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.

THANKS

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

First and foremost, thanks to:

KEV

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
Matt
Will
Götz

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

Ansible RHEL Automation Workshop

Introduction to Ansible RHEL Automation for System Administrators and Operators

Red Hat

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:

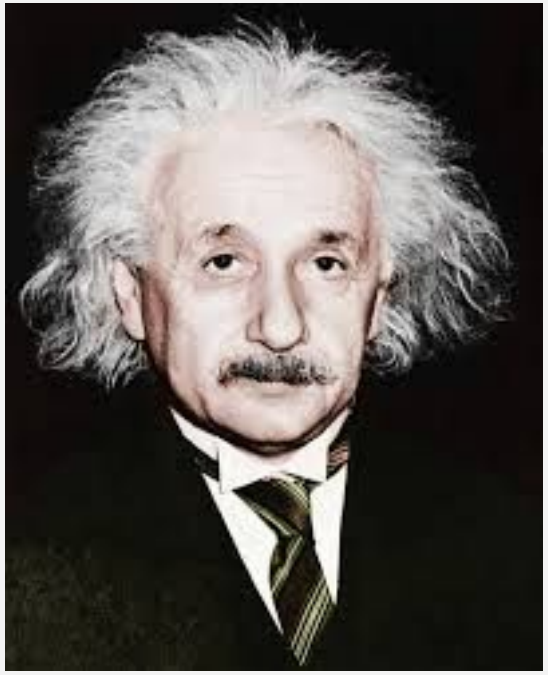
- What Ansible Automation is
- What it can do

Red Hat
Ansible
Automation

Red Hat

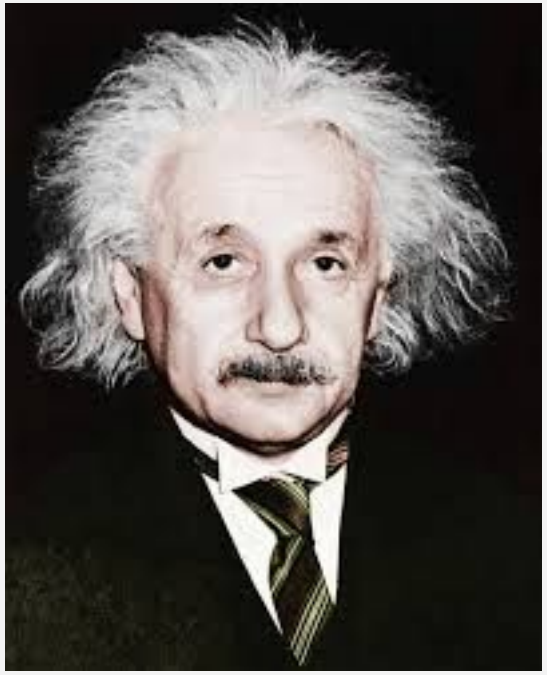
"Insanity is doing the same thing over and over again and expecting different results."

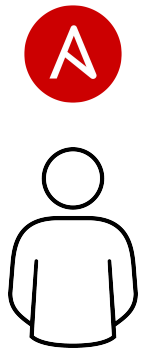
Albert Einstein



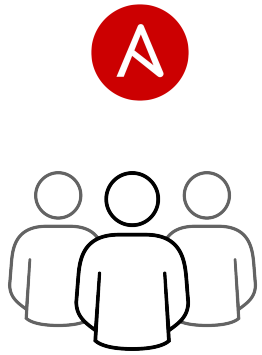
"Insanity is doing the same thing over and over again manually when you could have automated it with Ansible."

probably not Albert Einstein

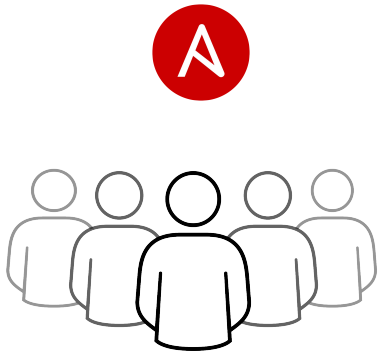




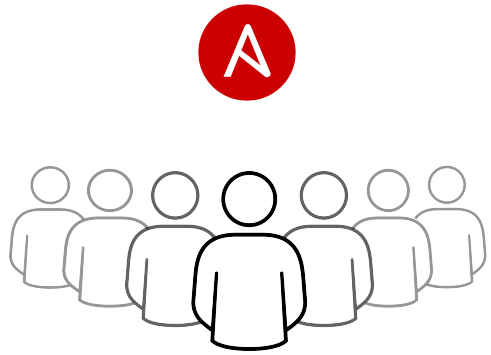
Automation happens when one person meets a problem they never want to solve again



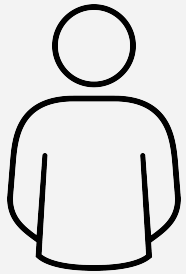
ACCELERATE



INTEGRATE



COLLABORATE



When we talk about **“repeatable processes,”** unfortunately, we’re often thinking of manual steps to assure we end up at the same destination as before.

Getting an IP Assigned

Contact Robert at **ext 2491** and request an IP in production servers range. Tell him which data center and he’ll assign one in the correct subnet.

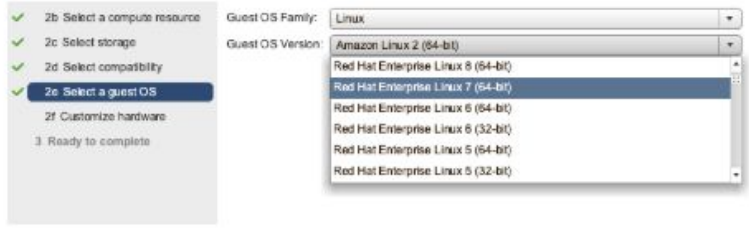
(this goes the same for dev and test)

Production servers range: 10.210.

Building The New Server

- In VMware VCenter right-click the CLUSTER and select “New Virtual Machine”
- Give the new server a name that’s compliant with the **VM Production Server naming standards and acceptable naming policy**, per IT management. Non-compliant names will be logged and potentially disconnected.

- Select VMNFS storage

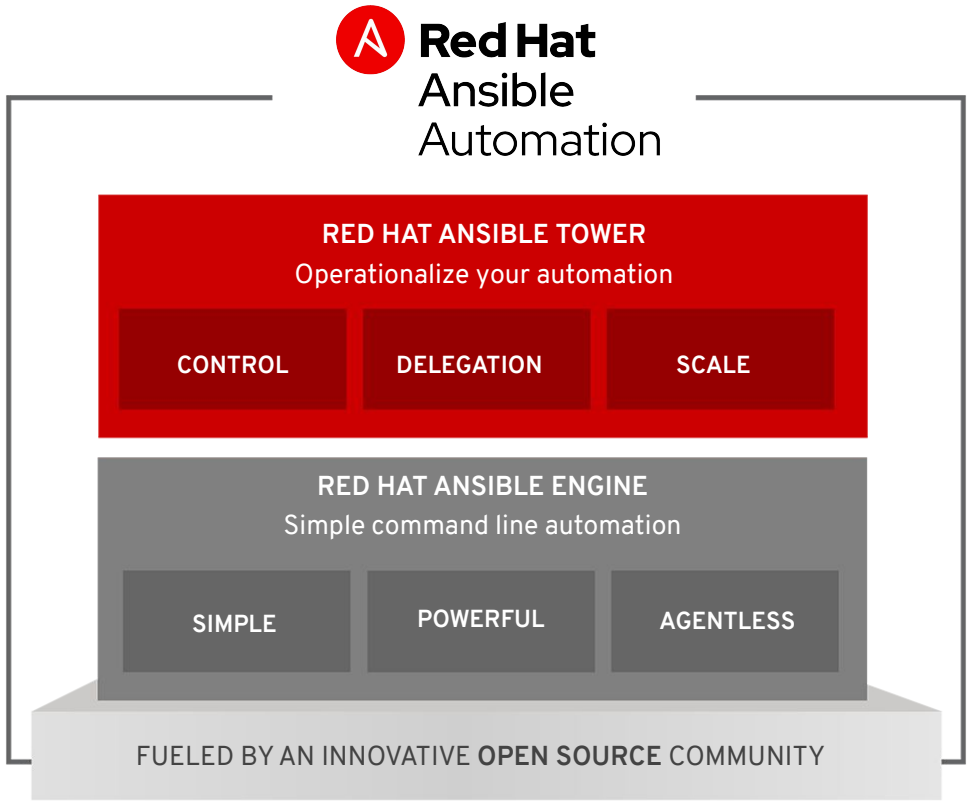


What is Ansible Automation?

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

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

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



Red Hat

Why Ansible?

Simple

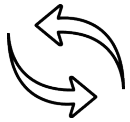
Human readable automation

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

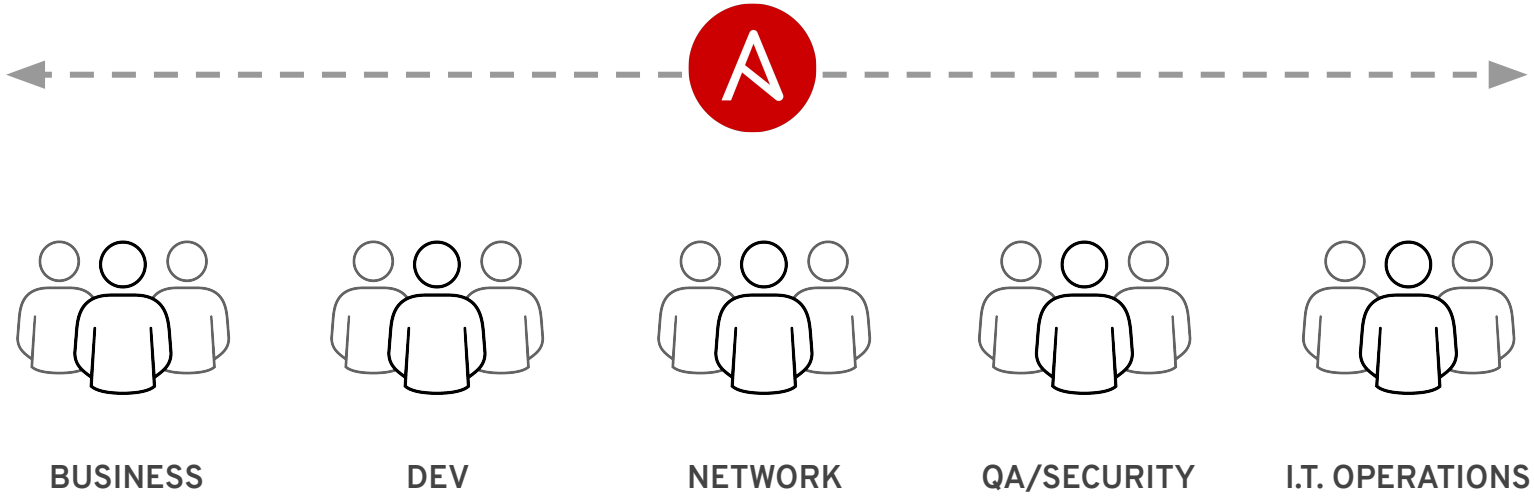
No agents to exploit or update

Get started immediately

More efficient & more secure

Red Hat

Ansible Automation works across teams



Red Hat

What Can I Stick Together With This Glue?

The bottle gives you suggestions as to what materials work best

Do this...

GLUE TWO THINGS TOGETHER AND MAKE THEM STICK

On these...

Wood	Stone	Ceramics	Bricks	Metal
Glass	Foam	And more...		

Red Hat

What Can I Automate Using Ansible?

Automate the deployment and management of your entire IT footprint.

Do this...

Orchestration	Configuration Management	Application Deployment	Provisioning	Continuous Delivery	Security and Compliance

On these...

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

Red Hat

Ansible automates technologies you use

Time to automate is measured in minutes

Cloud

AWS
Azure
Digital Ocean
Google
OpenStack
Rackspace
+more

Operating Systems

Rhel And Linux
Unix
Windows
+more

Virt & Container

Docker
VMware
RHV
OpenStack
OpenShift
+more

Storage

Netapp
Red Hat Storage
Infinidat
+more

Windows

ACLs
Files
Packages
IIS
Regedits
Shares
Services
Configs
Users
Domains
+more

Network

Arista
A10
Cumulus
Bigswitch
Cisco
Cumulus
Dell
F5
Juniper
Palo Alto
OpenSwitch
+more

Devops

Jira
GitHub
Vagrant
Jenkins
Bamboo
Atlassian
Subversion
Slack
Hipchat
+more

Monitoring

Dynatrace
Airbrake
BigPanda
Datadog
LogicMonitor
Nagios
New Relic
PagerDuty
Sensu
StackDriver
Zabbix
+more

Endless Use Cases For Ansible

- Ansible is NOT just a Config Management Tool.
- Ansible is NOT just an Application Deployment Tool.
- Ansible is NOT just a Provisioning Tool.
- Ansible is NOT just a CI/CD Tool.
- Ansible is NOT just an Audit and Compliance Tool.
- Ansible is NOT just an Orchestration Tool.

Ansible is a powerful automation engine...
with strong use cases for all of the above tasks.

Red Hat

Section 1
Engine

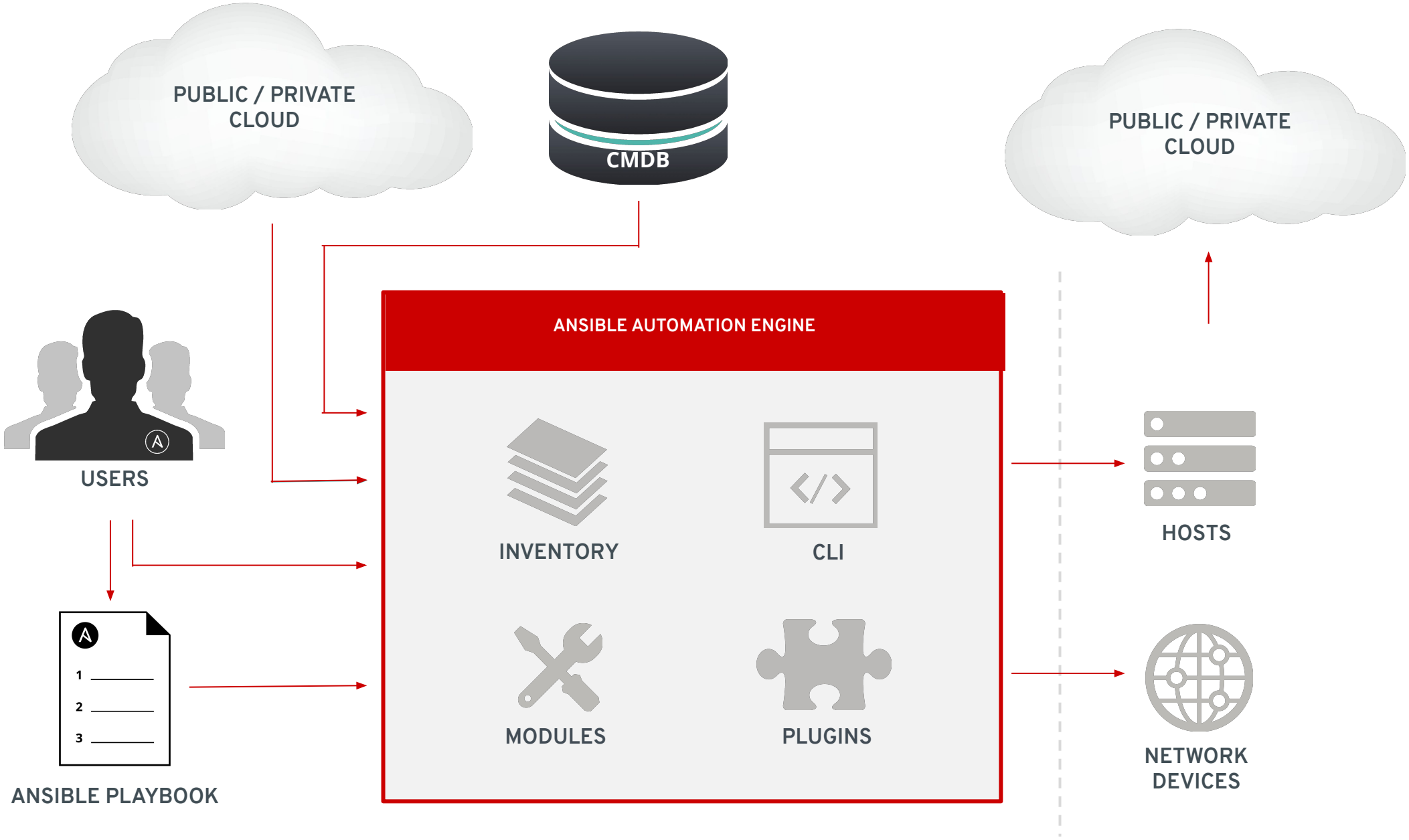
Exercise 1.1

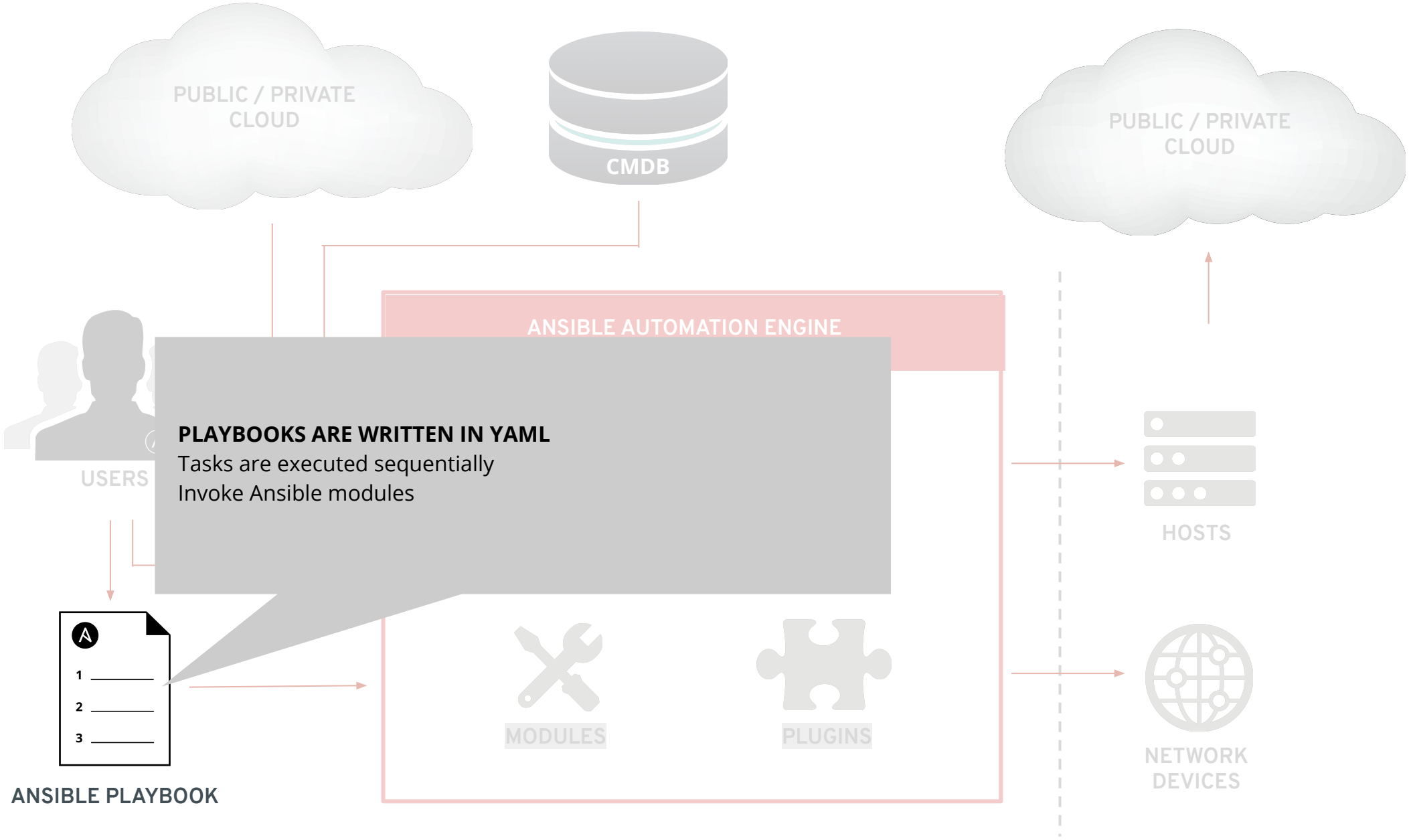
Topics Covered:

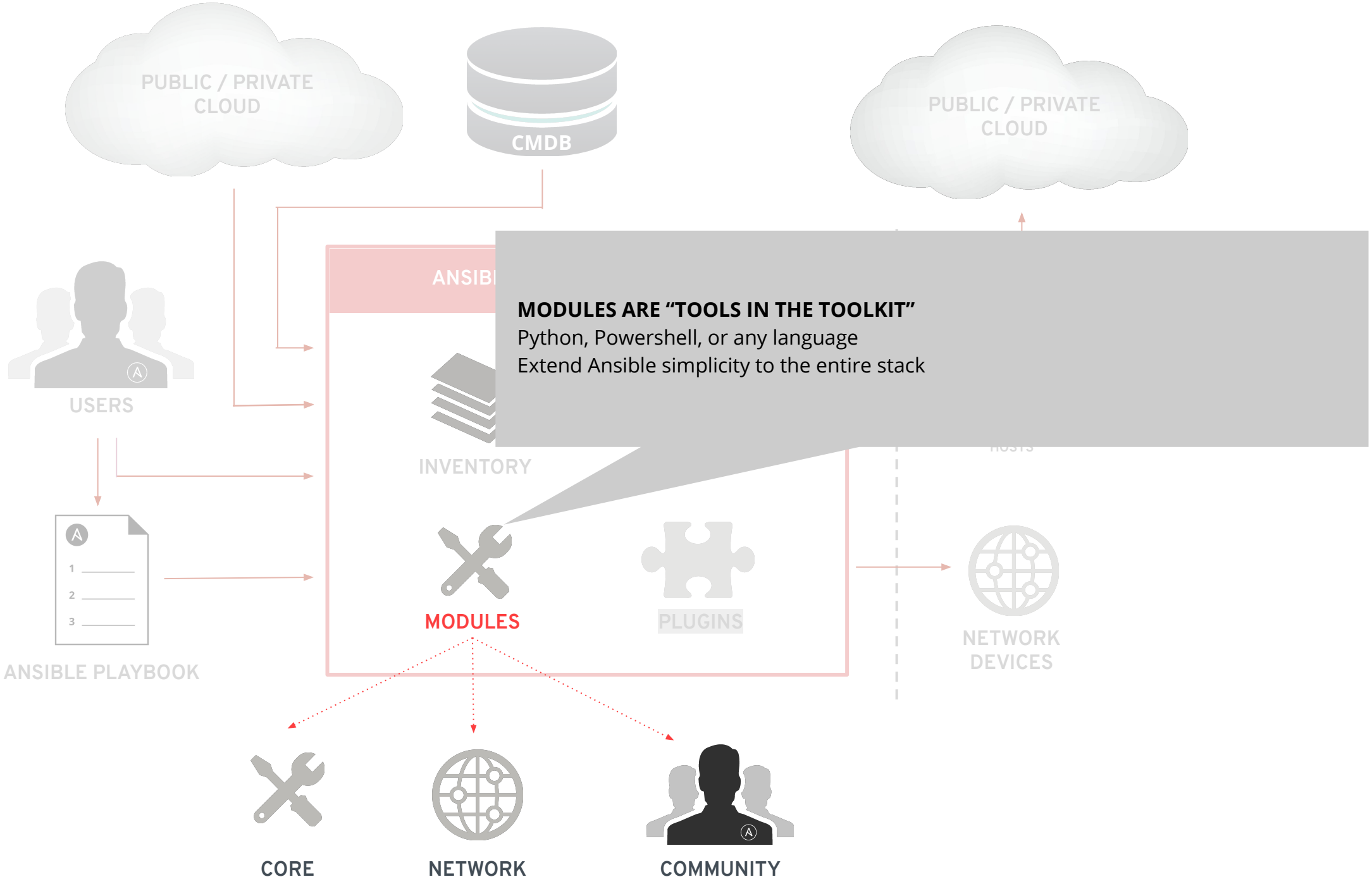
- Understanding the Ansible Infrastructure
- Check the prerequisites

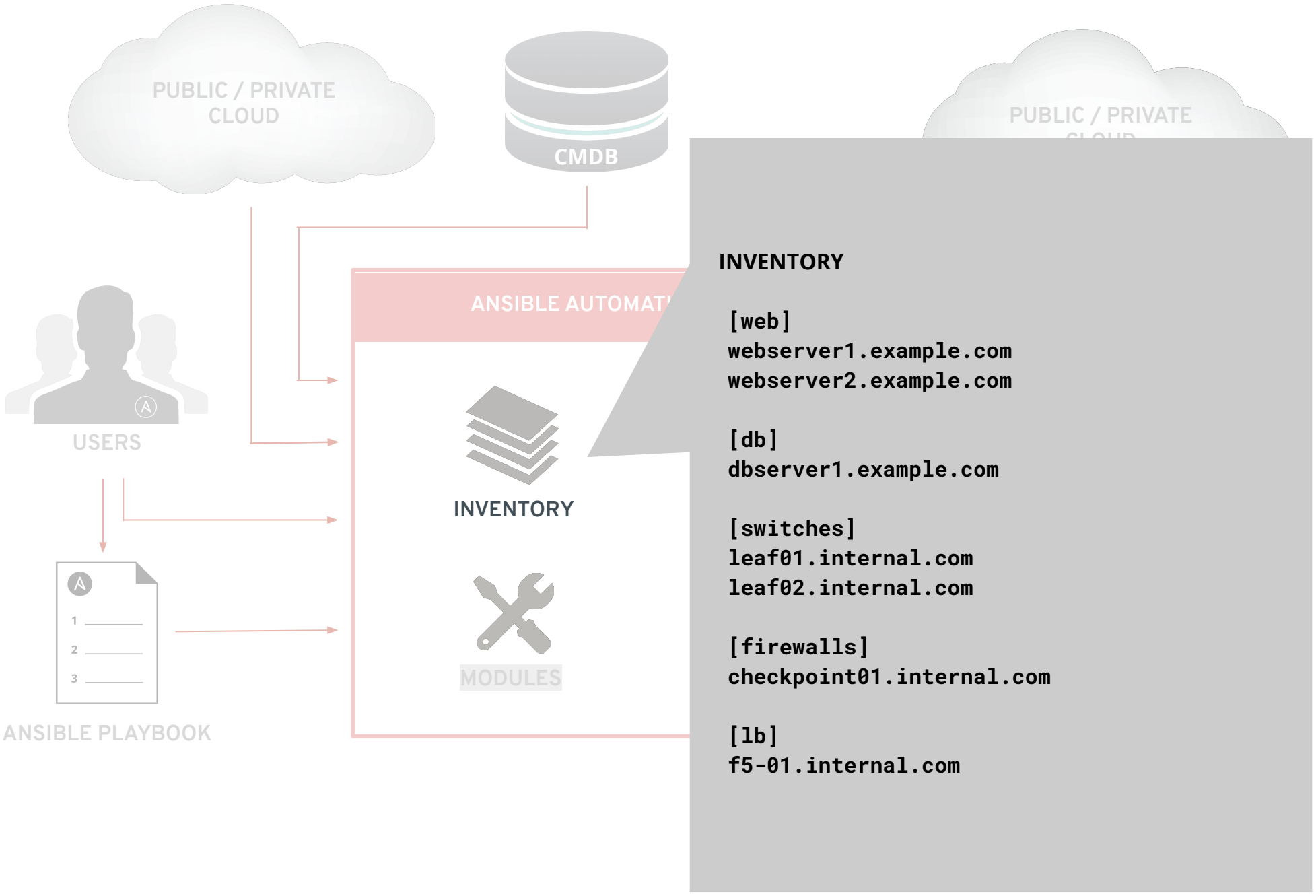
Red Hat
Ansible
Automation

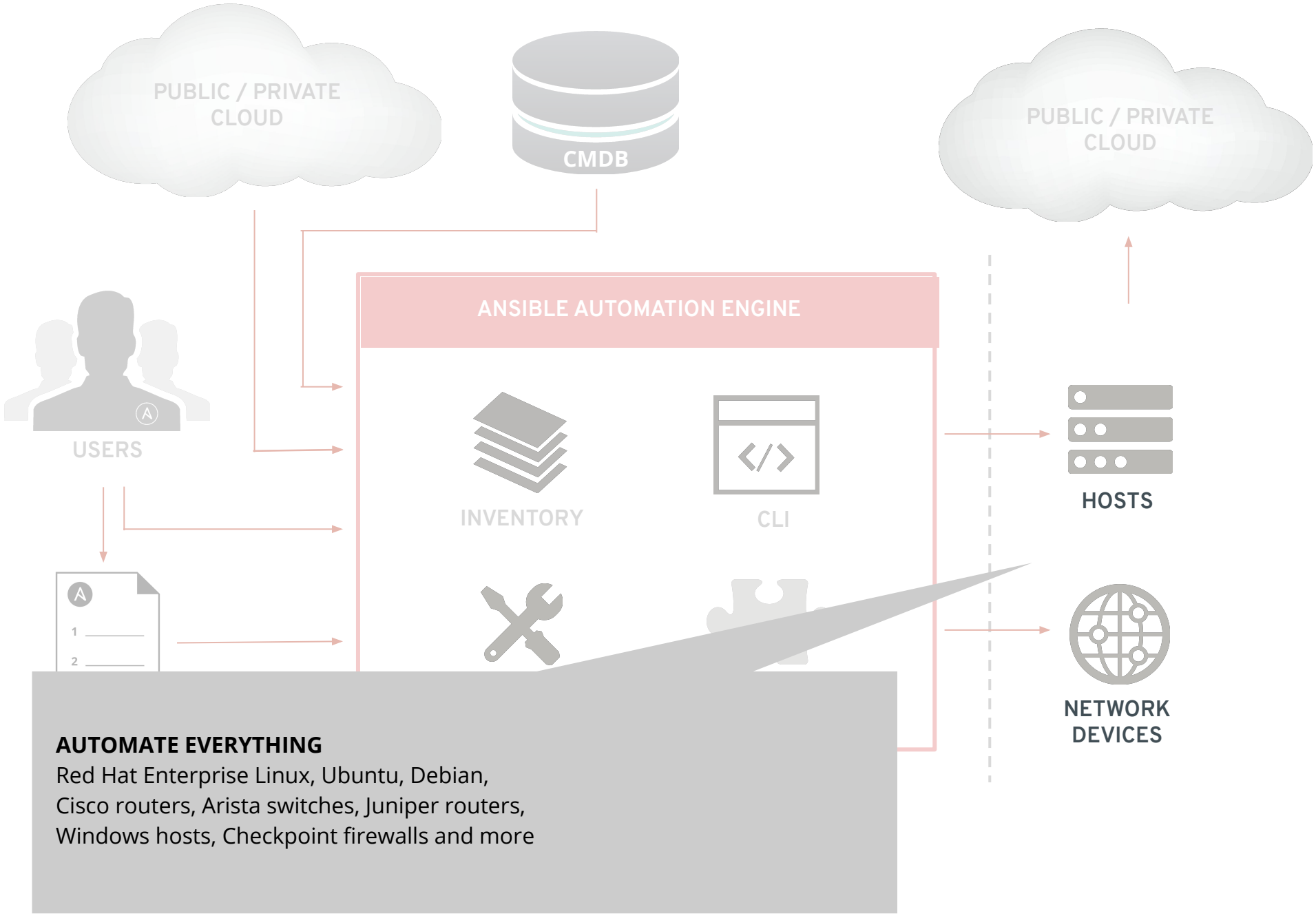
Red Hat


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


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
PLAYBOOKS ARE WRITTEN IN YAML
Tasks are executed sequentially
Invoke Ansible modules
HOSTS
1
2
3
MODULES
PLUGINS
NETWORK DEVICES
ANSIBLE PLAYBOOK


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
USERS
INVENTORY
MODULES ARE "TOOLS IN THE TOOLKIT"
Python, Powershell, or any language
Extend Ansible simplicity to the entire stack
MODULES
PLUGINS
NETWORK DEVICES
ANSIBLE PLAYBOOK
CORE
NETWORK
COMMUNITY


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE
ANSIBLE AUTOMAT
USERS
INVENTORY
MODULES
ANSIBLE PLAYBOOK
1
2
3
INVENTORY
[web]
webserver1.example.com
webserver2.example.com
[db]
dbserver1.example.com
[switches]
leaf01.internal.com
leaf02.internal.com
[firewalls]
checkpoint01.internal.com
[lb]
f5-01.internal.com


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
INVENTORY
CLI
HOSTS
NETWORK DEVICES
AUTOMATE EVERYTHING
Red Hat Enterprise Linux, Ubuntu, Debian,
Cisco routers, Arista switches, Juniper routers,
Windows hosts, Checkpoint firewalls and more

LINUX AUTOMATION

150+

Linux Modules

AUTOMATE EVERYTHING LINUX

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

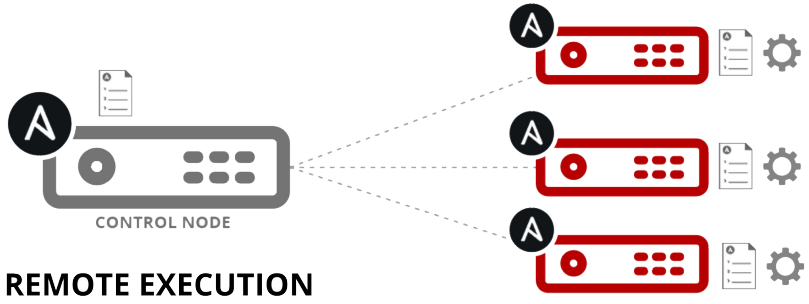
ONLY REQUIREMENTS:
Python 2 (2.6 or later)
or Python 3 (3.5 or later)

ansible.com/get-started

Red Hat

How Ansible Linux Automation works

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



LINUX HOSTS

Red Hat

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

Exercise Time - Do Exercise 1.1 Now In Your Lab Environment!

Exercise 1.2

Topics Covered:

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

Red Hat
Ansible
Automation

Red Hat

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 - 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=kev
ansible ssh private key file=/home/kev/.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=/tmpwsdir

[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
```

Host variables apply to the host and override group vars

Group variables apply for all devices in that group

Red Hat

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
```

Understanding Inventory - Groups

There is always a group called **"all"** by default

```
[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
… and about another 100 lines
```

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 “web1”
[user@ansible] $ ansible web1 -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)

... thousands of modules...
```

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*
```

Exercise Time - Do Exercise 1.2 Now In Your Lab Environment!

Red Hat

Exercise 1.3

Topics Covered:

- Playbooks basics
- Running a playbook

Red Hat
Ansible
Automation

An Ansible Play in an Ansible Playbook

A play

Another play

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

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

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.

Connections:

`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)

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

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)

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

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)

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

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	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.

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
```

Ansible Tasks Using Modules:

```
---
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] ******************************************
ok: [web2]
ok: [web1]
ok: [web3]

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

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

PLAY RECAP **********************************************************************
webservers                 : ok=3    changed=3 unreachable=0   failed=0
```

The “Setup” module

The “yum” module

The “copy” module

The “service” module

Exercise Time - Do 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

Red Hat

An Ansible Playbook Variable Example

```
---
- hosts: all

  vars:
    var_one: one is the loneliest number
    var_two: two can be as sad as one
    var_three: three dog night said that
    var_four: "{{ var_three }} {{ var_one }}"
    var_five: "and that {{ var_two }}."
```

three dog night said that one is the loneliest number
and that two can be as sad as one.

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)*

Facts

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

Gather facts on target machine

```
$ ansible -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"
        ],
```

Exercise Time - Do Exercise 1.4 Now In Your Lab Environment!

Red Hat

Exercise 1.5

Topics Covered:

- Conditionals
- Handlers
- Loops

Red Hat
Ansible
Automation

Advanced Playbooks: Conditionals via VARS

Choose your own adventure, based on variables, facts and more!

```
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
```

Advanced Playbooks: Conditionals via FACTS

Choose your own adventure, based on variables, facts and more!

```
tasks:
- name: Install apache
  apt:
    name: {{ item }}
    state: latest
  with_items:
   - apache2
  when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'

- name: Install httpd
  yum:
    name: {{ item }}
    state: latest
  with_items:
   - httpd
  when: ansible_distribution == 'Red Hat Enterprise Linux'
```

Advanced Playbooks: (this is not an example of) Handler Tasks

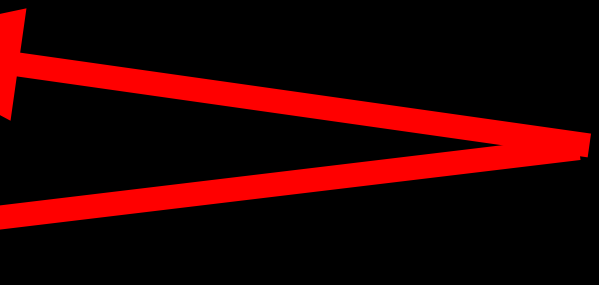
This is NOT a handler task, but has similar function

```
- 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
```

Advanced Playbooks: 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
  when: httpd_results.changed
```



Advanced Playbooks: Handler Tasks

What happens when a handler task is called?

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

```
TASK [Ensure httpd package is present] ***************************************
ok: [web2]
ok: [web1]

TASK [Standardized index.html file] ******************************************
ok: [web2]
ok: [web1]

NOTIFIED: [restart_httpd] *** ************************************************
ok: [web2]
ok: [web1]
```

UNCHANGED

CHANGED

HANDLER RUNS ONCE

Advanced Playbooks: Handler Tasks

What happens when a handler task is called more than once?

```
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] ***************************************
ok: [web2]
ok: [web1]

TASK [Standardized index.html file] ******************************************
ok: [web2]
ok: [web1]

NOTIFIED: [restart_httpd] *** ************************************************
ok: [web2]
ok: [web1]
```

CHANGED

CHANGED

HANDLER RUNS ONCE

Advanced Playbooks: Handler Tasks

What happens when no tasks notify a handler task?

```
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 one** of these tasks notifies of a **changed** result, the handler task ***does not run.***

```
TASK [Ensure httpd package is present] ******************************************
ok: [web2]
ok: [web1]

TASK [Standardized index.html file] *********************************************
ok: [web2]
ok: [web1]

PLAY RECAP ********************************************************************
web2       : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web1       : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

UNCHANGED

UNCHANGED

HANDLER DOESN'T RUN AT ALL

Advanced Playbooks: Variables & Loops

Using loops to save time with tasks

```
- yum:
    name: httpd
    state: latest

- yum:
    name: httpd-tools
    state: latest

- yum:
    name: mysql-server
    state: latest

- yum:
    name: php56-mysql
    state: latest
```

Red Hat

Advanced Playbooks: Variables & Loops

Using loops to save time with tasks

```
- name: ensure a list of packages installed
  yum:
    name: "{{ packages }}"
    state: latest
  vars:
    packages:
    - httpd
    - httpd-tools
    - mysql-server
    - php56-mysqlnd
    - php56-common
    - php56-xml
```

Advanced Playbooks: Variables & Loops

Using loops to save time with tasks

```
vars:
  bad_packages:
  - "@^gnome-desktop-environment"
  - make
  - gcc
  - tftp-server
  - telnet-server

tasks:
- name: list of bad packages are not present
  yum:
    name: "{{ bad_packages }}"
    state: absent
  check_mode: yes
```

Exercise Time - Do Exercise 1.5 Now In Your Lab Environment!

Exercise 1.6

Topics Covered:

- Templates

Red Hat
Ansible
Automation

Red Hat

Advanced Playbooks: 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 }}
```

Exercise Time - Do 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

Red Hat

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

Exercise Time - Do Exercise 1.7 Now In Your Lab Environment!

Exercise 1.8

Topics Covered:

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

Red Hat
Ansible
Automation

You are on your own!

You know it all - now use it!

Exercise Time - Do Exercise 1.8 Now In Your Lab Environment!

Red Hat

Section 2
Tower

Red Hat

Exercise 2.1

Topics Covered:

- Introduction to Tower

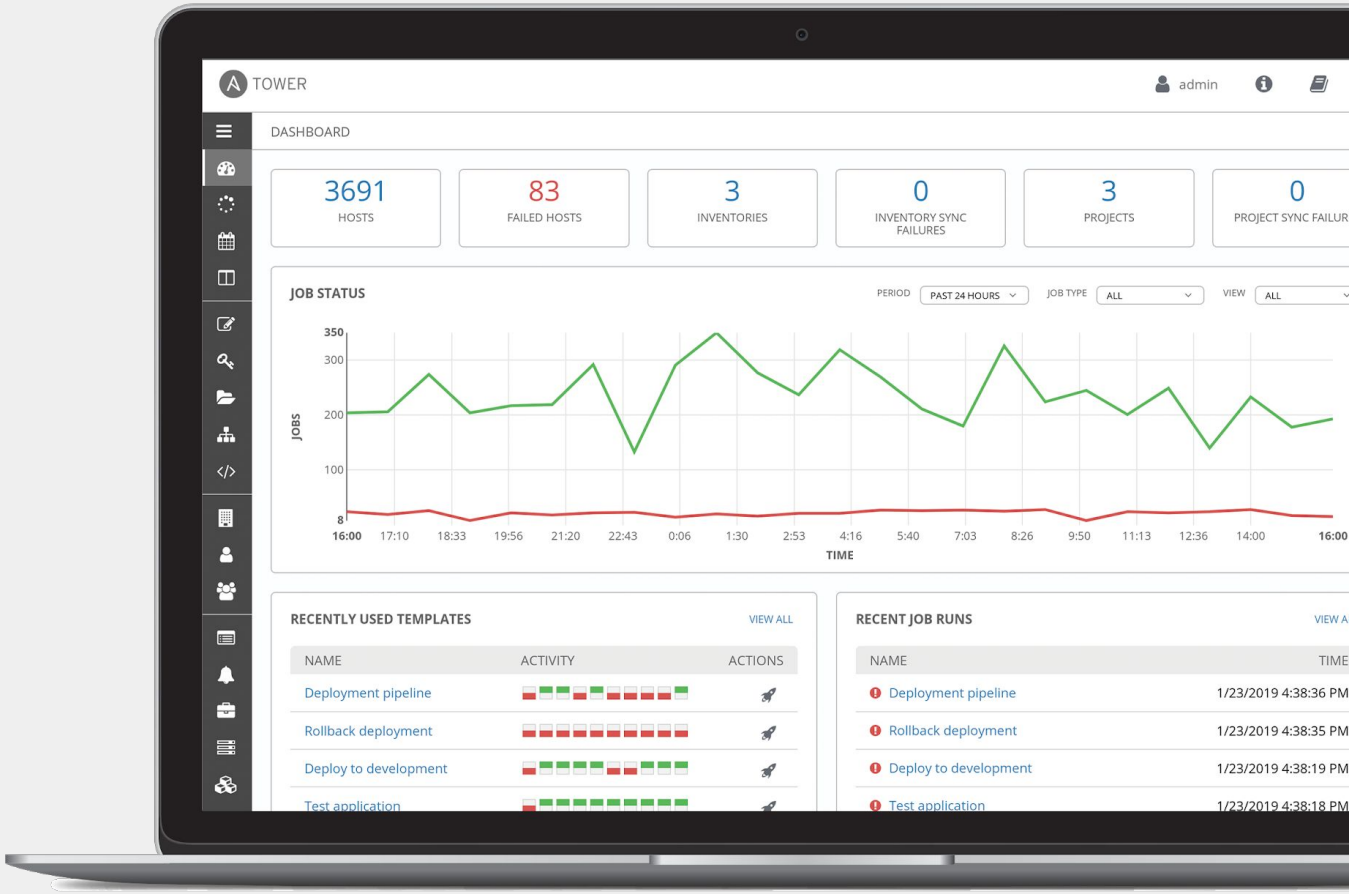
Red Hat
Ansible
Automation

Red Hat

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

Red Hat Ansible Tower

RBAC

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

Push button

An intuitive user interface experience makes it easy for novice users to execute playbooks you allow them access to.

RESTful API

With an API first mentality every feature and function of Tower can be API driven. Allow seamless integration with other tools like ServiceNow and Infoblox.

Workflows

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

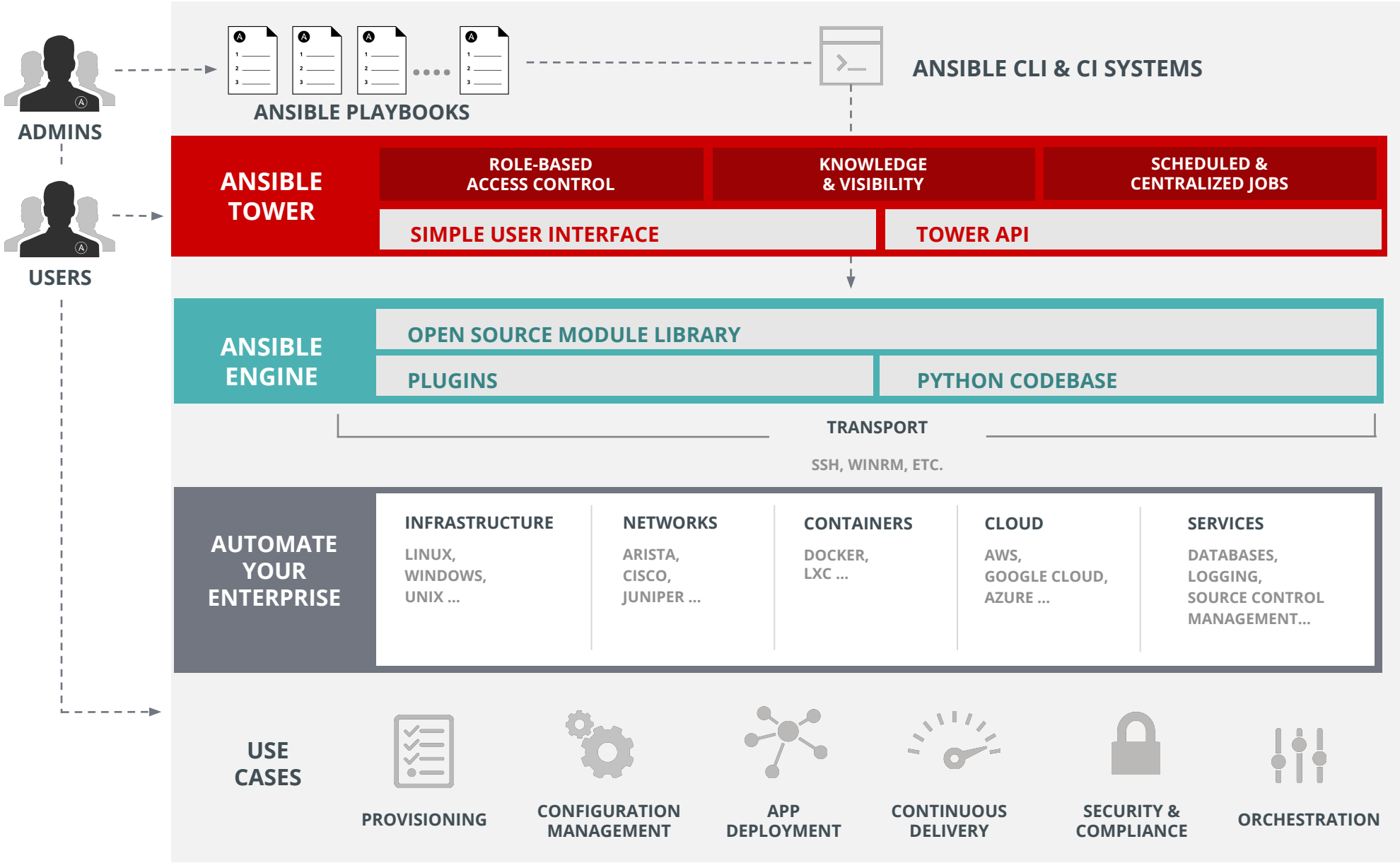
Enterprise integrations

Integrate with enterprise authentication like TACACS+, RADIUS, Azure AD. Setup token authentication with OAuth 2. Setup notifications with PagerDuty, Slack and Twilio.

Centralized logging

All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Ansible Tower's API.

Red Hat


ADMINS
USERS
ANSIBLE PLAYBOOKS
ANSIBLE CLI & CI SYSTEMS
ANSIBLE TOWER
ROLE-BASED ACCESS CONTROL
KNOWLEDGE & VISIBILITY
SCHEDULED & CENTRALIZED JOBS
SIMPLE USER INTERFACE
TOWER API
ANSIBLE ENGINE
OPEN SOURCE MODULE LIBRARY
PLUGINS
PYTHON CODEBASE
TRANSPORT
SSH, WINRM, ETC.
AUTOMATE YOUR ENTERPRISE
INFRASTRUCTURE
LINUX, WINDOWS, UNIX ...
NETWORKS
ARISTA, CISCO, JUNIPER ...
CONTAINERS
DOCKER, LXC ...
CLOUD
AWS, GOOGLE CLOUD, AZURE ...
SERVICES
DATABASES, LOGGING, SOURCE CONTROL MANAGEMENT...
USE CASES
PROVISIONING
CONFIGURATION MANAGEMENT
APP DEPLOYMENT
CONTINUOUS DELIVERY
SECURITY & COMPLIANCE
ORCHESTRATION

Exercise Time - Do Exercise 2.1 Now In Your Lab Environment!

Red Hat

Exercise 2.2

Topics Covered:

- Inventories
- Credentials

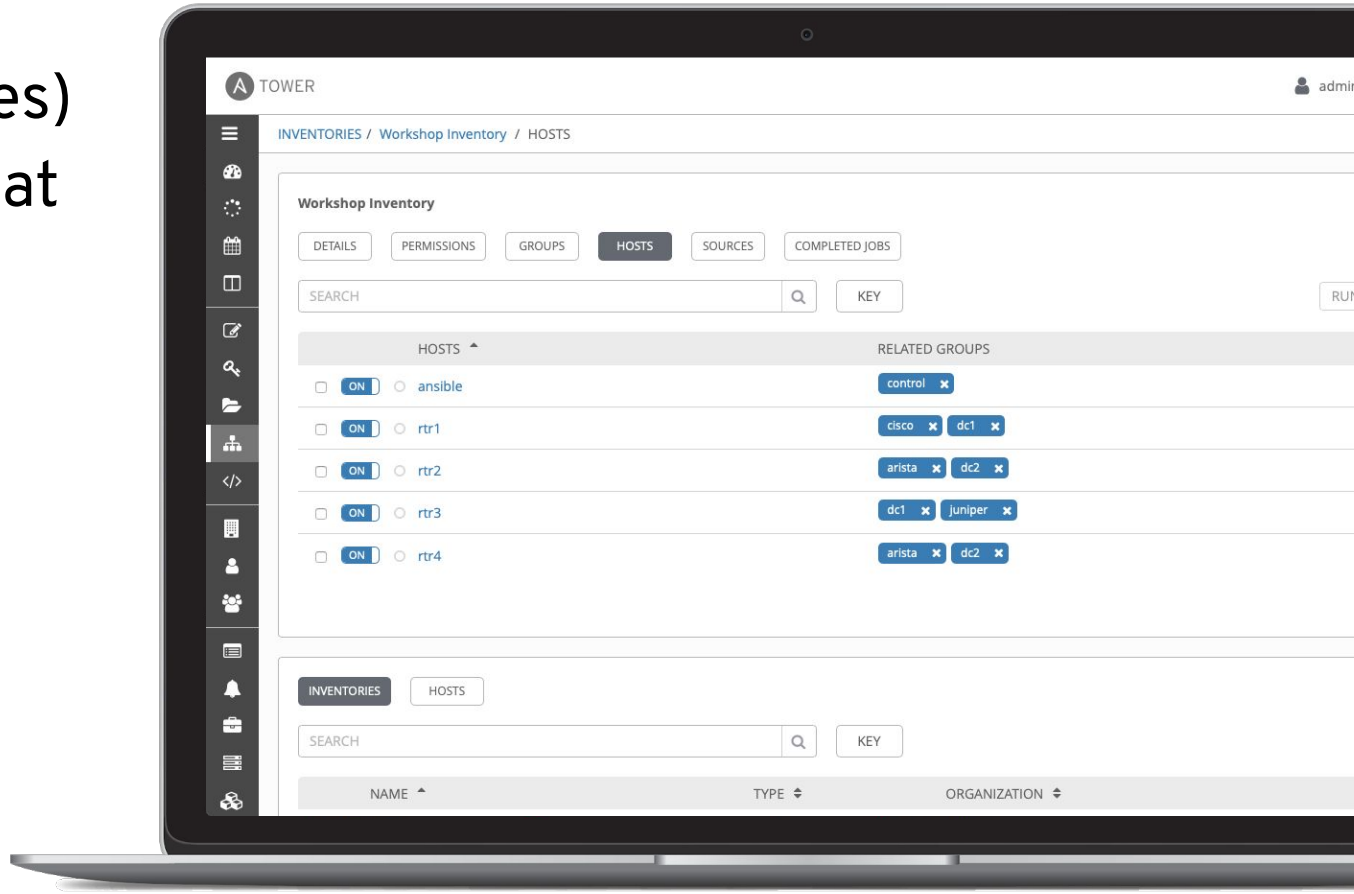
Red Hat
Ansible
Automation

Red Hat

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



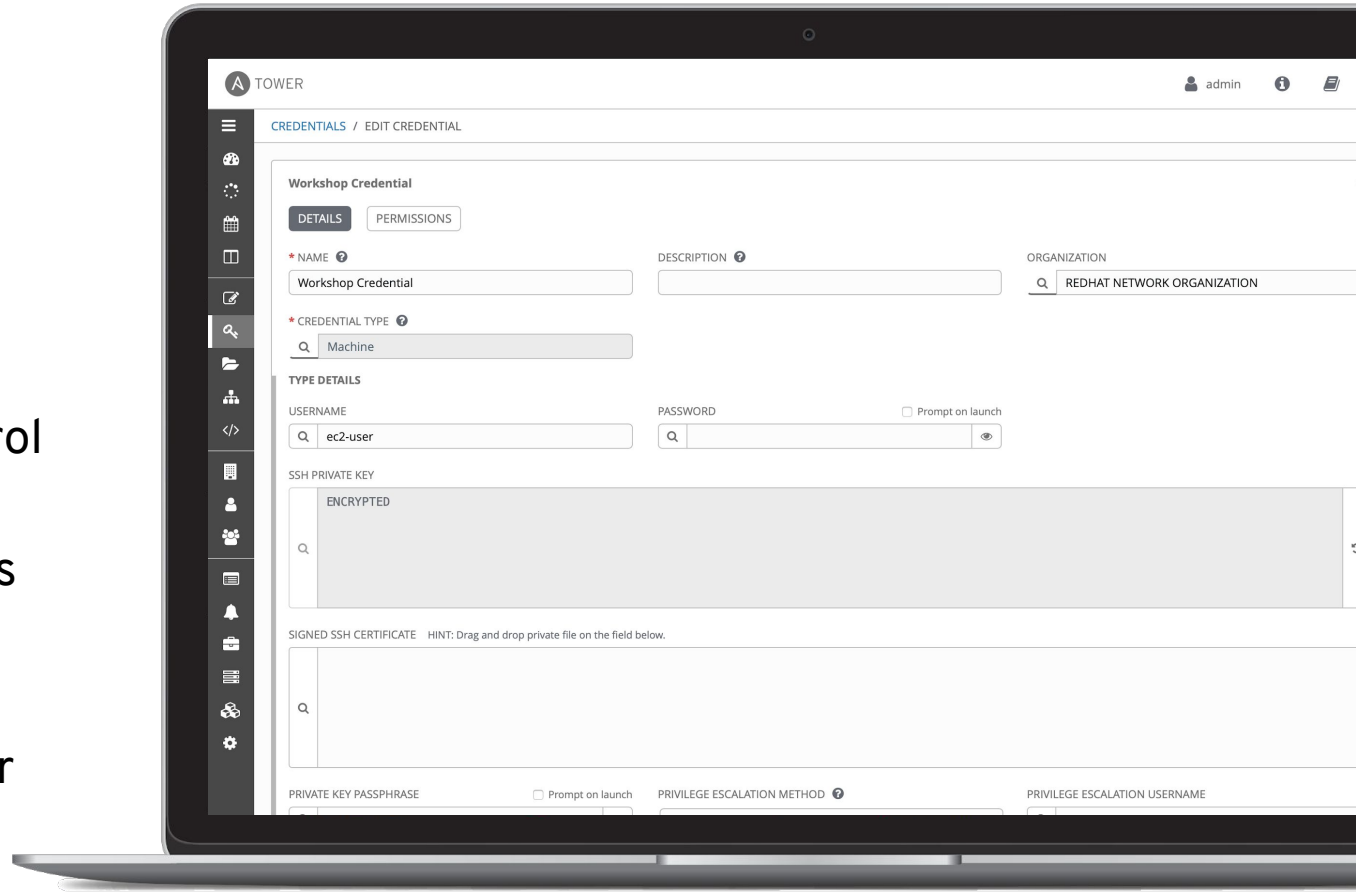
Red Hat

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

Exercise Time - Do Exercise 2.2 Now In Your Lab Environment!

Red Hat

Exercise 2.3

Topics Covered:

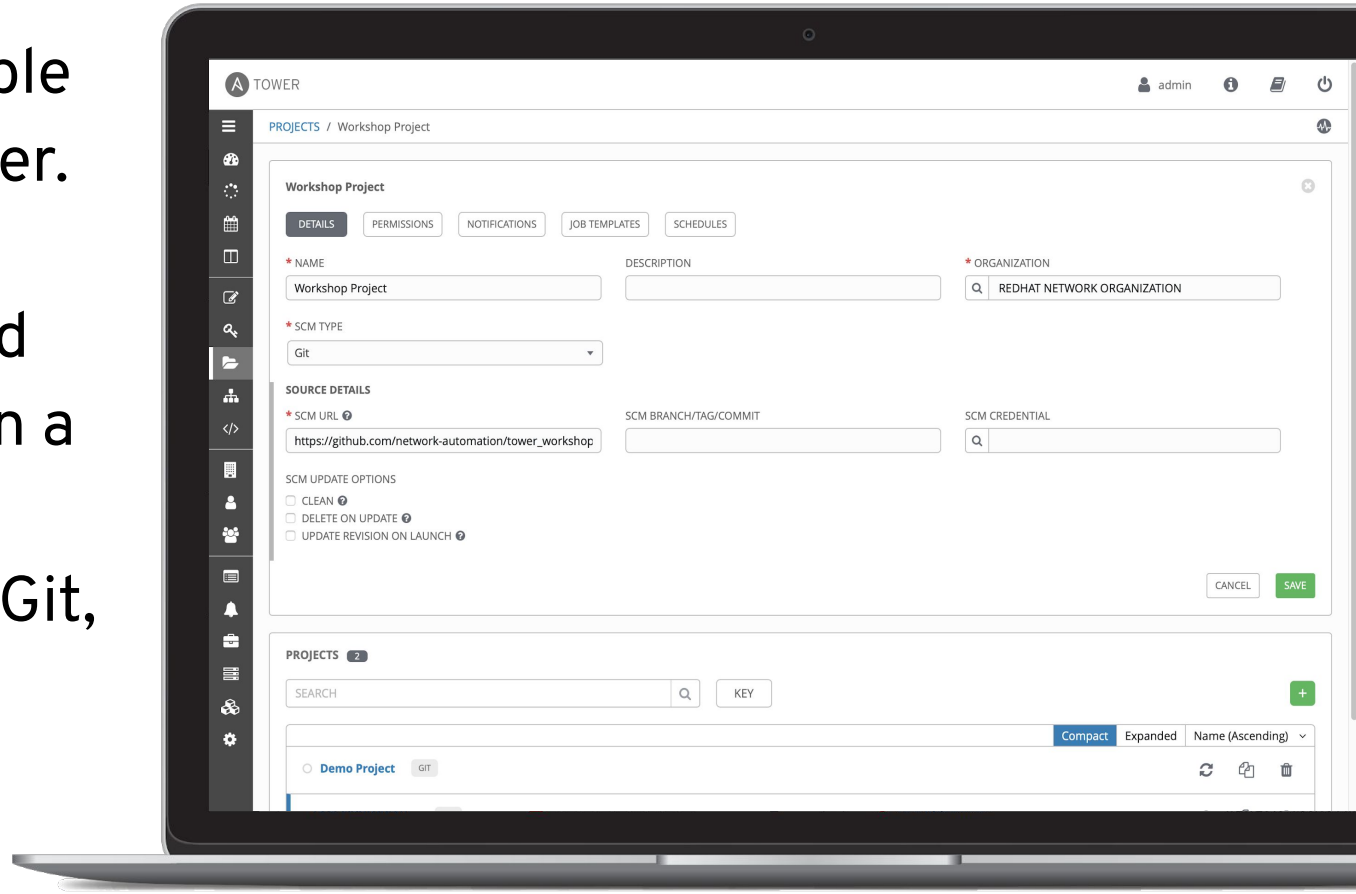
- Projects
- Job Templates

Red Hat
Ansible
Automation

Projects

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

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



Red Hat

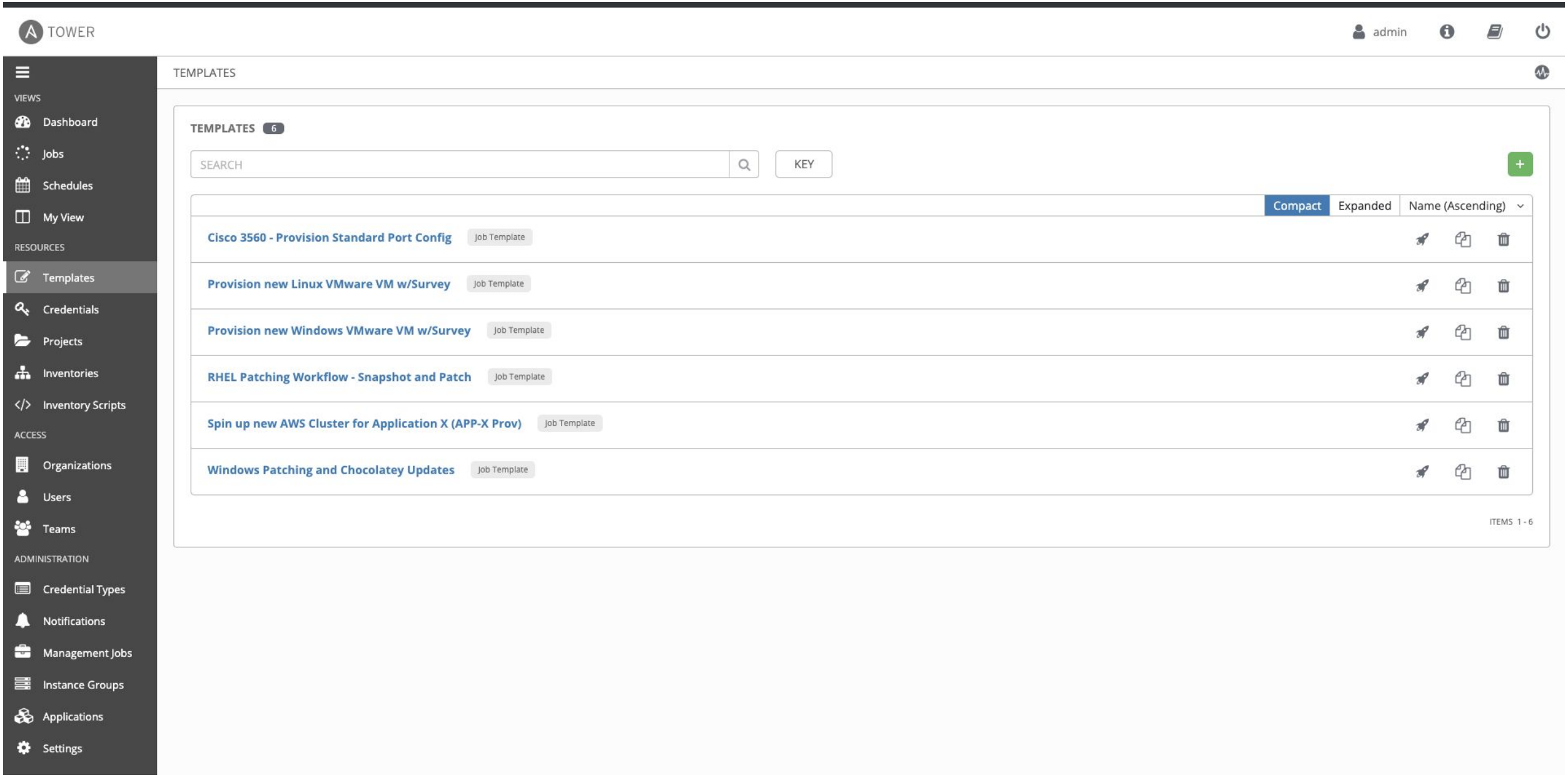
Job Templates

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

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

A **Job Template** requires:

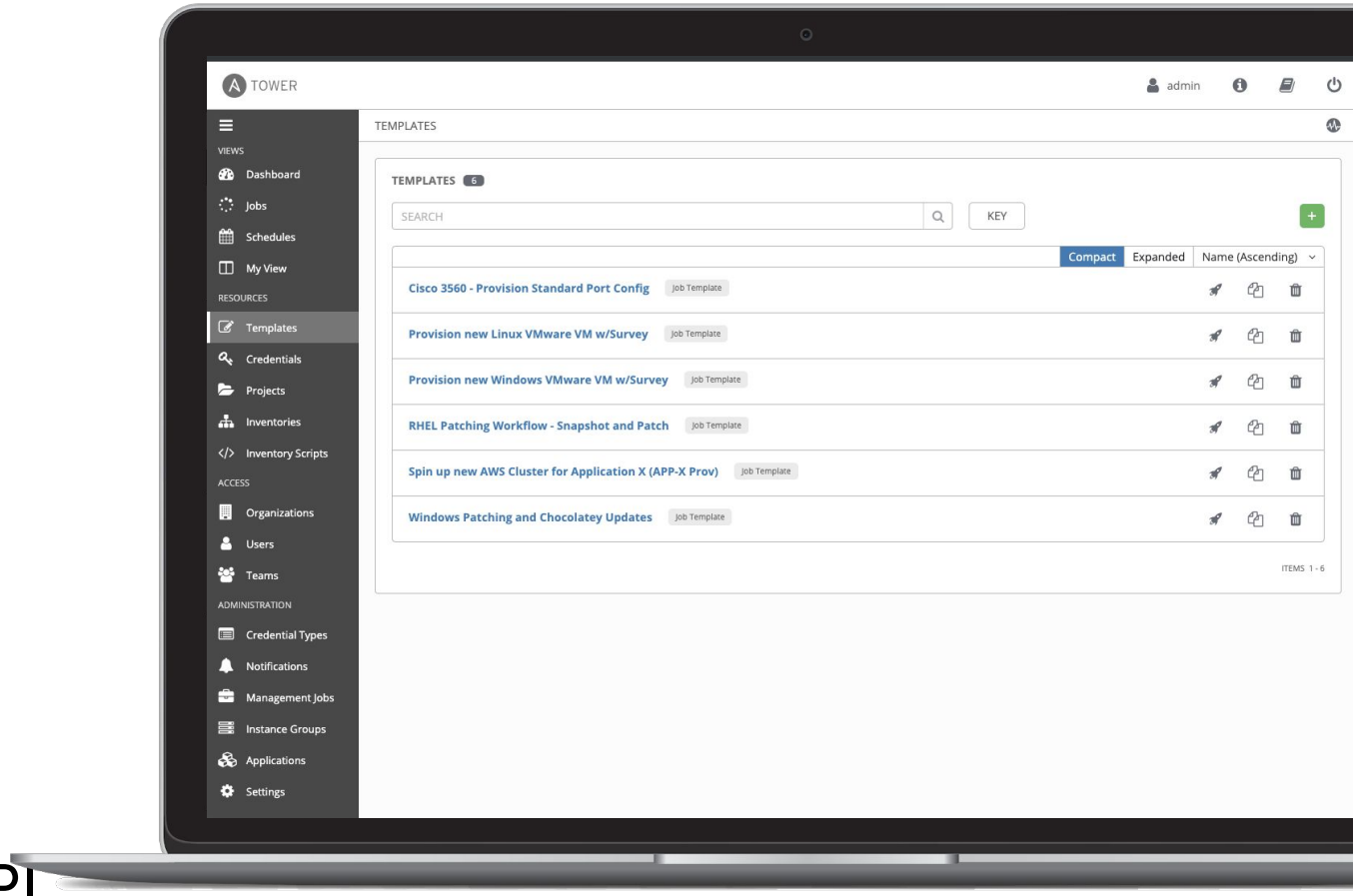
- An **Inventory** to run the job against
- A **Credential** to login to devices.
- A **Project** which contains Ansible Playbooks


TOWER
admin
TEMPLATES
VIEWS
Dashboard
Jobs
Schedules
My View
RESOURCES
Templates
Credentials
Projects
Inventories
Inventory Scripts
ACCESS
Organizations
Users
Teams
ADMINISTRATION
Credential Types
Notifications
Management Jobs
Instance Groups
Applications
Settings
TEMPLATES 6
SEARCH
KEY
Compact
Expanded
Name (Ascending)
Cisco 3560 - Provision Standard Port Config
Job Template
Provision new Linux VMware VM w/Survey
Job Template
Provision new Windows VMware VM w/Survey
Job Template
RHEL Patching Workflow - Snapshot and Patch
Job Template
Spin up new AWS Cluster for Application X (APP-X Prov)
Job Template
Windows Patching and Chocolatey Updates
Job Template
ITEMS 1 - 6

Job Templates

A Job Template is where all the pieces come together, defining how your Ansible job will run. A Job Template is made up of:

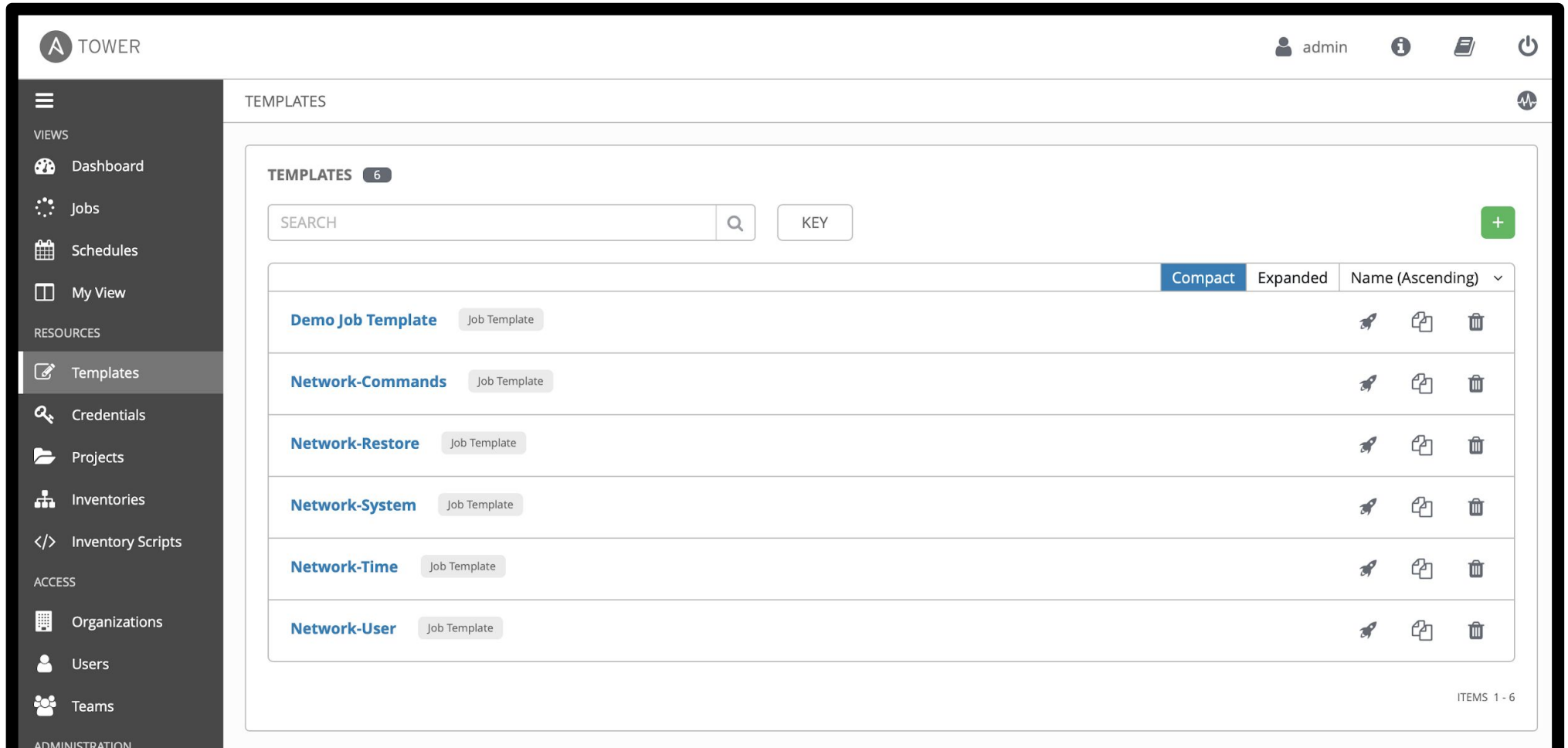
- Inventory
- Project (containing a playbook)
- Credentials
- Survey or optional vars
- Jobs can be launched via GUI or API



Red Hat

Expanding on Job Templates

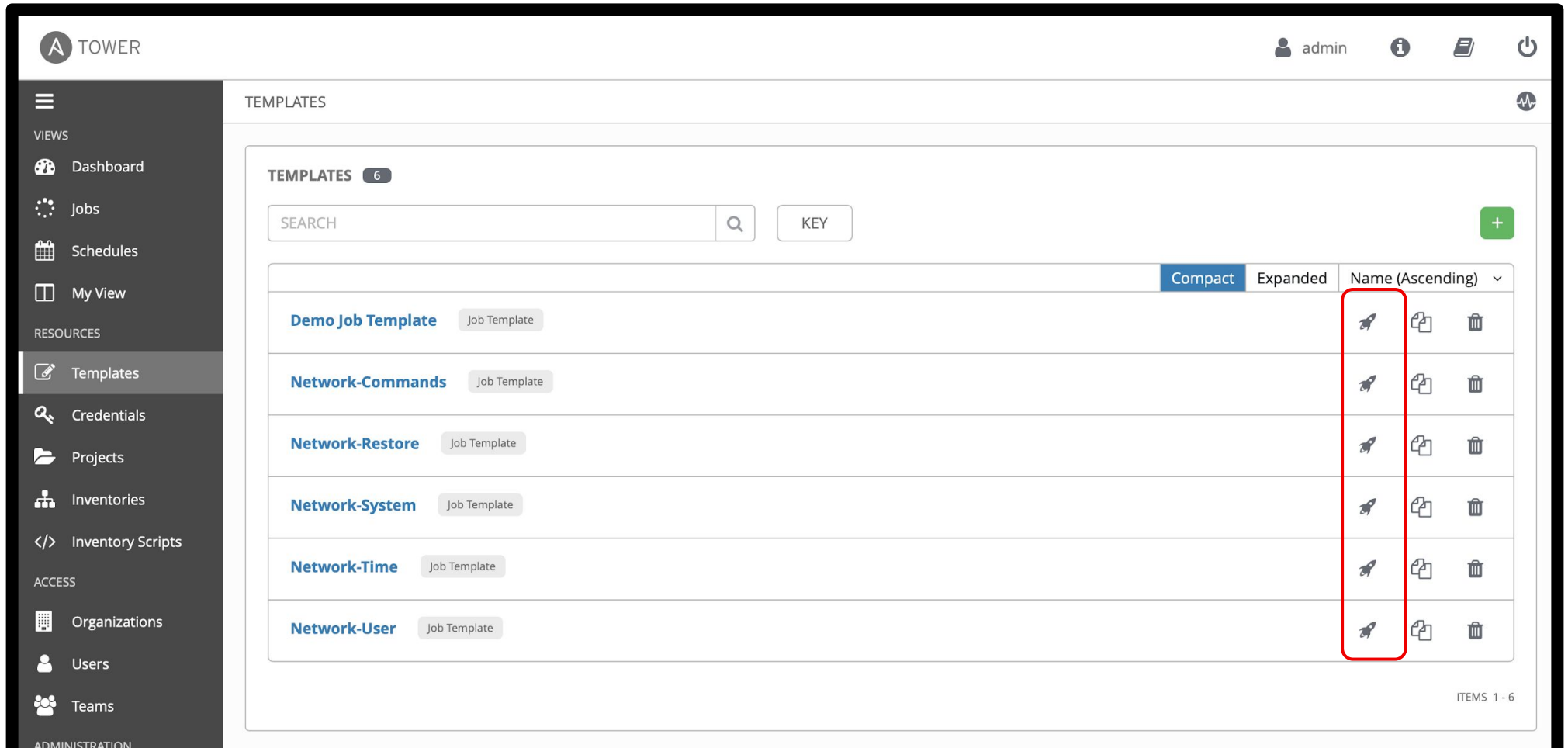
Job Templates can be found and created by clicking the **Templates** button under the *RESOURCES* section on the left menu.



Red Hat

Executing an existing Job Template

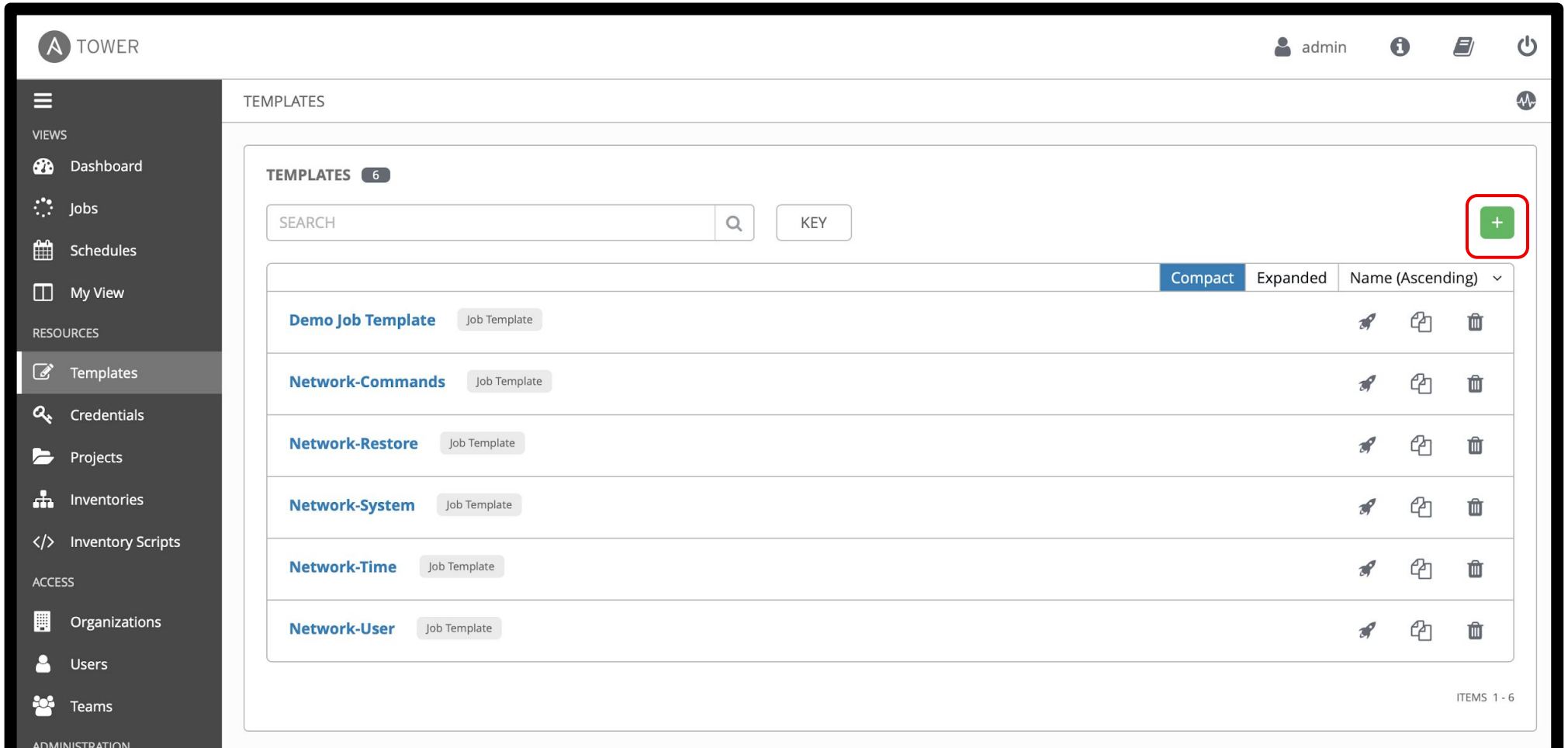
Job Templates can be launched by clicking the **rocketship button** for the corresponding Job Template



Red Hat

Creating a new Job Template (1/2)

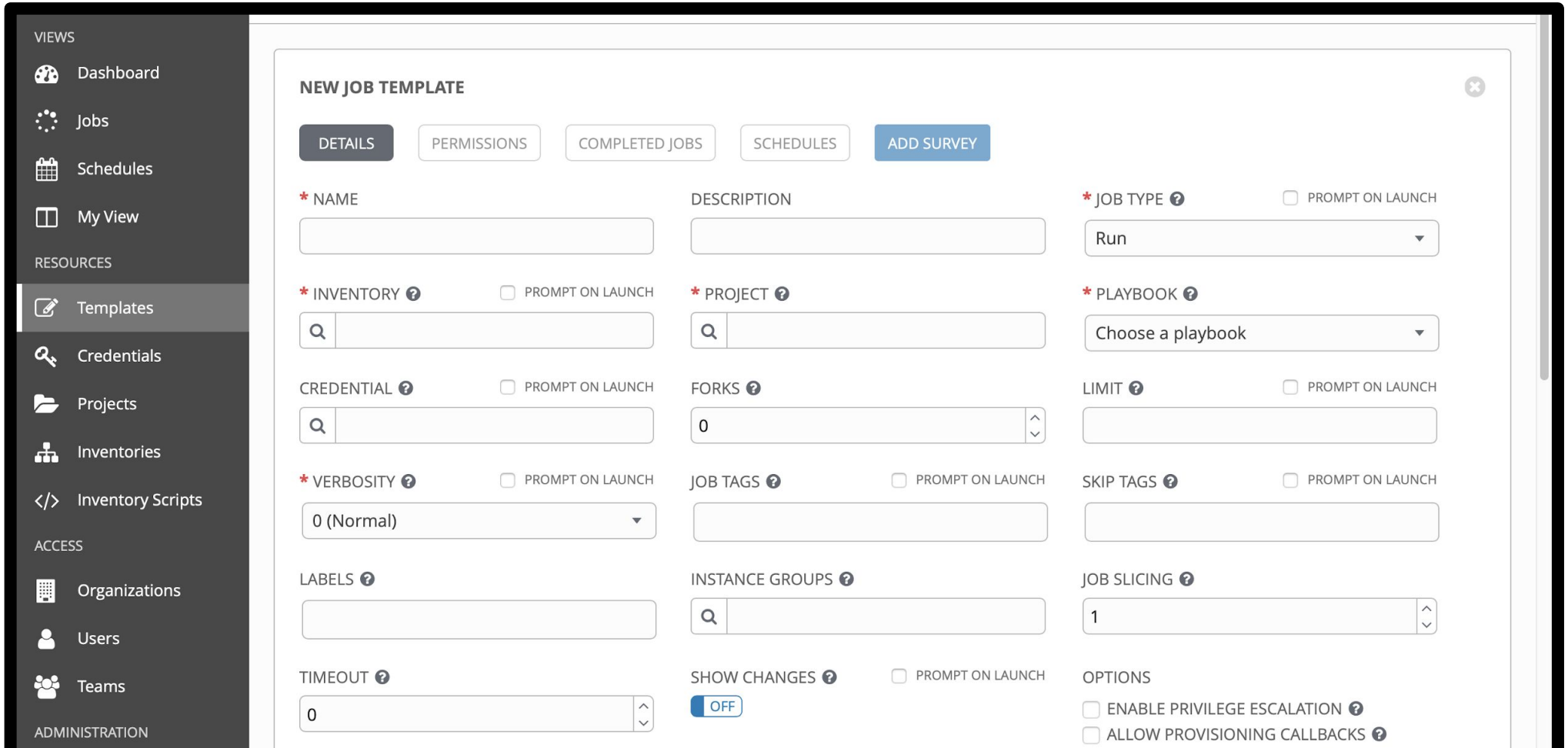
New Job Templates can be created by clicking the **plus button** 



Red Hat

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 .



Red Hat

Exercise Time - Do Exercise 2.3 Now In Your Lab Environment!

Red Hat

Exercise 2.4

Topics Covered:

- Surveys

Red Hat
Ansible
Automation

Red Hat

Surveys

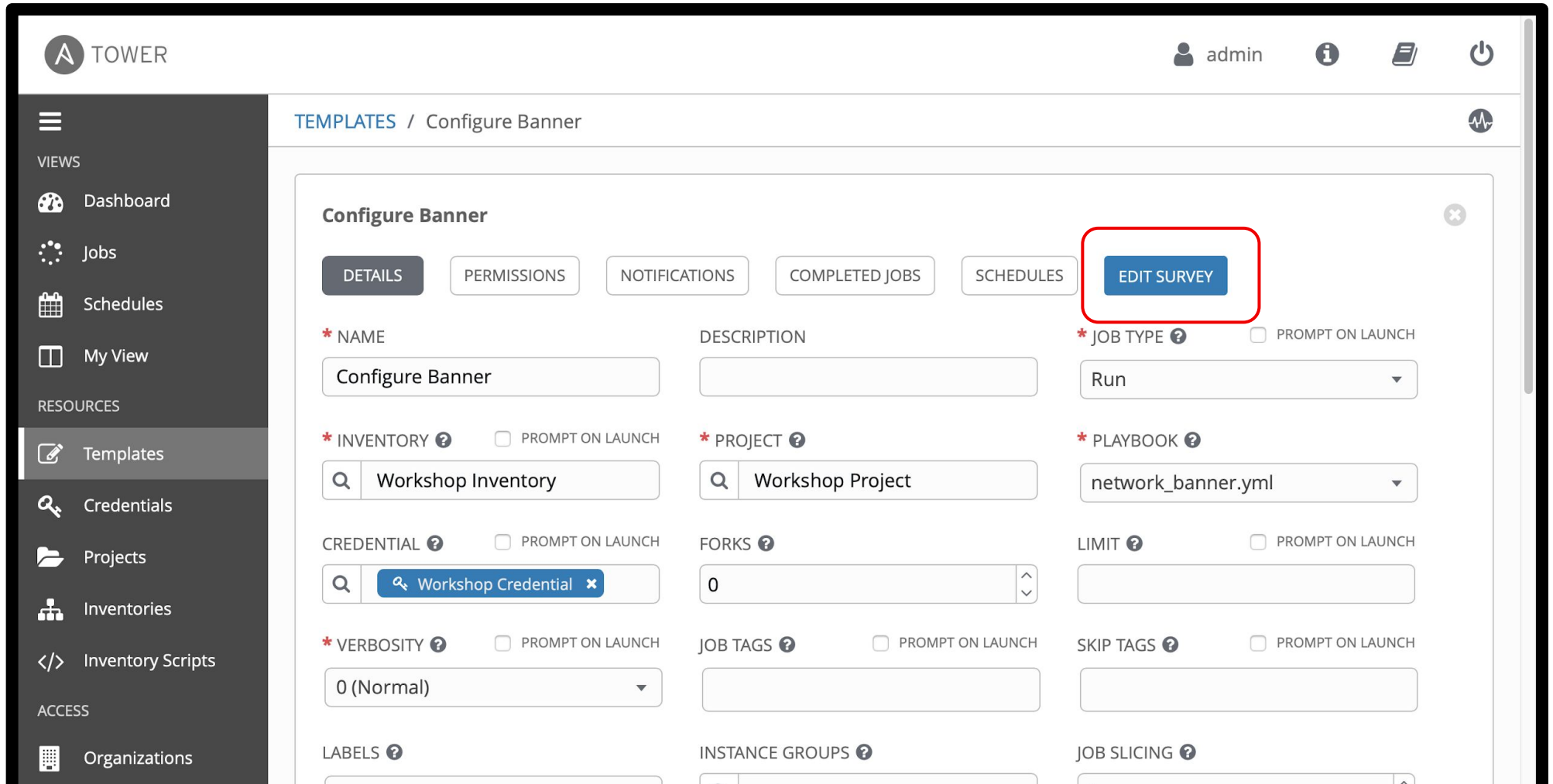
Tower surveys allow you to configure how a job runs via a series of questions, making it simple to customize your jobs in a user-friendly way.

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

Creating a Survey (1/2)

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

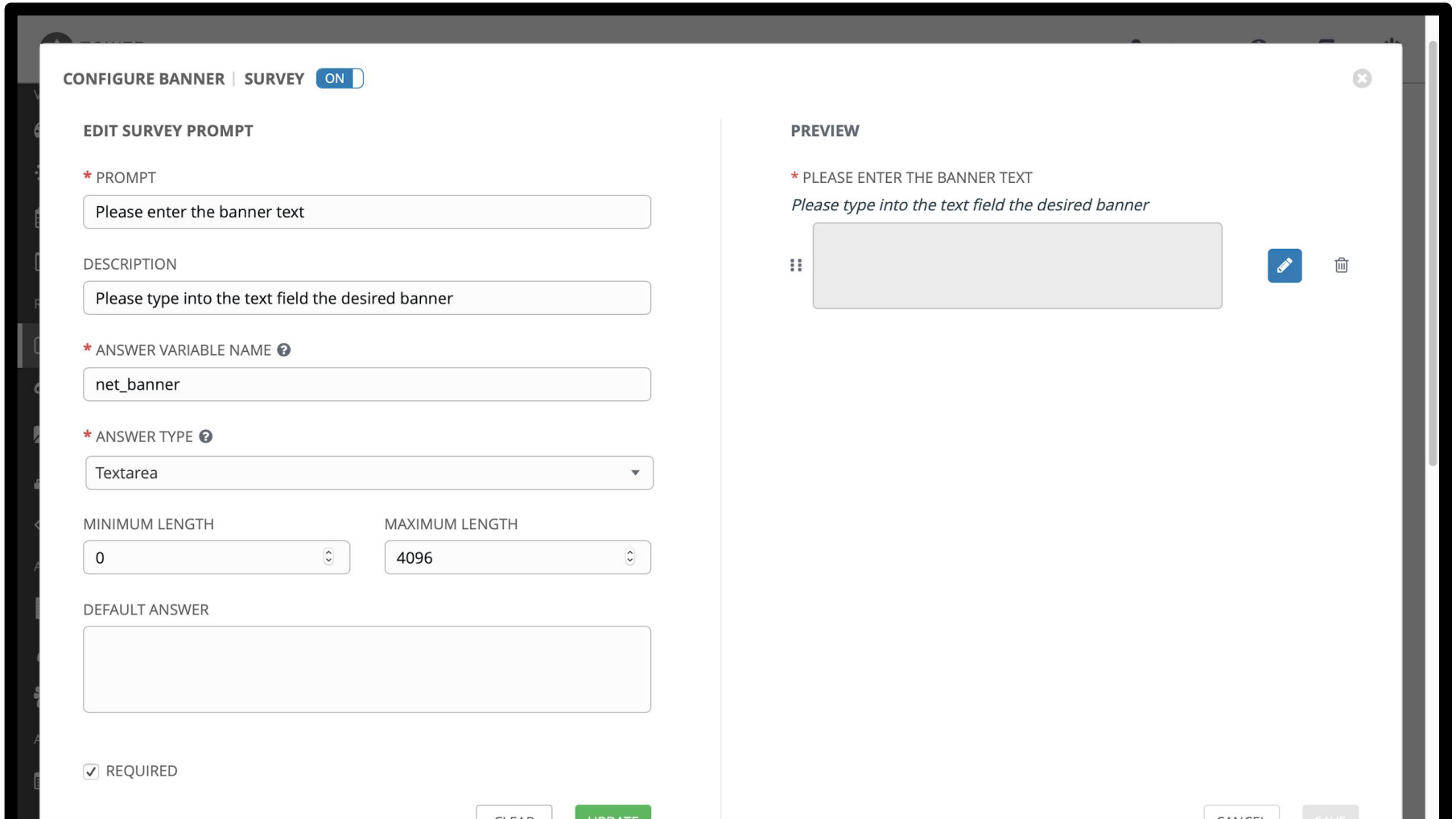
Click the button to open the Add Survey window.



Red Hat

Creating a Survey (2/2)

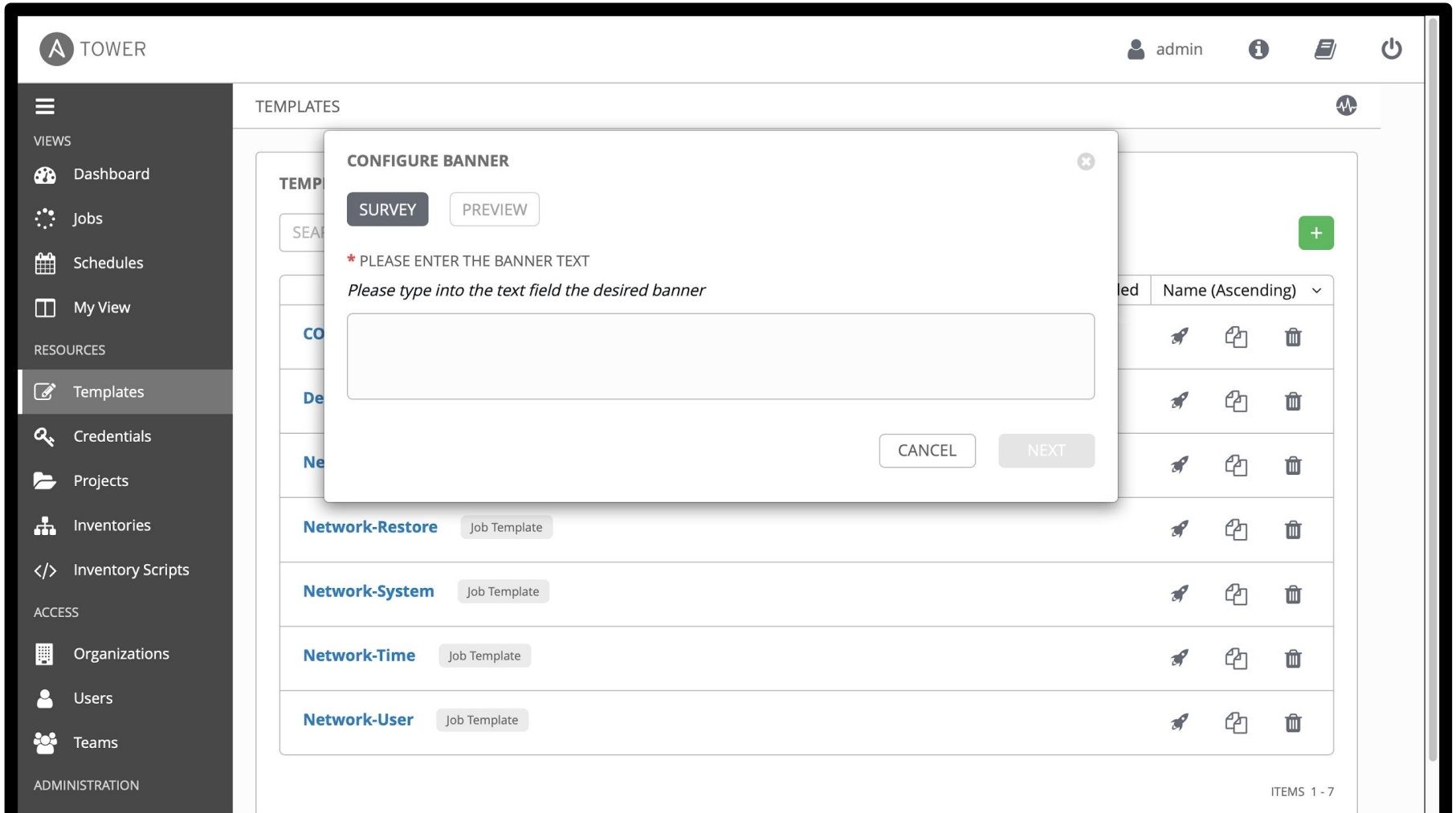
The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.



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.



Red Hat

Exercise Time - Do Exercise 2.4 Now In Your Lab Environment!

Red Hat

Exercise 2.5

Topics Covered:

- Role based access control

Red Hat
Ansible
Automation

Role Based Access Control (RBAC)

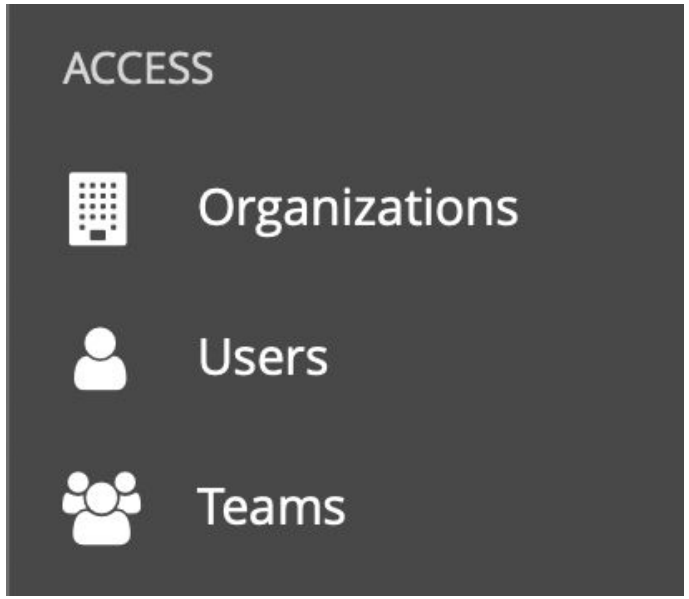
Role-Based Access Controls (RBAC) are built into Ansible Tower and allow administrators to delegate access to inventories, organizations, and more. These controls allow Ansible Tower to help you increase security and streamline management of your Ansible automation.

User Management

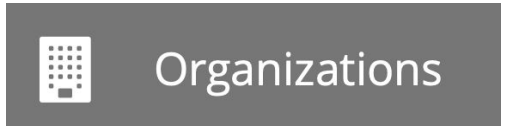
- An **organization** is a logical collection of users, teams, projects, inventories and more. All entities belong to an organization with the exception of users.

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

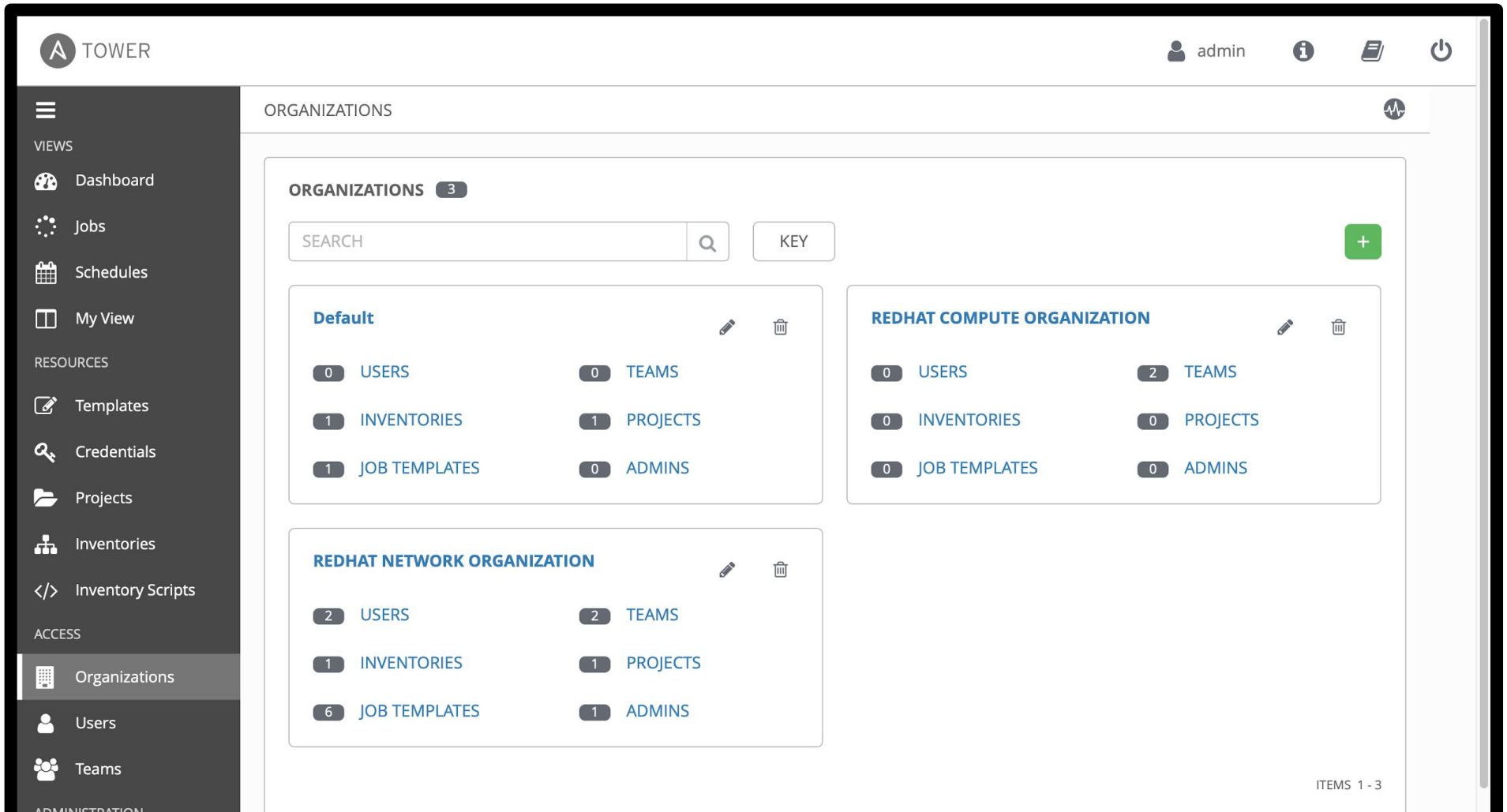
- **Teams** provide a means to implement role-based access control schemes and delegate responsibilities across organizations.



Viewing Organizations

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

in the left menu

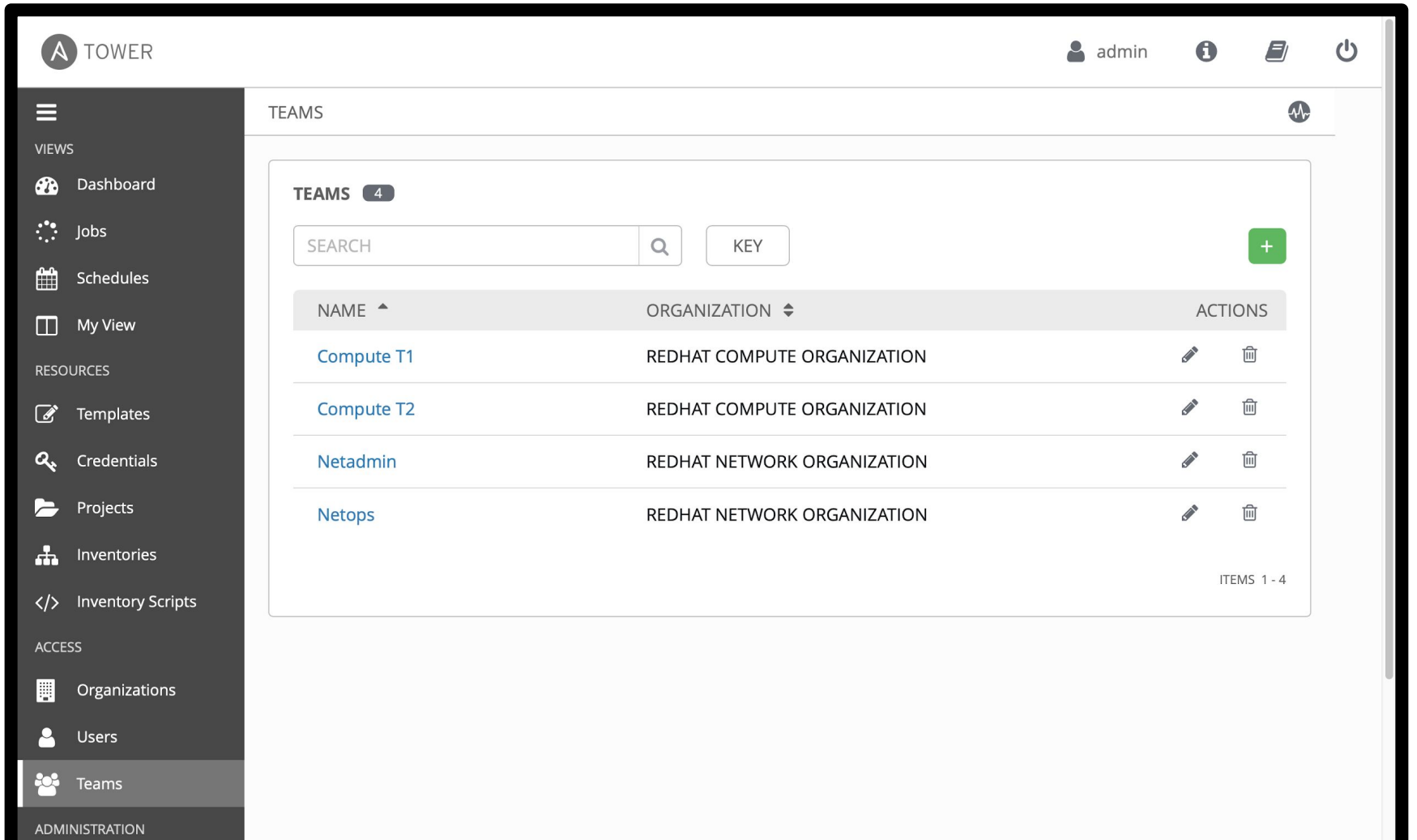


Red Hat

Viewing Teams

Clicking on the **Teams** button 

in the left menu will open up the Teams window

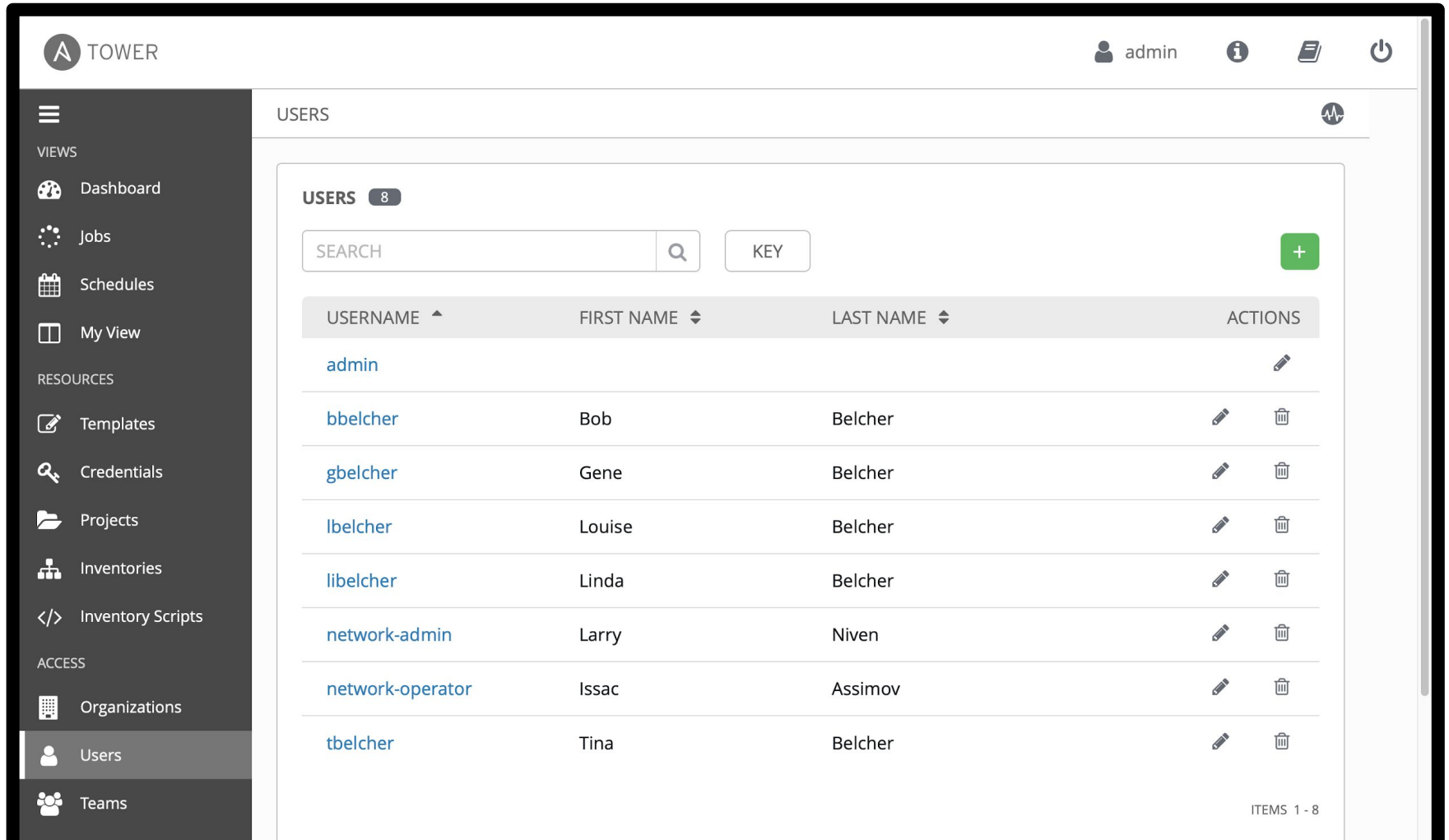


Red Hat

Viewing Users

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

in the left menu



Red Hat

Exercise Time - Do Exercise 2.5 Now In Your Lab Environment!

Red Hat

Exercise 2.6

Topics Covered:

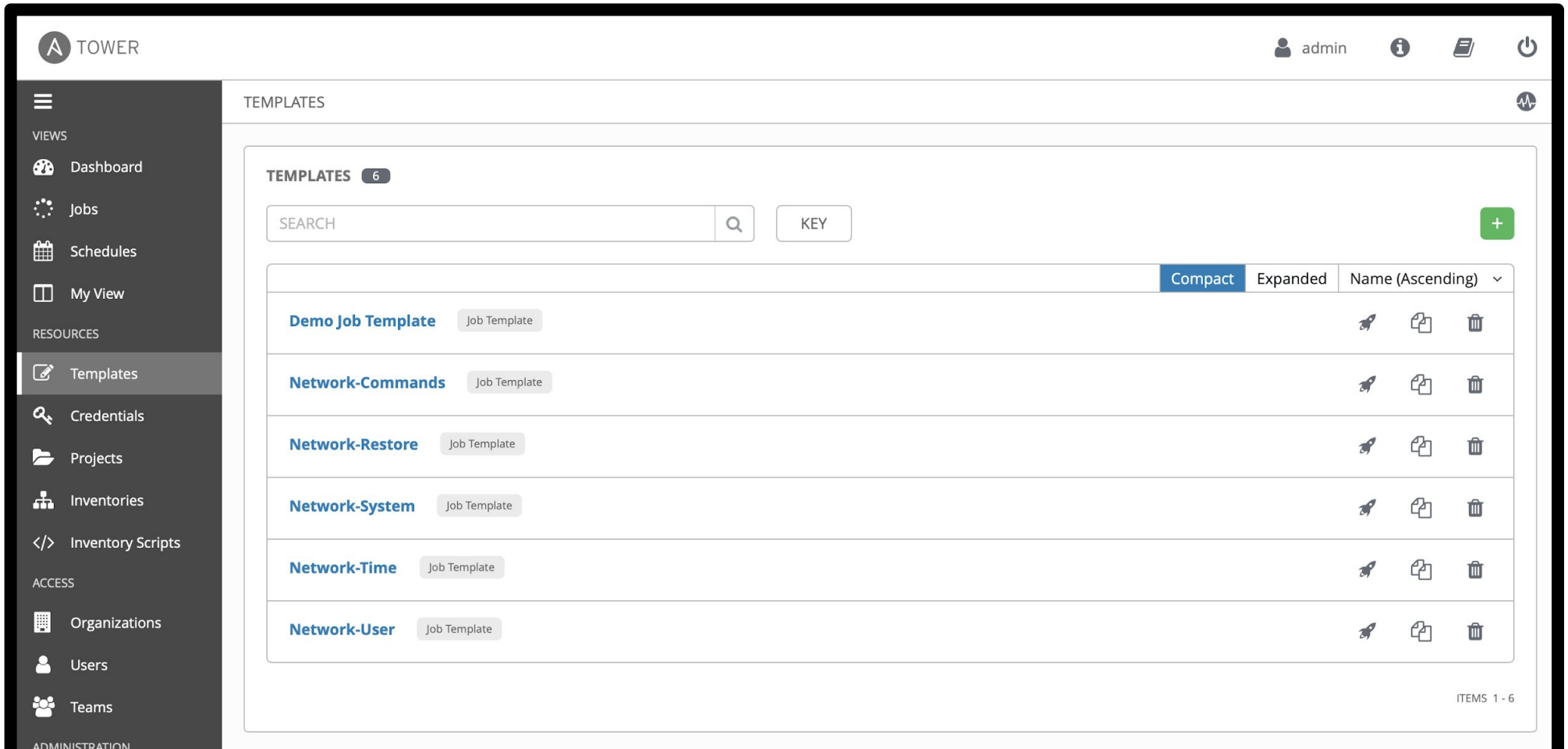
- Workflows

Red Hat
Ansible
Automation

Red Hat

Workflows

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

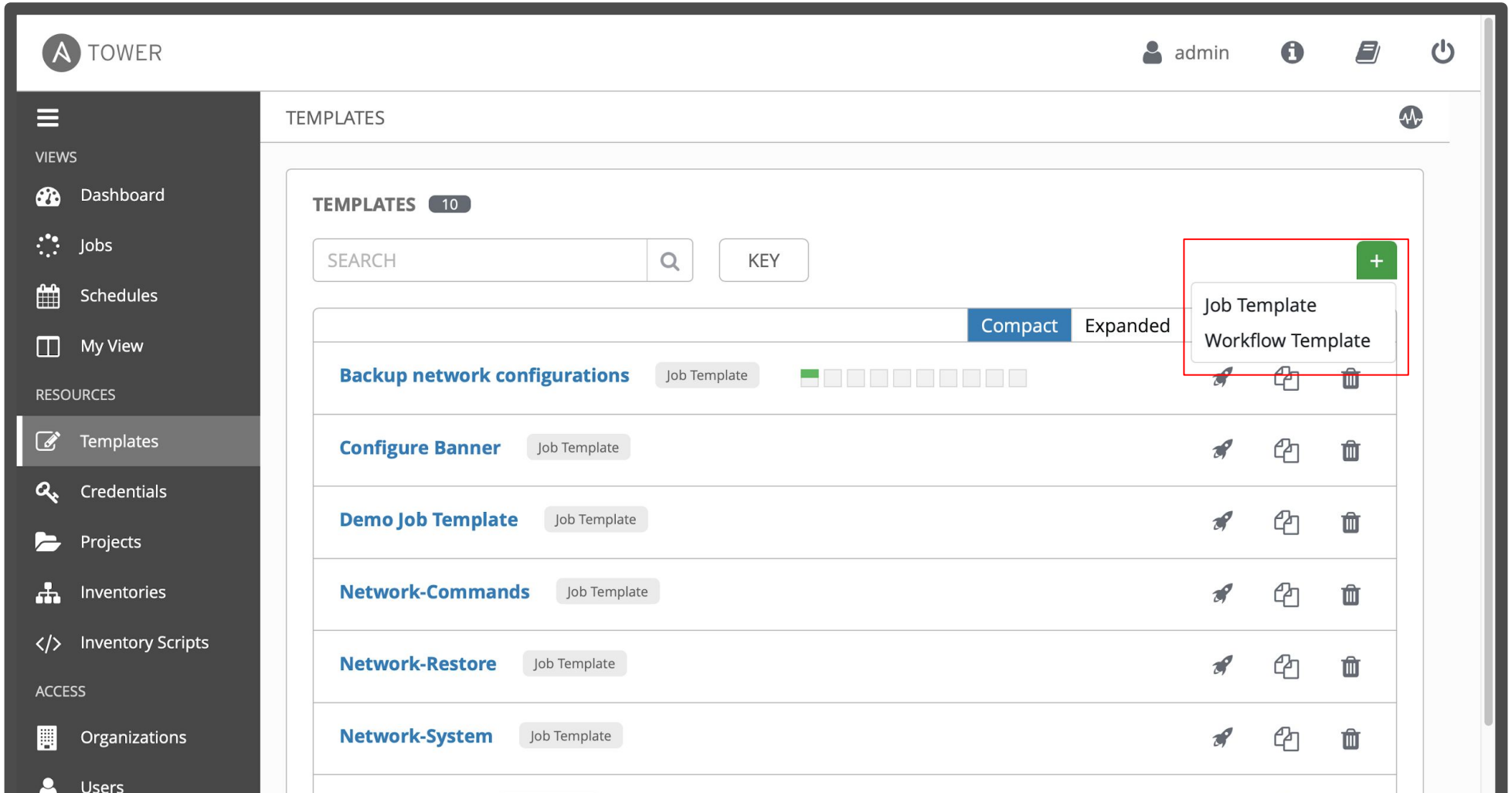


Red Hat

Adding a new Workflow Template

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

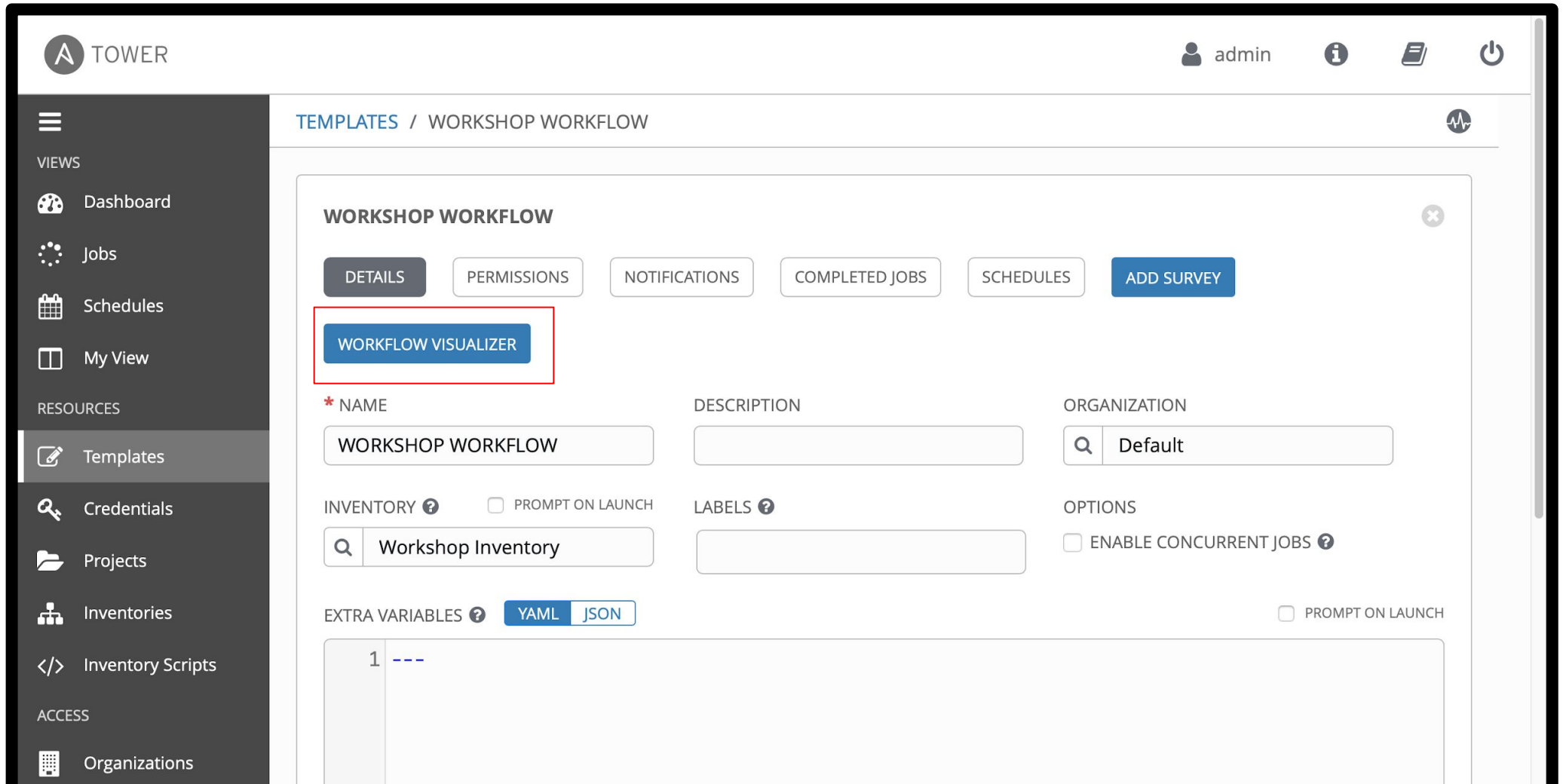
This time select the **Workflow Template**



Red Hat

Creating the Workflow

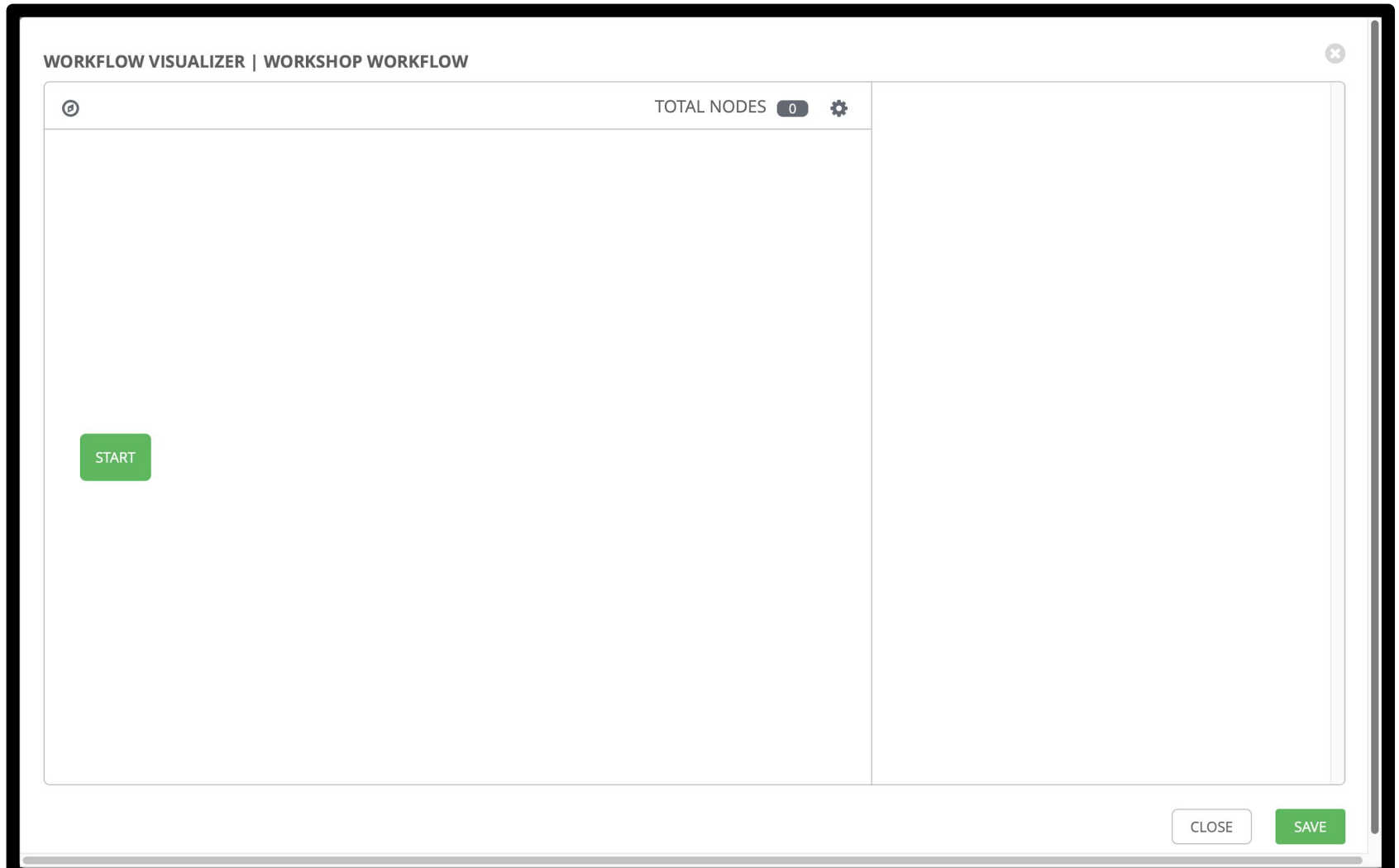
Fill out the required parameters and click **SAVE.** As soon as the Workflow Template is saved the WORKFLOW VISUALIZER will open.



Red Hat

Workflow Visualizer

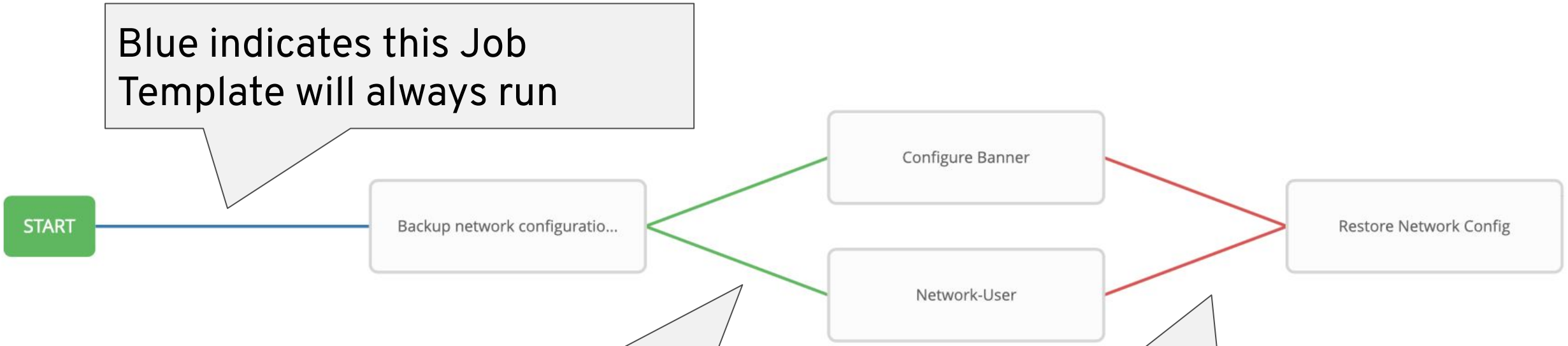
The workflow visualizer will start as a blank canvas.



Red Hat

Visualizing a Workflow

Workflows can branch out, or converge in.



Exercise Time - Do Exercise 2.6 Now In Your Lab Environment!

Red Hat

Exercise 2.7

Topics Covered:

- Wrap-up

Red Hat
Ansible
Automation

You are on your own!

You know it all - now use it!

Exercise Time - Do Exercise 2.7 Now In Your Lab Environment!

Red Hat

Thank you

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/ansibleautomation

twitter.com/ansible

github.com/ansible

Red Hat