

HOW TO USE THIS DECK

This slide deck is meant to accompany the Ansible RHEL workshop, both sections if needed.

Note that this deck is optional - the workshop content explains each and every Ansible idea in detail already.

HOW TO IMPROVE THIS DECK

The workshop is a collaborative effort. Help us to improve it! You can leave comments, and the BU will make sure to work on this. Tag for example Roland (Wolters) or Sean (Cavanaugh) to ensure that they pick it up.

Speaking about the BU: the fact that this deck is now owned by an organization and not individuals anymore hopefully ensures for the future that the deck stays up2date over time as the workshop develops.

THANKS

HUGE THANK YOU to the following people - without them, this deck would not have been possible.

First and foremost, thanks to:

Kevin “GoKEV” Holmes

He did the base work for this slide deck by migrating everything from ansible.red, and his fingerprint shows almost on each and every slide. Thank you so much for your cooperation and helping us and of course for submitting this in the first place.

But others should not go unmentioned:

Russell Pavlicek Matt St Onge Will Tome Götz

Thanks for providing input, helping proofread, error check, and keep Kev smiling when he needed to.



Red Hat

Ansible Automation Platform

Ansible Linux Automation Workshop

Introduction to Ansible for Red Hat Enterprise Linux Automation
for System Administrators and Operators



Housekeeping

- Timing
- Breaks
- Takeaways

What you will learn

- Introduction to Ansible Automation
- How it works
- Understanding modules, tasks & playbooks
- How to execute Ansible commands
- Using variables & templates
- Tower - where it fits in
- Basic usage of Tower
- Learn major Tower features: RBAC, workflows and so on

Introduction

Topics Covered:

- What is the Ansible Automation Platform?
- What can it do?



Red Hat
Ansible Automation
Platform



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

Teams are automating...



Lines Of Business



Network



Security



Operations

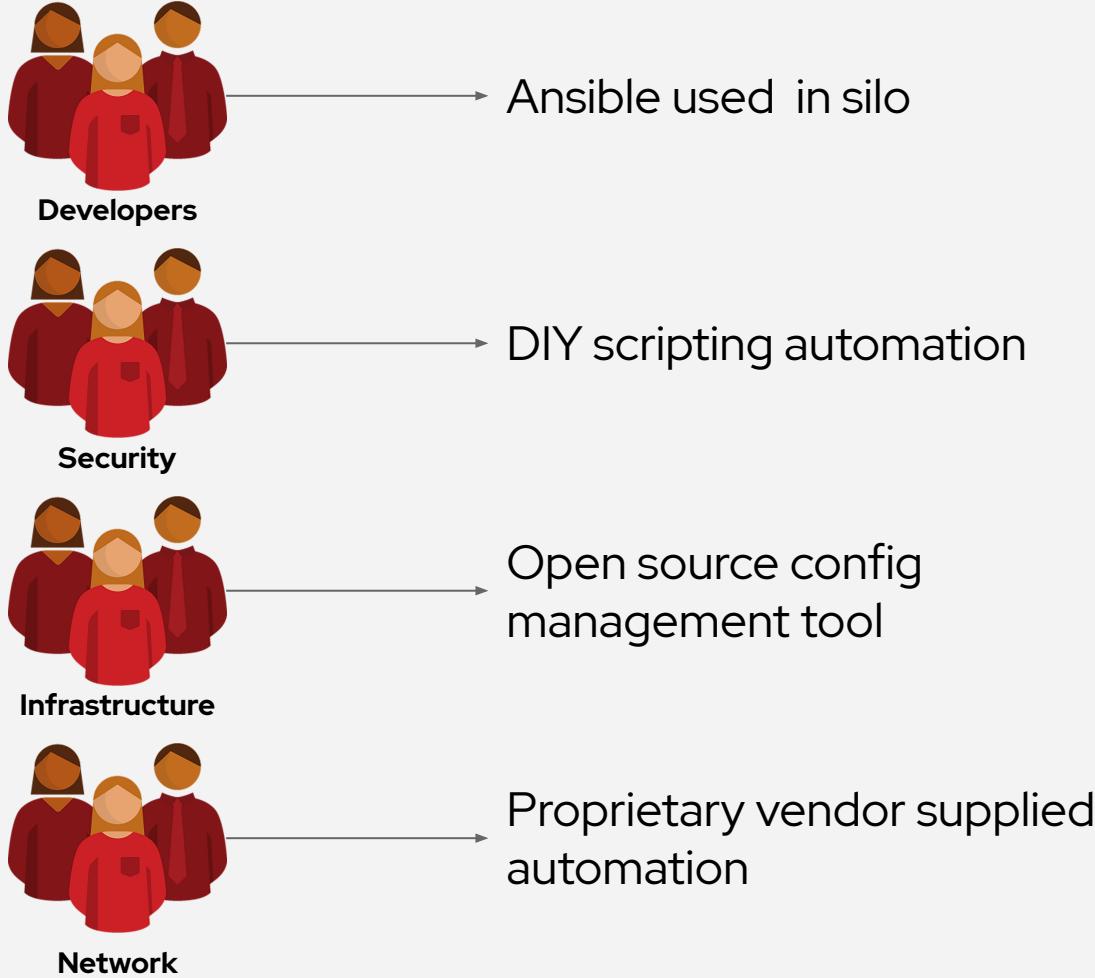


Developers



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

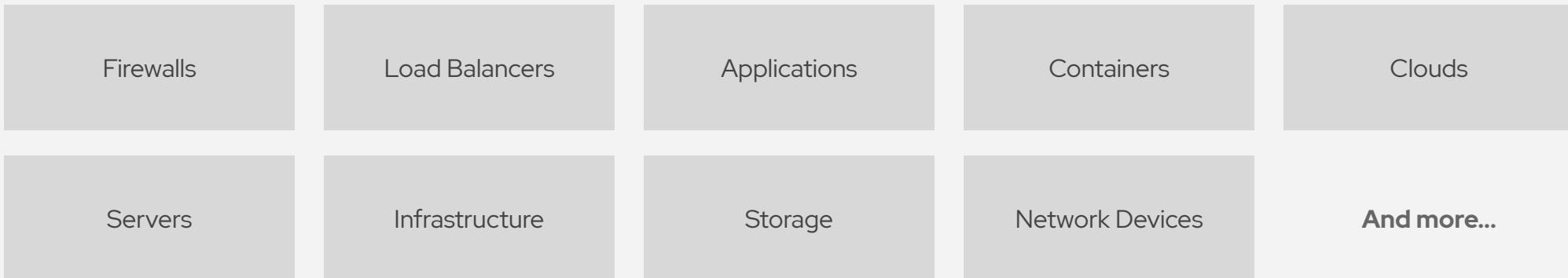
What can I do using Ansible?

Automate the deployment and management of your entire IT footprint.

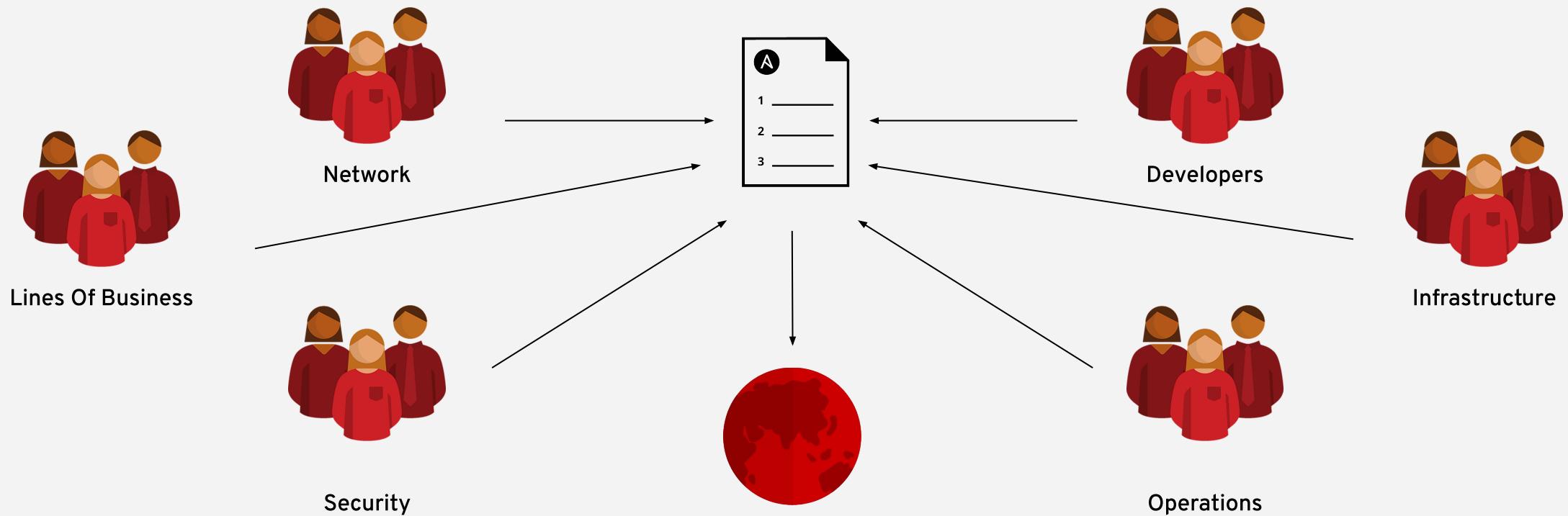
Do this...



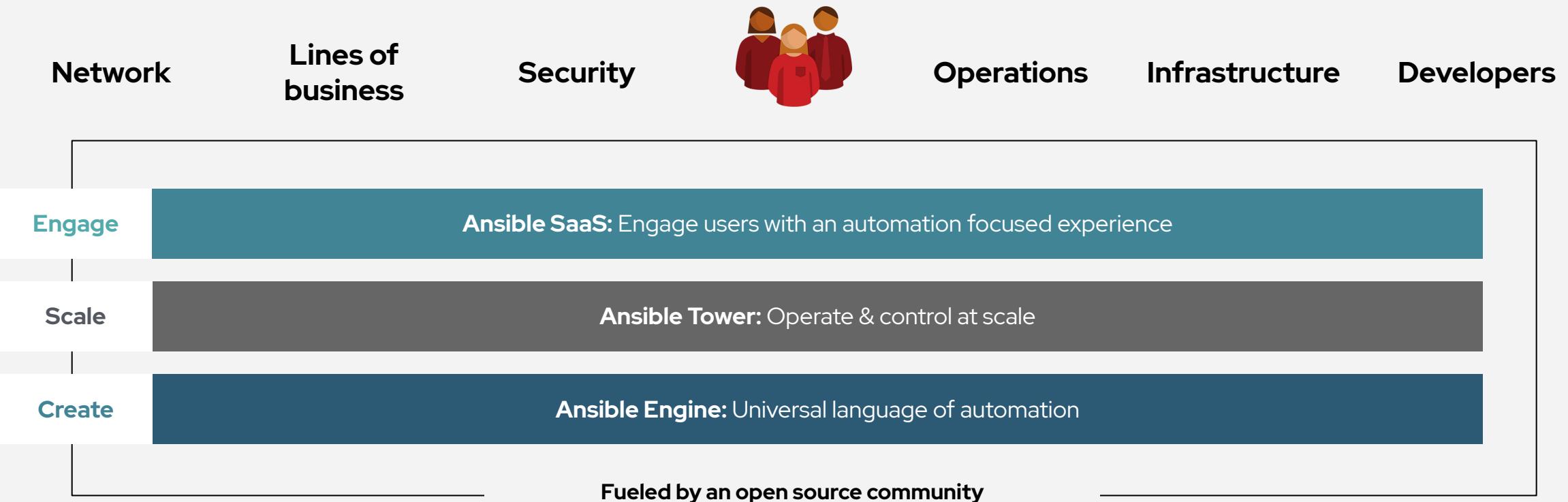
On these...



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



Section 1

Ansible Engine



Red Hat
Ansible Automation
Platform

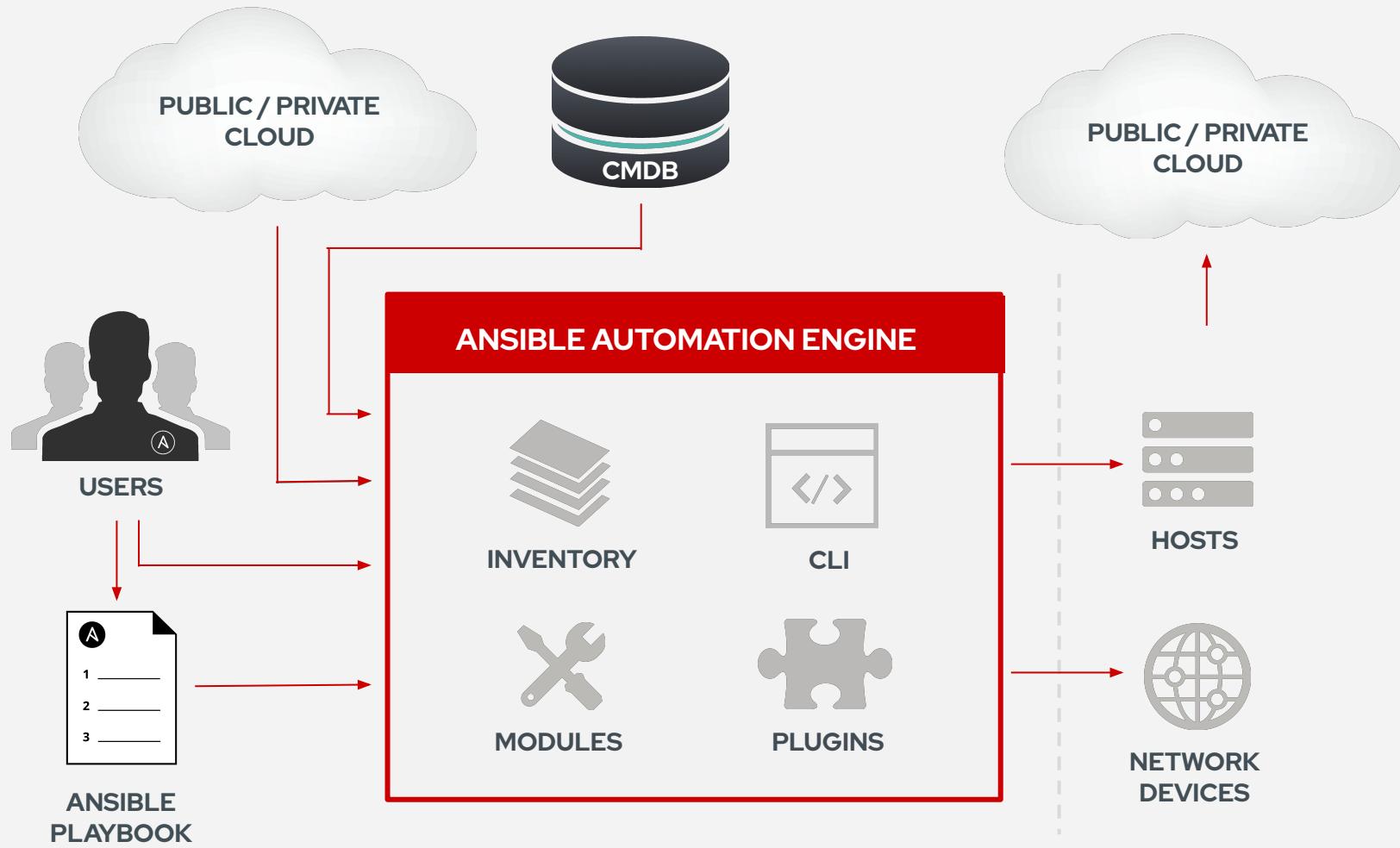
Exercise 1.1

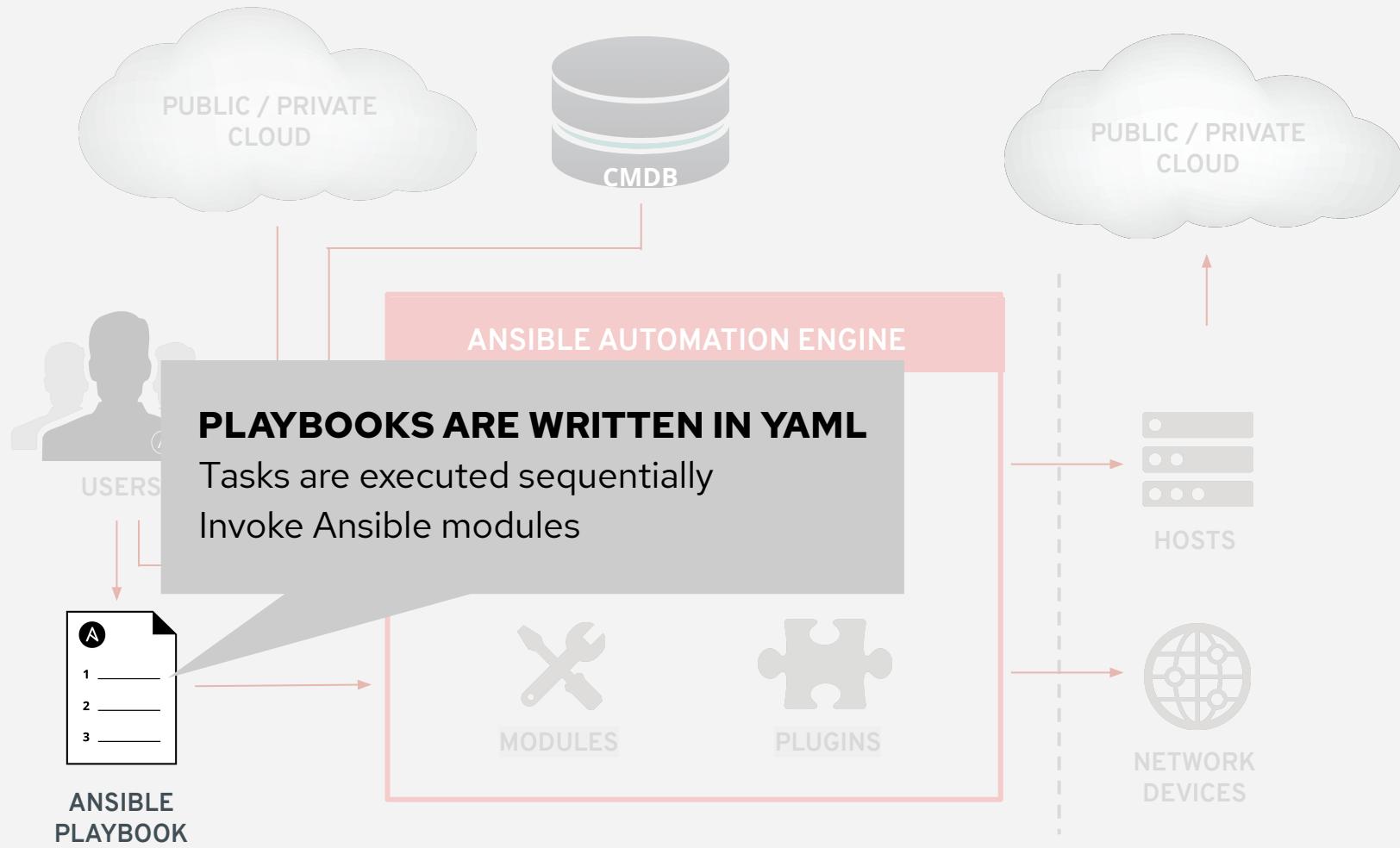
Topics Covered:

- Understanding the Ansible Infrastructure
- Check the prerequisites



Red Hat
Ansible Automation
Platform



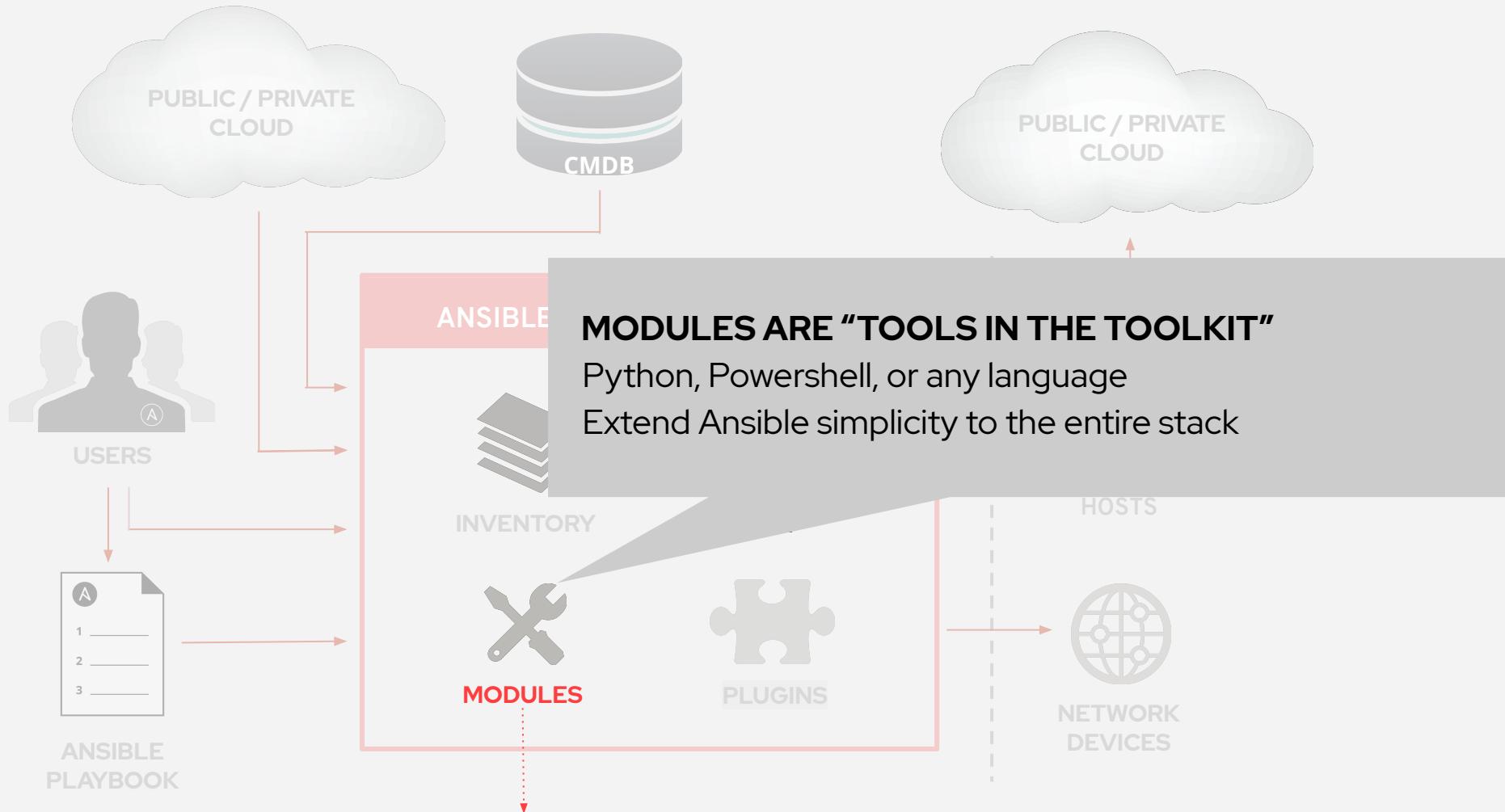


```
---
```

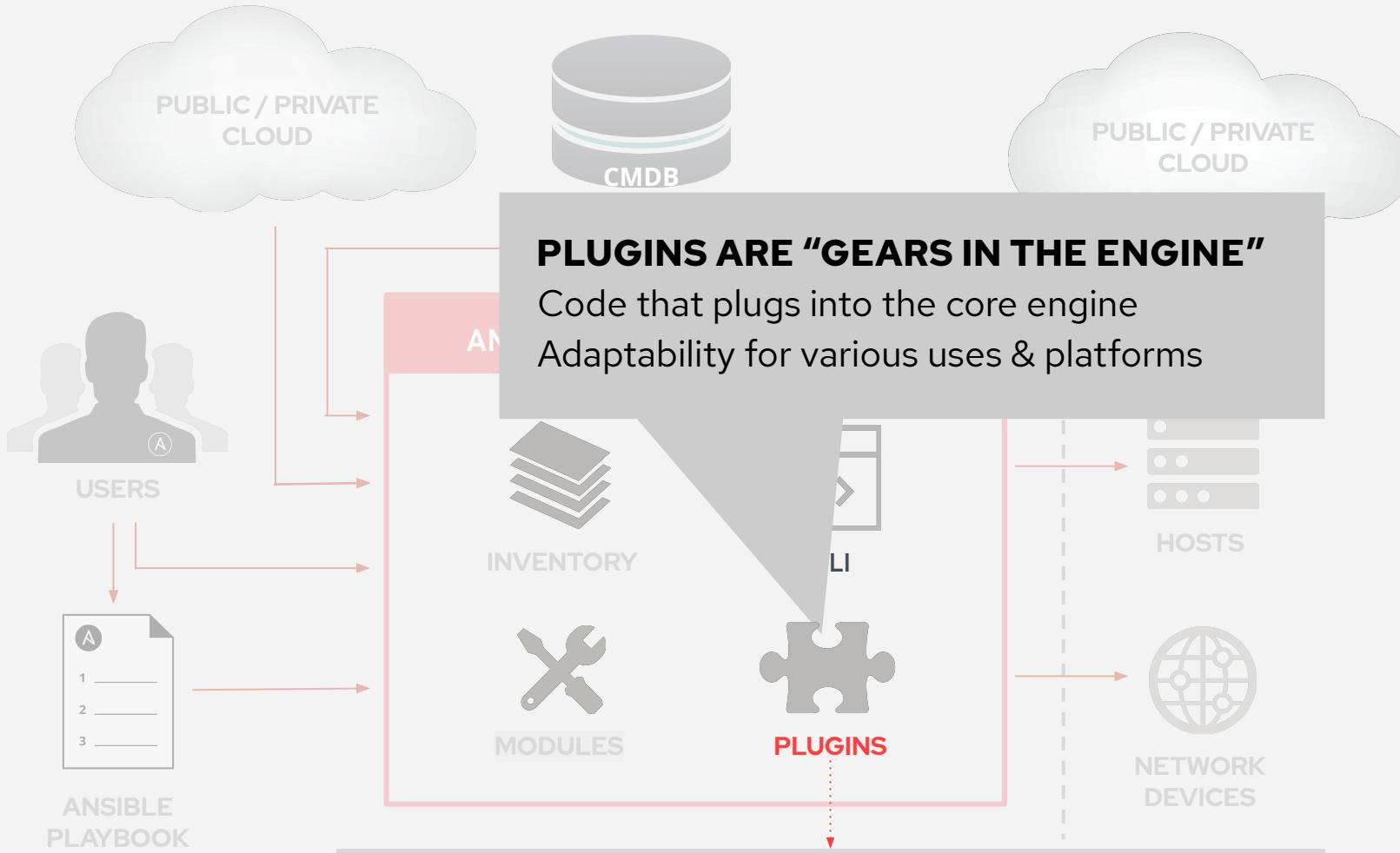
- **name: install and start apache**
hosts: web
become: yes

tasks:

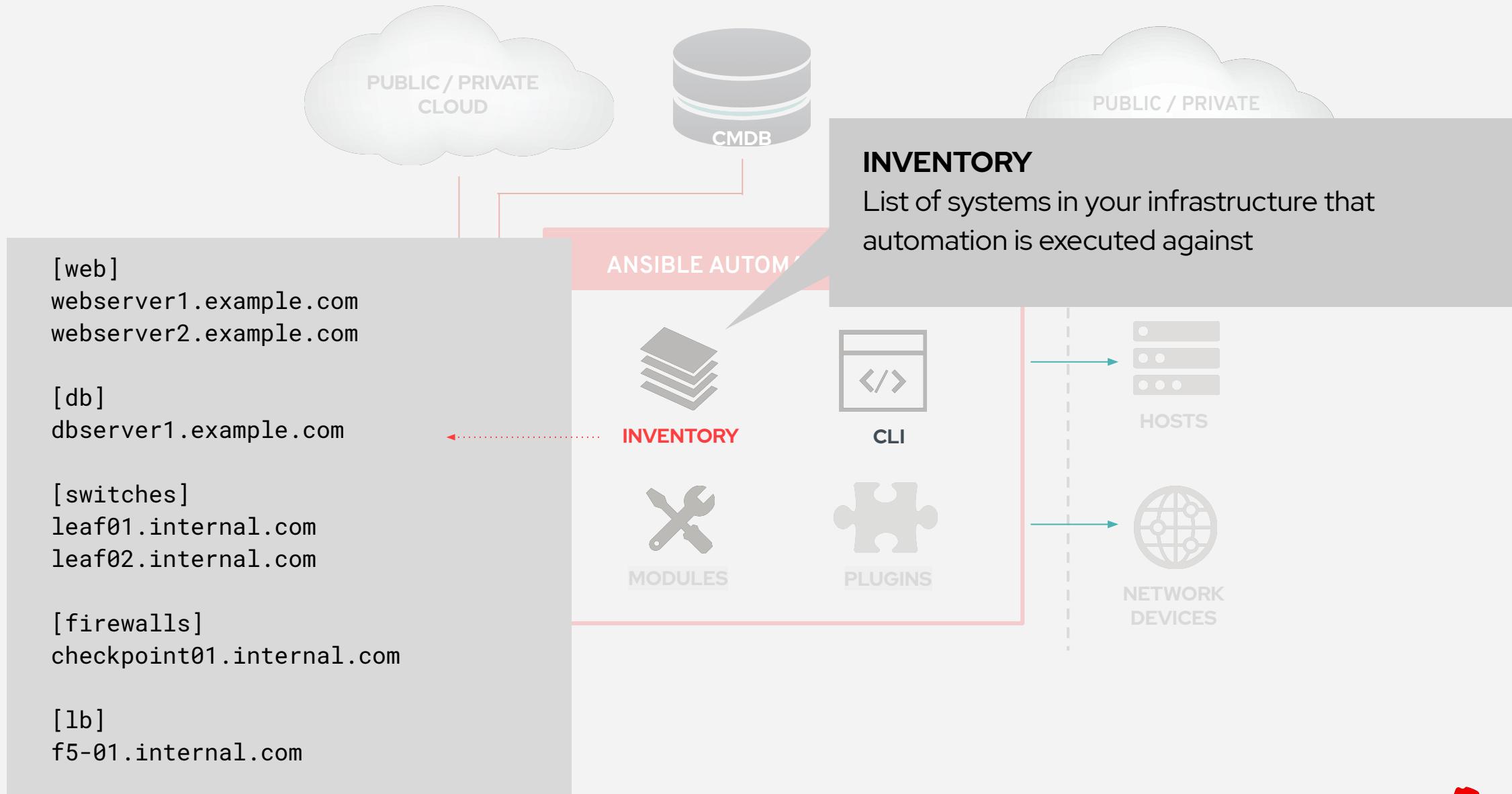
- **name: httpd package is present**
yum:
 name: httpd
 state: latest
- **name: latest index.html file is present**
template:
 src: files/index.html
 dest: /var/www/html/
- **name: httpd is started**
service:
 name: httpd
 state: started

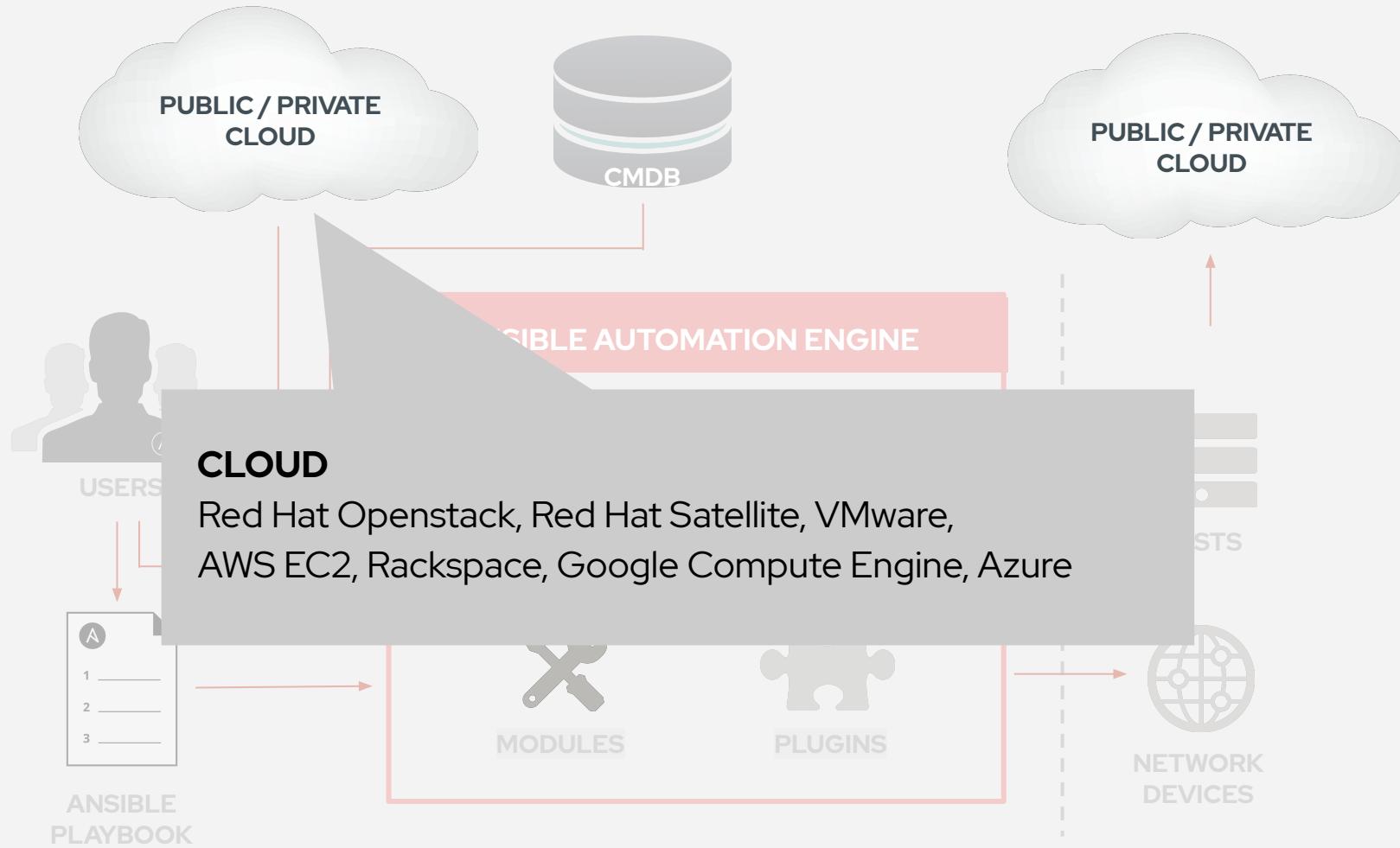


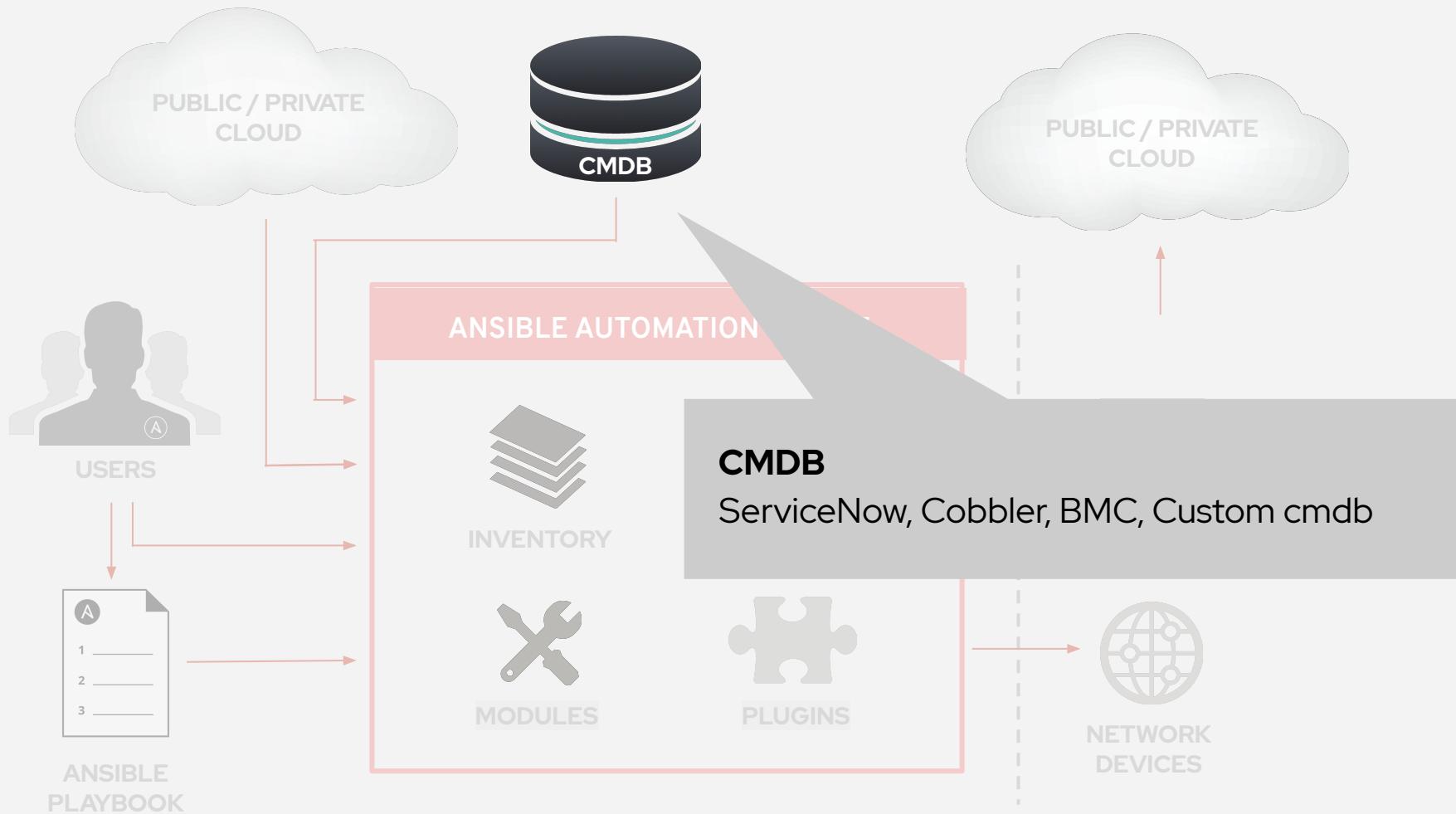
```
- name: latest index.html file is present
  template:
    src: files/index.html
    dest: /var/www/html/
```

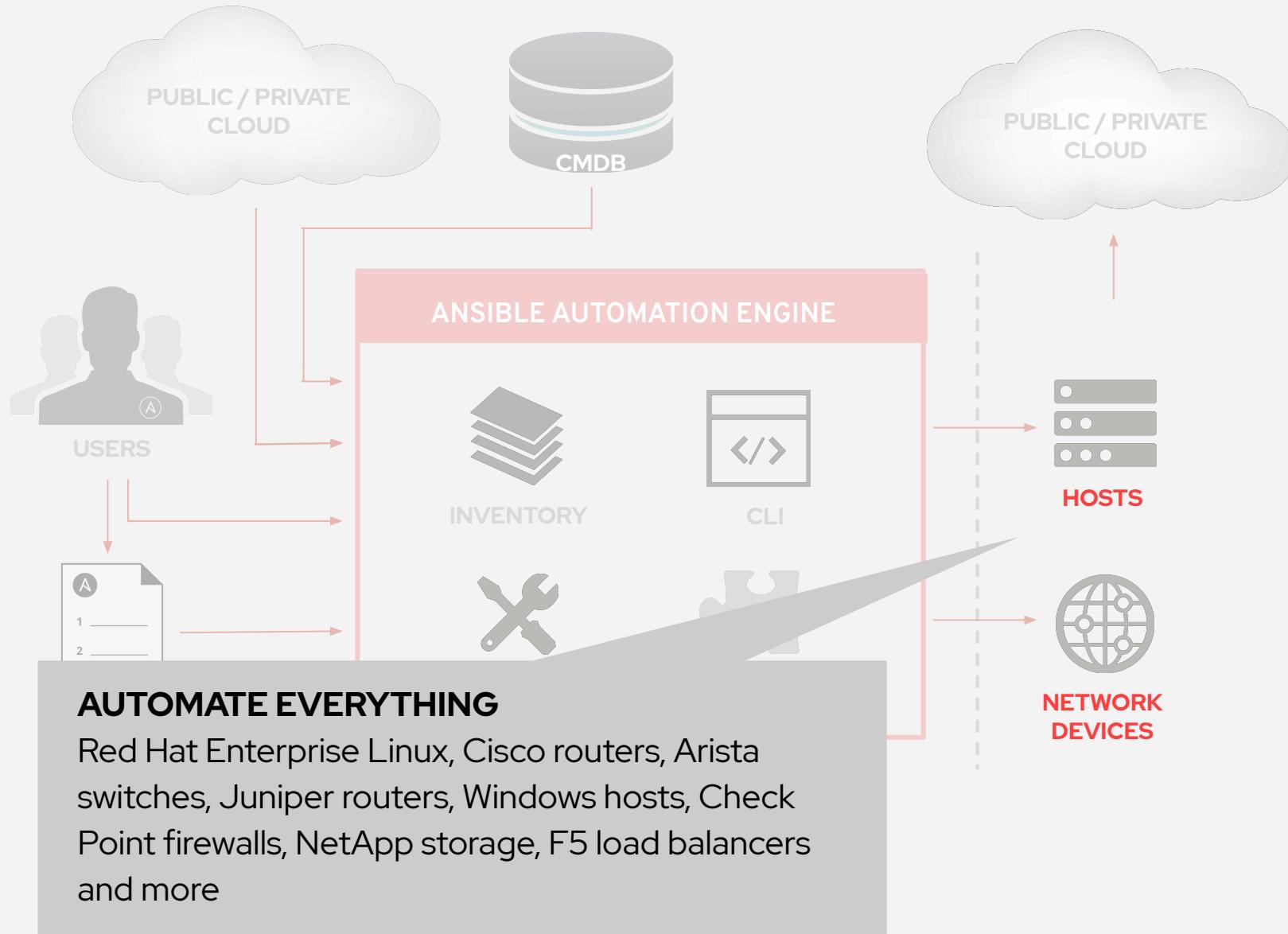


```
{ { some_variable | to_nice_yaml } }
```









LINUX AUTOMATION

150+
Linux Modules

**AUTOMATE EVERYTHING
LINUX**

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

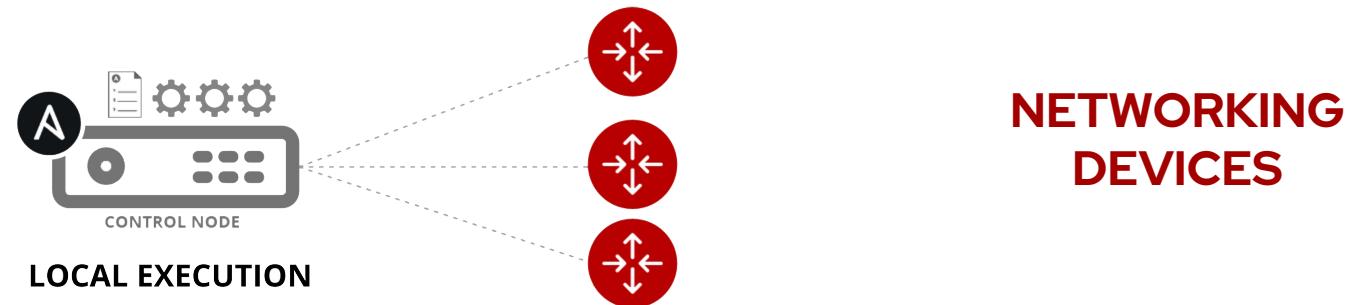
ONLY REQUIREMENTS:
Python 2 (2.6 or later)
or Python 3 (3.5 or later)

ansible.com/get-started



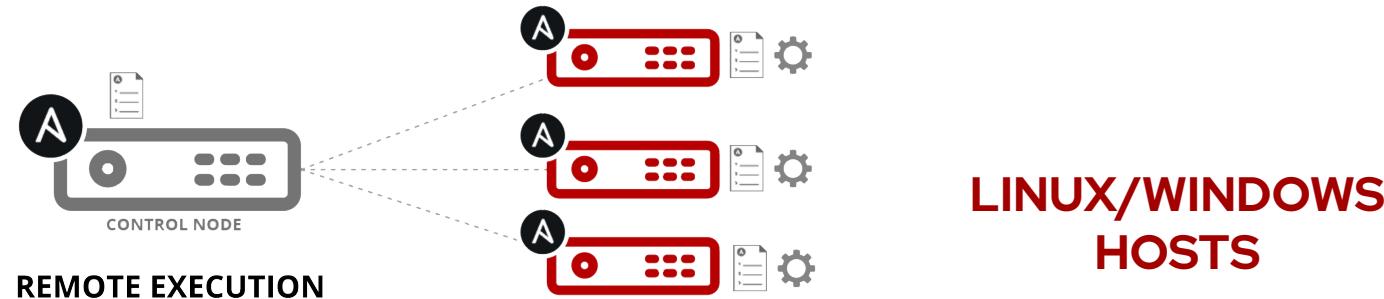
How Ansible Network Automation works

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**

Verify Access

- Follow the steps to access environment
- Use the IP provided to you, the script only has example IP
- Which editor do you use on command line?
If you don't know, we have a short intro



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 1.1 now in your lab environment



Red Hat



Red Hat
Ansible Automation
Platform

Exercise 1.2

Topics Covered:

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

Inventory

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

Understanding Inventory - Basic

```
# Static inventory example:  
[myservers]  
10.42.0.2  
10.42.0.6  
10.42.0.7  
10.42.0.8  
10.42.0.100  
host.example.com
```

Understanding Inventory - Basic

[app1srv]

```
appserver01 ansible_host=10.42.0.2  
appserver02 ansible_host=10.42.0.3
```

[web]

```
node-[1:30] ansible_host=10.42.0.[31:60]
```

[web:vars]

```
apache_listen_port=8080  
apache_root_path=/var/www/mywebdocs/
```

[all:vars]

```
ansible_user=kev  
ansible_ssh_private_key_file=/home/kev/.ssh/id_rsa
```

Understanding Inventory - Variables

[app1srv]

```
appserver01 ansible host=10.42.0.2  
appserver02 ansible_host=10.42.0.3
```

[web]

```
node-[1:30] ansible_host=10.42.0.[31:60]
```

[web:vars]

```
apache_listen_port=8080  
apache_root_path=/var/www/mywebdocs/
```

[all:vars]

```
ansible_user=ender  
ansible_ssh_private_key_file=/home/ender/.ssh/id_rsa
```

Understanding Inventory - Variable Precedence

[webservers]

```
web01 ansible_host=52.14.208.176 tmp_dir=/tempdir  
web02 ansible_host=52.14.208.179 tmp_dir=/tmpwkdir
```

[appservers]

```
app01 ansible_host=18.221.195.152  
app02 ansible_host=18.188.124.127
```

[loadbalancers]

```
balancer01 ansible_host=3.15.11.56
```

[webservers:vars]

```
ansible_user=ec2-user  
ansible_notify_owner=frances  
apache_max_clients=288
```

Understanding Inventory - Variable Precedence

[webservers]

```
web01 ansible host=52.14.208.176 tmp_dir=/tempdir  
web02 ansible_host=52.14.208.179 tmp_dir=/tmpwsdir
```

[appservers]

```
app01 ansible_host=18.221.195.152  
app02 ansible_host=18.188.124.127
```

Host variables apply to the host and override group vars

[loadbalancers]

```
balancer01 ansible_host=3.15.11.56
```

[webservers:vars]

```
ansible user=ec2-user  
ansible notify owner=frances  
apache_max_clients=288
```

Group variables apply for all devices in that group

Understanding Inventory - Groups

[nashville]

bnaapp01

bnaapp02

[atlanta]

atlapp03

atlapp04

[south:children]

atlanta

nashville

hsvapp05

Configuration File

- Basic configuration for Ansible
- Can be in multiple locations, with different precedence
- Here: `.ansible.cfg` in the home directory
- Configures where to find the inventory

The Ansible Configuration

Configuration files will be searched for in the following order:

- **ANSIBLE_CONFIG** (environment variable if set)
- **ansible.cfg** (in the current directory)
- **~/.ansible.cfg** (in the home directory)
- **/etc/ansible/ansible.cfg** (installed as Ansible default)

First Ad-Hoc Command: ping

- Single Ansible command to perform a task quickly directly on command line
- Most basic operation that can be performed
- Here: an example Ansible ping - not to be confused with ICMP

```
$ ansible all -m ping
```

Ad-Hoc Commands ping

```
# Check connections (submarine ping, not ICMP)
[user@ansible] $ ansible all -m ping
```

```
web1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python":
"/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

The Ansible Command

Some basics to keep you from getting stuck

--help (Display some basic and extensive options)

```
[user@ansible ~]$ ansible --help
Usage: ansible <host-pattern> [options]
```

Define and run a single task 'playbook' against a set of hosts

Options:

```
-a MODULE_ARGS, --args=MODULE_ARGS
                      module arguments
--ask-vault-pass      ask for vault password
-B SECONDS, --background=SECONDS
<<<snippet, output removed for brevity>>>
```

Ad-Hoc Commands

Here are some common options you might use:

-m MODULE_NAME , --module-name=MODULE_NAME

Module name to execute the ad-hoc command

-a MODULE_ARGS , --args=MODULE_ARGS

Module arguments for the ad-hoc command

-b , --become

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

-e EXTRA_VARS , --extra-vars=EXTRA_VARS

Set additional variables as key=value or YAML/JSON

Ad-Hoc Commands

Here are some common options you might use:

```
# Check connections to all (submarine ping, not ICMP)  
[user@ansible] $ ansible all -m ping
```

```
# Run a command on all the hosts in the web group  
[user@ansible] $ ansible web -m command -a "uptime"
```

```
# Collect and display known facts for server "webl"  
[user@ansible] $ ansible webl -m setup
```

Ansible Modules

Using **ansible-doc** to list all modules

```
[user@ansible ~]$ ansible-doc --list

a10_server                                Manage A10 Networks... server object.
a10_server_axapi3                            Manage A10 Networks... devices
a10_service_group                           Manage A10 Networks... service groups
a10_virtual_server                          Manage A10 Networks... virtual server...
aci_aaa_user                                 Manage AAA users (aaa:User)
aci_aep                                     Manage attachable Access Entity Profile...
aci_aep_to_domain                           Bind AEPs to Physical or Virtual Domains...
aci_ap                                      Manage top level Application Profile...
aci_bd                                       Manage Bridge Domains (BD) objects...
aci_bd_subnet                               Manage Subnets (fv:Subnet)
aci_bd_to_l3out                             Bind Bridge Domain to L3 Out (fv:RsBDToOut)
.

<<snippet, output removed for brevity>>>
```

Ansible Modules

Using **ansible-doc** to specify one module

```
[user@ansible ~]$ ansible-doc copy
> COPY      (/usr/lib/python2.7/site-packages/ansible/modules/files/copy.py)
```

The `copy' module copies a file from the local or remote machine to a location on the remote machine. Use the [fetch] module to copy files from remote locations to the local box. If you need variable interpolation in copied files, use the [template] module. For Windows targets, use the [win_copy] module instead.

* note: This module has a corresponding action plugin.

OPTIONS (= is mandatory):

- attributes

Attributes the file or directory should have. To get supported flags look at the man page for `chattr' on the target system. This string should contain the attributes in the same order as the one displayed by `lsattr'.
`=' operator is assumed as default, otherwise `+' or `-' operators need to be included in the string.

(Aliases: attr) [Default: (null)]

version_added: 2.3

Ansible Modules

"I can't find a module that does what I need it to do!"

command

shell

raw

script*



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 1.2 now in your lab environment



Red Hat

Exercise 1.3

Topics Covered:

- Playbooks basics
- Running a playbook



Red Hat
Ansible Automation
Platform

An Ansible Play in an Ansible Playbook

A play

```
- hosts: db
vars:
  software:
    - mariadb-server
roles:
  - install_wordpress_db

- hosts: web
vars:
  software:
    - httpd
    - php
roles:
  - install_wordpress_web
```

Another
play

An Ansible Play in an Ansible Playbook

A play

```
- hosts: db
  vars:
    software:
      - mariadb-server
  roles:
    - install_wordpress_db

- hosts: web
  vars:
    software:
      - httpd
      - php
  roles:
    - install_wordpress_web
```

Another play

An Ansible Play in an Ansible Playbook

A play

```
- hosts: db
vars:
  software:
    - mariadb-server
roles:
  - install_wordpress_db

- hosts: web
vars:
  software:
    - httpd
    - php
roles:
  - install_wordpress_web
```

Another
play

An Ansible Play (Common Elements)

This is not an exhaustive list, but contains most of the elements you will commonly see in an Ansible play.

hosts	The declarative list of hosts or groups against which this play will run.
connection	Allows you to change the connection plugin used for tasks to execute on the target.
port	Used to override the default port used in a connection.
remote_user	User to define / override which user is connecting to the remote system
become	Boolean that controls if privilege escalation is used or not on Task execution. (also become_flags , become_user , become_method)

An Ansible Play (Common Elements)

Information Handling:

name	Identifier. Can be used for documentation, in or tasks/handlers.
gather_facts	Boolean (default yes) allows the bypass of fact gathering. This can speed up connection time where facts are not needed in a playbook. This refers to the content retrieved by the setup module.
no_log	Boolean that controls information disclosure and logging.
ignore_errors	Boolean. When set to yes , errors will be ignored unless absolutely fatal to the playbook execution
check_mode	Also known as “dry run” mode, will evaluate but not execute. For modules that support check mode, the module will report the expected result without making any changes as a result of the tasks.

An Ansible Play (Common Elements)

Inventory Handling:

order

Controls the sorting of hosts as they are used for executing the play. Possible values are inventory (default), sorted, reverse_sorted, reverse_inventory and shuffle.

Variable Handling:

vars

Dictionary/map of variables

vars_files

List of files that contain vars to include in the play.

vars_prompt

list of variables to prompt for on launch.

An Ansible Play (Common Elements)

Task Handling:

pre_tasks	A list of tasks to execute before roles.
roles	List of roles to be imported into the play
tasks	Main list of tasks to execute in the play, they run after roles and before post_tasks .
post_tasks	A list of tasks to execute after the tasks section.
handlers	Like regular tasks, but referenced by a globally unique name, and notified by other tasks. If nothing notifies a handler, it will not run. Regardless of how many tasks notify a handler, it will run only once.

Common Ansible Play Elements: Hosts

```
- name: install a LAMP stack
  hosts: web,db,appserver01
  become: yes
  vars:
    my_greeting: Welcome to my awesome page
    favorite_food: fried pickles

  roles:
    - install_lamp_elements

  tasks:
    - name: write the index file
      copy:
        content: "{{ my_greeting }}. Enjoy some {{ favorite_food }}"
        dest: /var/www/html/index.html
      notify: reload_apache

  handlers:
    - name: reload_apache
      service:
        name: httpd
        state: reloaded
```



Common Ansible Play Elements: Hosts

```
- name: install a LAMP stack
hosts: web,db,appserver01
become: yes
vars:
  my_greeting: Welcome to my awesome page
  favorite_food: fried pickles

roles:
  - install_lamp_elements

tasks:
- name: write the index file
  copy:
    content: "{{ my_greeting }}. Enjoy some {{ favorite_food }}"
    dest: /var/www/html/index.html
  notify: reload_apache

handlers:
- name: reload_apache
  service:
    name: httpd
    state: reloaded
```

Common Ansible Play Elements: Hosts

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
  
- name: Ensure latest index.html file is present  
  copy:  
    src: files/index.html  
    dest: /var/www/html/  
  
- name: Restart httpd  
  service:  
    name: httpd  
    state: restart
```

Common Ansible Play Elements: Hosts

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
  
- name: Ensure latest index.html file is present  
  copy:  
    src: files/index.html  
    dest: /var/www/html/  
  
- name: Restart httpd  
  service:  
    name: httpd  
    state: restart
```

Common Ansible Play Elements: Hosts

```
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest

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

- name: Restart httpd
  service:
    name: httpd
    state: restart
```

Common Ansible Play Elements: Hosts

```
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest

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

- name: Restart httpd
  service:
    name: httpd
    state: restart
```

Running an Ansible Playbook:

The many colors of Ansible

A task executed as expected, no change was made.

A task executed as expected, making a change

General text information and headers

A conditional task was skipped

A bug or deprecation warning

A task failed to execute successfully

Running an Ansible Playbook

```
[user@ansible] $ ansible-playbook apache.yml

PLAY [webservers] ****
TASK [Gathering Facts] ****
ok: [web2]
ok: [web1]
ok: [web3]

TASK [Ensure httpd package is present] ****
changed: [web2]
changed: [web1]
changed: [web3]

TASK [Ensure latest index.html file is present] ****
changed: [web2]
changed: [web1]
changed: [web3]

TASK [Restart httpd] ****
changed: [web2]
changed: [web1]
changed: [web3]

PLAY RECAP ****
web2          : ok=1    changed=3  unreachable=0   failed=0
web1          : ok=1    changed=3  unreachable=0   failed=0
web3          : ok=1    changed=3  unreachable=0   failed=0
```

Running an Ansible Playbook

```
[user@ansible] $ ansible-playbook apache.yml
```

```
PLAY [webservers] ****
```

```
TASK [Gathering Facts] ****
```

```
ok: [web2]
```

```
ok: [web1]
```

```
ok: [web3]
```

```
TASK [Ensure httpd package is present] ****
```

```
changed: [web2]
```

```
changed: [web1]
```

```
changed: [web3]
```

```
TASK [Ensure latest index.html file is present] ****
```

```
changed: [web2]
```

```
changed: [web1]
```

```
changed: [web3]
```

```
TASK [Restart httpd] ****
```

```
changed: [web2]
```

```
changed: [web1]
```

```
changed: [web3]
```

```
PLAY RECAP ****
```

web2	:	ok=1	changed=3	unreachable=0	failed=0
web1	:	ok=1	changed=3	unreachable=0	failed=0
web3	:	ok=1	changed=3	unreachable=0	failed=0



setup module

yum module

copy module

service module



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 1.3 now in your lab environment



Red Hat

Exercise 1.4

Topics Covered:

- Working with variables
- What are facts?



Red Hat
Ansible Automation
Platform

An Ansible Playbook Variable Example

```
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }} {{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{var_three}}"
```

An Ansible Playbook Variable Example

```
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }} {{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{var_three}}"
```

ansible is awesome



Facts

- Just like variables, really...
- ...but: coming from the host itself!
- Check them out with the setup module

```
"ansible_facts": {  
    "ansible_default_ipv4": {  
        "address": "10.41.17.37",  
        "macaddress": "00:69:08:3b:a9:16",  
        "interface": "eth0",  
    ...  
}
```

Gather facts on target machine

```
$ ansible localhost -m setup
localhost | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.122.1",
            "172.21.208.111"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::8f31:b68d:f487:2775"
        ],
    }
}
```

Ansible Variables and Facts

```
"ansible_facts": {  
    "ansible_default_ipv4": {  
        "address": "10.41.17.37",  
        "macaddress": "00:69:08:3b:a9:16",  
        "interface": "eth0",  
    ...  
}
```

A variable, defined
in our playbook

```
vars:  
    mynewip: 10.7.62.39
```

This is a template
file for **ifcfg-eth0**,
using a mix of
discovered facts and
variables to write
the static file.

```
DEVICE="{{ ansible_default_ipv4.interface }}"  
ONBOOT=yes  
HWADDR={{ ansible_default_ipv4.macaddress }}  
TYPE=Ethernet  
BOOTPROTO=static  
IPADDR={{ mynewip }}
```

Variable Precedence

Ansible can work with metadata from various sources as variables. Different sources will be overridden in an order of precedence.

1. extra vars (**Highest - will override anything else**)
2. task vars (overridden only for the task)
3. block vars (overridden only for tasks in block)
4. role and include vars
5. play vars_files
6. play vars_prompt
7. play vars
8. set_facts
9. registered vars
10. host facts
11. playbook host_vars
12. playbook group_vars
13. inventory host_vars
14. inventory group_vars
15. inventory vars
16. role defaults (**Lowest - will be overridden by anything else listed here**)

Ansible Inventory - Managing Variables In Files

```
[user@ansible ~] $ tree /somedir  
  
/somedir  
├── group_vars  
│   └── applsrv  
│       └── db  
│       └── web  
└── inventory  
    └── host_vars  
        └── app01  
        └── app02  
        └── app03
```

Ansible Inventory - Managing Variables In Files

```
[user@ansible ~] $ tree  
/somedir
```

```
/somedir  
├── group vars  
│   └── app1srv  
│   └── db  
│   └── web  
└── inventory  
    └── host vars  
        └── app01  
        └── app02  
        └── app03
```

```
[user@ansible ~] $ cat /somedir/inventory  
  
[web]  
node-[1:30] ansible_host=10.42.0.[31:60]
```

```
[appxsrv]  
app01  
app02  
app03
```

```
[user@ansible ~] $ cat /somedir/group vars/web  
  
apache_listen_port: 8080  
apache_root_path: /var/www/mywebdocs/
```

```
[user@ansible ~] $ cat /somedir/host vars/app01  
  
owner_name: Chris P. Bacon  
owner_contact: 'cbacon@mydomain.tld'  
server_purpose: Application X
```



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 1.4 now in your lab environment



Red Hat

Exercise 1.5

Topics Covered:

- Conditionals
- Handlers
- Loops



Red Hat
Ansible Automation
Platform

Conditionals via VARS

```
vars:  
  my_mood: happy  
  
tasks:  
- name: conditional task, based on my_mood var  
  debug:  
    msg: "Come talk to me. I am {{ my_mood }}!"  
  when: my_mood == "happy"
```

Conditionals with variables

```
vars:  
  my_mood: happy  
  
tasks:  
- name: conditional task, based on my_mood var  
  debug:  
    msg: "Come talk to me. I am {{ my_mood }}!"  
  when: my_mood == "happy"
```

Alternatively

```
debug:  
  msg: "Feel free to interact. I am {{ my_mood }}"  
when: my_mood != "grumpy"
```

Conditionals with facts

```
tasks:  
- name: Install apache  
  apt:  
    name: apache2  
    state: latest  
  when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'  
  
- name: Install httpd  
  yum:  
    name: httpd  
    state: latest  
  when: ansible_distribution == 'RedHat'
```

Using the previous task state

This is NOT a handler task, but has similar function

- **name: Ensure httpd package is present**
yum:
 name: httpd
 state: latest
register: http_results
- **name: Restart httpd**
service:
 name: httpd
 state: restart
when: httpd_results.changed

Handler Tasks

A handler task is run when a referring task result shows a change

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
    notify: restart_httpd  
  
handlers:  
- name: restart_httpd  
  service:  
    name: httpd  
    state: restart
```

Handler Tasks

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
    notify: restart_httpd  
  
- name: Standardized index.html file  
  copy:  
    content: "This is my index.html file for {{ ansible_host }}"  
    dest: /var/www/html/index.html  
    notify: restart_httpd
```

If **either** task notifies a **changed** result, the handler will be notified **ONCE**.

```
TASK [Ensure httpd package is present] ****  
ok: [web2]  unchanged  
ok: [web1]  unchanged  
  
TASK [Standardized index.html file] ****  
changed: [web2]  
changed: [web1]  changed  
  
NOTIFIED: [restart_httpd] ***  
changed: [web2]  
changed: [web1]
```

handler runs once

Handler Tasks

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
    notify: restart_httpd  
  
- name: Standardized index.html file  
  copy:  
    content: "This is my index.html file for {{ ansible_host }}"  
    dest: /var/www/html/index.html  
    notify: restart_httpd
```

If **both** of these tasks notifies of a **changed** result, the handler will be notified **ONCE**.

```
TASK [Ensure httpd package is present] ****  
changed: [web2]  
changed: [web1] changed  
  
TASK [Standardized index.html file] ****  
changed: [web2]  
changed: [web1] changed  
  
NOTIFIED: [restart_httpd] ***  
changed: [web2]  
changed: [web1]
```

handler runs once

Handler Tasks

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
    notify: restart_httpd  
  
- name: Standardized index.html file  
  copy:  
    content: "This is my index.html file for {{ ansible_host }}"  
    dest: /var/www/html/index.html  
    notify: restart_httpd
```

If **neither** task notifies a **changed** result, the handler **does not run**.

```
TASK [Ensure httpd package is present] ****  
ok: [web2]  
ok: [web1]  unchanged  
  
TASK [Standardized index.html file] ****  
ok: [web2]  
ok: [web1]  unchanged  
  
PLAY RECAP ****  
web2      : ok=2  changed=0  nreachable=0  failed=0  skipped=0  rescued=0  ignored=0  
web1      : ok=2  changed=0  nreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```



Variables & Loops

Great opportunity to use a loop

```
---
```

- **name:** Ensure users
 - hosts:** node1
 - become:** yes

- tasks:**
 - **name:** Ensure user is present
 - user:**
 - name:** dev_user
 - state:** present

 - **name:** Ensure user is present
 - user:**
 - name:** qa_user
 - state:** present

 - **name:** Ensure user is present
 - user:**
 - name:** prod_user
 - state:** present

Variables & Loops

Using loops to simplify tasks

```
---
```

- **name:** Ensure users
 - hosts:** node1
 - become:** yes

tasks:

- **name:** Ensure users are present
 - user:**
 - name:** "{{item}}"
 - state:** present

loop:

- dev_user
- qa_user
- prod_user



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 1.5 now in your lab environment



Red Hat

Exercise 1.6

Topics Covered:

- Templates



Red Hat
Ansible Automation
Platform

Variables & Templates

Using a system fact or declared variable to write a file

```
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80
    http_docroot: /var/www/mysite.com

  tasks:
    - name: Verify correct config file is present
      template:
        src: templates/httpd.conf.j2
        dest: /etc/httpd/conf/httpd.conf
```

Variables & Templates

Using a system fact or declared variable to write a file

```
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80
    http_docroot: /var/www/mysite.com

  tasks:
    - name: Verify correct config file is present
      template:
        src: templates/httpd.conf.j2
        dest: /etc/httpd/conf/httpd.conf
```

```
## Excerpt from httpd.conf.j2

# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
# Listen 80    ## original line
Listen {{ http_port }}

# DocumentRoot: The directory out of which you will serve your
# documents.
# DocumentRoot "/var/www/html"
DocumentRoot {{ http_docroot }}
```



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 1.6 now in your lab environment



Red Hat

Exercise 1.7

Topics Covered:

- What are roles?
- How they look like
- Galaxy



Red Hat
Ansible Automation
Platform

Roles

- Roles: Think Ansible packages
- Roles provide Ansible with a way to load tasks, handlers, and variables from separate files.
- Roles group content, allowing easy sharing of code with others
- Roles make larger projects more manageable
- Roles can be developed in parallel by different administrators

Better start using roles now!

Role structure

- Defaults: default variables with lowest precedence (e.g. port)
- Handlers: contains all handlers
- Meta: role metadata including dependencies to other roles
- Tasks: plays or tasks
Tip: It's common to include tasks in main.yml with "when" (e.g. OS == xyz)
- Templates: templates to deploy
- Tests: place for playbook tests
- Vars: variables (e.g. override port)

```
user/
  └── defaults
      └── main.yml
  └── handlers
      └── main.yml
  └── meta
      └── main.yml
  └── README.md
  └── tasks
      └── main.yml
  └── templates
  └── tests
      └── inventory
          └── test.yml
  └── vars
      └── main.yml
```

Ansible Galaxy

Sharing
Content

Community

Roles, and
more

v1 - Set config file to use on boot.

1. Write multiple configuration files.
 - For each environment/region.
2. Inspect metadata on boot and use the matching config file.

v1 - Set config file to use on boot.

1. Write multiple configuration files.
 - For each environment/region.
2. Inspect metadata on boot and use the matching config file.



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 1.7 now in your lab environment



Red Hat



Red Hat
Ansible Automation
Platform

Exercise 1.8

Topics Covered:

- A bonus lab - try it on your own, and when time permits



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 1.8 now in your lab environment



Red Hat

Section 2

Ansible Tower



Red Hat
Ansible Automation
Platform

Exercise 2.1

Topics Covered:

- Introduction to Tower

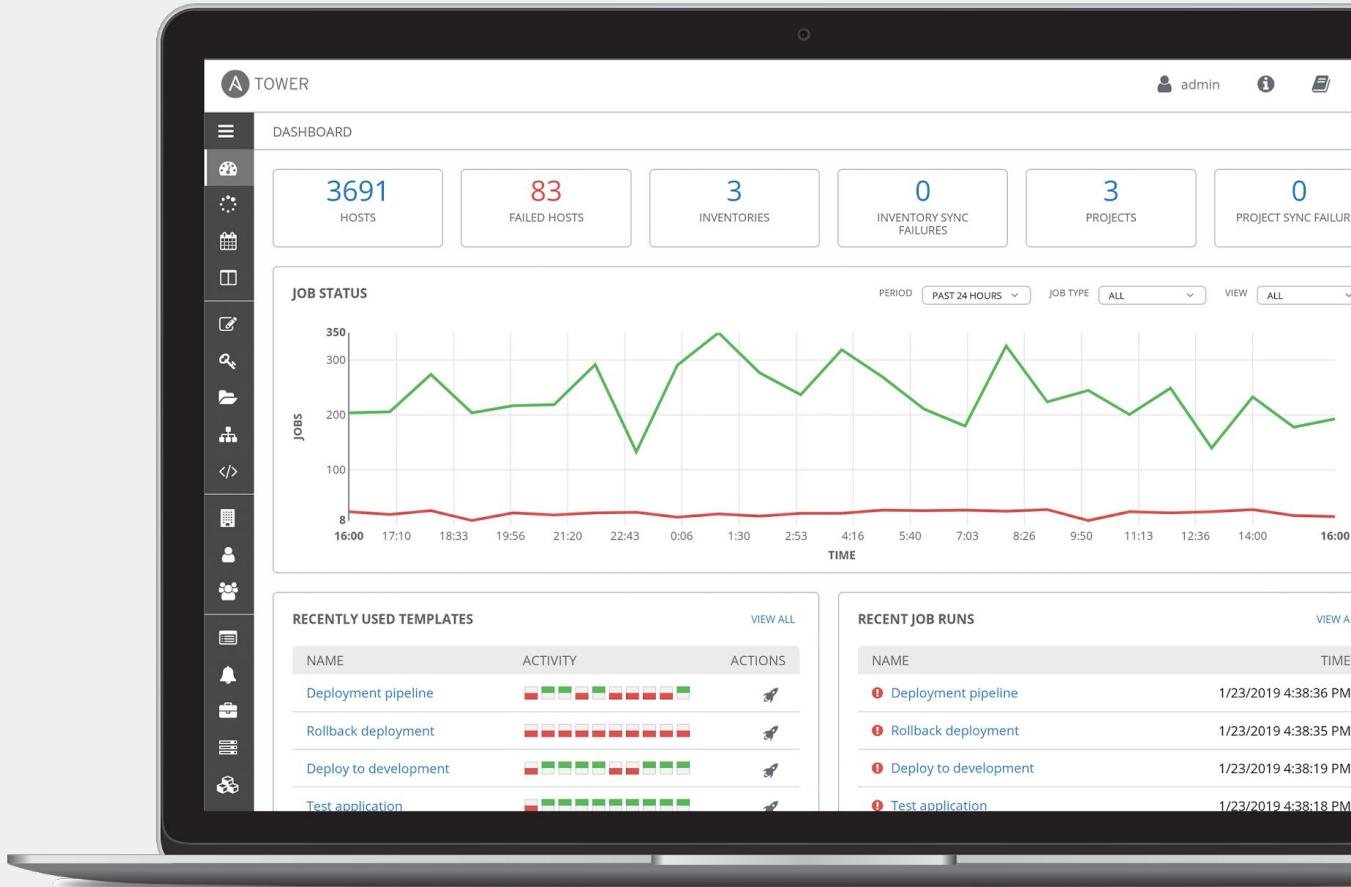


Red Hat
Ansible Automation
Platform

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.

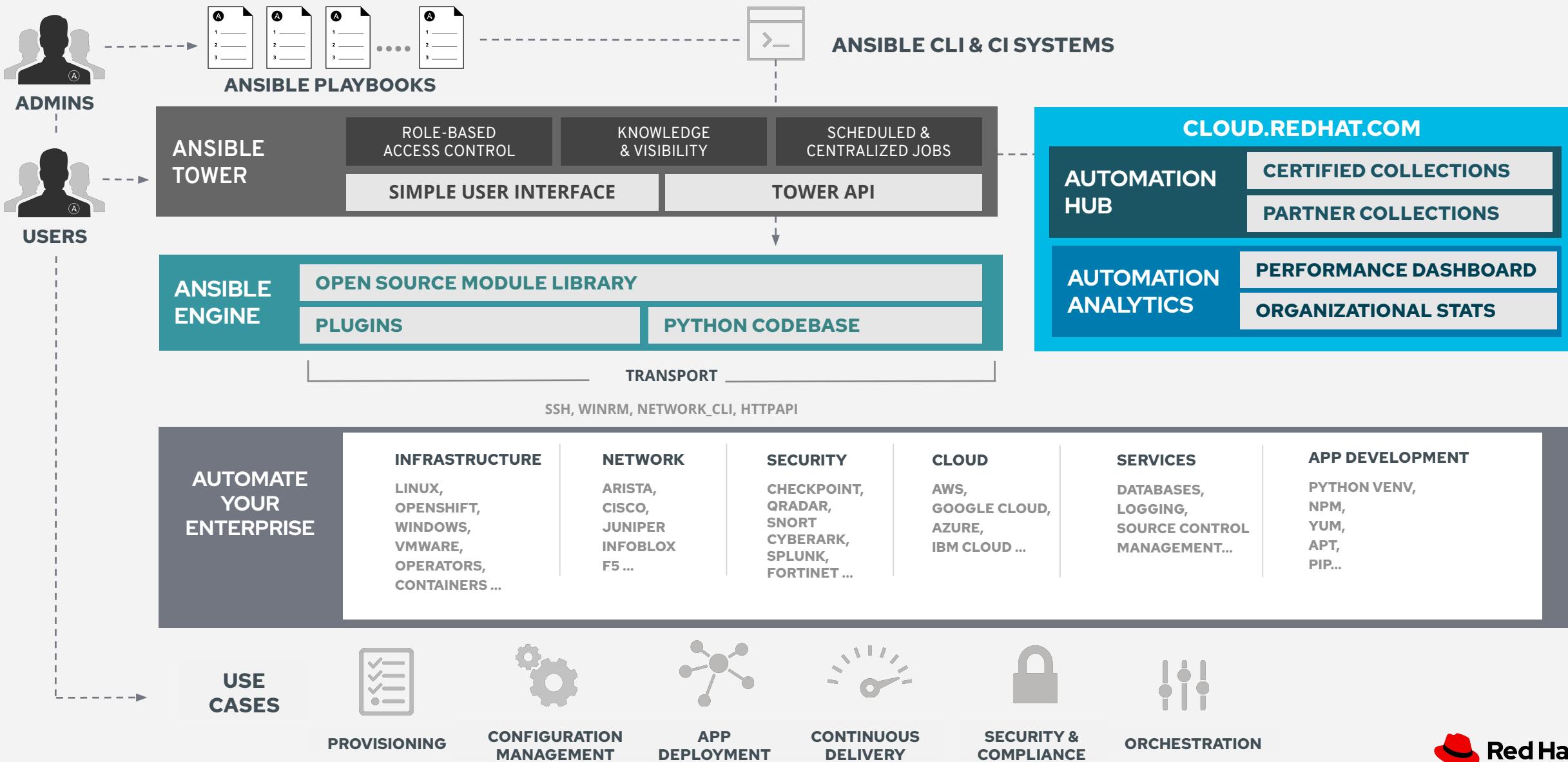
Centralized logging

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 Automation Platform

Lab Time

Complete exercise 2.1 now in your lab environment



Red Hat

Exercise 2.2

Topics Covered:

- Inventories
- Credentials



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

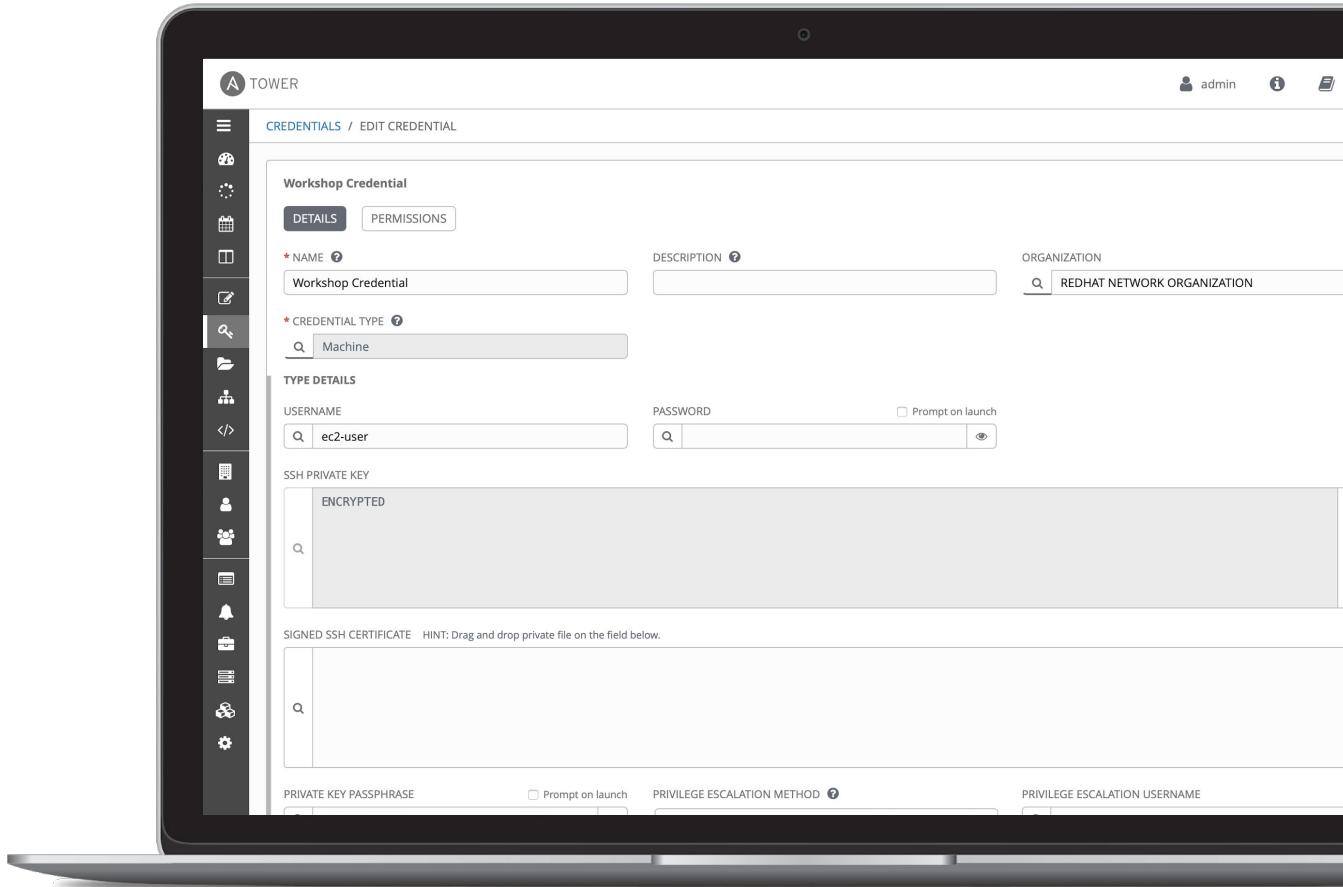
The screenshot shows the Ansible Tower interface on a laptop screen. The main window displays the 'Workshop Inventory' host list. The left sidebar has icons for inventories, hosts, permissions, groups, sources, and completed jobs. The top navigation bar shows 'INVENTORIES / Workshop Inventory / HOSTS'. The main content area has tabs for 'DETAILS', 'PERMISSIONS', 'GROUPS', 'HOSTS' (which is selected), 'SOURCES', and 'COMPLETED JOBS'. A search bar and a 'KEY' button are also present. The 'HOSTS' table lists five hosts: 'ON' (radio button) and 'ansible' (radio button); 'ON' (radio button) and 'rtr1' (radio button); 'ON' (radio button) and 'rtr2' (radio button); 'ON' (radio button) and 'rtr3' (radio button); and 'ON' (radio button) and 'rtr4' (radio button). To the right of the hosts, under 'RELATED GROUPS', are several group names with 'X' buttons: 'control' (under ansible), 'cisco' and 'dc1' (under rtr1), 'arista' and 'dc2' (under rtr2), 'dc1' and 'juniper' (under rtr3), and 'arista' and 'dc2' (under rtr4). Below this, there is another section with tabs for 'INVENTORIES' and 'HOSTS', a search bar, and filters for '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.





Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 2.2 now in your lab environment



Red Hat

Exercise 2.3

Topics Covered:

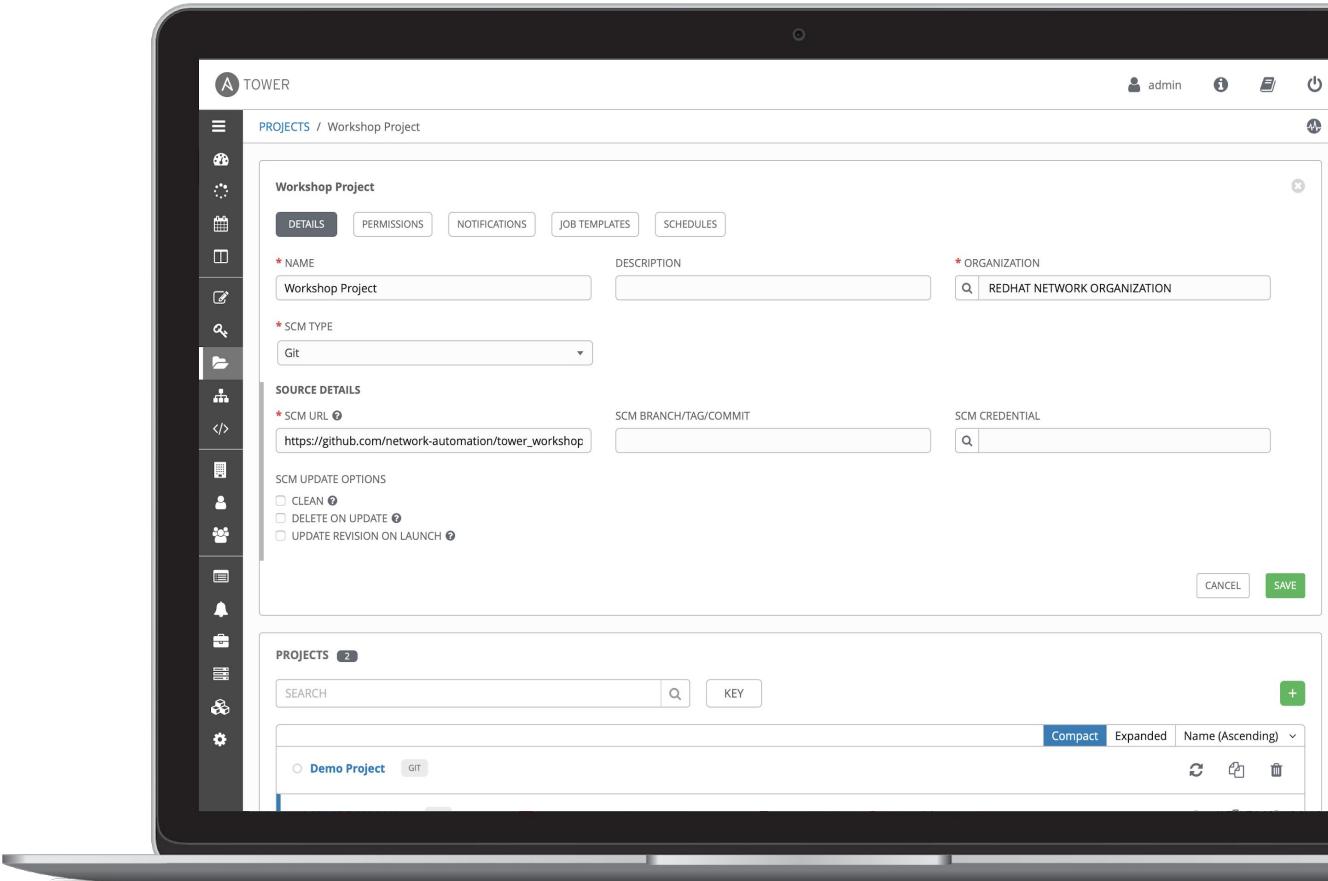
- Projects
- Job Templates



Project

A project is a logical collection of Ansible Playbooks, represented in Ansible Tower.

You can manage Ansible Playbooks and playbook directories by placing them in a source code management system supported by Ansible Tower, including Git, Subversion, and Mercurial.



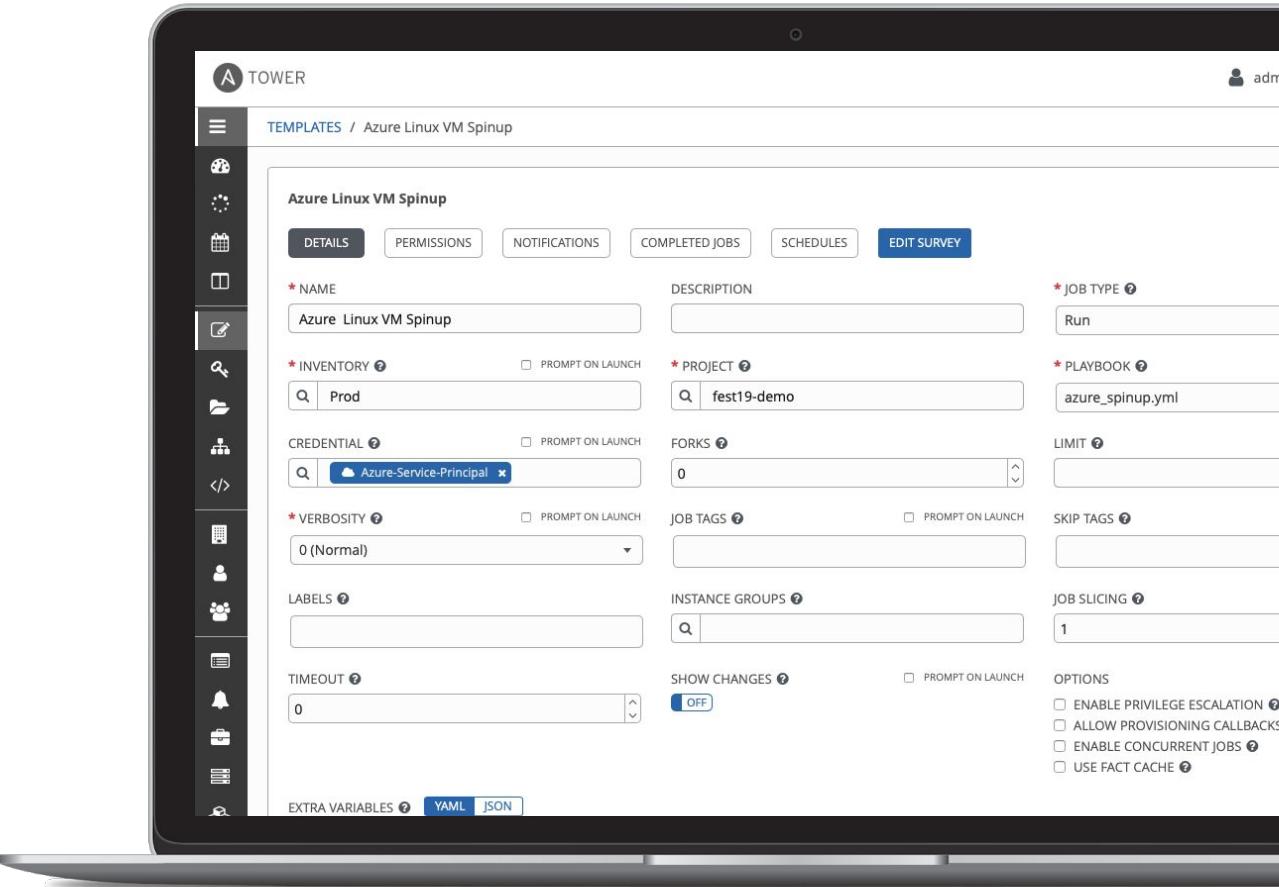
Job Templates

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

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

A **Job Template** requires:

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



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



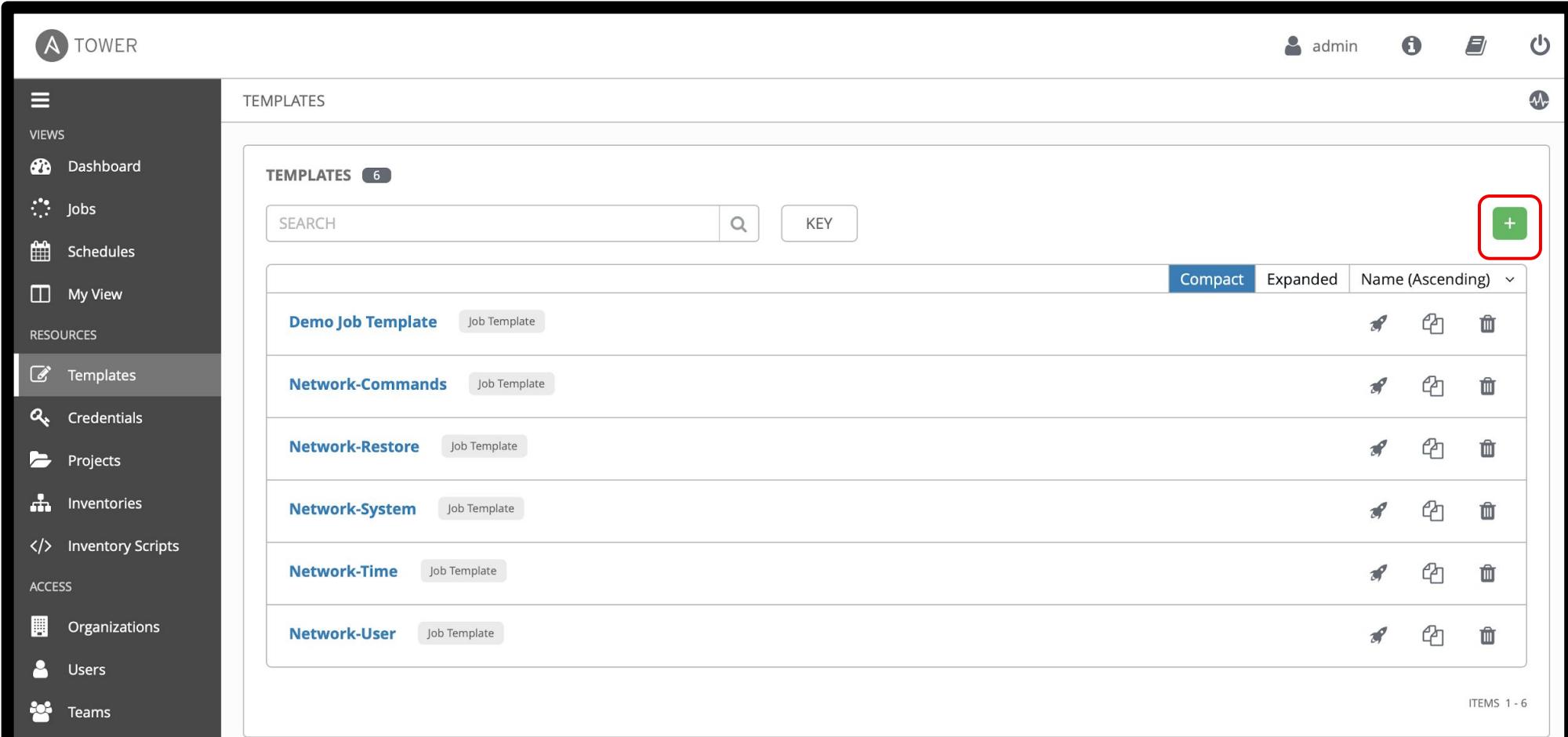
The screenshot shows the Tower interface with the 'TEMPLATES' view selected. The left sidebar includes options like 'Dashboard', 'Jobs', 'Schedules', 'My View', 'Templates' (which is currently selected), 'Credentials', 'Projects', 'Inventories', 'Inventory Scripts', 'Organizations', 'Users', and 'Teams'. 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 bottom right corner of the interface shows the text '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.

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

Lab Time

Complete exercise 2.3 now in your lab environment



Red Hat

Exercise 2.4

Topics Covered:

- Surveys



Red Hat
Ansible Automation
Platform

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.

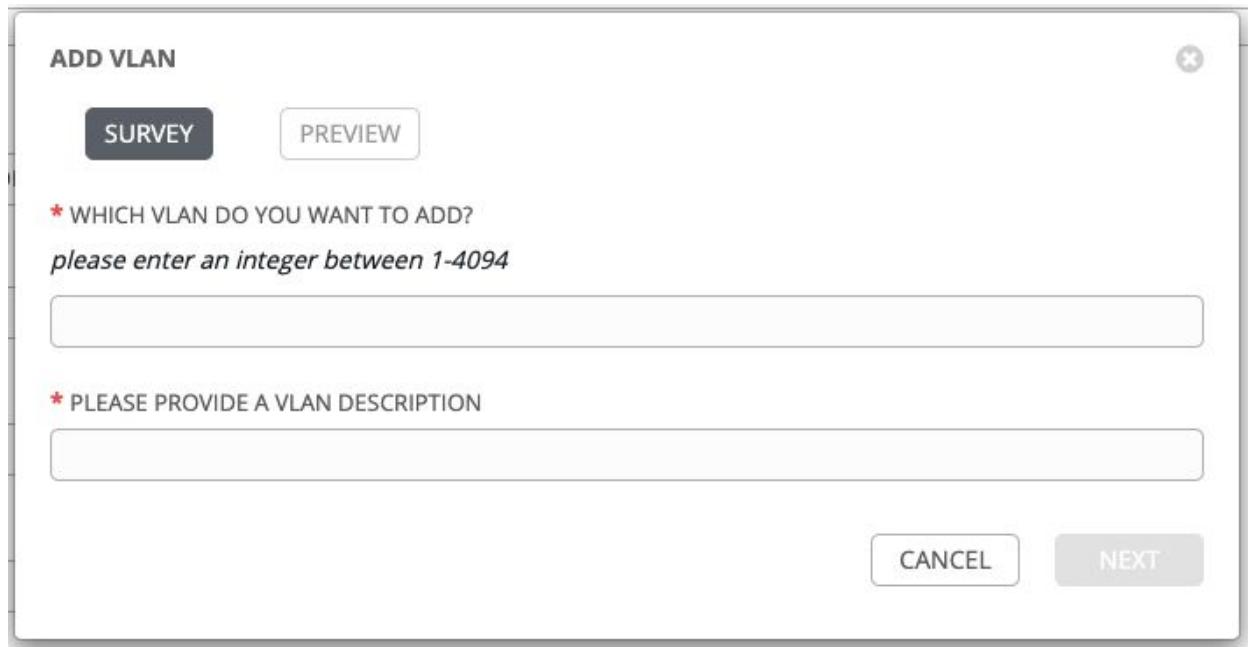
ADD VLAN

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 fields.

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. At the top left, there's a 'CONFIGURE BANNER | SURVEY ON' button. The main area is titled 'EDIT SURVEY PROMPT' and contains the following fields:

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

At the bottom left, there's a checked checkbox labeled "REQUIRED". At the bottom right, there are "CLEAR" and "UPDATE" buttons. On the right side of the window, there's a 'PREVIEW' section with a large text input field containing "Please enter the banner text" and "Please type into the text field the desired banner". This preview field has a blue edit icon and a trash icon. Below the preview is a small 'X' icon.

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' panel shows the survey prompt with the same text and instructions, along with edit and delete icons.

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

(edit) (delete)

CLEAR UPDATE CANCEL SAVE

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 with a green '+' button at the top. The items are listed by name (Ascending): Network-Restore, Network-System, Network-Time, and Network-User. Each item has three icons: a rocket (Edit), a thumbs up (Approve), and a trash can (Delete).



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 2.4 now in your lab environment



Red Hat

Exercise 2.5

Topics Covered:

- Role based access control



Red Hat
Ansible Automation
Platform

Role Based Access Control (RBAC)

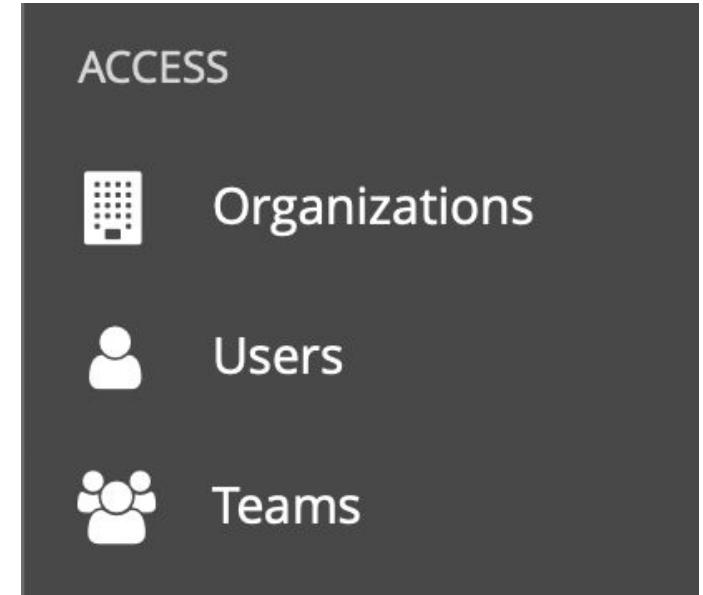
Role-Based Access Controls (RBAC) are built into Ansible Tower and allow administrators to delegate access to inventories, organizations, and more.

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 corner 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

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), Resources (Templates, Credentials, Projects, Inventories), ACCESS (Organizations, Users), and ADMINISTRATION (Teams). The 'Teams' option is highlighted with a grey background. The main content area has a header 'TEAMS' with a count of 4. Below the header is a search bar and a 'KEY' button. A table lists four teams: 'Compute T1' and 'Compute T2' belong to 'REDHAT COMPUTE ORGANIZATION', while 'Netadmin' and 'Netops' belong to 'REDHAT NETWORK ORGANIZATION'. Each team entry has edit and delete icons in the 'ACTIONS' column. At the bottom right of the table, it says 'ITEMS 1 - 4'.

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 with "TOWER" and "admin". Below the header is a "USERS" section with a search bar, a "KEY" button, and a green "+" button. A table lists eight users: admin, bbelcher, gbelcher, lbelcher, libelcher, network-admin, network-operator, and tbelcher. Each user row includes columns for Username, First Name, Last Name, and Actions (edit and delete icons). At the bottom of the table, it says "ITEMS 1 - 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	



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 2.5 now in your lab environment



Red Hat

Exercise 2.6

Topics Covered:

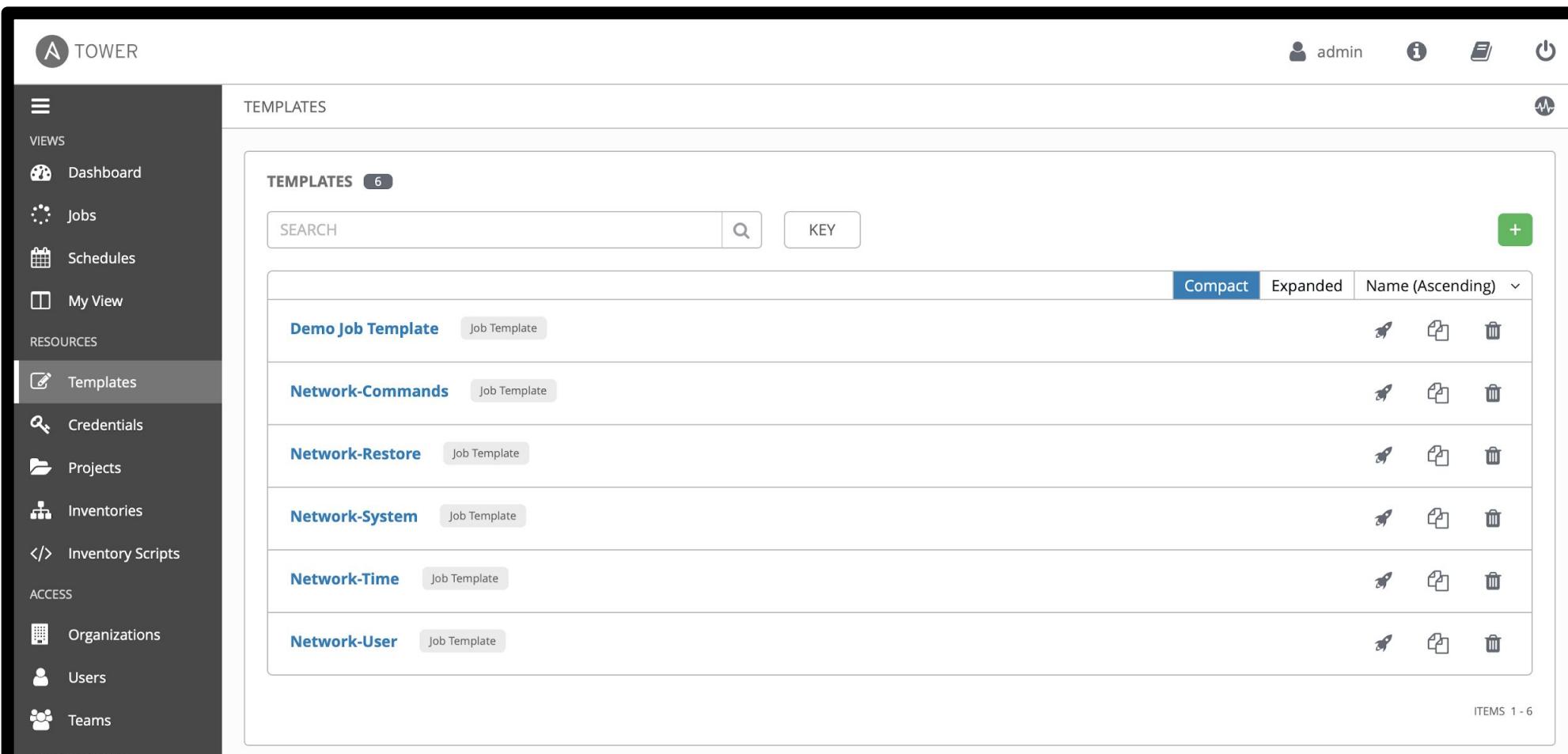
- Workflows



Red Hat
Ansible Automation
Platform

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. On the far left is a dark sidebar with a navigation menu:

- VIEWS: Dashboard, Jobs, Schedules, My View
- RESOURCES: **Templates** (selected), Credentials, Projects, Inventories, Inventory Scripts
- ACCESS: Organizations, Users, Teams
- ADMINISTRATION

The main content area is titled "TEMPLATES" and shows a list of six items:

NAME	TYPED AS	OPTIONS
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	

At the bottom right of the main content area, it says "ITEMS 1 - 6".

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 in the top right corner of the template list. The dropdown also lists 'Job Template'. The interface has a dark mode theme with light-colored cards for each template.

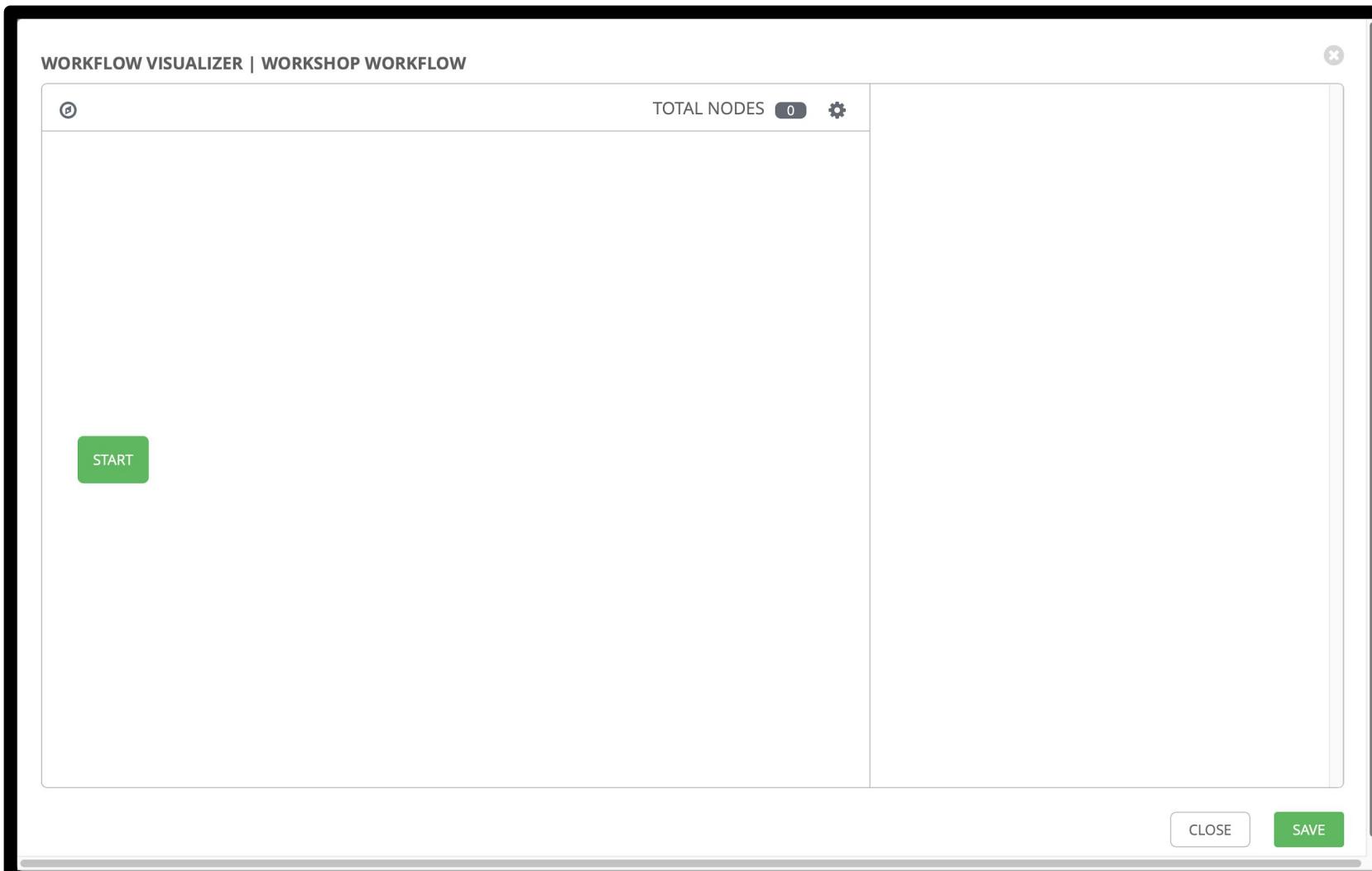
Creating the Workflow

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 'TEMPLATES / WORKSHOP WORKFLOW' screen for a template named 'WORKSHOP WORKFLOW'. A red box highlights the 'WORKFLOW VISUALIZER' button, which is located above the 'NAME', 'DESCRIPTION', and 'ORGANIZATION' fields. The 'NAME' field contains 'WORKSHOP WORKFLOW', 'DESCRIPTION' is empty, and 'ORGANIZATION' is set to 'Default'. Below these fields are sections for 'INVENTORY', 'LABELS', 'OPTIONS', and 'EXTRA VARIABLES'. The 'OPTIONS' section includes a checkbox for 'ENABLE CONCURRENT JOBS'. At the bottom, there are 'YAML' and 'JSON' tabs, and a 'PROMPT ON LAUNCH' checkbox. The top right shows the user 'admin' and various navigation icons.

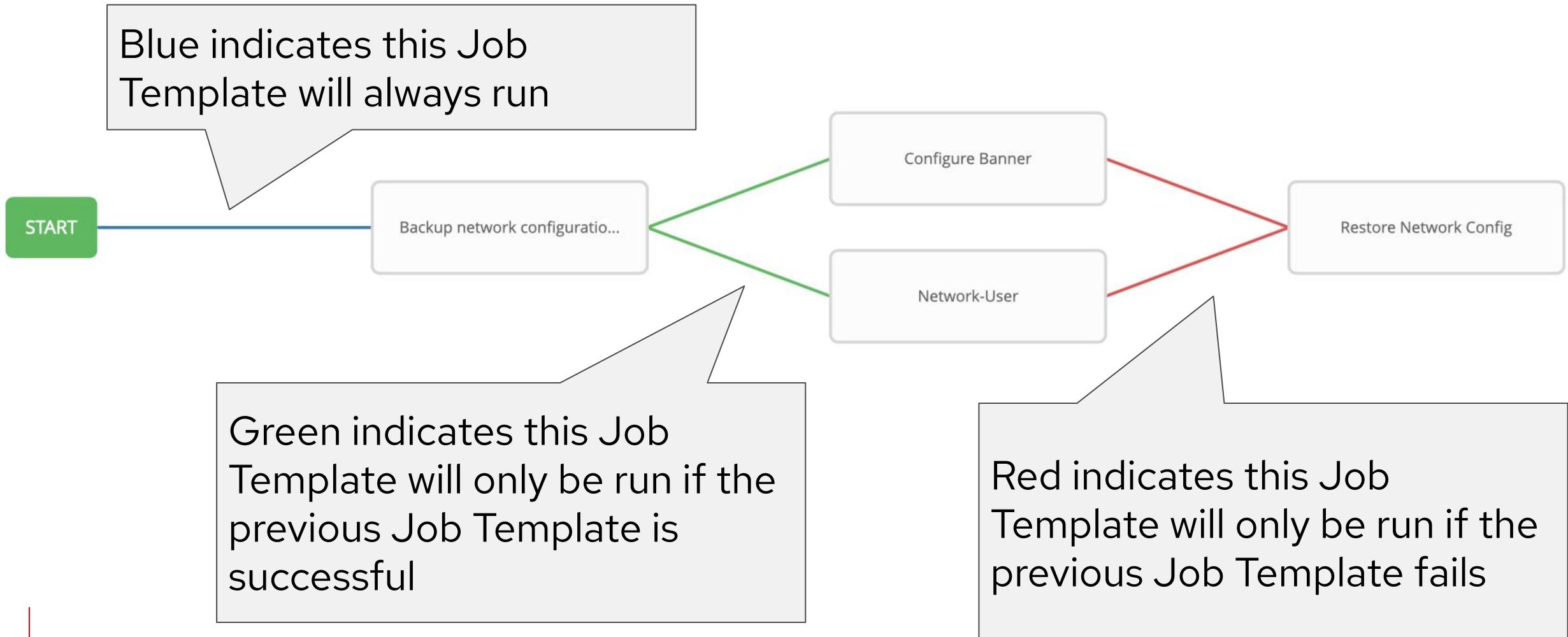
Workflow Visualizer

The workflow visualizer will start as a blank canvas.



Visualizing a Workflow

Workflows can branch out, or converge in.





Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 2.6 now in your lab environment



Red Hat

Exercise 2.7

Topics Covered:

- Wrap-up



Red Hat
Ansible Automation
Platform



Red Hat

Ansible Automation Platform

Lab Time

Complete exercise 2.7 now in your lab environment



Red Hat

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](#)

Thank you



linkedin.com/company/red-hat



youtube.com/AnsibleAutomation



facebook.com/ansibleautomation



twitter.com/ansible



github.com/ansible