



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

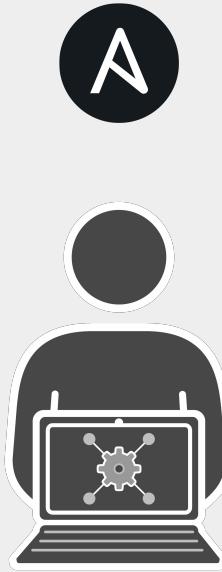
- Introduction to Ansible automation
- How Ansible works for network automation
- Understanding Ansible modules and playbooks
- Executing Ansible playbooks to:
 - Make configuration changes
 - Gather information (Ansible facts)
- Using Jinja to template network configurations
- Using Ansible Tower to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

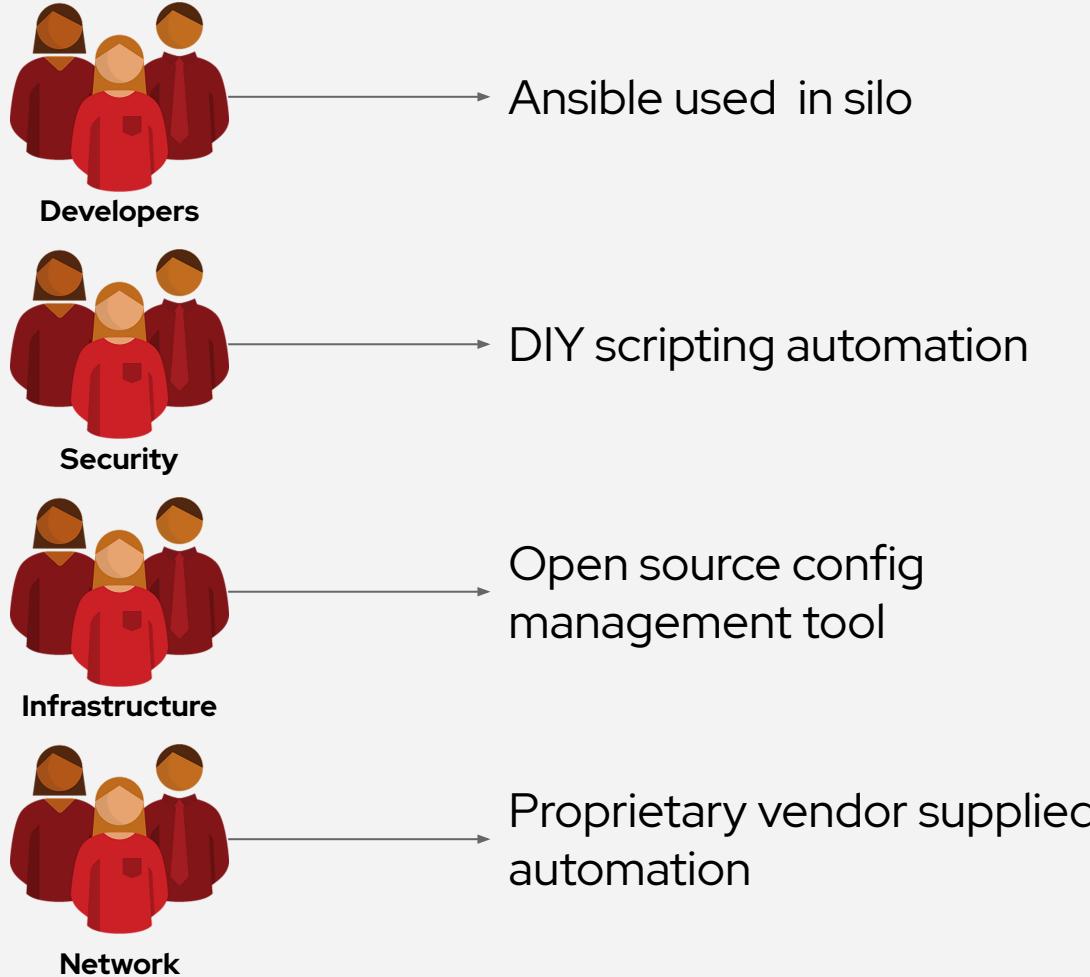


Developers



Infrastructure

Ad-hoc Automation is happening in silos



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

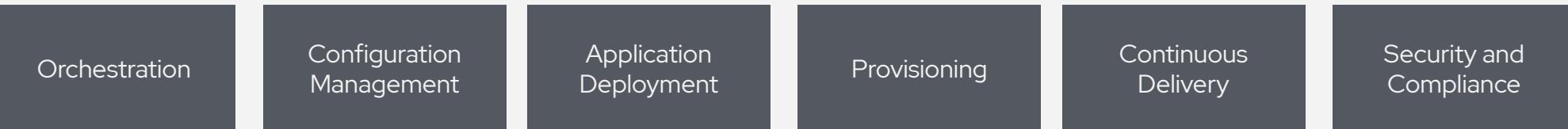
Get started immediately

More efficient & more secure

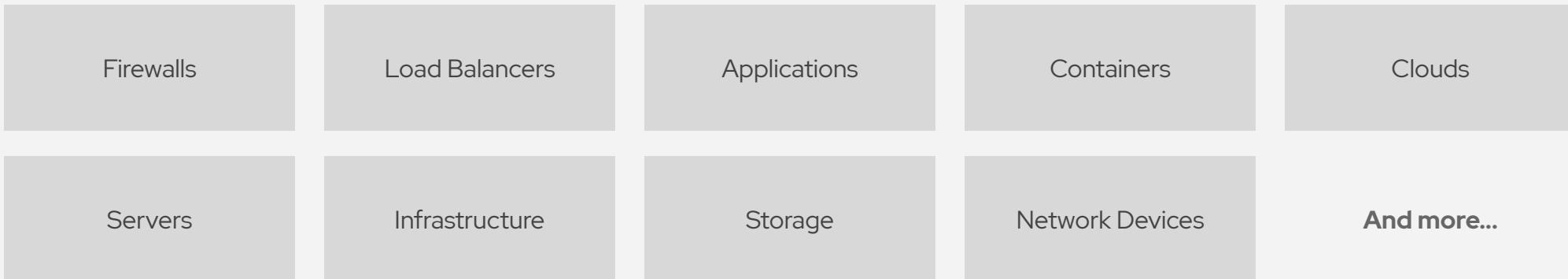
What can I do using Ansible?

Automate the deployment and management of your entire IT footprint.

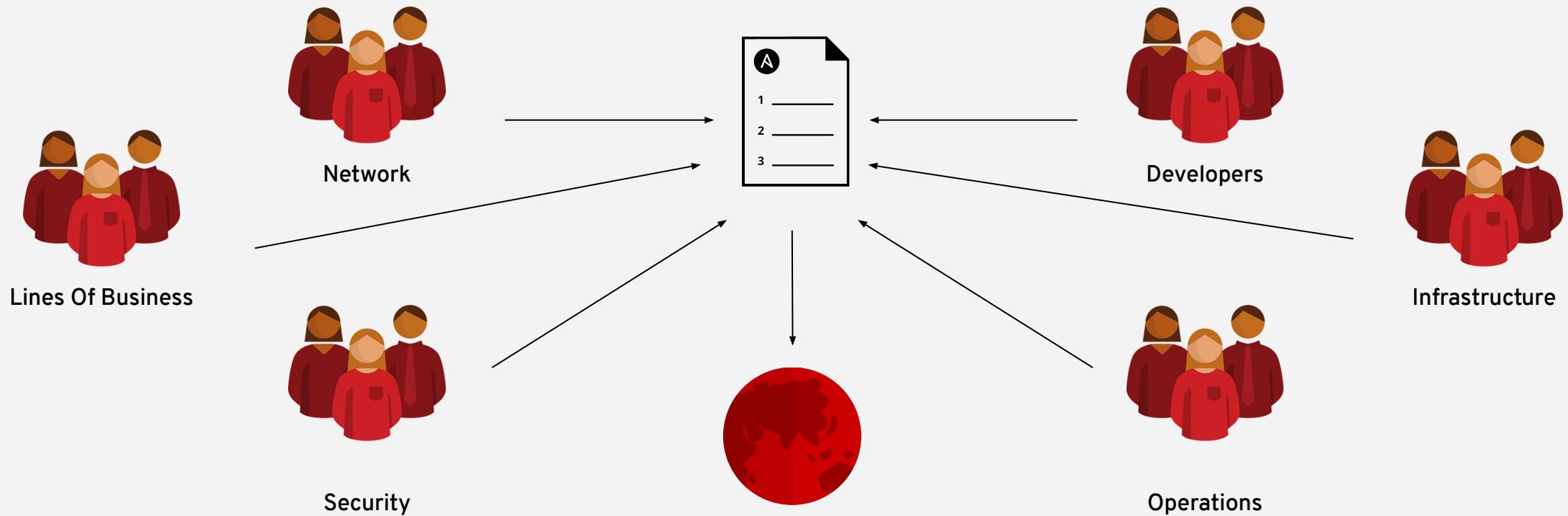
Do this...



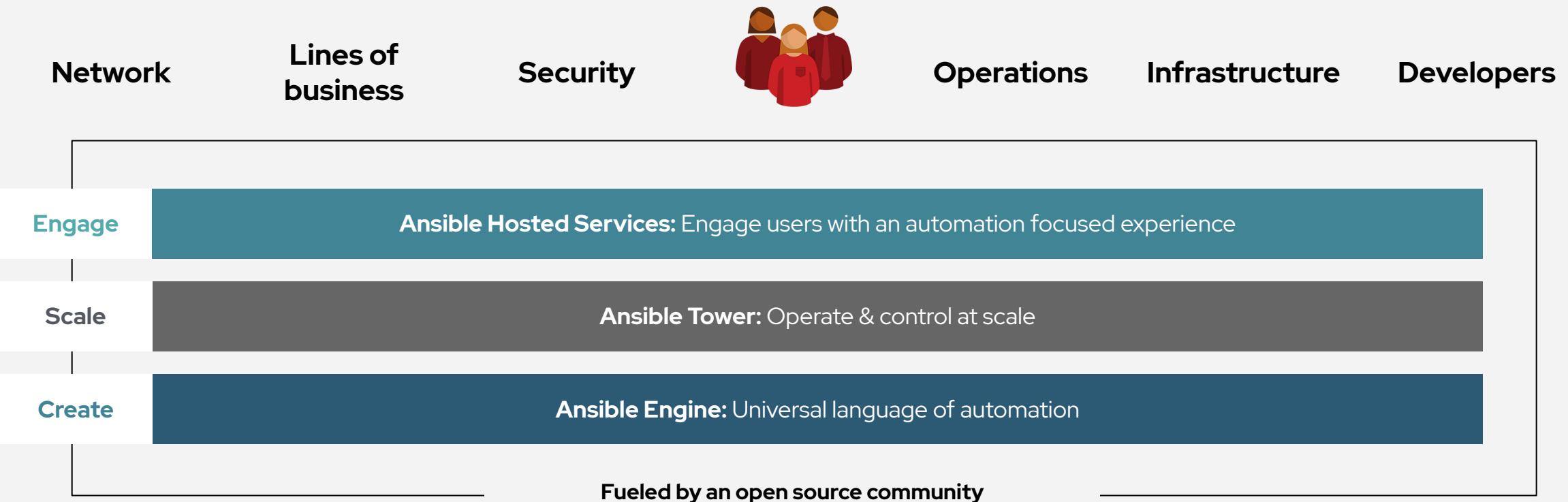
On these...



When automation crosses teams, you need an automation platform



Red Hat Ansible Automation Platform



Ansible automates technologies you use

Time to automate is measured in minutes

Cloud	Virt & Container	Windows	Network	Security	Monitoring
AWS	Docker	ACLs	A10	Checkpoint	Dynatrace
Azure	VMware	Files	Arista	Cisco	Datadog
Digital Ocean	RHV	Packages	Aruba	CyberArk	LogicMonitor
Google	OpenStack	IIS	Cumulus	F5	New Relic
OpenStack	OpenShift	Regedits	Bigswitch	Fortinet	Sensu
Rackspace	+more	Shares	Cisco	Juniper	+more
+more		Services	Dell	IBM	
Operating Systems	Storage	Configs	Extreme	Palo Alto	Devops
RHEL	Netapp	Users	F5	Snort	Jira
Linux	Red Hat Storage	Domains	Lenovo	+more	GitHub
Windows	Infinidat	+more	MikroTik		Vagrant
+more	+more		Juniper		Jenkins
			OpenSwitch		Slack
			+more		+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

SOURCE: "The Total Economic Impact™ Of Red Hat Ansible Tower, a June 2018 commissioned study conducted by Forrester Consulting on behalf of Red Hat."
redhat.com/en/engage/total-economic-impact-ansible-tower-20180710





Red Hat

Ansible Automation Platform

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)

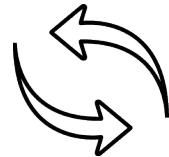


SIMPLE

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+

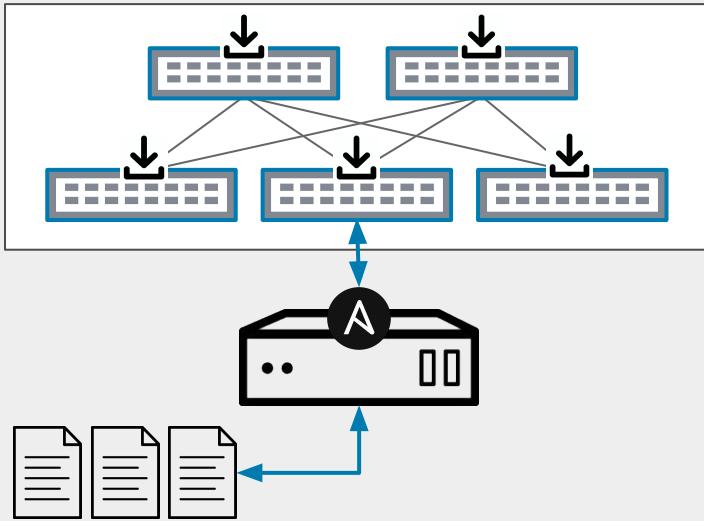
Network
Platforms

1000+

Network
Modules

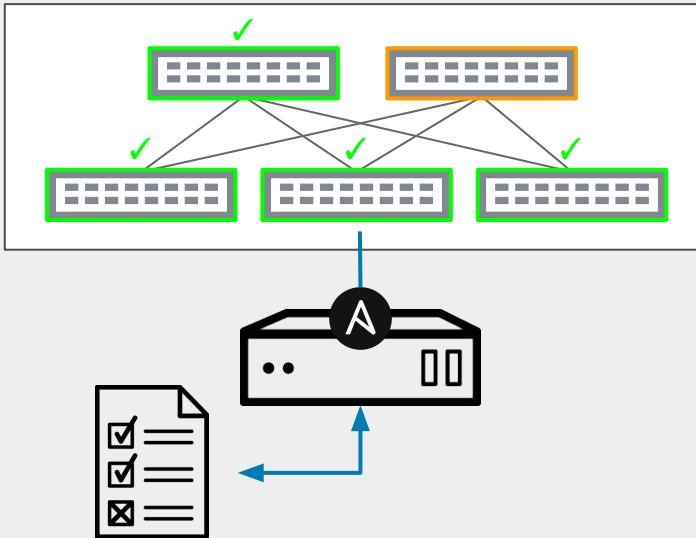
ansible.com/for/networks

Common use cases



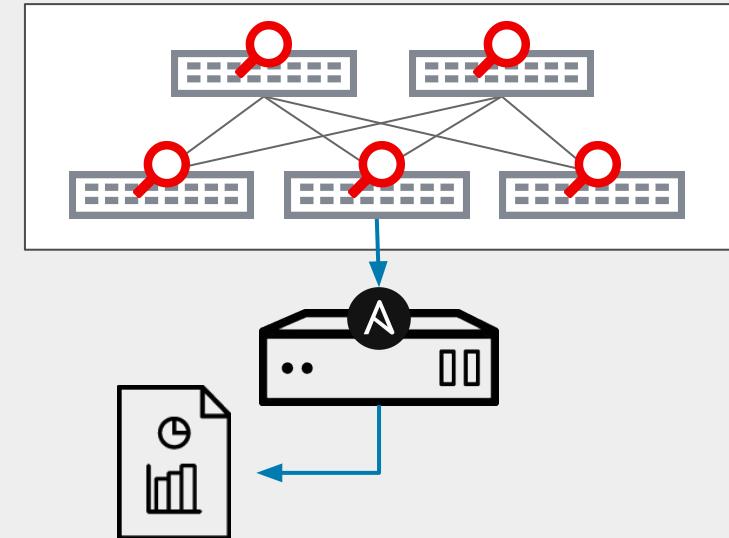
Backup and Restore

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



Configuration Compliance

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

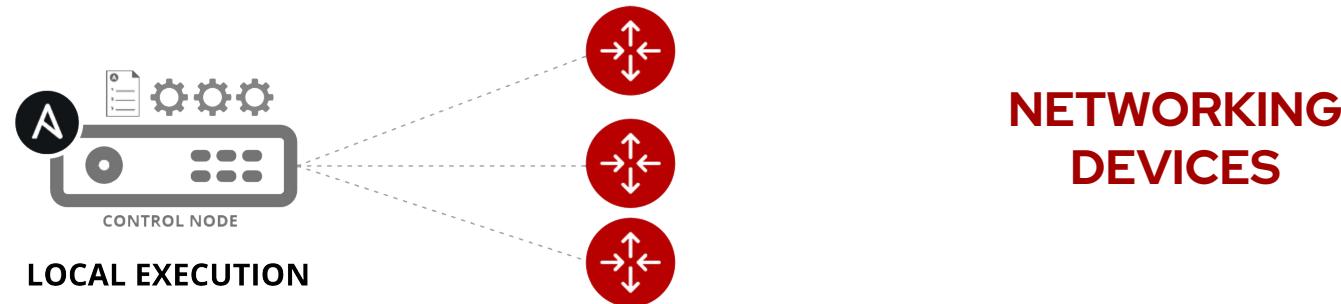


Dynamic Documentation

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

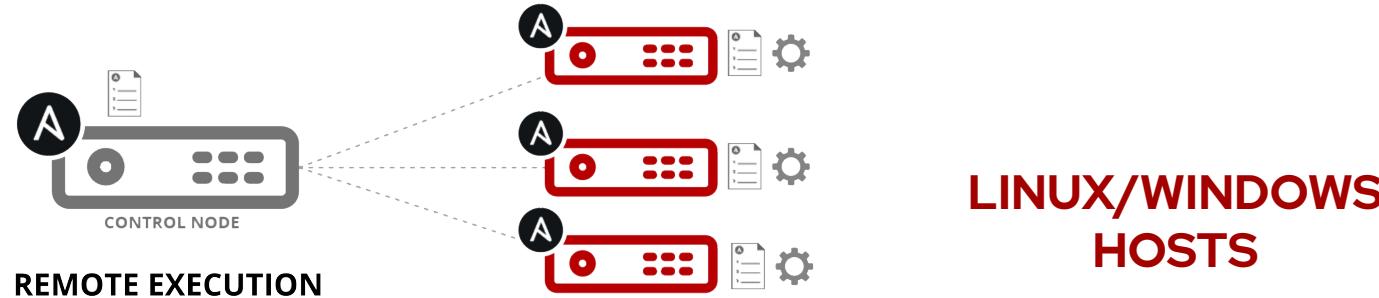
How Ansible Network Automation works

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**



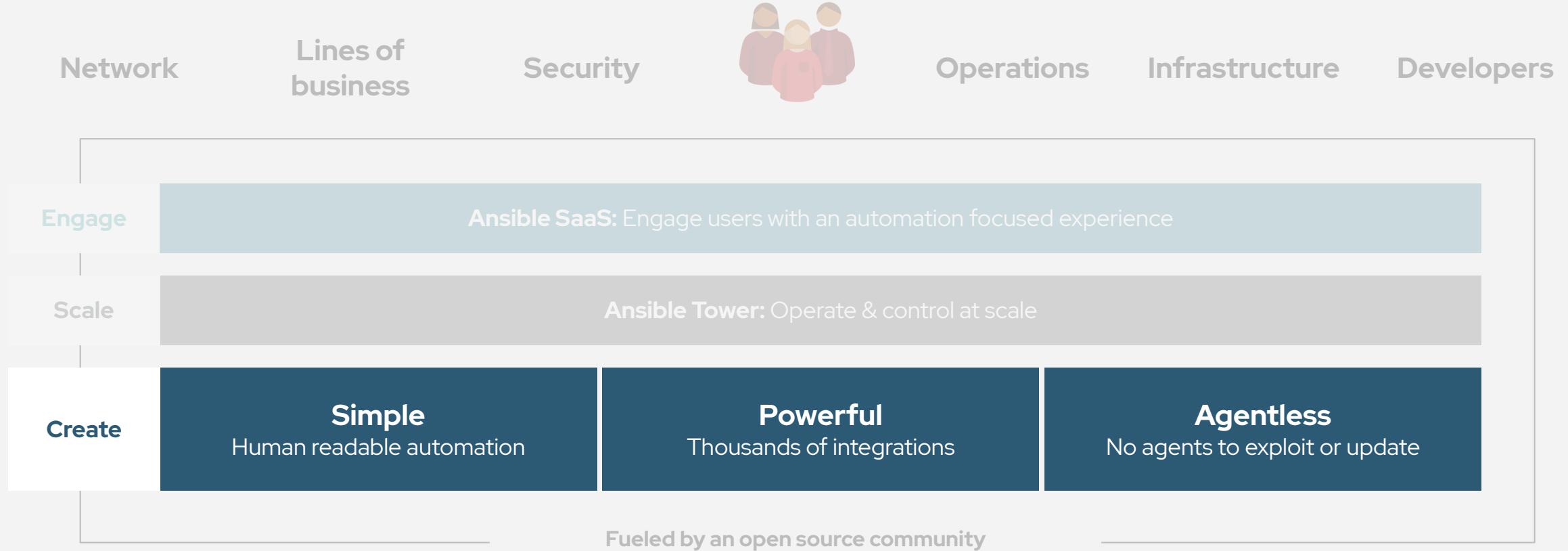
Red Hat

Ansible Automation Platform

Red Hat Ansible Engine:
Universal language
of automation



Red Hat Ansible Automation Platform

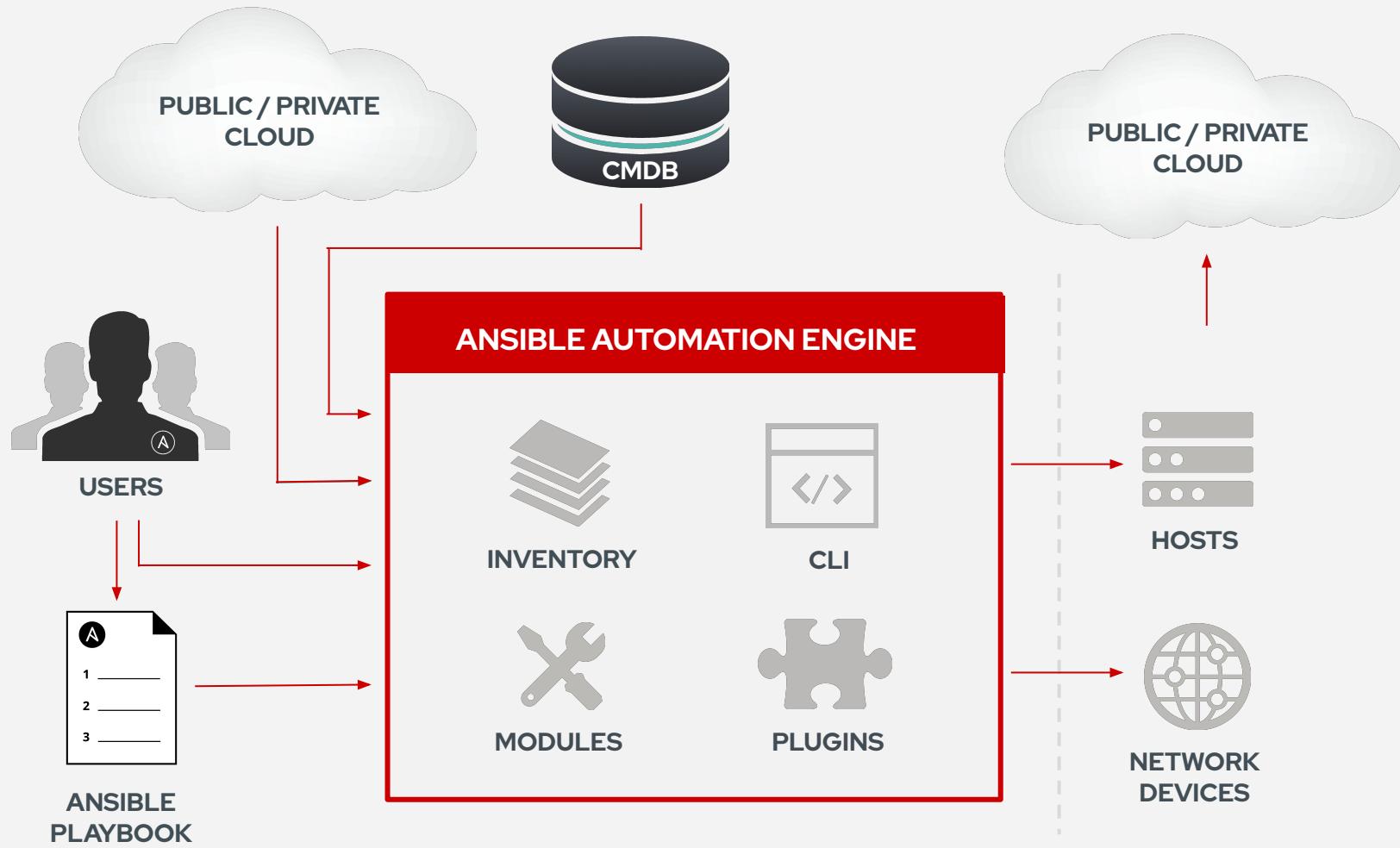


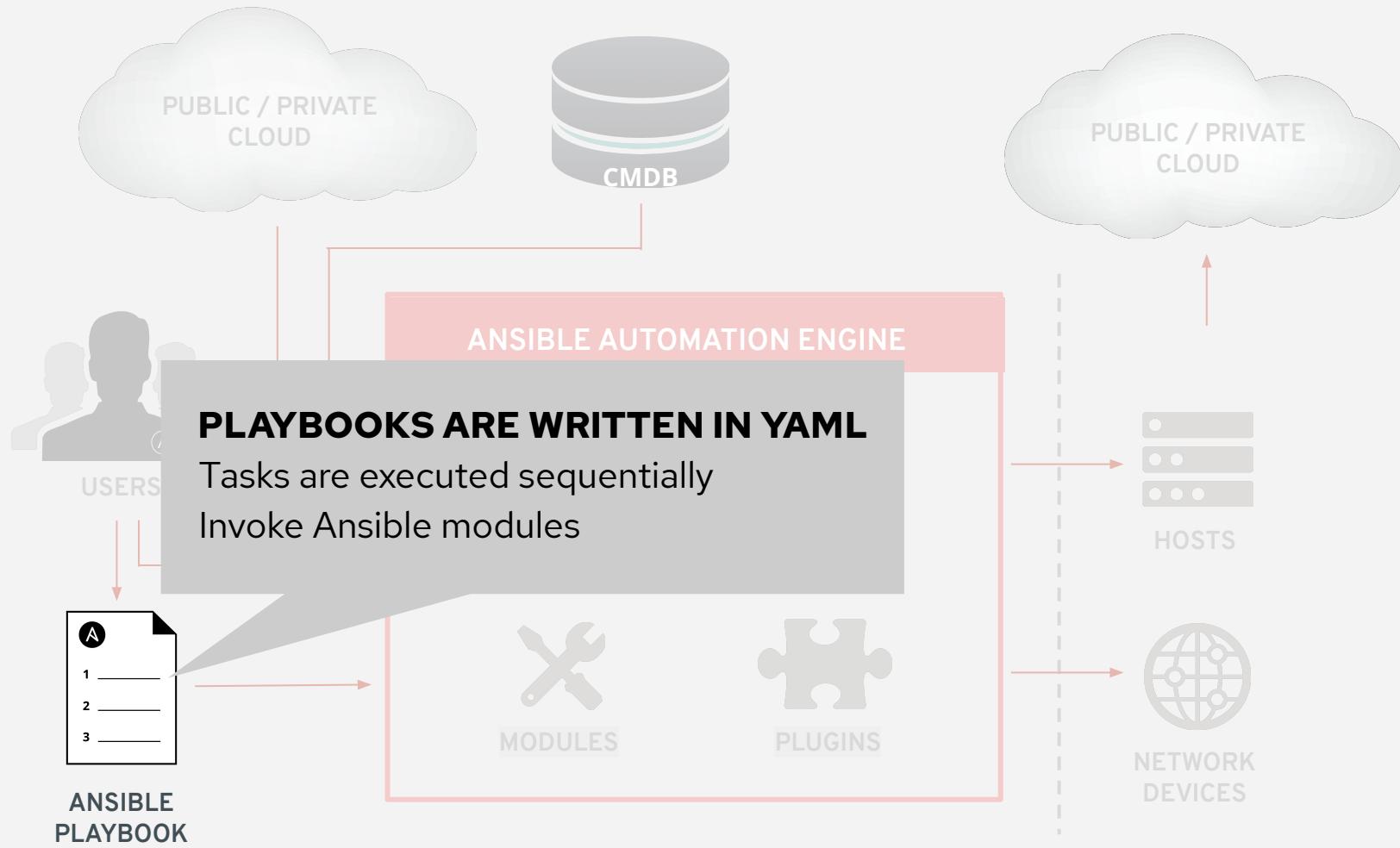


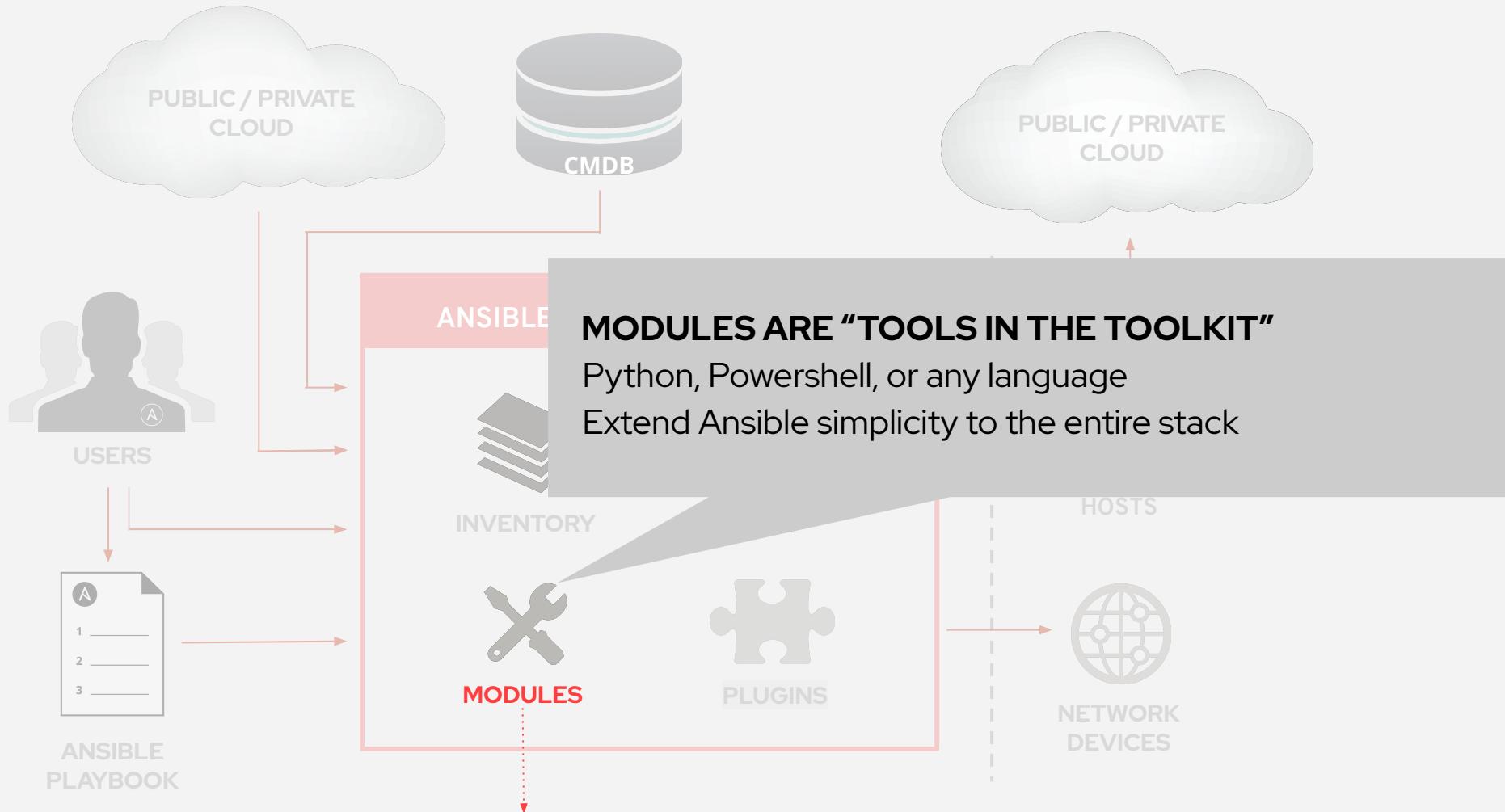
Exercise 1

Topics Covered:

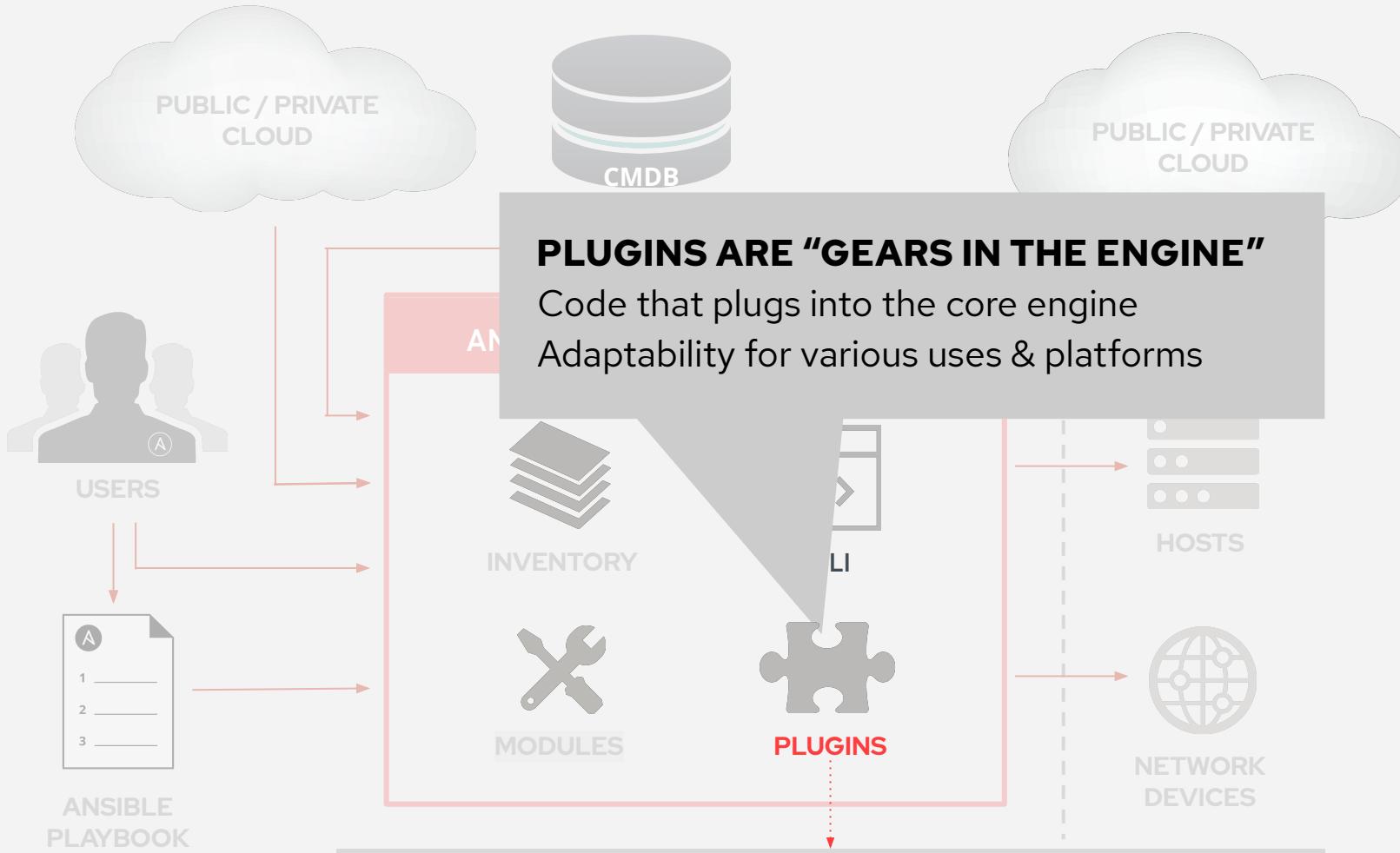
- Understanding Inventory
- An example Ansible Playbook



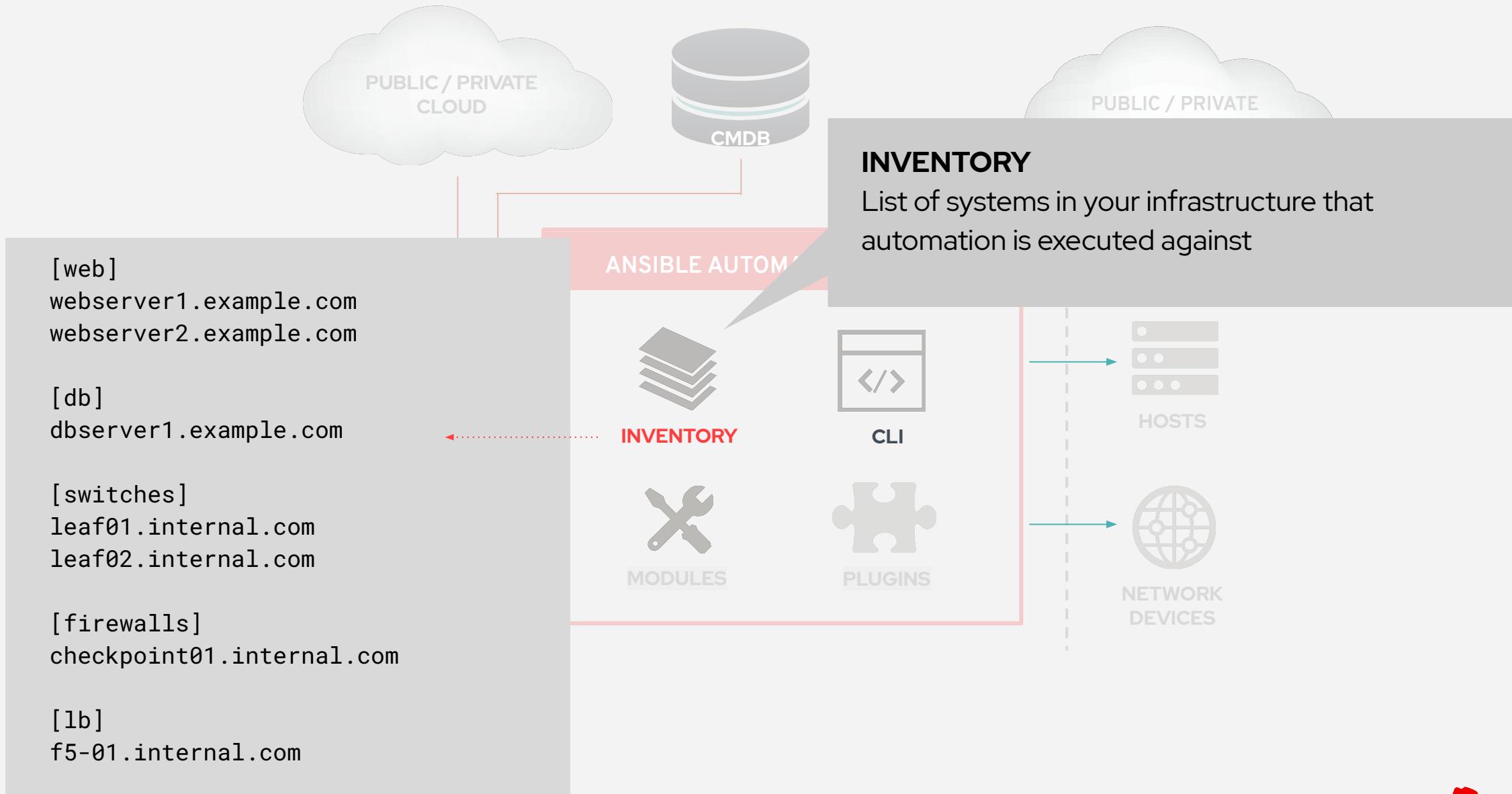




```
- name: latest index.html file is present
  template:
    src: files/index.html
    dest: /var/www/html/
```



```
{ { some_variable | to_nice_yaml } }
```



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 for version 2.8. The top navigation bar includes links for ANSIBLEFEST, PRODUCTS, COMMUNITY, WEBINARS & TRAINING, and BLOG. The main content area is titled "Ansible for Network Automation". It explains that Ansible Network modules extend automation to network administrators and teams, mentioning configuration, testing, validation, discovery, and drift correction. It provides links to "Getting Started with Ansible for Network Automation" and "Advanced Topics with Ansible for Network Automation". Below this, it lists network modules maintained by the Ansible community. A sidebar on the left contains links for Installation, Upgrade & Configuration, Using Ansible, Contributing to Ansible, Extending Ansible, Common Ansible Scenarios, and Ansible for Network Automation. The "Ansible for Network Automation" section in the sidebar is expanded, showing sub-links for Getting Started, Advanced Topics, Developer Guide, and Reference & Appendices. A search bar at the bottom right allows users to search the 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 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 atomic 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 reload 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

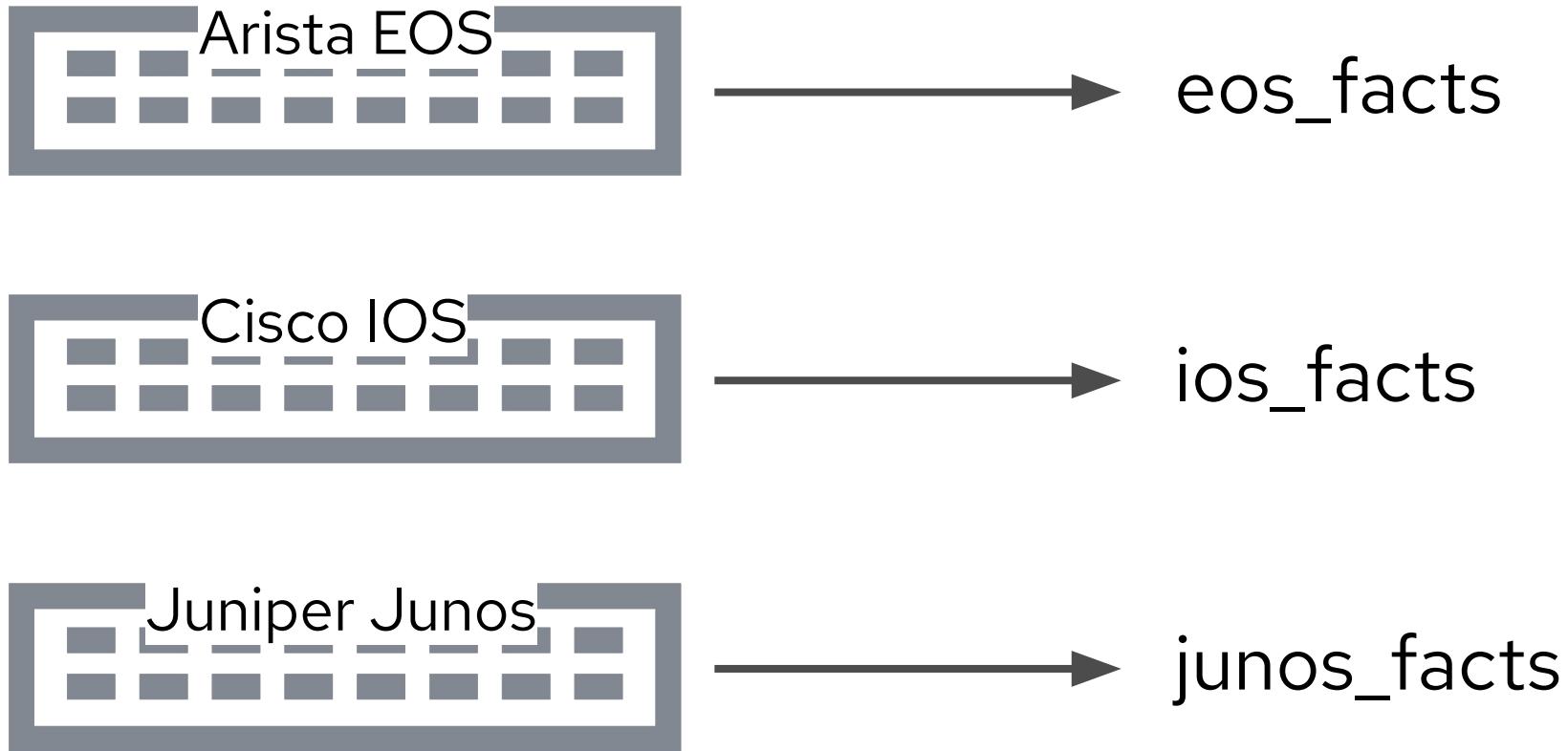
```
# List out all modules installed
$ ansible-doc -l
...
ios_banner                                Manage multiline banners on Cisco IOS devices
ios_command                                 Run commands on remote devices running Cisco IOS
ios_config                                  Manage Cisco IOS configuration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



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 mcpred
..
.
<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



Red Hat



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:

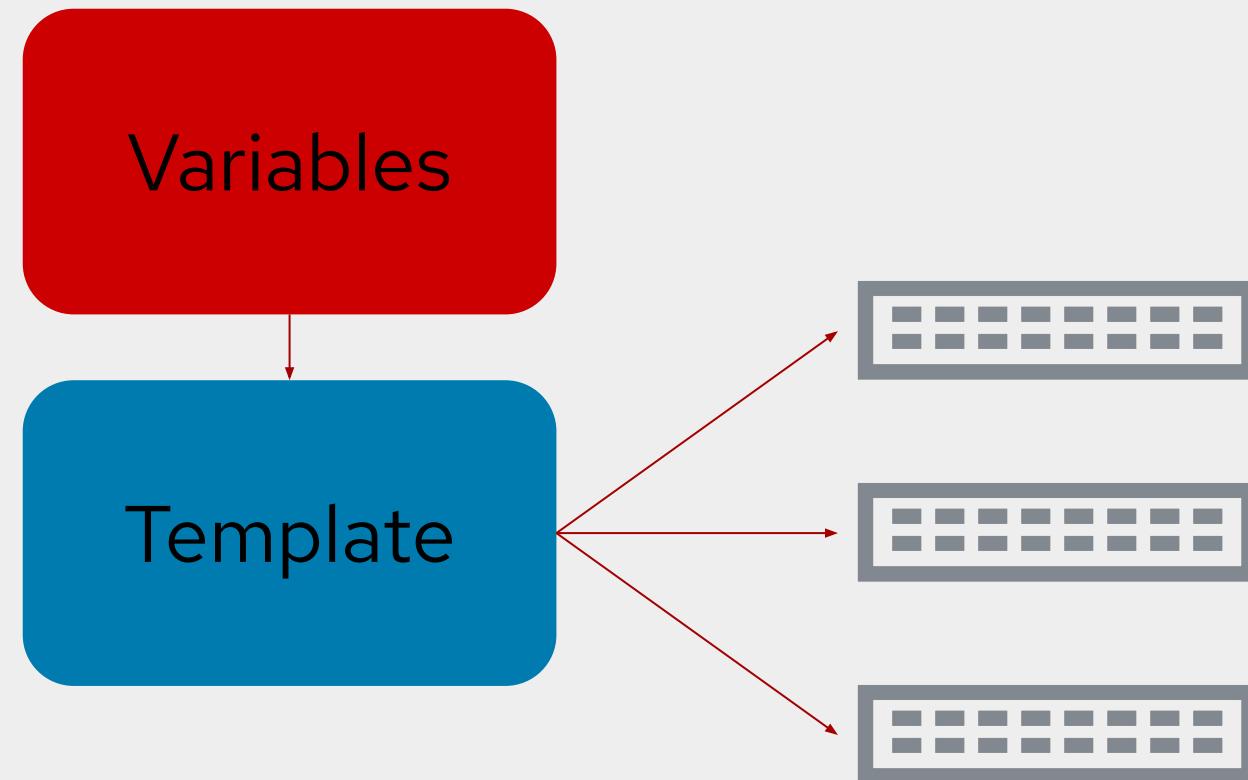
iosxr_config:

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

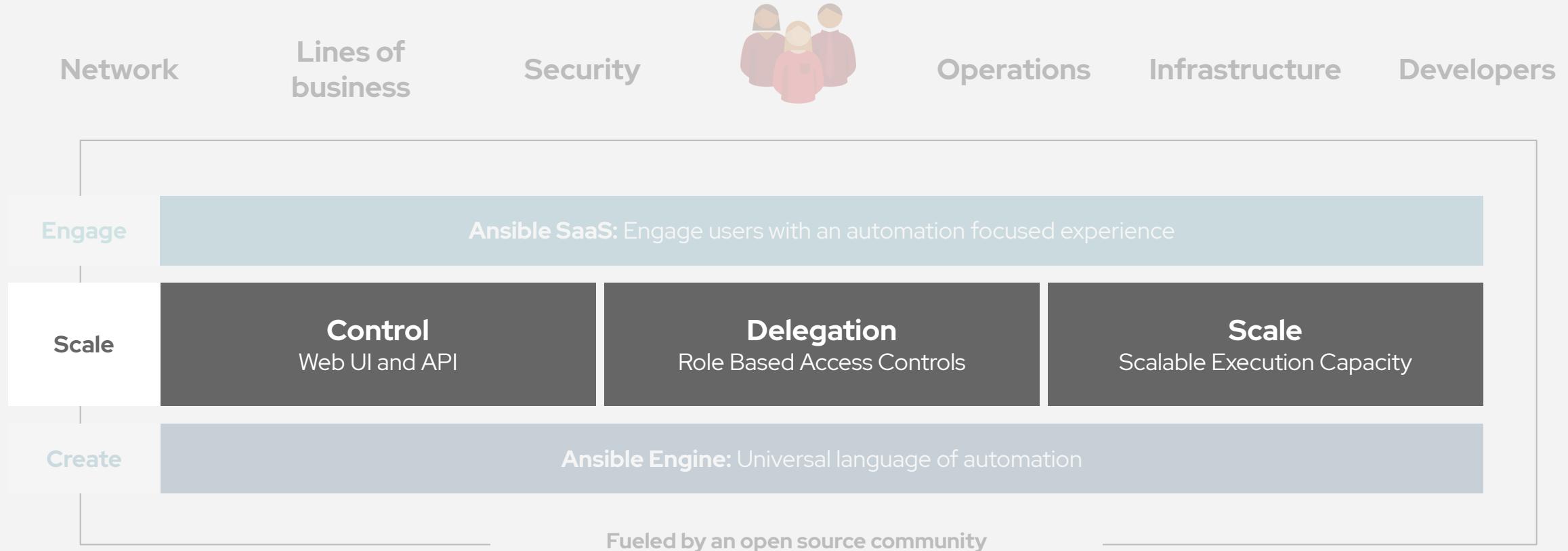


Tower Introduction

Topics Covered:

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

Red Hat Ansible Automation Platform



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.

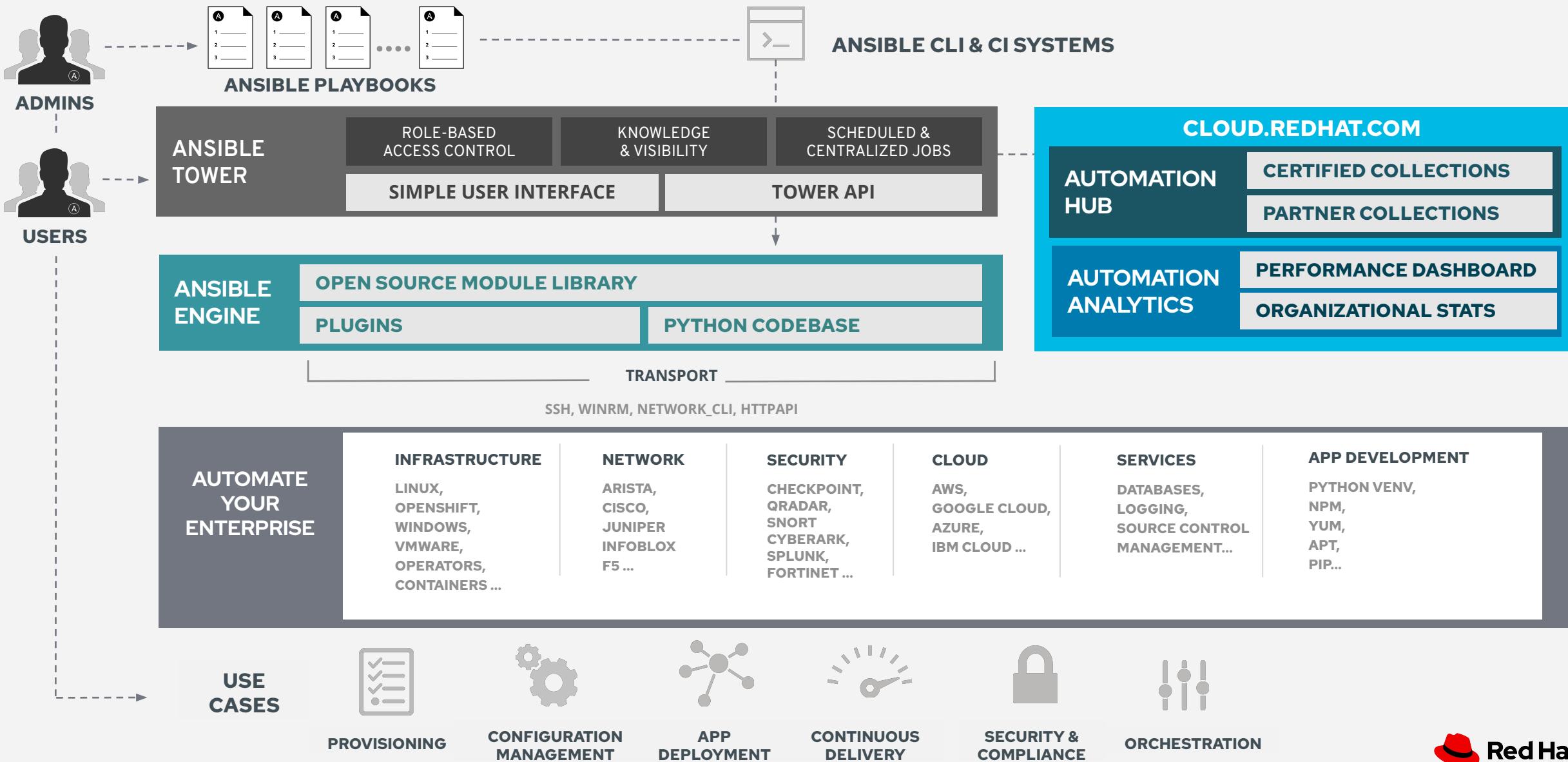
Centralized logging

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

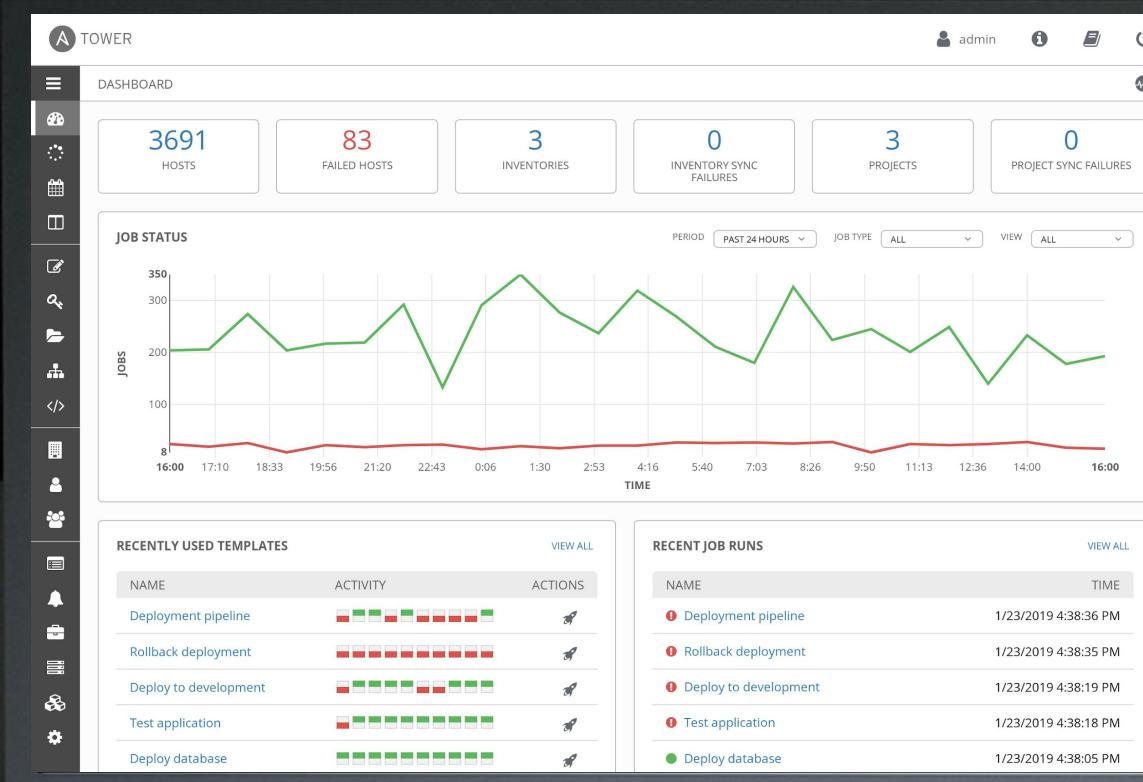




Red Hat Ansible Tower

FEATURE OVERVIEW:

Job Template



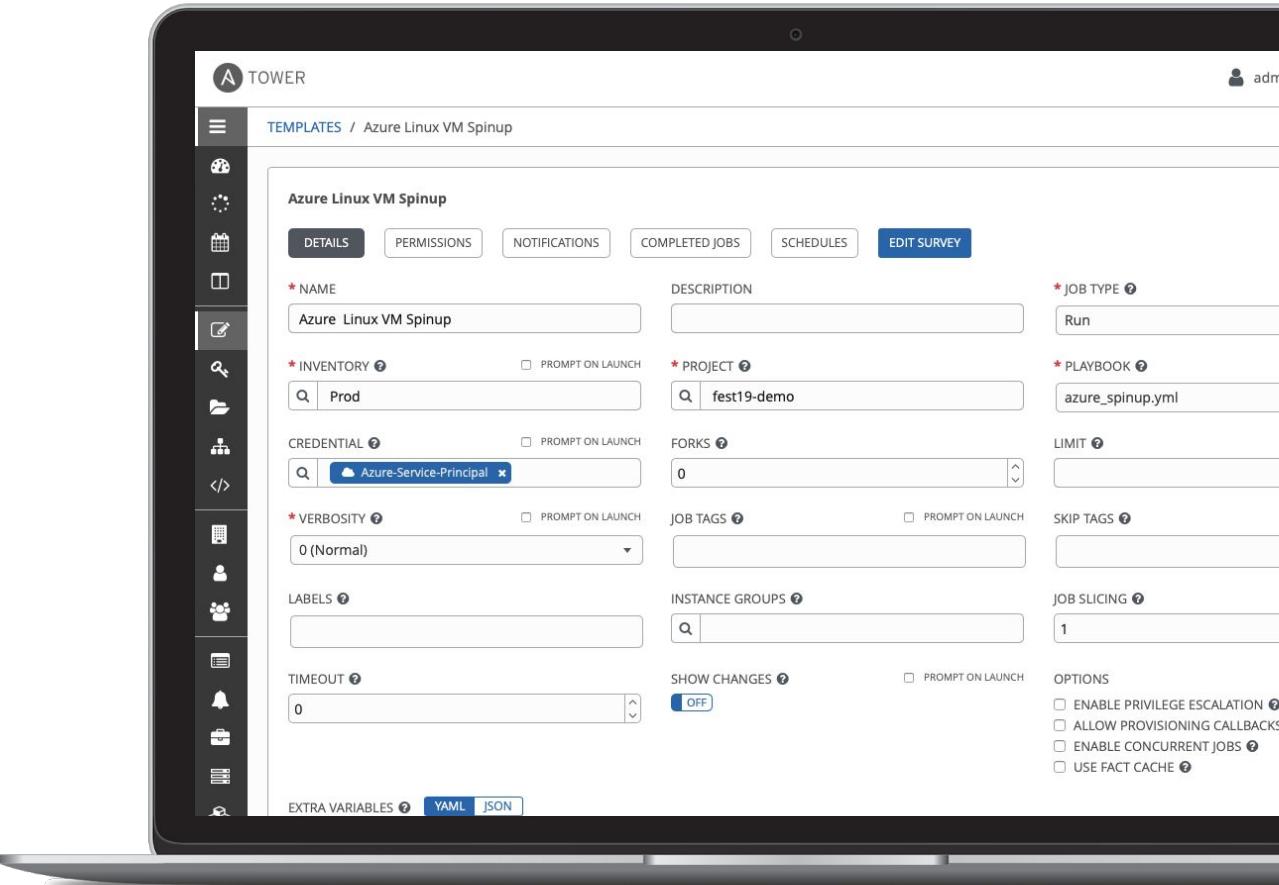
Job Templates

Everything in Ansible Tower revolves around the concept of a **Job Template**. Job Templates allow Ansible Playbooks to be controlled, delegated and scaled for an organization.

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

A **Job Template** requires:

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



Inventory

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

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

The screenshot shows the Ansible Tower interface for managing inventories. The top navigation bar includes 'INVENTORIES / Workshop Inventory / HOSTS'. On the left is a sidebar with various icons for managing projects, hosts, roles, variables, and more. The main content area displays the 'Workshop Inventory' with a list of hosts:

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

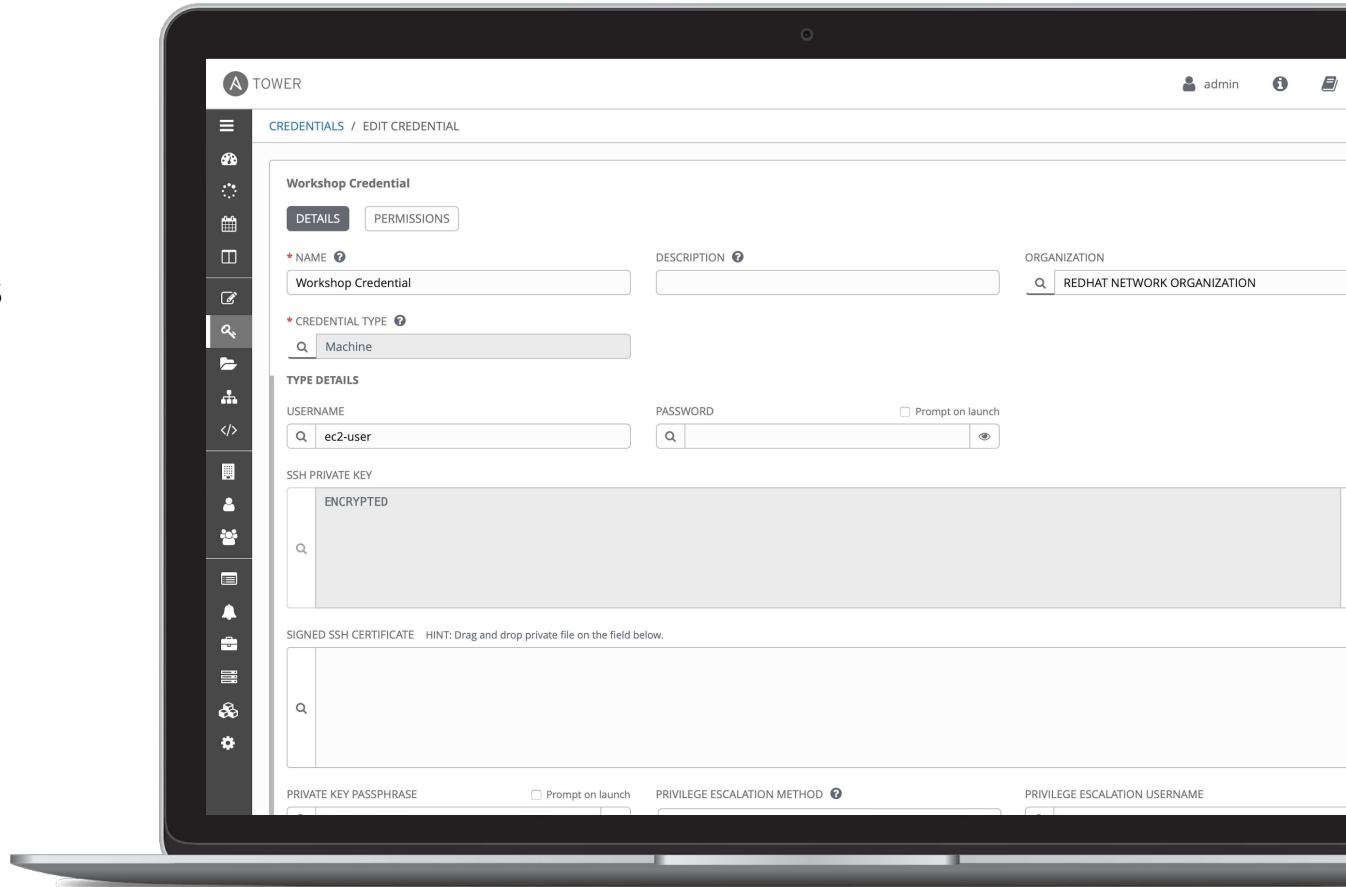
Below this, there's another section for managing inventories, with tabs for 'INVENTORIES' and 'HOSTS', and filters for 'SEARCH', 'NAME', 'TYPE', and 'ORGANIZATION'.

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

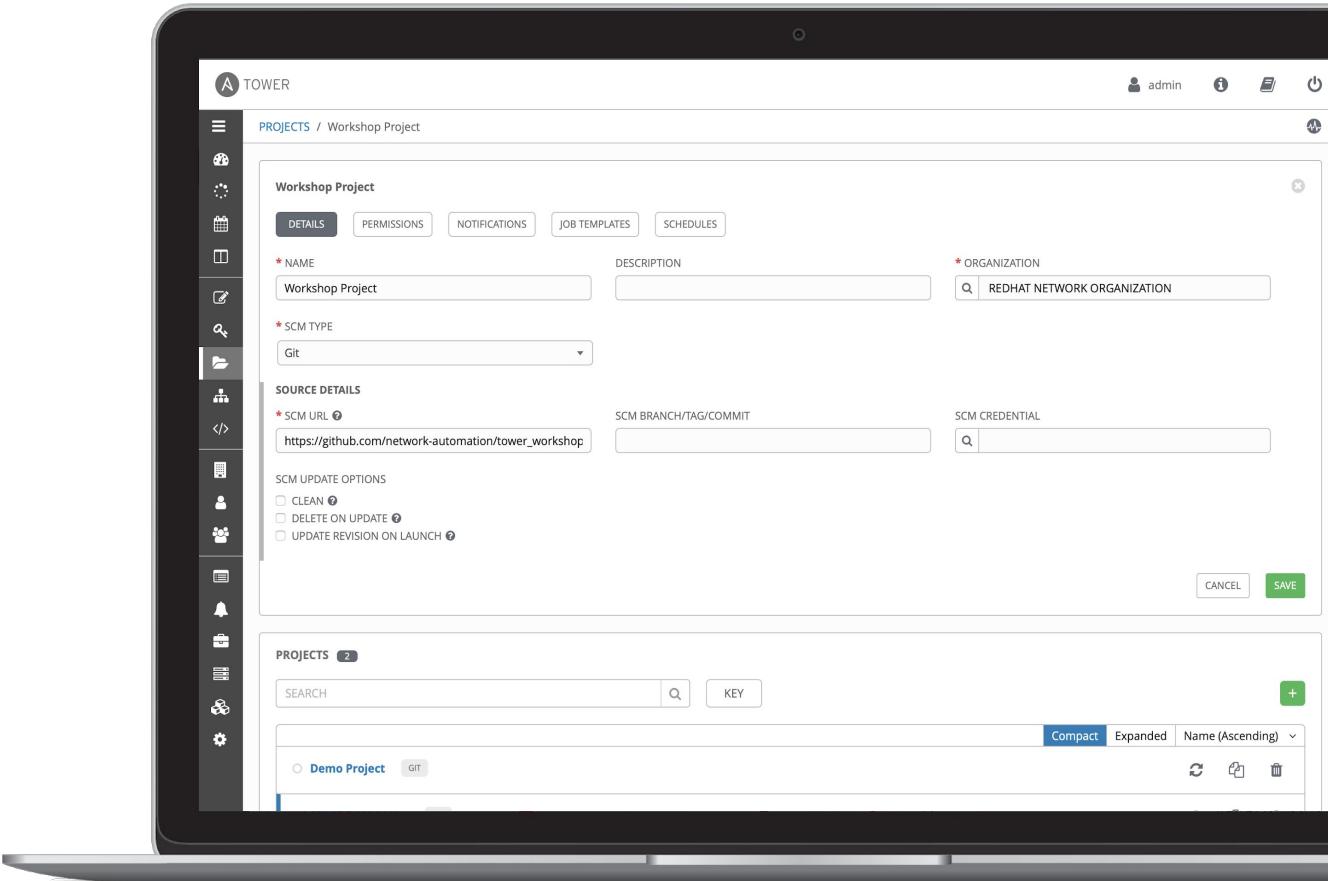
Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.



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 Platform

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 shows the Ansible Tower web interface. The left sidebar has a dark theme with white icons and labels. The 'RESOURCES' section is expanded, showing 'Templates' as the active item, which is highlighted with a light blue background. Other items in this section include 'Credentials', 'Projects', 'Inventories', 'Inventory Scripts', 'Organizations', 'Users', and 'Teams'. The main content area is titled 'TEMPLATES' and shows a list of six job templates: 'Demo Job Template', 'Network-Commands', 'Network-Restore', 'Network-System', 'Network-Time', and 'Network-User'. Each template entry includes a small thumbnail icon, the template name, a 'Job Template' badge, and three action icons: a rocket (Run), a document (Edit), and a trash can (Delete). Above the list are search and key filters, and below it are sorting options ('Compact', 'Expanded', 'Name (Ascending)'). The top right corner shows a user profile for 'admin' and various system status icons. At the bottom right, there is a note 'ITEMS 1 - 6'.

Executing an existing Job Template

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



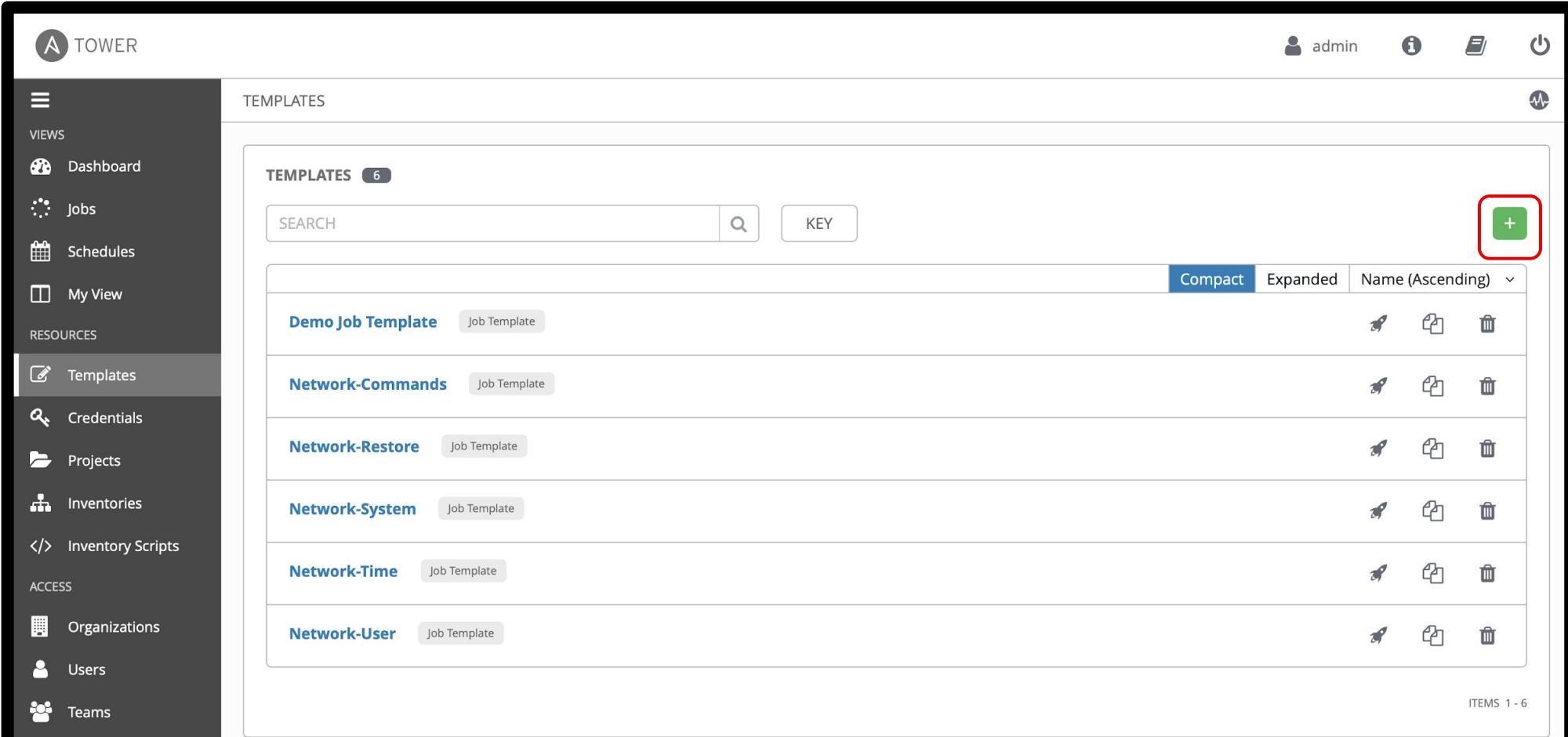
The screenshot shows the Tower interface with the 'TEMPLATES' view selected. The left sidebar includes options like 'Dashboard', 'Jobs', 'Schedules', 'My View', 'Templates' (which is currently selected), 'Credentials', 'Projects', 'Inventories', 'Inventory Scripts', 'Organizations', 'Users', and 'Teams'. The main area displays a list of six job templates:

Template Name	Type	Action Icons
Demo Job Template	Job Template	Rocketship, Copy, Delete
Network-Commands	Job Template	Rocketship, Copy, Delete
Network-Restore	Job Template	Rocketship, Copy, Delete
Network-System	Job Template	Rocketship, Copy, Delete
Network-Time	Job Template	Rocketship, Copy, Delete
Network-User	Job Template	Rocketship, Copy, Delete

A red box highlights the 'Rocketship' icon in the first row's actions column. The bottom right corner of the interface shows '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 Ansible 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 'Templates' link is currently selected. The main content area is titled 'TEMPLATES' and shows a list of six existing job templates: 'Demo Job Template', 'Network-Commands', 'Network-Restore', 'Network-System', 'Network-Time', and 'Network-User'. Each template entry includes a 'Job Template' badge, a search bar, and three icons for edit, copy, and delete. In the top right corner of the template list area, there is a green button with a white plus sign (+). This button is highlighted with a red square, indicating it is the target for creating a new job template. The top right of the interface also features user information ('admin') and system status icons.

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.

The screenshot shows the 'New Job Template' configuration window. The left sidebar contains navigation links for Views (Dashboard, Jobs, Schedules, My View), Resources (Templates, Credentials, Projects, Inventories, Inventory Scripts), Access (Organizations, Users, Teams), and Administration. The 'Templates' link is currently selected. The main window has tabs for DETAILS, PERMISSIONS, COMPLETED JOBS, SCHEDULES, and ADD SURVEY. The DETAILS tab is active. It includes fields for NAME, DESCRIPTION, and JOB TYPE (Run). Other sections include INVENTORY, PROJECT, PLAYBOOK, CREDENTIAL, FORKS, LIMIT, VERBOSITY, JOB TAGS, SKIP TAGS, LABELS, INSTANCE GROUPS, JOB SLICING, TIMEOUT, SHOW CHANGES, and OPTIONS (Enable Privilege Escalation, Allow Provisioning Callbacks).

NEW JOB TEMPLATE

DETAILS PERMISSIONS COMPLETED JOBS SCHEDULES ADD SURVEY

* NAME

DESCRIPTION

* JOB TYPE Run

* INVENTORY PROMPT ON LAUNCH

* PROJECT PROMPT ON LAUNCH

* PLAYBOOK Choose a playbook

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 PROMPT ON LAUNCH

JOB SLICING 1

TIMEOUT PROMPT ON LAUNCH

SHOW CHANGES OFF PROMPT ON LAUNCH

OPTIONS

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.

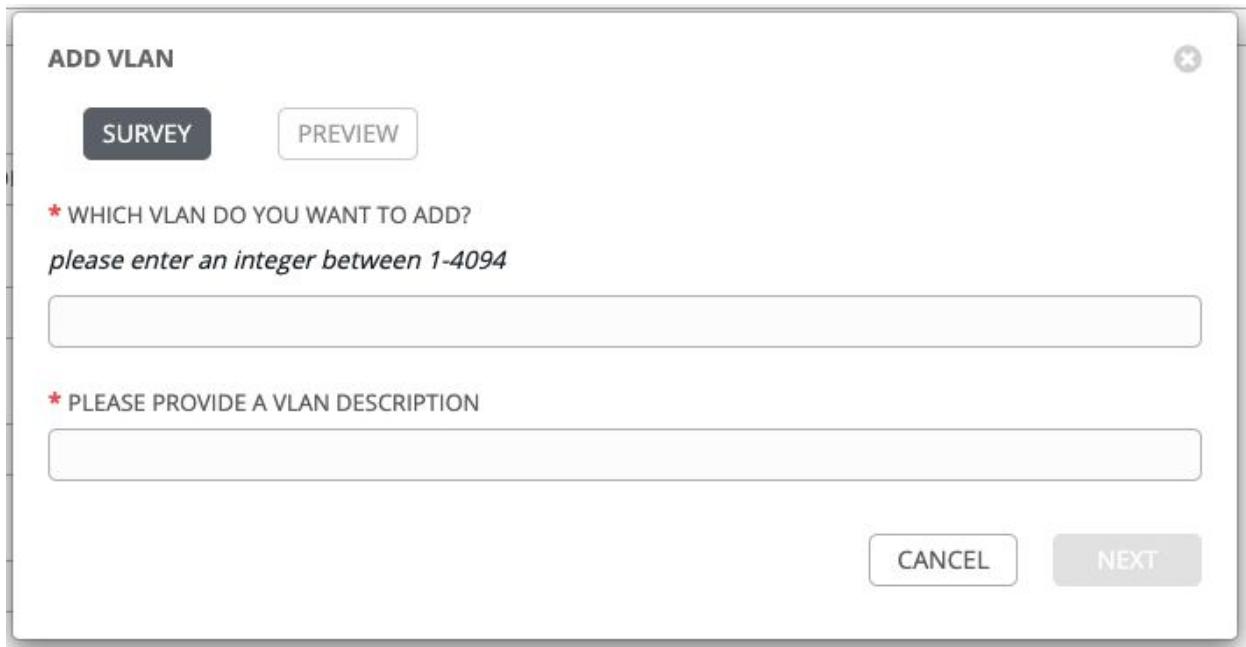
ADD VLAN

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

ADD SURVEY

Click the button to open the Add Survey window.

The screenshot shows the Ansible Tower web interface. On the left is a dark sidebar with navigation links: Views, Dashboard, Jobs, Schedules, My View, Resources, Templates (which is selected), Credentials, Projects, Inventories, Inventory Scripts, Access, and Organizations. The main content area has a title 'TEMPLATES / Configure Banner'. A modal window titled 'Configure Banner' is open. Inside the modal, there are several configuration sections: 'DETAILS' (selected), 'PERMISSIONS', 'NOTIFICATIONS', 'COMPLETED JOBS', 'SCHEDULES', and 'EDIT SURVEY' (which is highlighted with a red rectangle). Below these are fields for 'NAME' ('Configure Banner'), 'DESCRIPTION', 'JOB TYPE' ('Run'), 'INVENTORY' ('Workshop Inventory'), 'PROJECT' ('Workshop Project'), 'PLAYBOOK' ('network_banner.yml'), 'CREDENTIAL' ('Workshop Credential'), 'FORKS' ('0'), 'LIMIT' (empty), 'VERBOSITY' ('0 (Normal)'), 'JOB TAGS' (empty), 'SKIP TAGS' (empty), and 'LABELS' (empty). There are also 'PROMPT ON LAUNCH' checkboxes for various parameters.

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 shows the 'Edit Survey Prompt' configuration window. It includes fields for 'PROMPT' (containing 'Please enter the banner text'), 'DESCRIPTION' (containing 'Please type into the text field the desired banner'), 'ANSWER VARIABLE NAME' (set to 'net_banner'), 'ANSWER TYPE' (set to 'Textarea'), 'MINIMUM LENGTH' (set to 0), 'MAXIMUM LENGTH' (set to 4096), and a 'DEFAULT ANSWER' field. A 'REQUIRED' checkbox is checked. On the right, a 'PREVIEW' section shows a placeholder text area with the prompt and description. The Red Hat logo is visible in the bottom right corner.

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

Red Hat

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 shows the TOWER interface. On the left is a dark sidebar with various navigation options: Views, Dashboard, Jobs, Schedules, My View, Templates (which is selected), Credentials, Projects, Inventories, Inventory Scripts, Organizations, Users, and Teams. The main area is titled 'TEMPLATES' and lists four job templates: 'Network-Restore' (Job Template), 'Network-System' (Job Template), 'Network-Time' (Job Template), and 'Network-User' (Job Template). To the right of these templates is a 'CONFIGURE BANNER' dialog box. The dialog has tabs for 'SURVEY' (which is selected) and 'PREVIEW'. It contains a text field with the placeholder 'Please type into the text field the desired banner' and a note '* PLEASE ENTER THE BANNER TEXT'. Below the text field are 'CANCEL' and 'NEXT' buttons. To the right of the dialog is a list of items, each with a green '+' button, a name, and three icons: a rocket, a clipboard, and a trash can. The names listed are 'Network-Restore', 'Network-System', 'Network-Time', and 'Network-User'. At the bottom of the page, it says 'ITEMS 1 - 7'.



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)

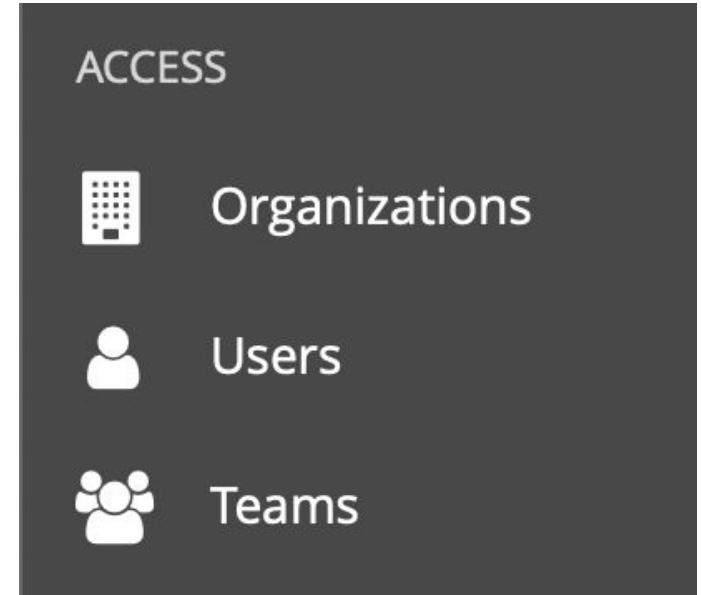
Role-Based Access Controls (RBAC) are built into Ansible Tower and allow administrators to delegate access to inventories, organizations, and more.

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



The screenshot shows the Red Hat Satellite 6 interface. On the left, there is a dark sidebar with various navigation options: Views (Dashboard, Jobs, Schedules, My View), Resources (Templates, Credentials, Projects, Inventories, Inventory Scripts), Access (Organizations, Users, Teams), and Administration (Administration). The "Organizations" option is highlighted with a light blue background. The main content area is titled "ORGANIZATIONS" and shows three organizations: "Default", "REDHAT COMPUTE ORGANIZATION", and "REDHAT NETWORK ORGANIZATION". Each organization card displays its name, a green "+" button to add more, and counts for Users, Teams, Inventories, Projects, Job Templates, and Admins. The "Default" organization has 0 users, 0 teams, 1 inventory, 1 project, 1 job template, and 0 admins. The "REDHAT COMPUTE ORGANIZATION" has 0 users, 2 teams, 0 inventories, 0 projects, 0 job templates, and 0 admins. The "REDHAT NETWORK ORGANIZATION" has 2 users, 2 teams, 1 inventory, 1 project, 6 job templates, and 1 admin. The top right of the interface shows the user "admin" and various system icons.

Organization	Users	Teams	Inventories	Projects	Job Templates	Admins
Default	0	0	1	1	1	0
REDHAT COMPUTE ORGANIZATION	0	2	0	0	0	0
REDHAT NETWORK ORGANIZATION	2	2	1	1	6	1

Viewing Teams

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



Teams

The screenshot shows the Ansible Tower web interface. On the left is a dark sidebar with various navigation options. At the bottom of this sidebar, the 'Teams' option is highlighted with a grey background. The main content area is titled 'TEAMS' and shows a list of four teams. Each team entry includes the name, organization, and actions (edit and delete). A green '+' button is available to add new teams. The interface is clean with a light grey background and white text for the main content area.

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

Viewing Users

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



in the left menu

The screenshot shows the Ansible Tower web interface. On the left, there is a dark sidebar with various navigation options: Views, Dashboard, Jobs, Schedules, My View, Templates, Credentials, Projects, Inventories, Inventory Scripts, Organizations, Users (which is highlighted in grey), and Teams. The main content area has a header bar with "TOWER" on the left, user information ("admin"), and a power button icon on the right. Below the header is a sub-header "USERS" with a count of "8". The main area is titled "USERS" and contains a table with the following data:

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	

At the bottom right of the table, it says "ITEMS 1 - 8".



Red Hat Ansible Automation Platform

Exercise 8 - Understanding RBAC

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

Approximate time: 15 mins



Red Hat



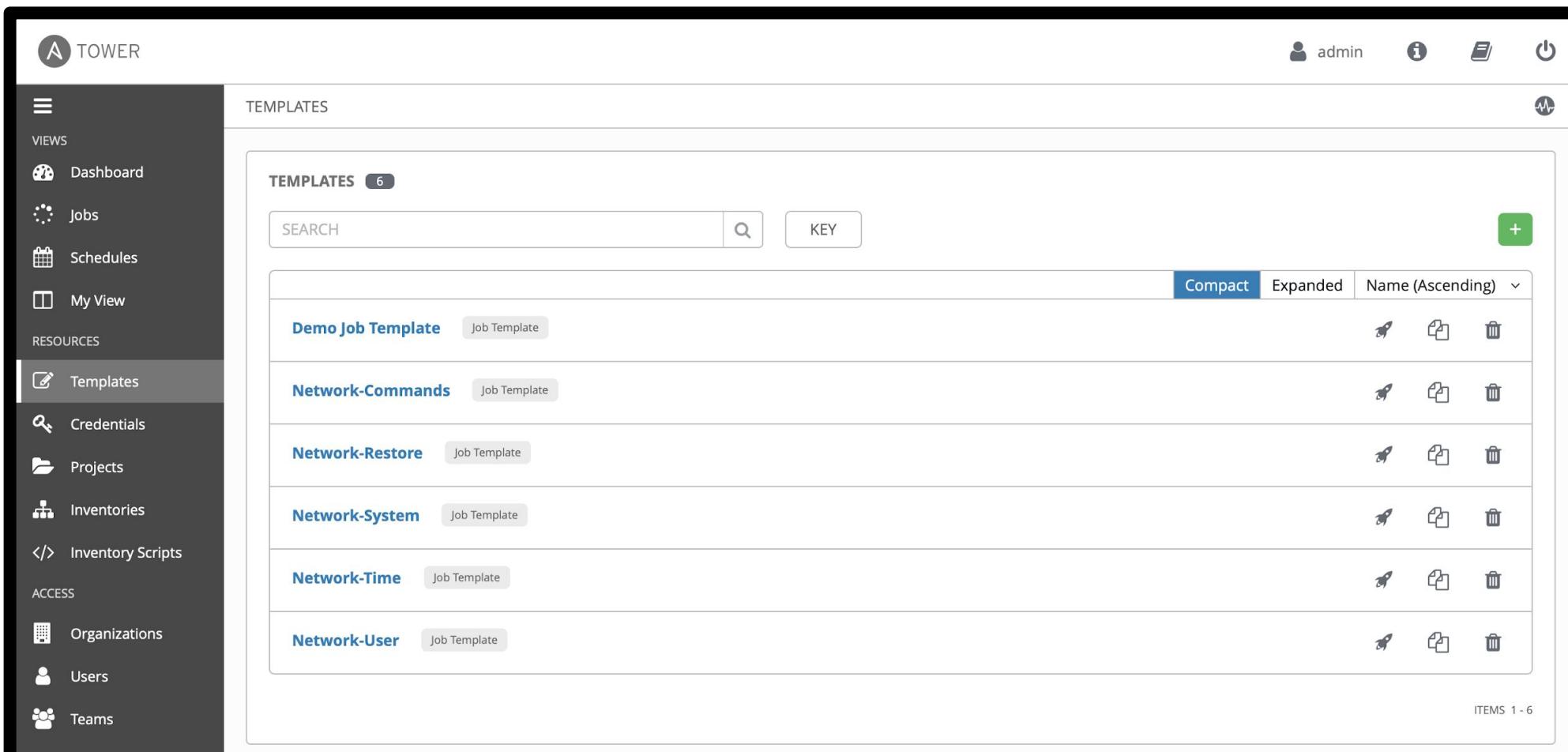
Exercise 9

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 shows the Ansible Tower web interface. The top navigation bar includes the TOWER logo, user info (admin), and various icons. The left sidebar has a dark theme with sections for VIEWS (Dashboard, Jobs, Schedules, My View), RESOURCES (Templates, Credentials, Projects, Inventories, Inventory Scripts), ACCESS (Organizations, Users, Teams), and ADMINISTRATION. The 'Templates' item is highlighted. The main content area is titled 'TEMPLATES' and shows a list of six items: 'Demo Job Template', 'Network-Commands', 'Network-Restore', 'Network-System', 'Network-Time', and 'Network-User', all categorized as 'Job Template'. Each item has a 'Run', 'Edit', and 'Delete' icon. The interface uses a clean, modern design with a light background and blue accents for selected items.

Template Name	Type	Actions
Demo Job Template	Job Template	Run Edit Delete
Network-Commands	Job Template	Run Edit Delete
Network-Restore	Job Template	Run Edit Delete
Network-System	Job Template	Run Edit Delete
Network-Time	Job Template	Run Edit Delete
Network-User	Job Template	Run Edit Delete

Adding a new Workflow Template

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

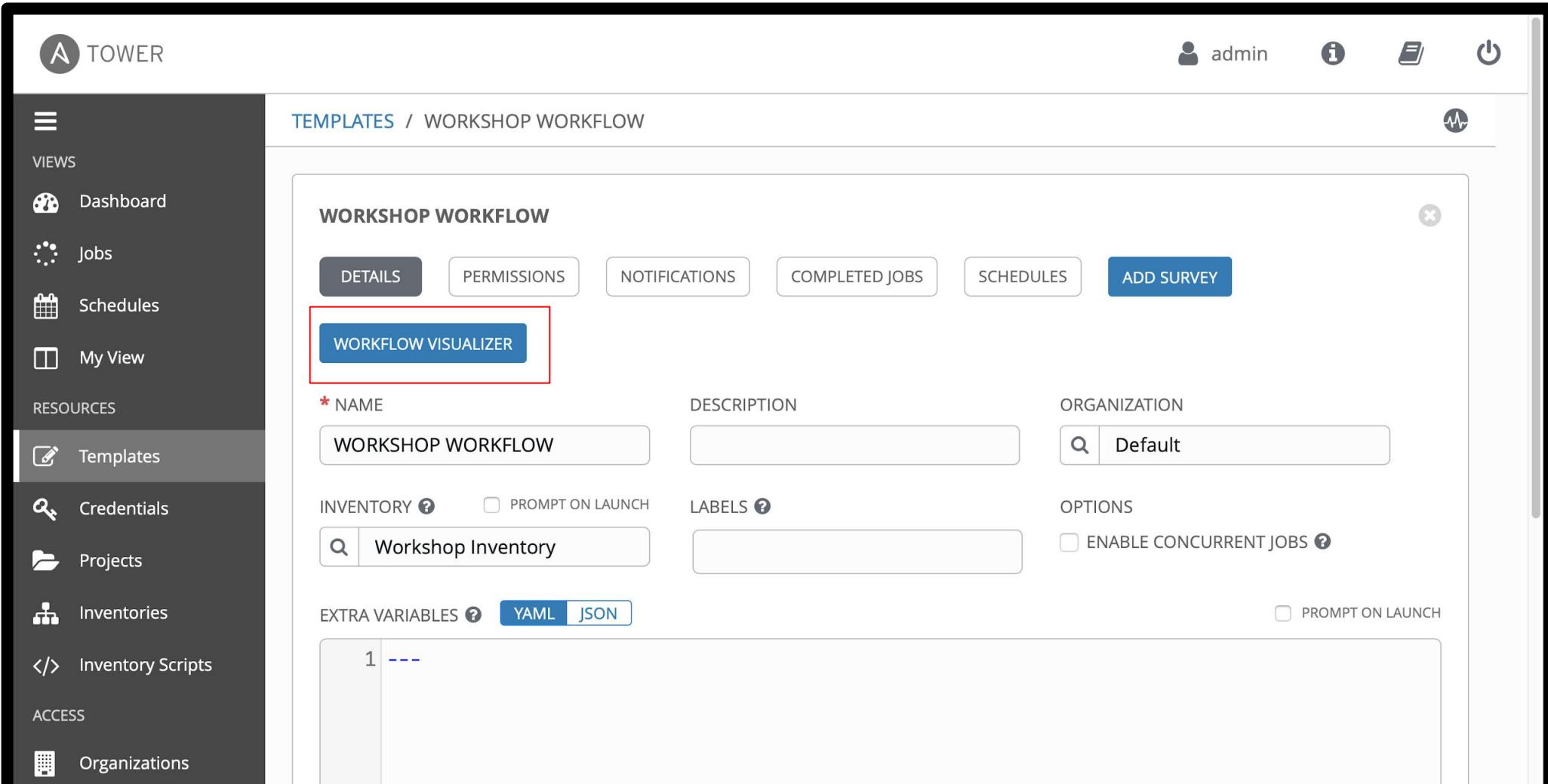


This time select the **Workflow Template**

A screenshot of the Ansible Tower web interface. The left sidebar shows navigation options like Dashboard, Jobs, Schedules, My View, Templates (which is selected), Credentials, Projects, Inventories, Inventory Scripts, Organizations, and Users. The main content area is titled 'TEMPLATES' and shows a list of templates. Each template entry includes a name, a 'Job Template' badge, a preview icon, and three action icons (rocket, copy, delete). A red box highlights the 'Workflow Template' option in the dropdown menu that appears when clicking the green '+' button. The dropdown also lists 'Job Template'. The interface has a dark mode theme with light-colored cards for each 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.



The screenshot shows the Tower interface for creating a new workflow template. The left sidebar has 'Templates' selected. The main area shows the 'WORKSHOP WORKFLOW' template details. The 'WORKFLOW VISUALIZER' button is highlighted with a red box.

WORKSHOP WORKFLOW

DETAILS **PERMISSIONS** **NOTIFICATIONS** **COMPLETED JOBS** **SCHEDULES** **ADD SURVEY**

WORKFLOW VISUALIZER

*** NAME**: WORKSHOP WORKFLOW

DESCRIPTION: (empty)

ORGANIZATION: Default

INVENTORY: Workshop Inventory

LABELS: (empty)

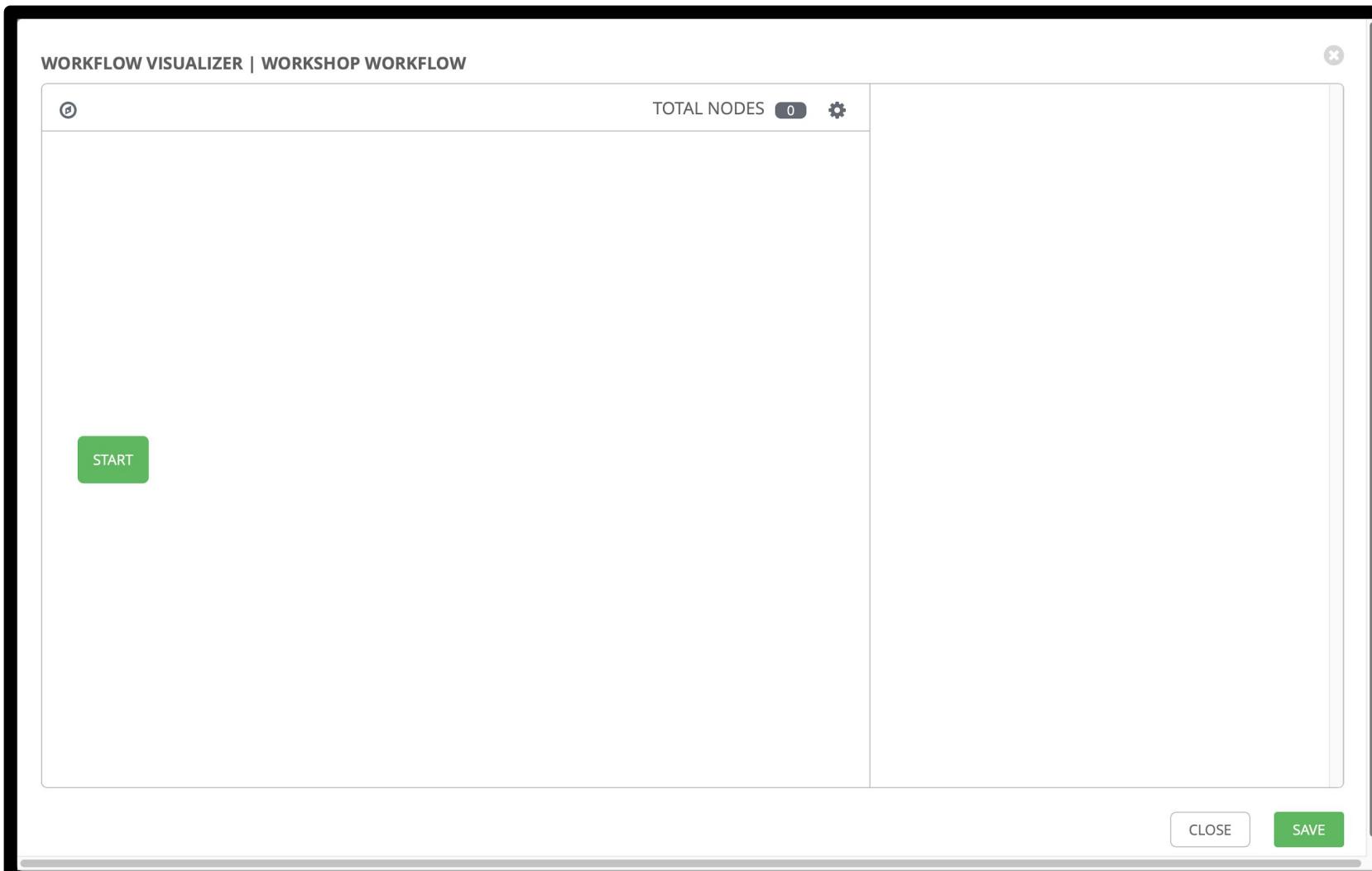
OPTIONS: ENABLE CONCURRENT JOBS

EXTRA VARIABLES: (empty)

YAML **JSON**

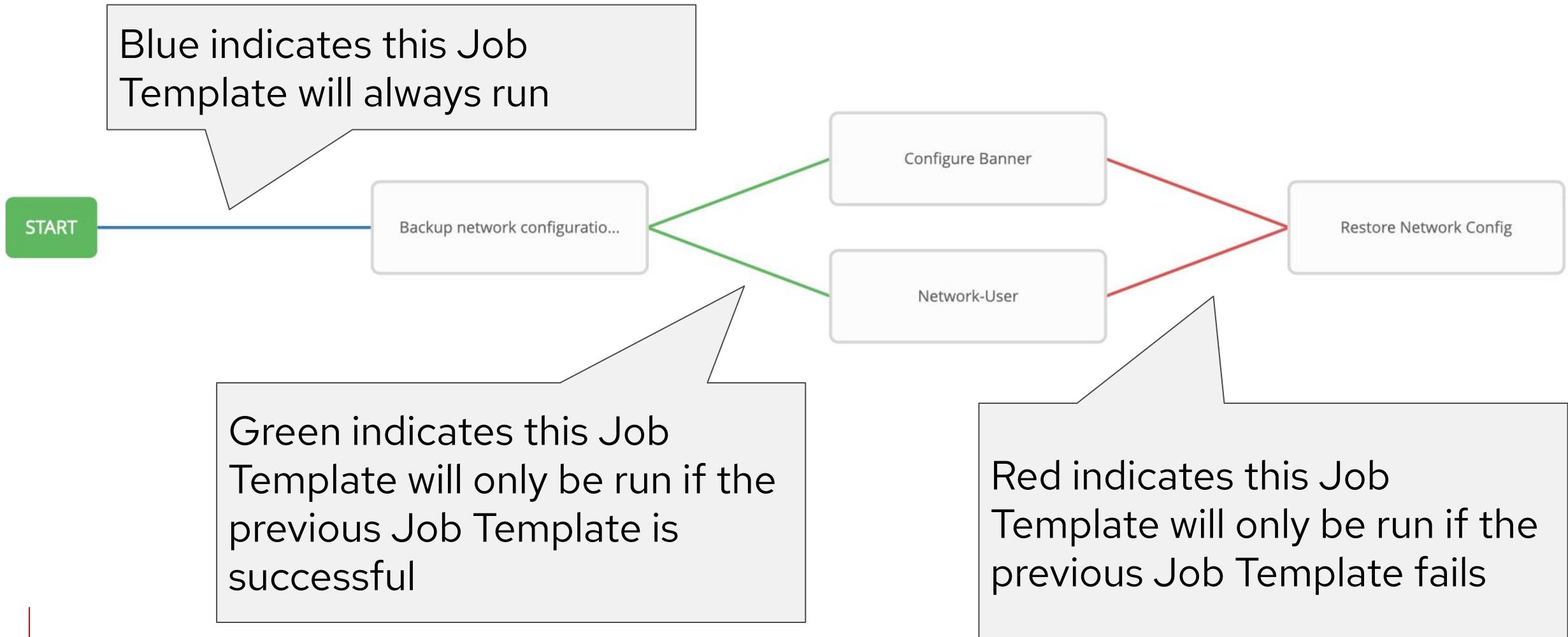
Workflow Visualizer

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>

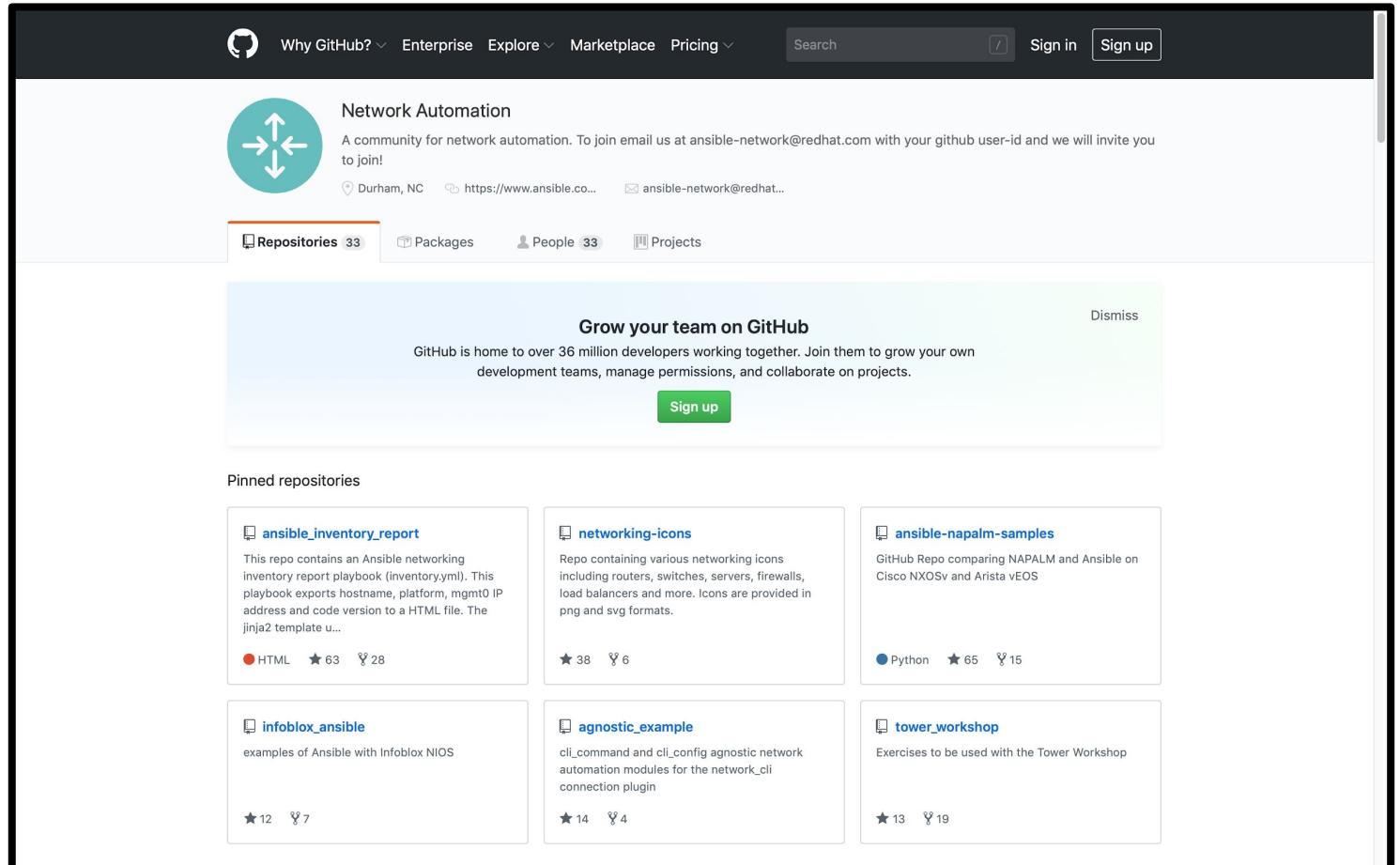
- **IRC**

#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





AnsibleFest

October 13-14, 2020 | Virtual Experience



Thank you



linkedin.com/company/red-hat



youtube.com/AnsibleAutomation



facebook.com/ansibleautomation



twitter.com/ansible



github.com/ansible