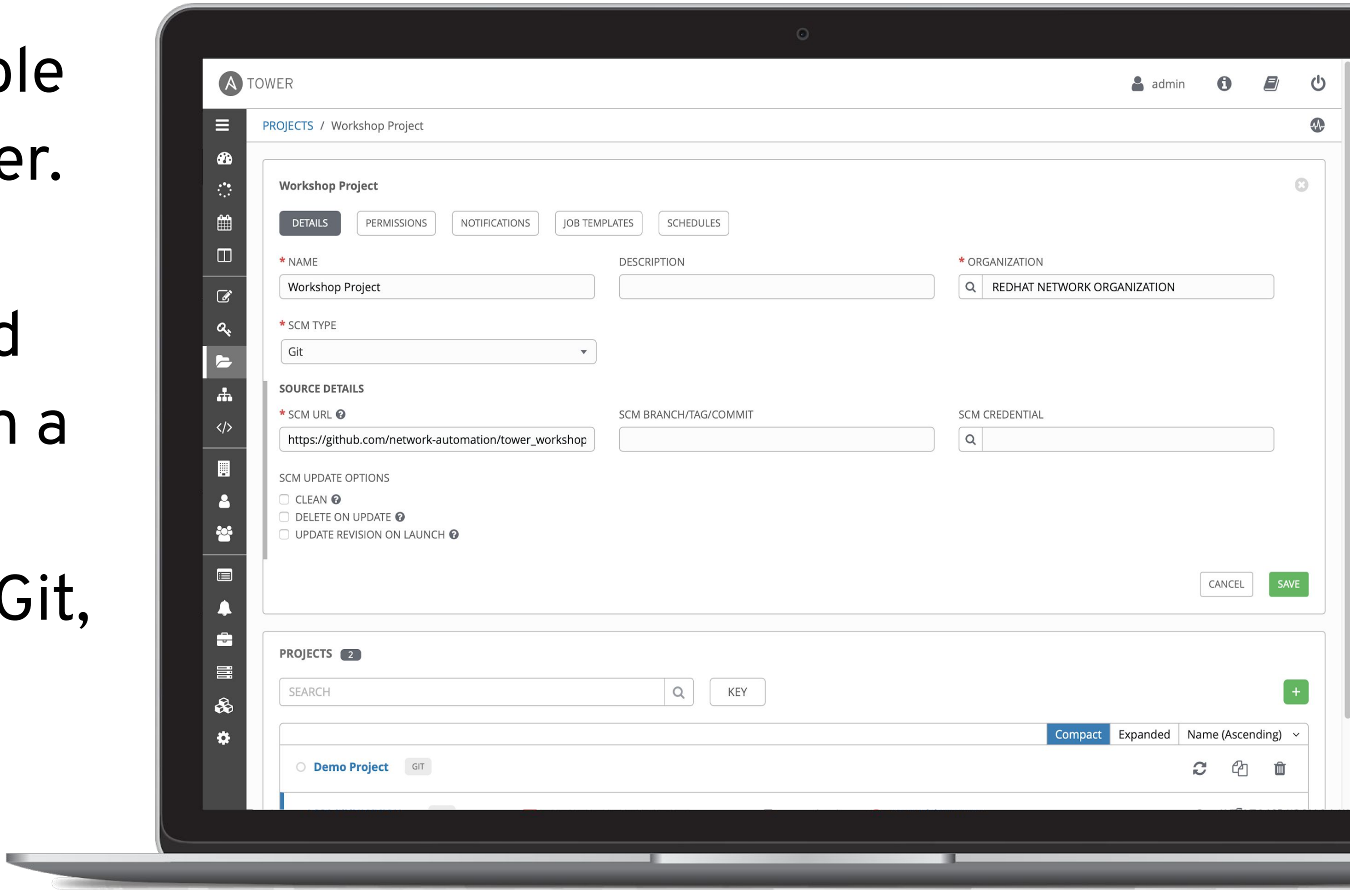
PRIVILEGE ESCALATION METHOD

PRIVILEGE ESCALATION USERNAME

Projects

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.



Workshop Project

* NAME: Workshop Project

DESCRIPTION:

* ORGANIZATION: REDHAT NETWORK ORGANIZATION

* SCM TYPE: Git

SOURCE DETAILS

* SCM URL: https://github.com/network-automation/tower_workshop

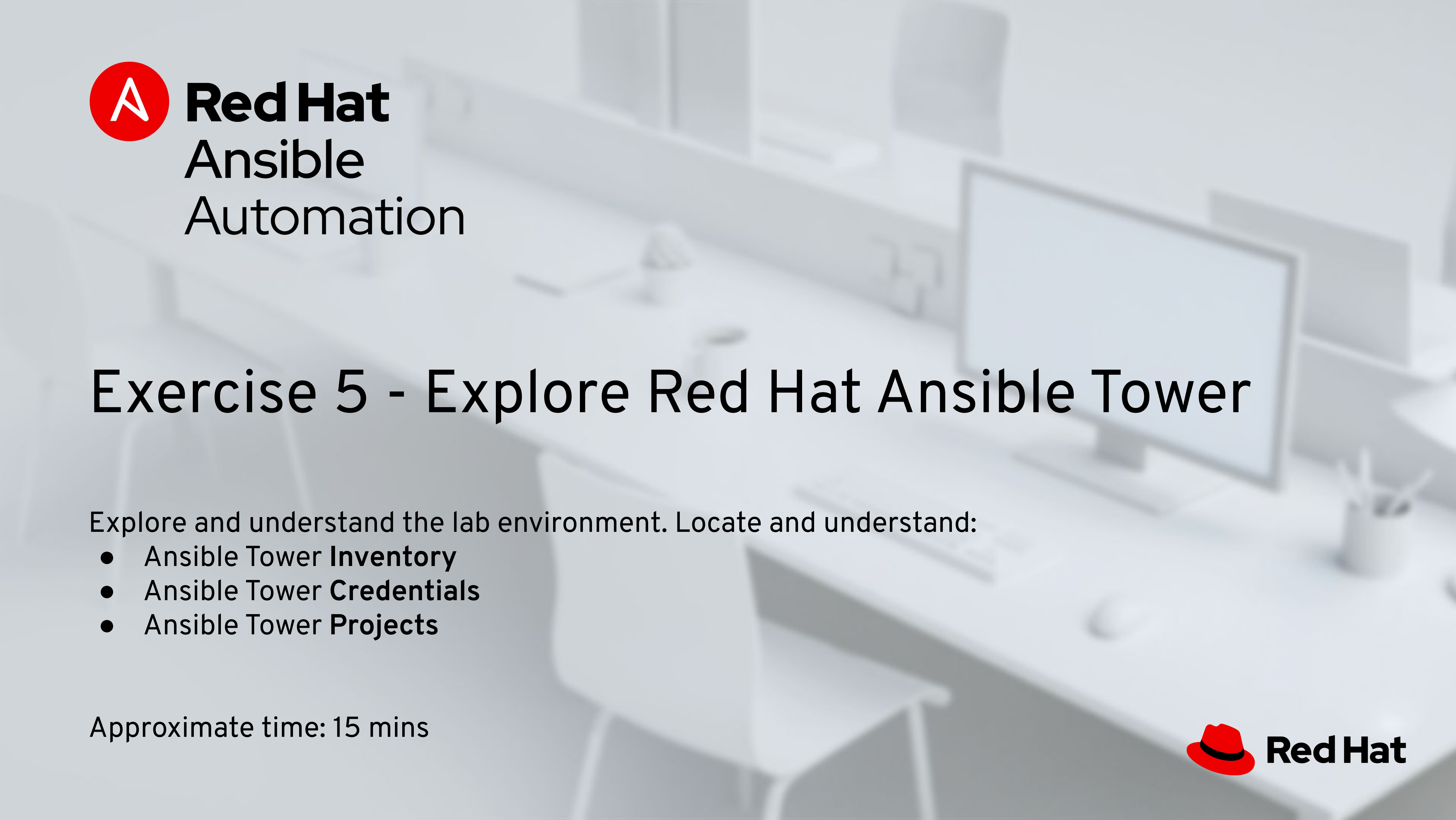
SCM BRANCH/TAG/COMMIT:

SCM CREDENTIAL:

SCM UPDATE OPTIONS

□ CLEAN
□ DELETE ON UPDATE
□ UPDATE REVISION ON LAUNCH

○ Demo Project GIT



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

Section 6

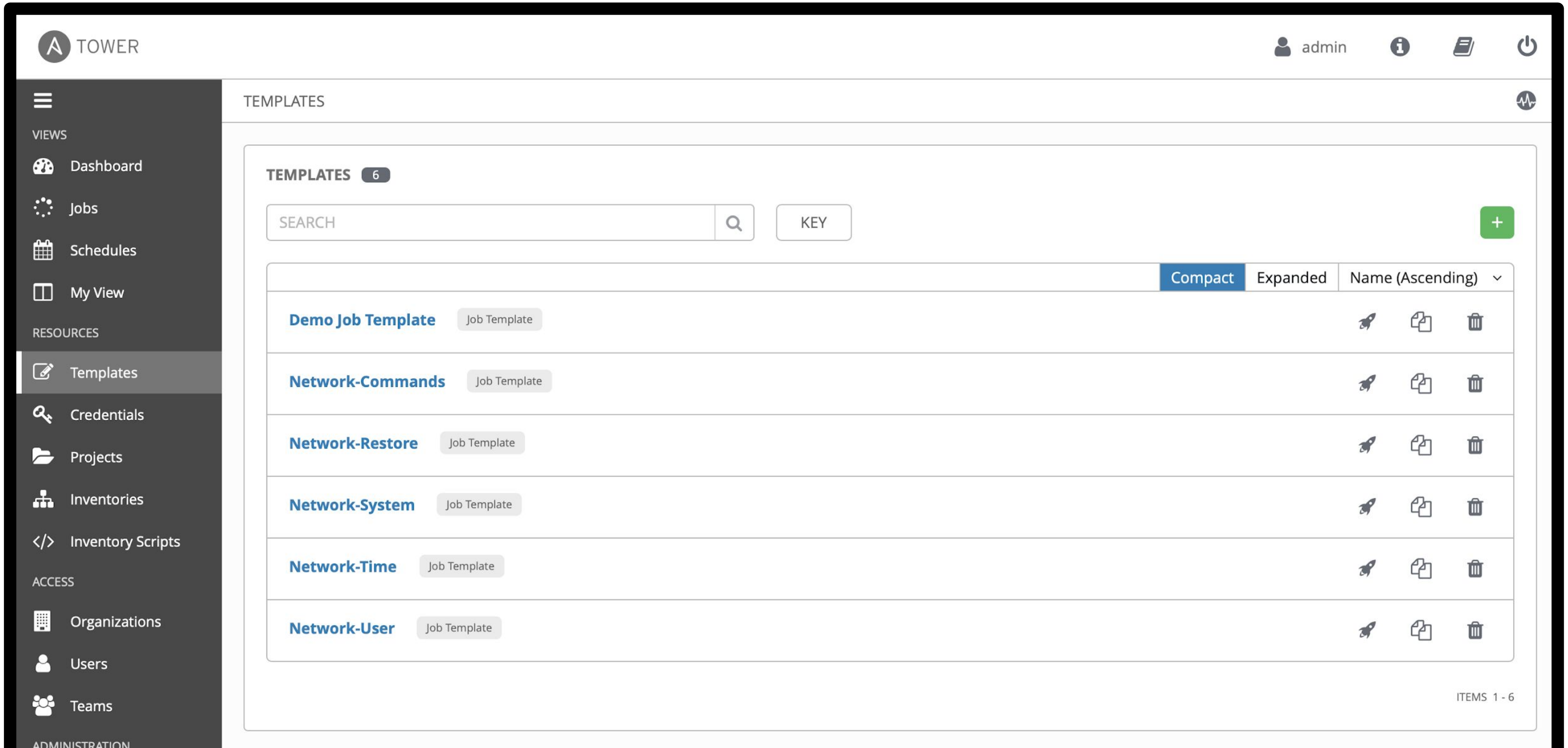
Topics Covered:

- Building a Job Template
- Executing a Job Template

Red Hat Ansible Tower

Expanding on Job Templates

Job Templates can be found and created by clicking the **Templates** button under the *RESOURCES* section on the left menu.



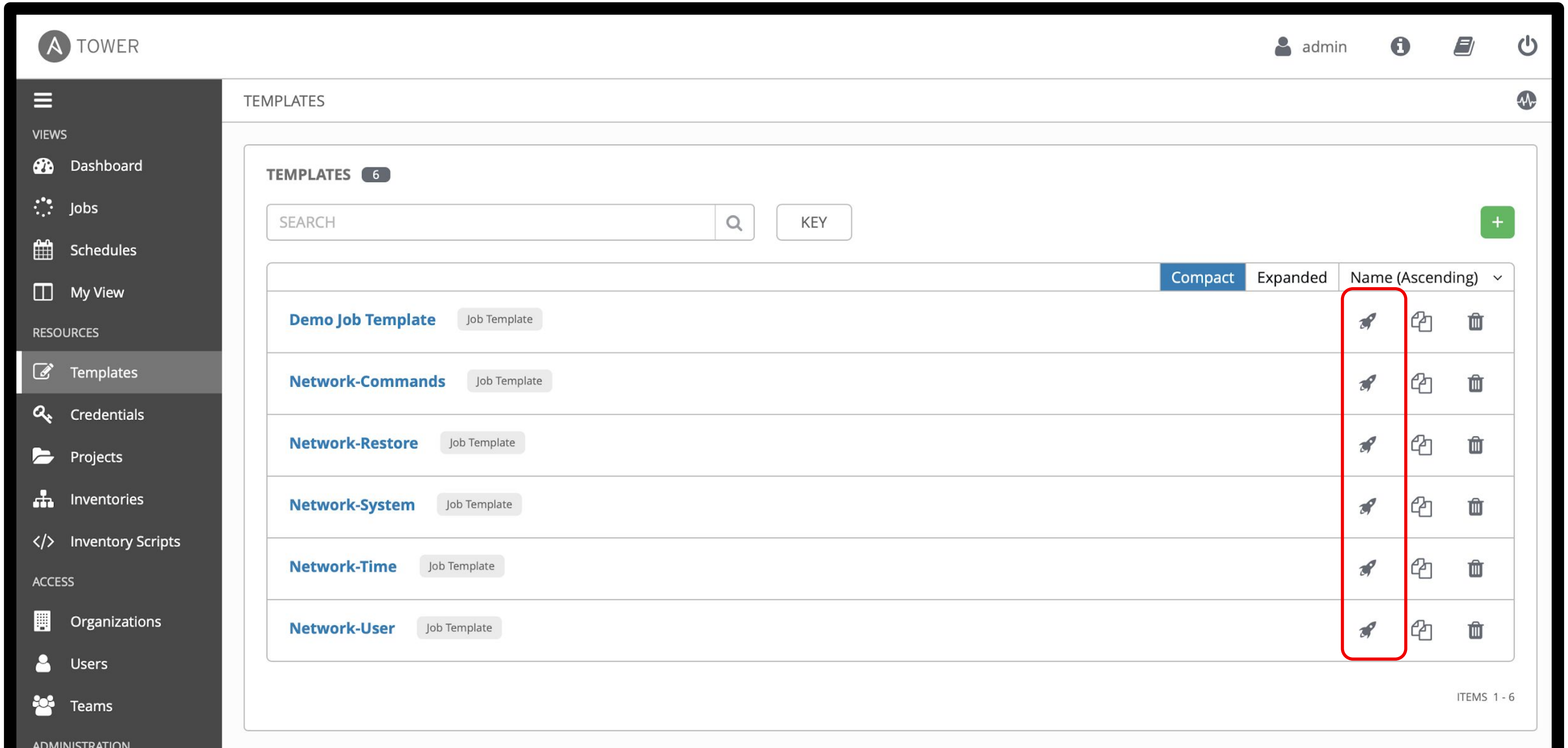
TEMPLATES 6

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

Executing an existing Job Template

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



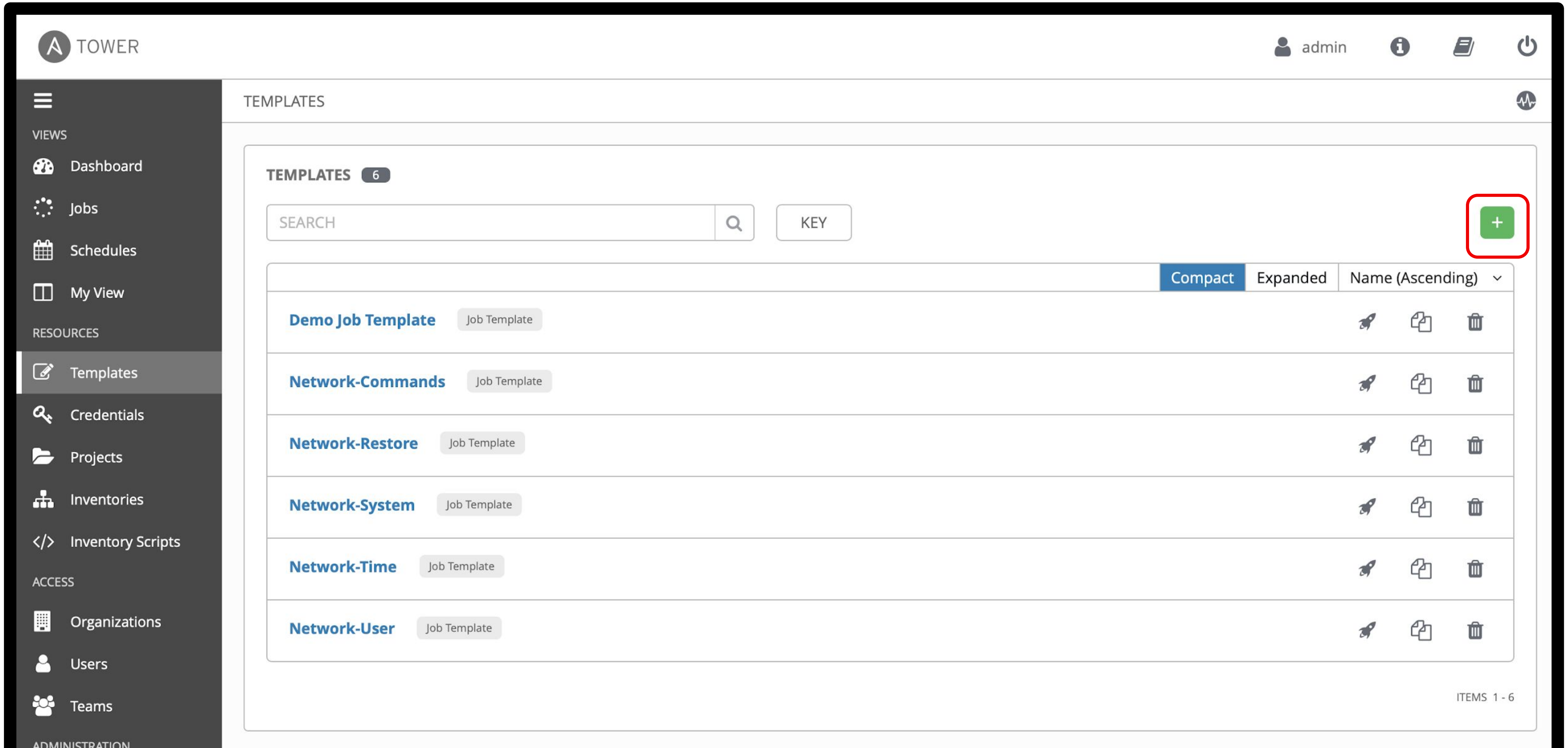
TEMPLATES 6

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** +



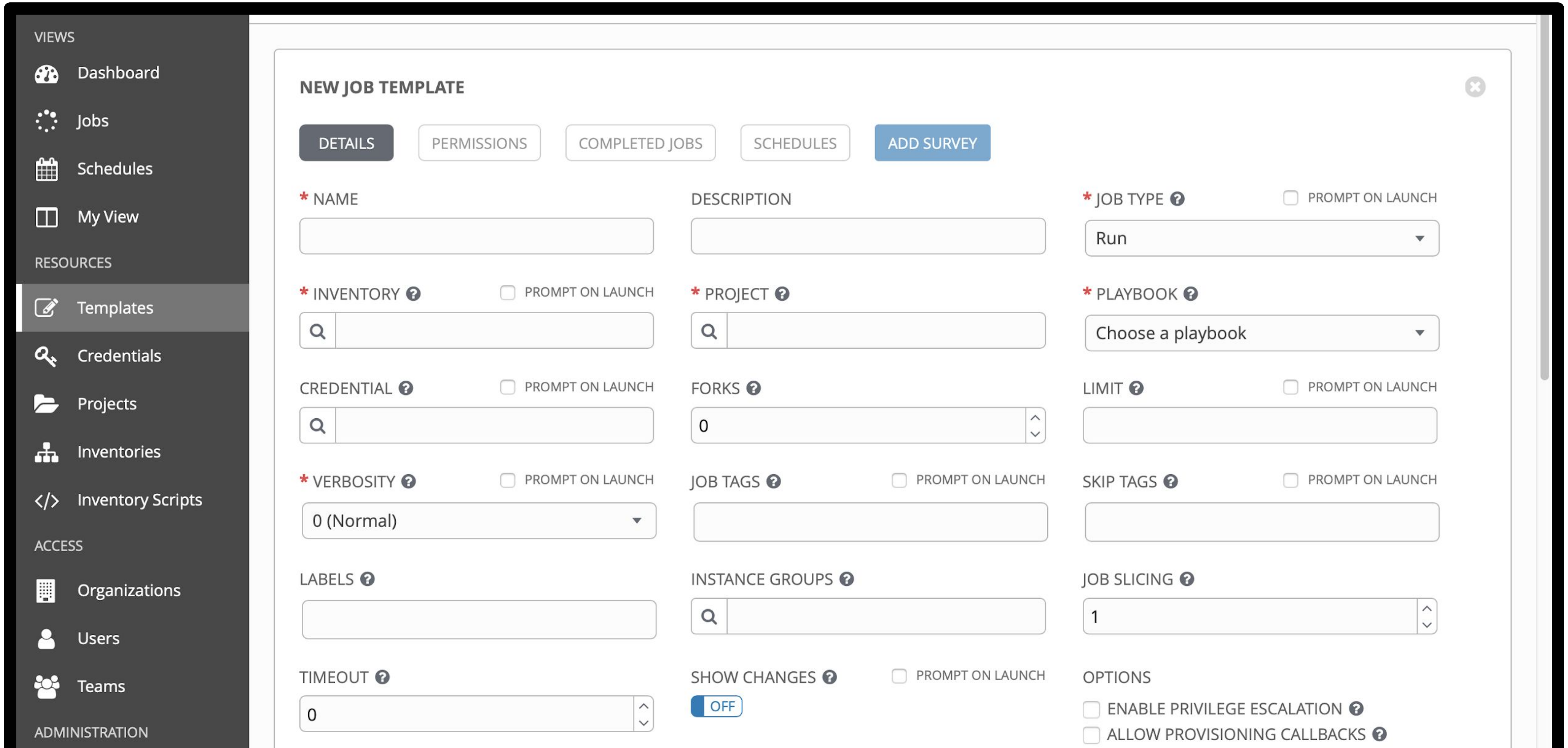
TEMPLATES 6

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

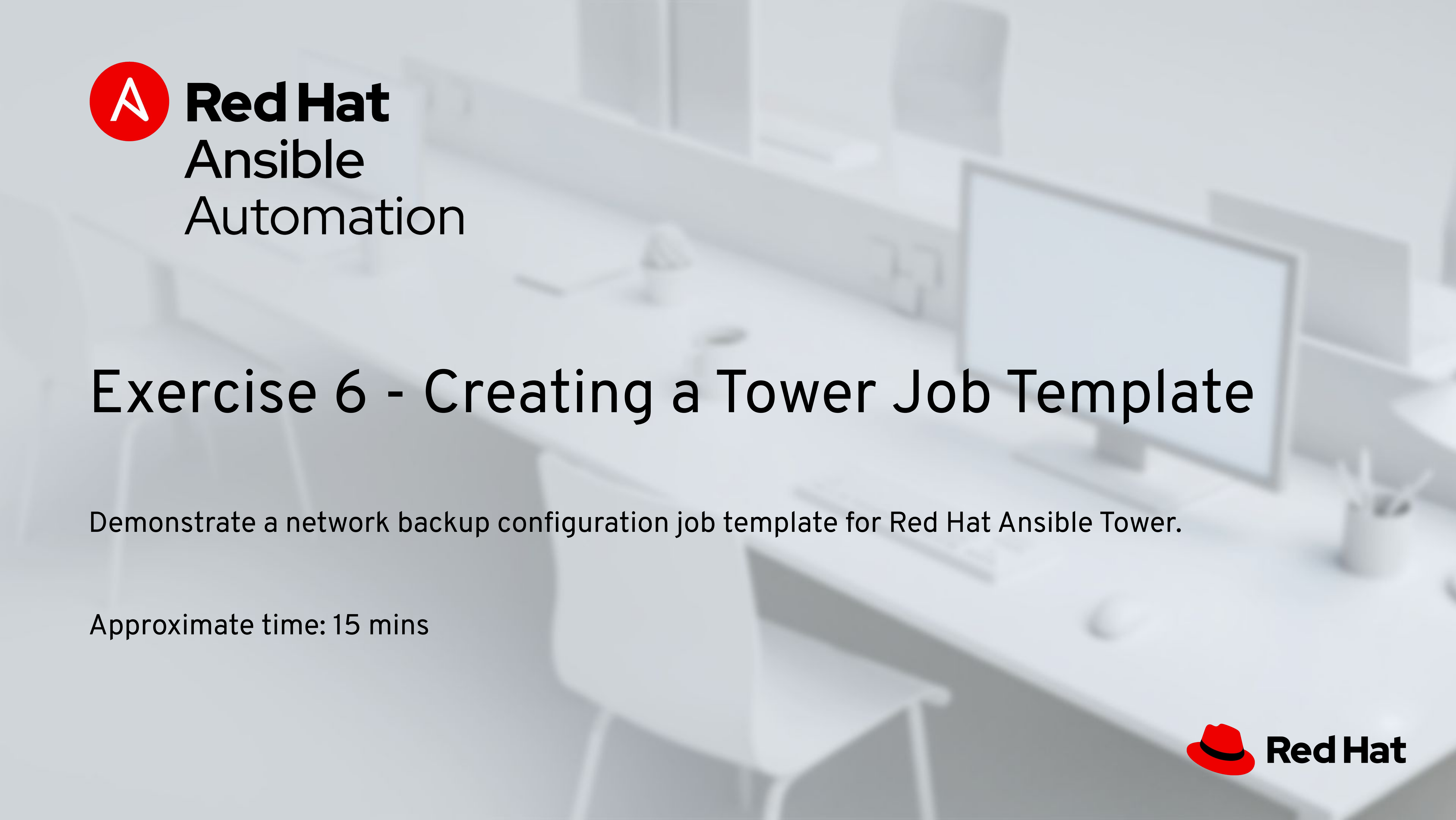
Creating a new Job Template (2/2)

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

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



Red Hat
Ansible
Automation

Exercise 6 - Creating a Tower Job Template

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

Approximate time: 15 mins

Red Hat

Section 7

Red Hat Ansible Tower

Topics Covered:

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

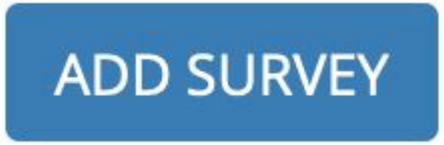
Surveys

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.

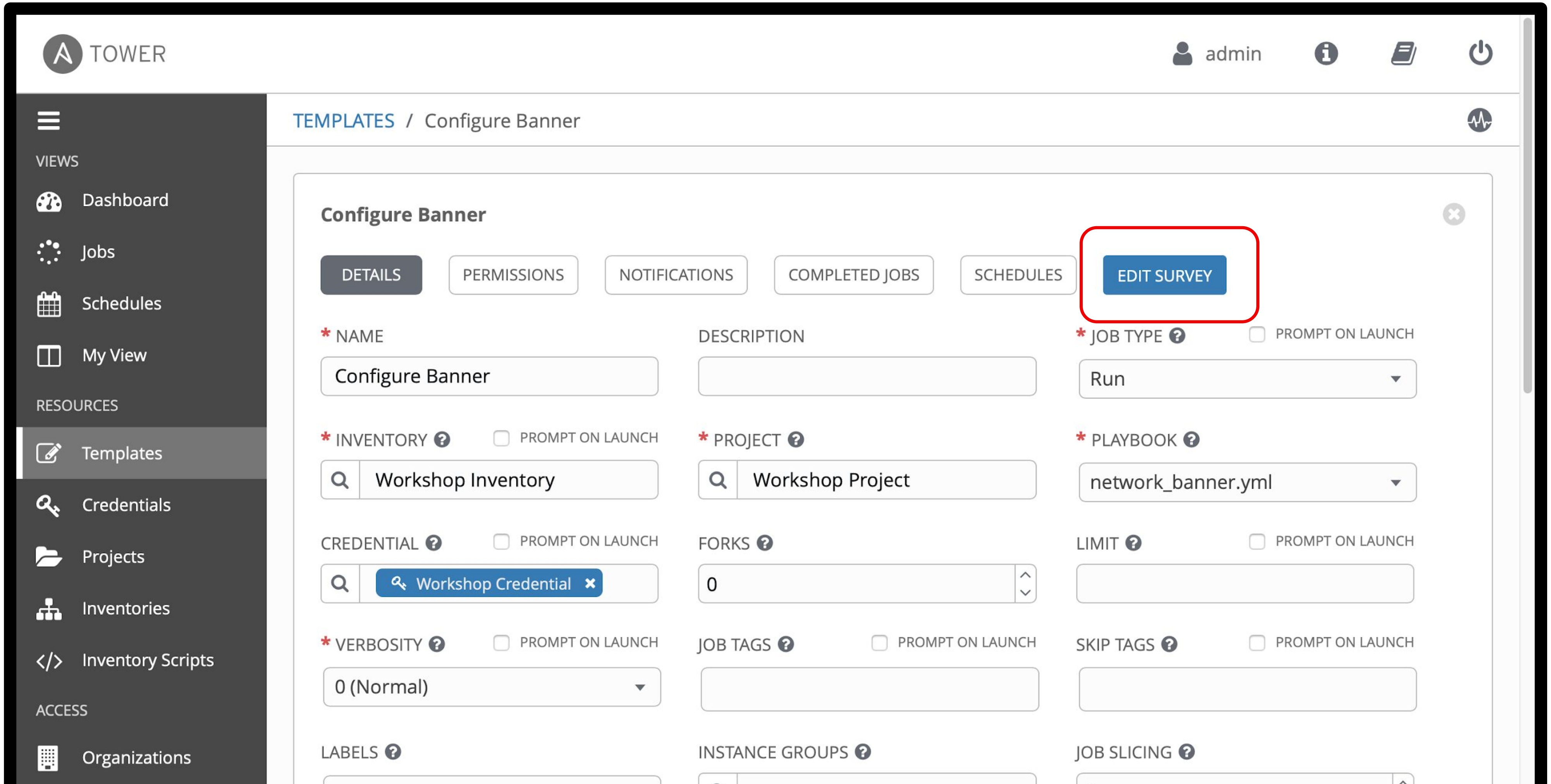
An Ansible Tower survey is a simple question-and-answer form that allows users to customize their job runs. Combine that with Tower's role-based access control, and you can build simple, easy self-service for your users.

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.

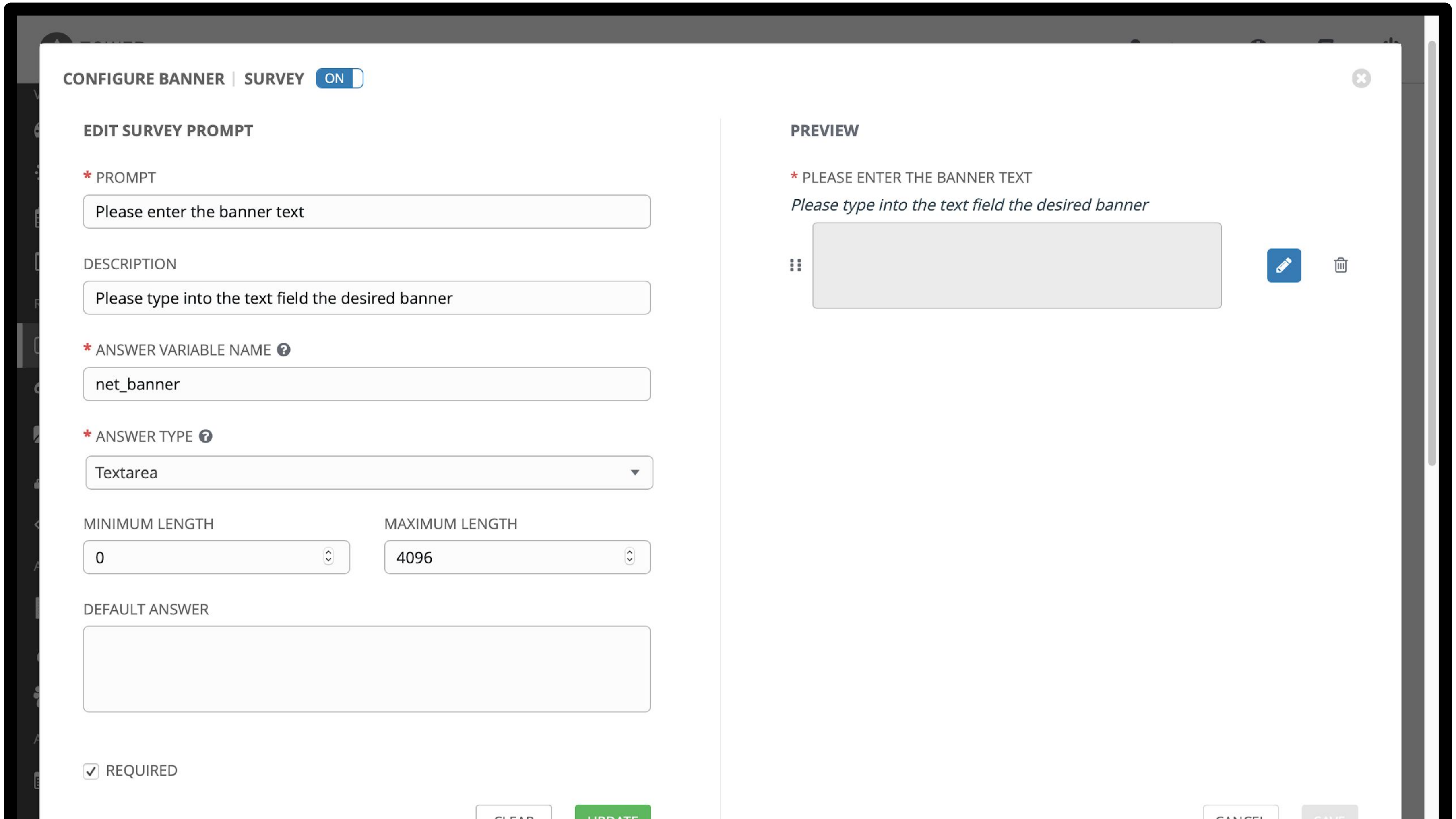


Configure Banner

* NAME: Configure Banner	DESCRIPTION:	* JOB TYPE: Run (□ PROMPT ON LAUNCH)
* INVENTORY: Workshop Inventory (□ PROMPT ON LAUNCH)	* PROJECT: Workshop Project	* PLAYBOOK: network_banner.yml
CREDENTIAL: Workshop Credential (□ PROMPT ON LAUNCH)	FORKS: 0	LIMIT: (□ PROMPT ON LAUNCH)
* VERBOSITY: 0 (Normal) (□ PROMPT ON LAUNCH)	JOB TAGS: (□ PROMPT ON LAUNCH)	SKIP TAGS: (□ PROMPT ON LAUNCH)
LABELS	INSTANCE GROUPS	JOB SLICING

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.



CONFIGURE BANNER | SURVEY ON

EDIT SURVEY PROMPT

* PROMPT
Please enter the banner text

DESCRIPTION
Please type into the text field the desired banner

* ANSWER VARIABLE NAME
net_banner

* ANSWER TYPE
Textarea

MINIMUM LENGTH
0

MAXIMUM LENGTH
4096

DEFAULT ANSWER

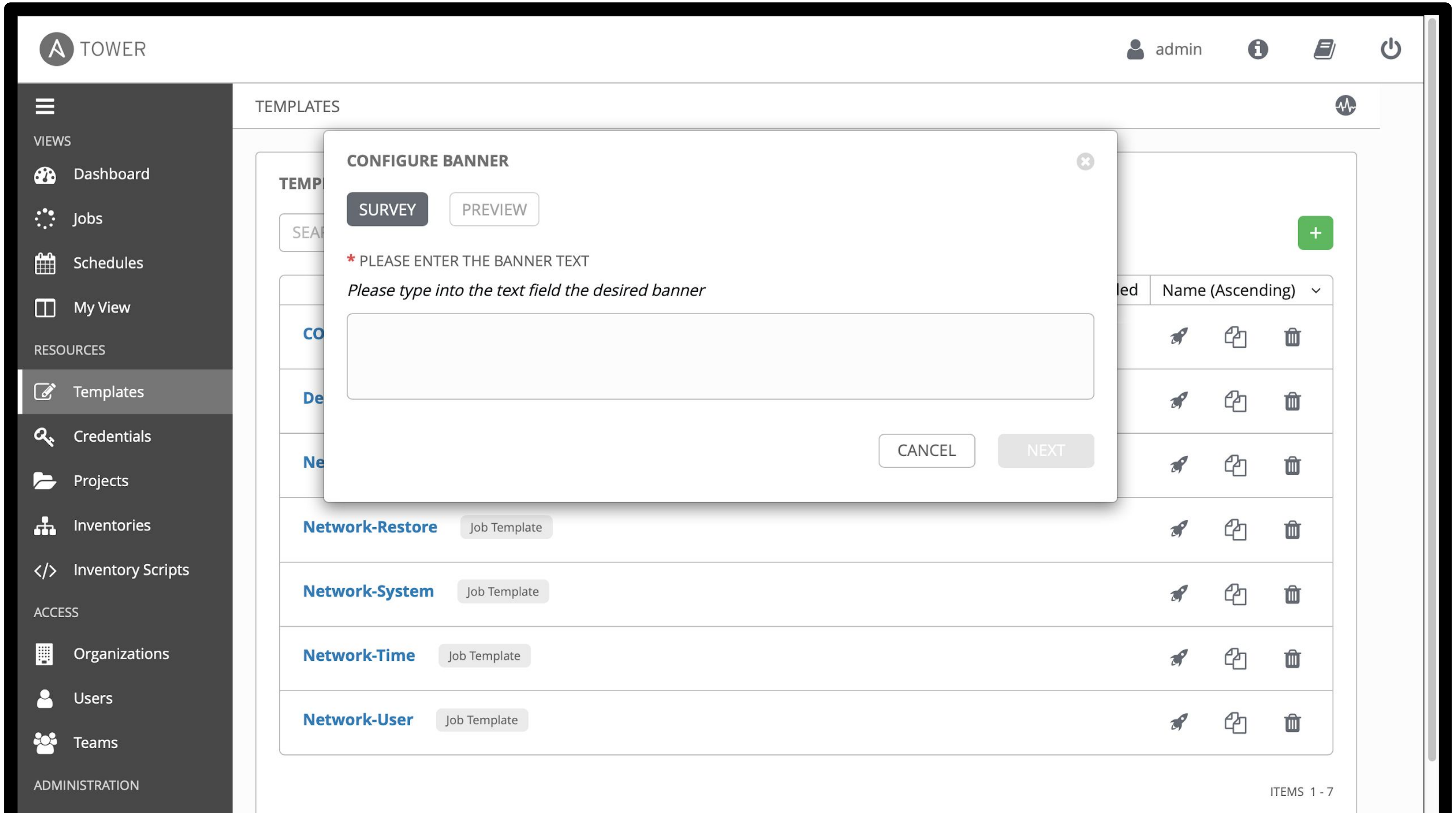
☑ REQUIRED

PREVIEW

* PLEASE ENTER THE BANNER TEXT
Please type into the text field the desired banner

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.



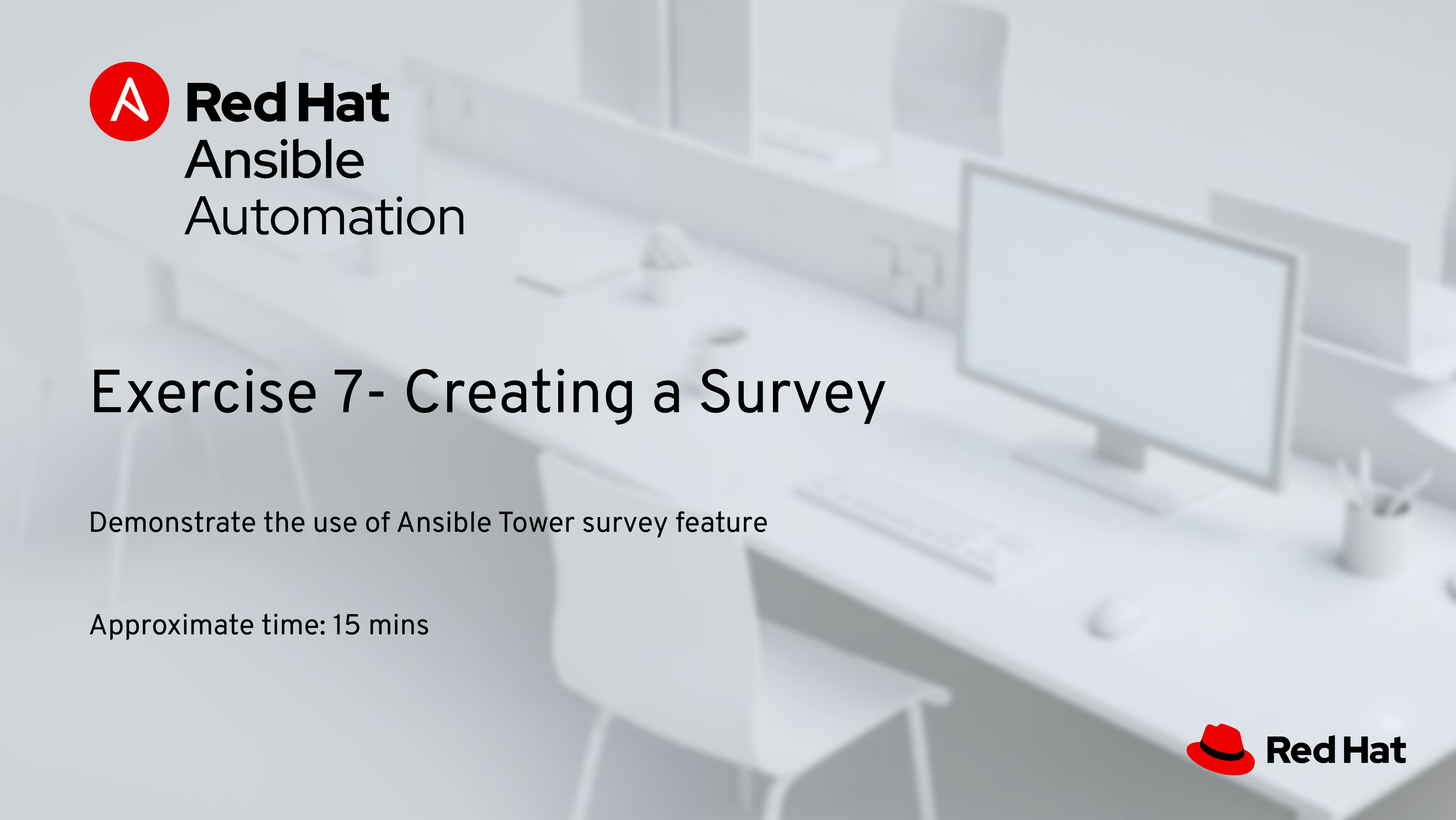
CONFIGURE BANNER

SURVEY PREVIEW

* PLEASE ENTER THE BANNER TEXT

Please type into the text field the desired banner

CANCEL NEXT



Red Hat
Ansible
Automation

Exercise 7- Creating a Survey

Demonstrate the use of Ansible Tower survey feature

Approximate time: 15 mins

Red Hat

Section 8

Topics Covered:

- Understanding Organizations
- Understanding Teams
- Understanding Users

Red Hat
Ansible
Tower

Role Based Access Control (RBAC)

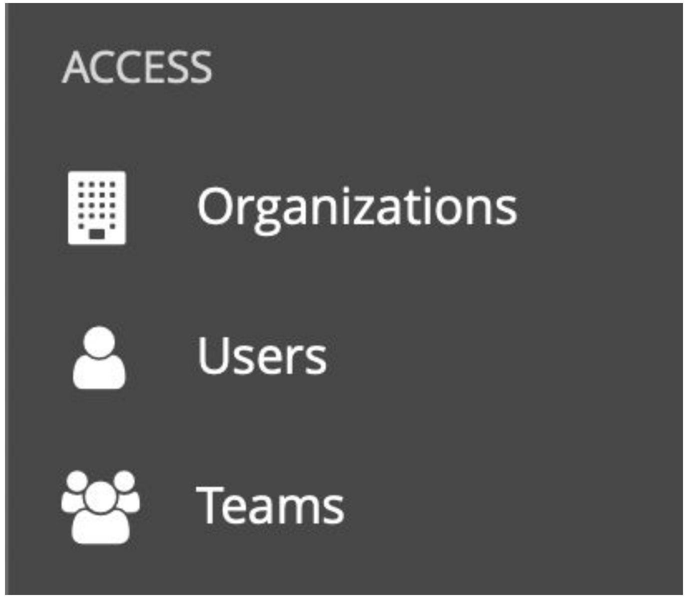
Role-Based Access Controls (RBAC) are built into Ansible Tower and allow administrators to delegate access to inventories, organizations, and more. These controls allow Ansible Tower to help you increase security and streamline management of your Ansible automation.

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.

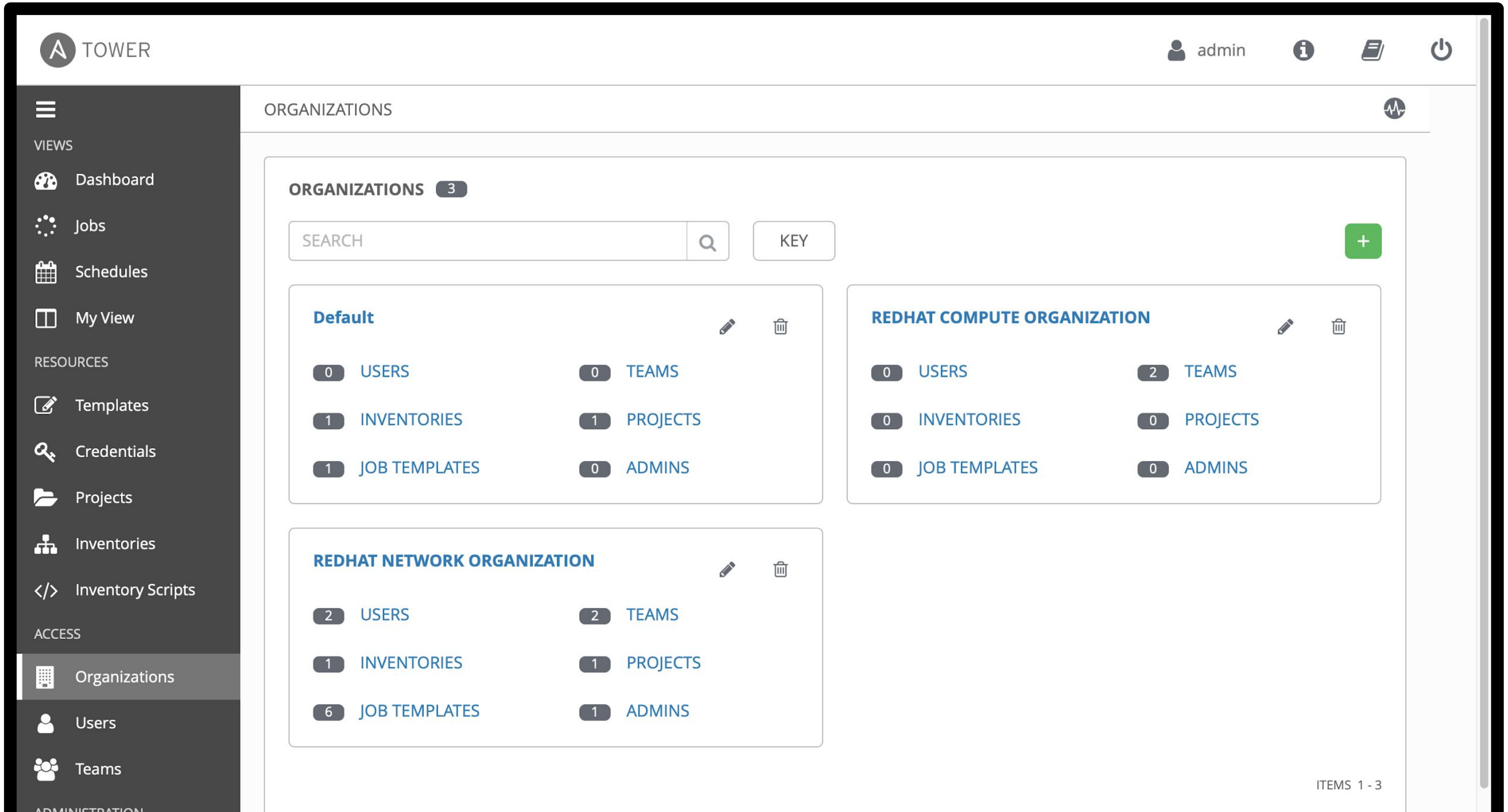
- A **user** is an account to access Ansible Tower and its services given the permissions granted to it.

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



Viewing Organizations

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



ORGANIZATIONS 3

Default

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

REDHAT COMPUTE ORGANIZATION

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

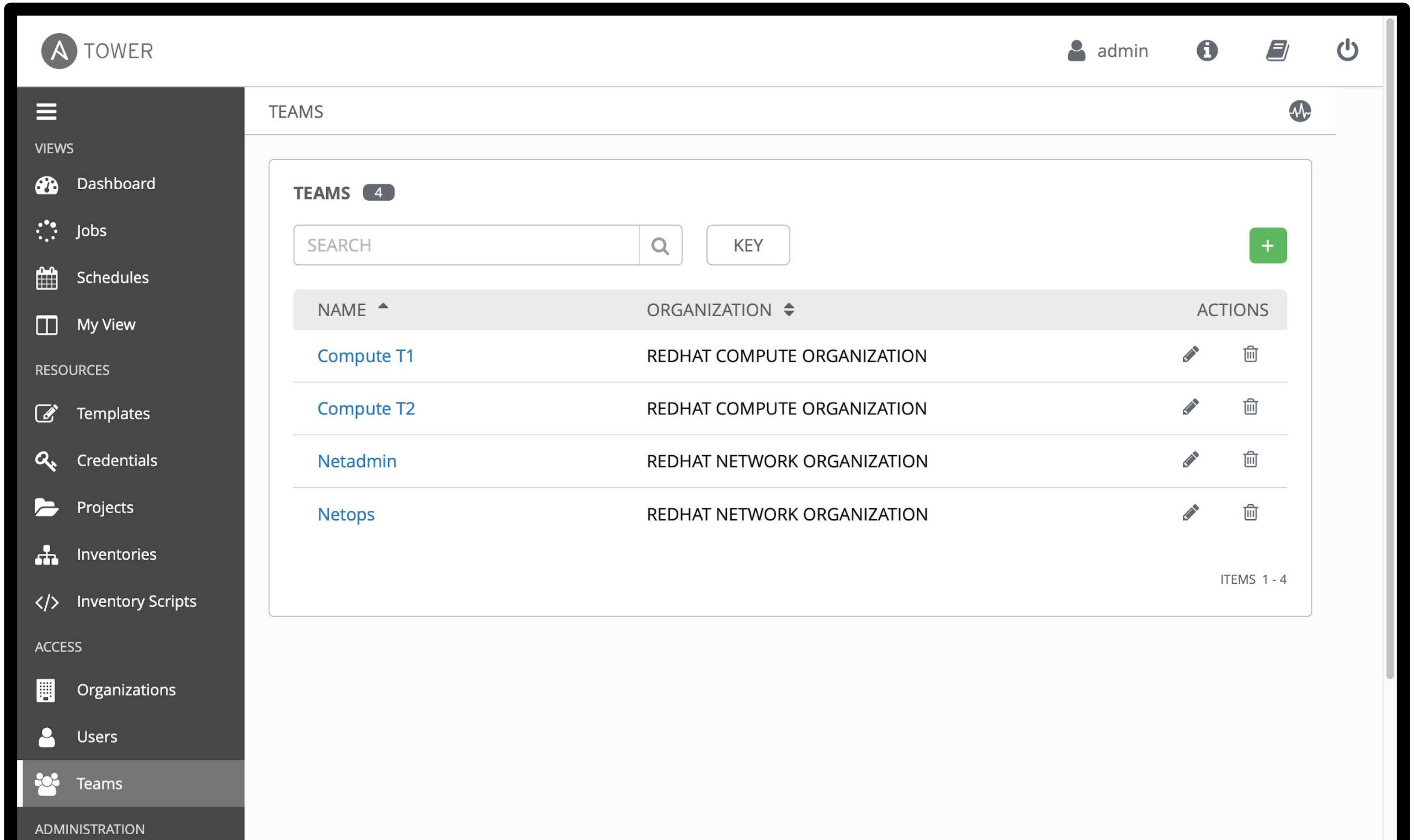
REDHAT NETWORK ORGANIZATION

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

ITEMS 1 - 3

Viewing Teams

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



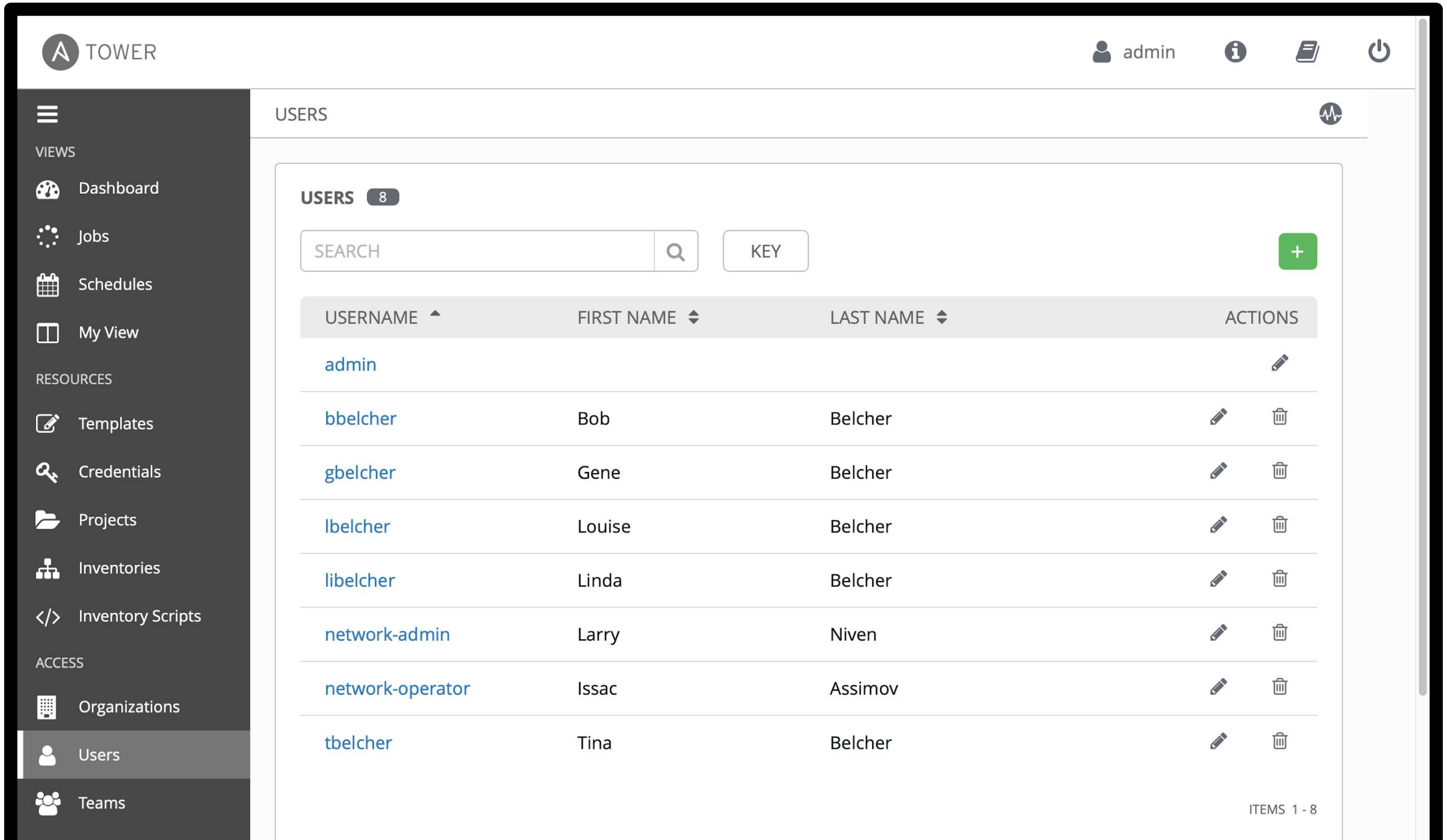
TEAMS 4

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 in the left menu will open up the Users window



USERS 8

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

Red Hat
Ansible
Automation

Exercise 8 - Understanding RBAC

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

Approximate time: 15 mins

Section 9

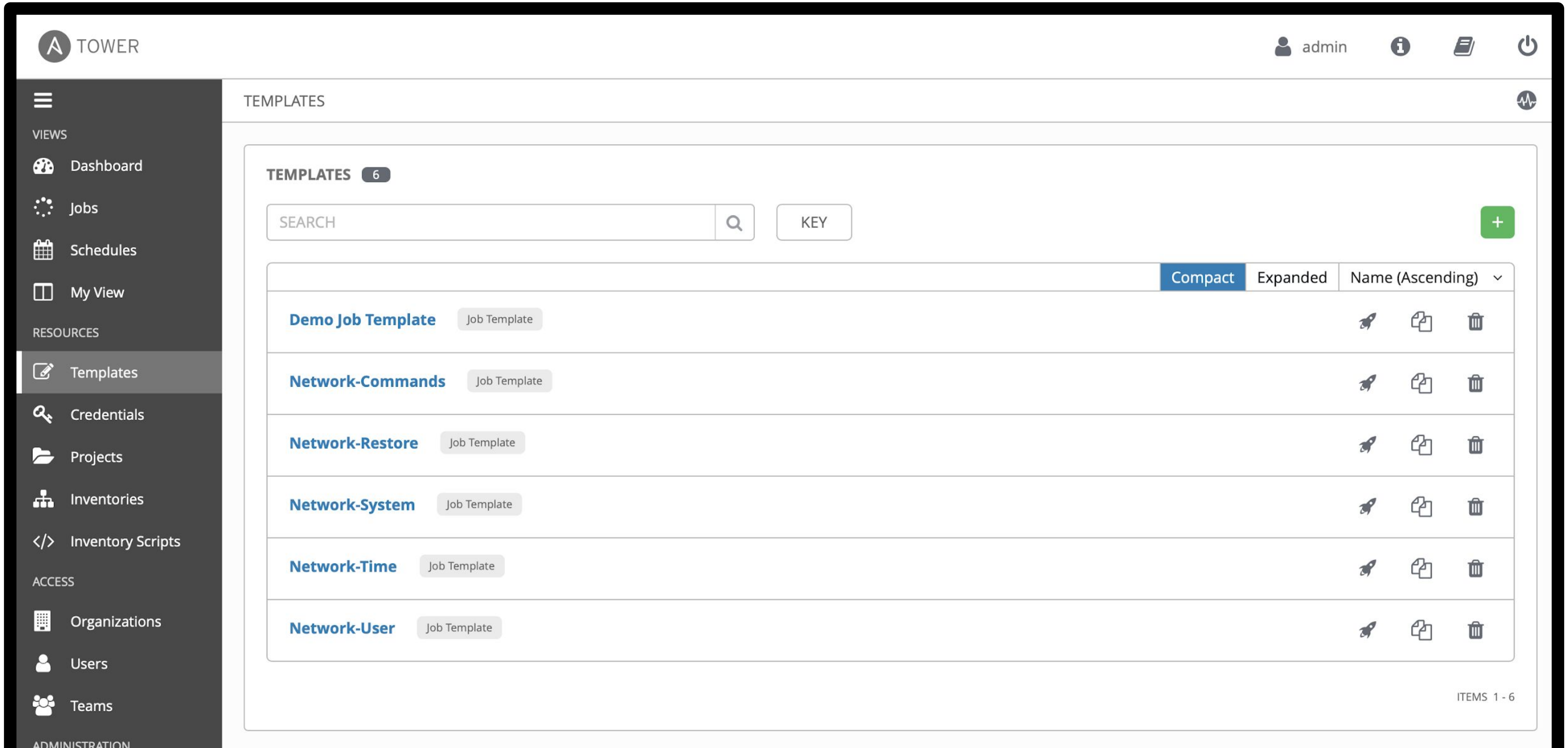
Red Hat Ansible Tower

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.

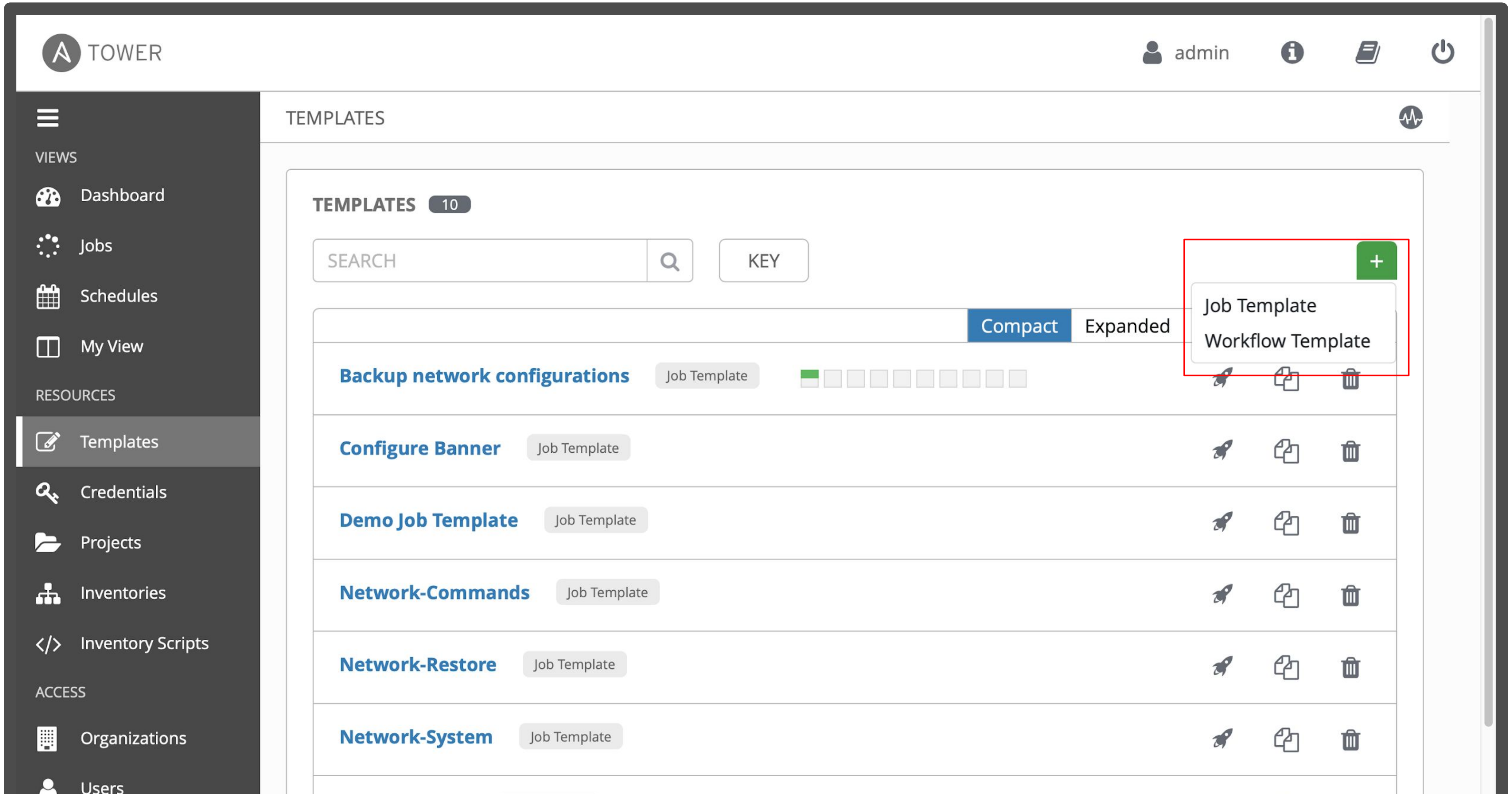


Demo Job Template	Job Template
Network-Commands	Job Template
Network-Restore	Job Template
Network-System	Job Template
Network-Time	Job Template
Network-User	Job Template

Adding a new Workflow Template

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

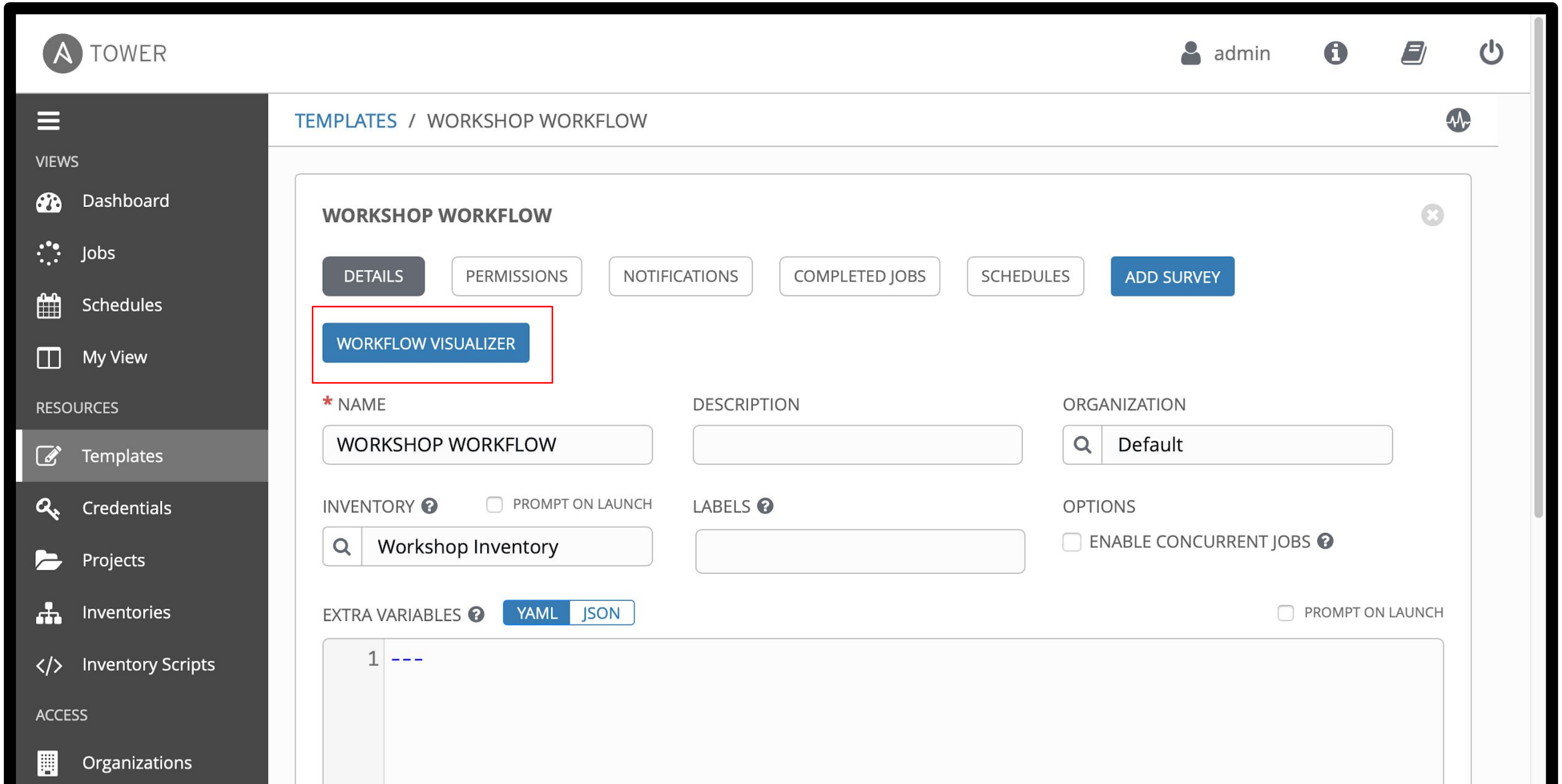
This time select the **Workflow Template**



Backup network configurations | Job Template
Configure Banner | Job Template
Demo Job Template | Job Template
Network-Commands | Job Template
Network-Restore | Job Template
Network-System | Job 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.



* NAME: WORKSHOP WORKFLOW

DESCRIPTION:

ORGANIZATION: Default

INVENTORY: Workshop Inventory ☐ PROMPT ON LAUNCH

LABELS:

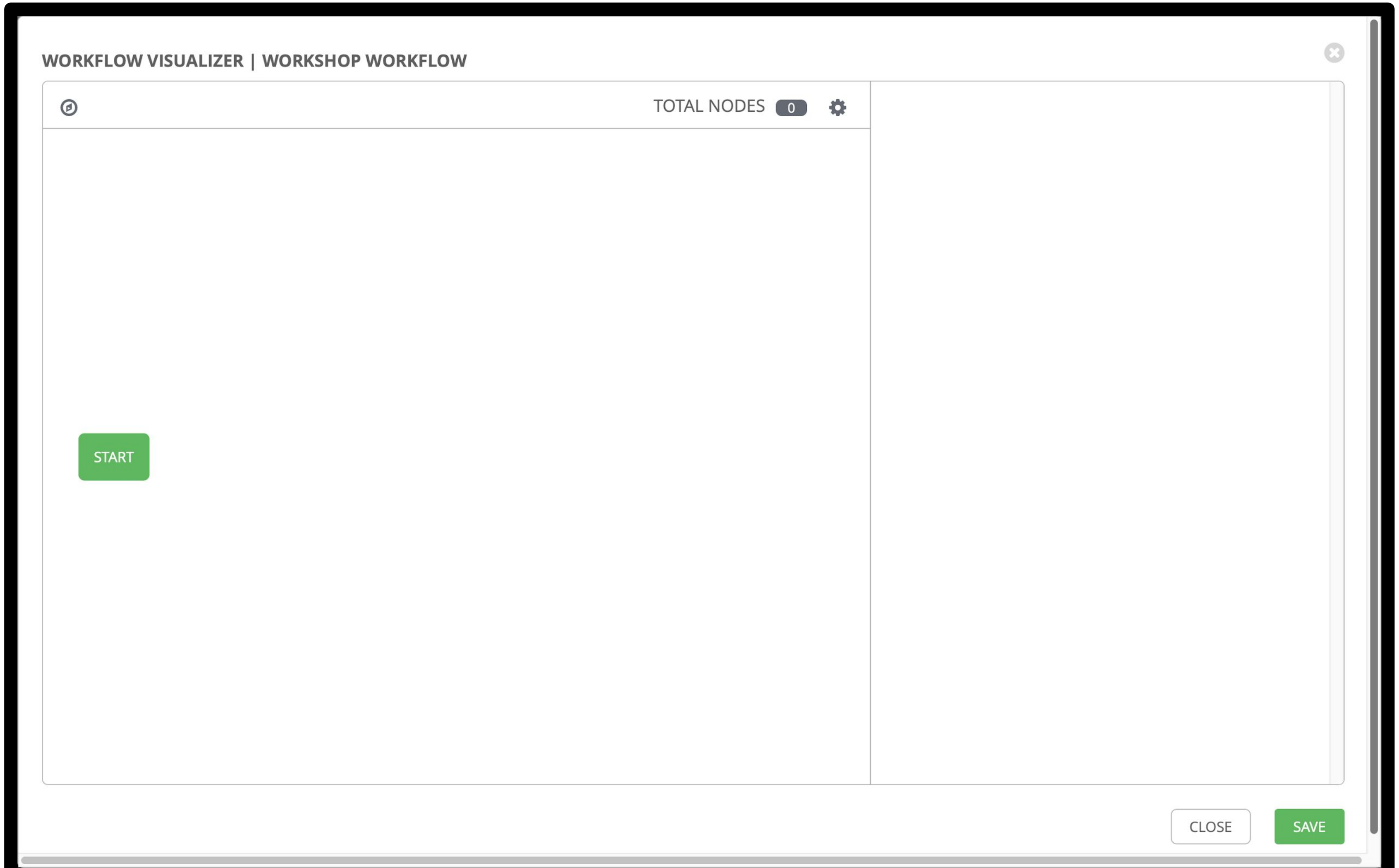
OPTIONS: ☐ ENABLE CONCURRENT JOBS

EXTRA VARIABLES ☐ PROMPT ON LAUNCH

```
---
```

Workflow Visualizer

The workflow visualizer will start as a blank canvas.

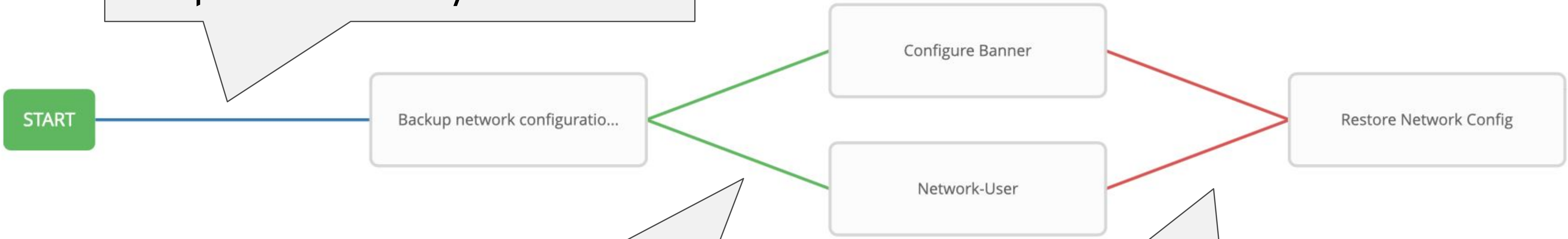


WORKFLOW VISUALIZER | WORKSHOP WORKFLOW

Visualizing a Workflow

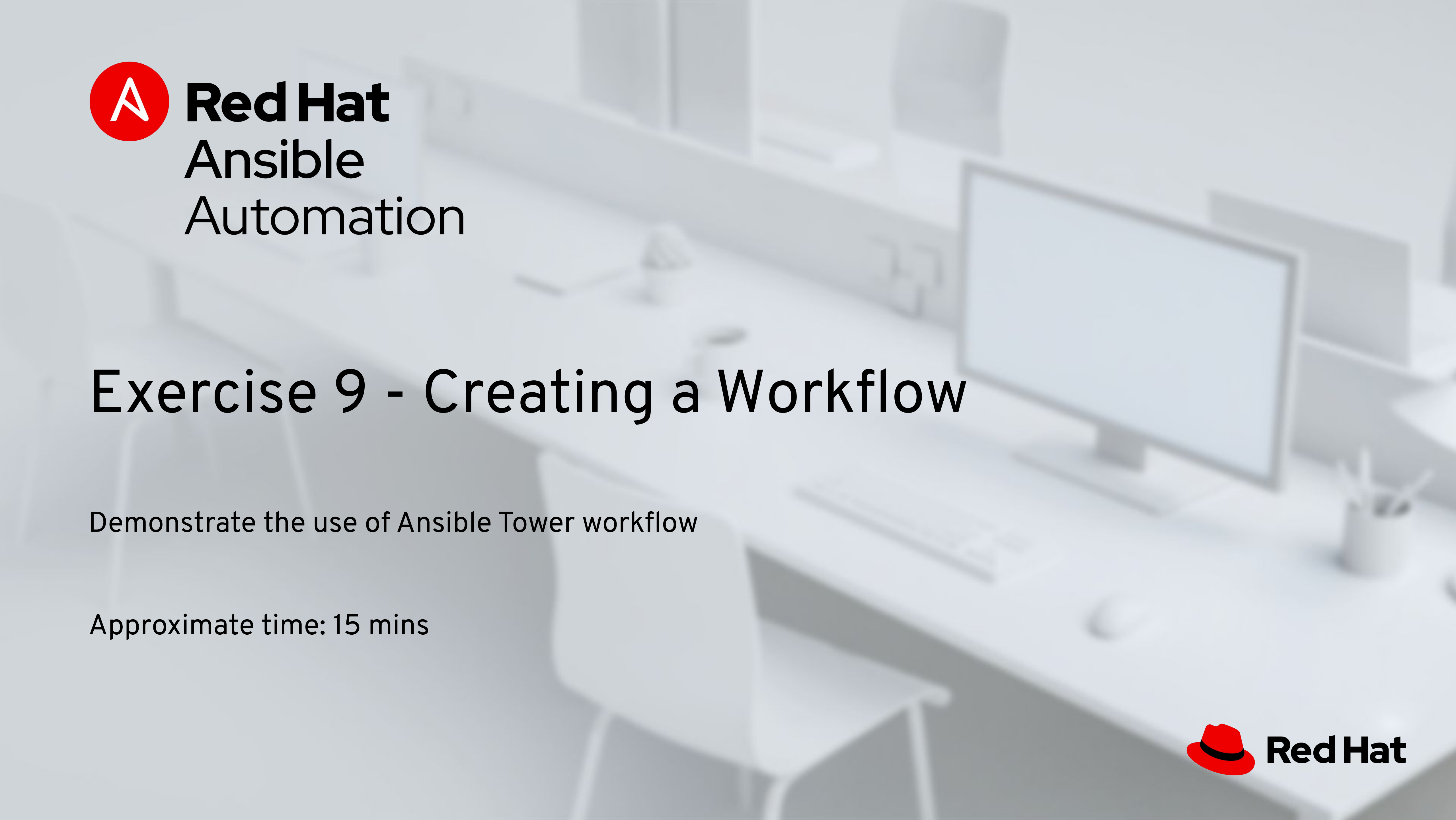
Workflows can branch out, or converge in.

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

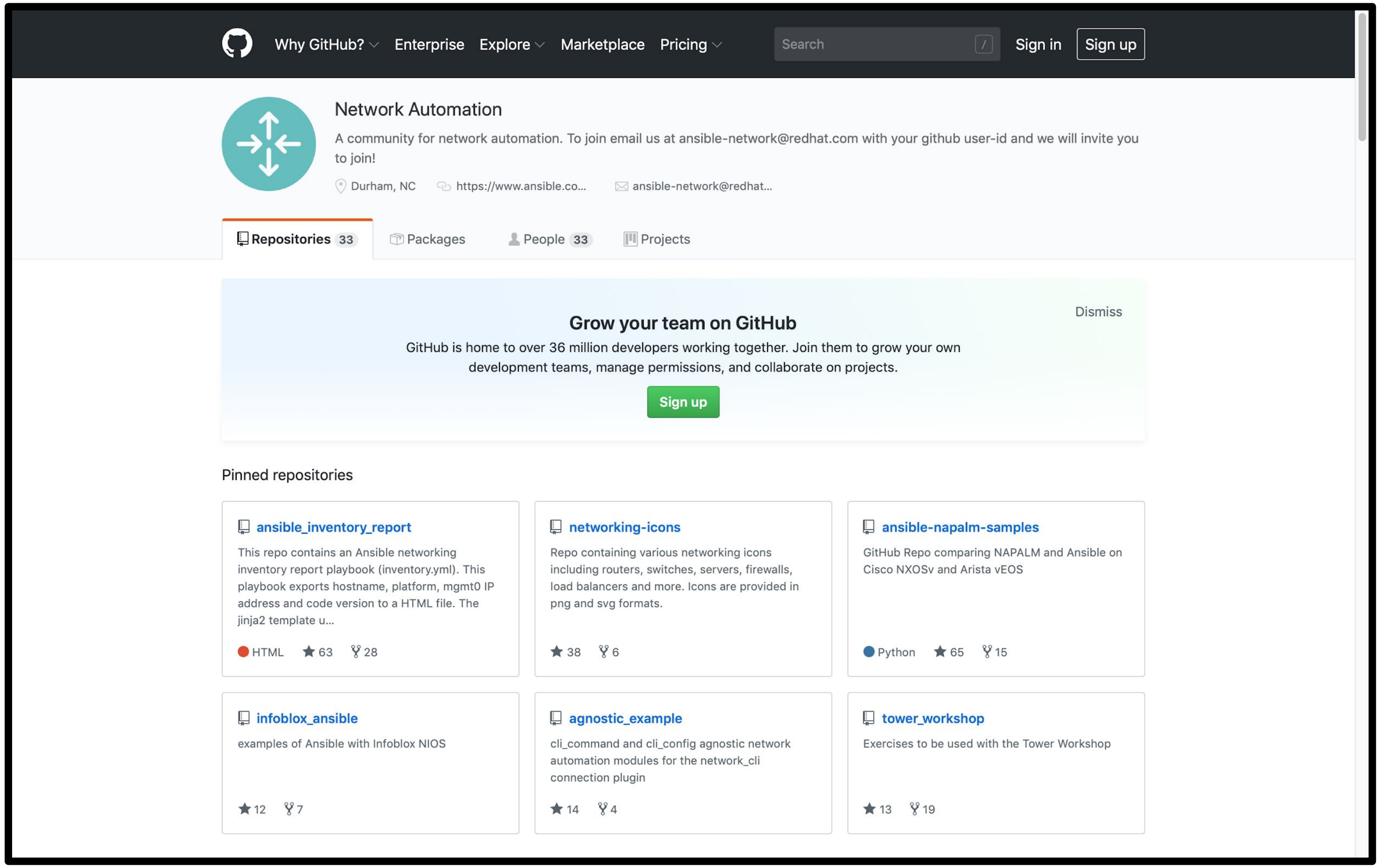
- **Slack**
  https://ansiblenetwork.slack.com
  Join by clicking here https://bit.ly/2OfNEBr

- **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



Network Automation

A community for network automation. To join email us at ansible-network@redhat.com with your github user-id and we will invite you to join!

Durham, NC https://www.ansible.co... ansible-network@redhat...

Grow your team on GitHub

GitHub is home to over 36 million developers working together. Join them to grow your own development teams, manage permissions, and collaborate on projects.

Pinned repositories

ansible_inventory_report
This repo contains an Ansible networking inventory report playbook (inventory.yml). This playbook exports hostname, platform, mgmt0 IP address and code version to a HTML file. The jinja2 template u...
HTML ★ 63 28

networking-icons
Repo containing various networking icons including routers, switches, servers, firewalls, load balancers and more. Icons are provided in png and svg formats.
★ 38 6

ansible-napalm-samples
GitHub Repo comparing NAPALM and Ansible on Cisco NXOSv and Arista vEOS
Python ★ 65 15

infoblox_ansible
examples of Ansible with Infoblox NIOS
★ 12 7

agnostic_example
cli_command and cli_config agnostic network automation modules for the network_cli connection plugin
★ 14 4

tower_workshop
Exercises to be used with the Tower Workshop
★ 13 19

CONFIDENTIAL Designator

Thank you

linkedin.com/company/red-hat

youtube.com/AnsibleAutomation

facebook.com/ansibleautomation

twitter.com/ansible

github.com/ansible