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.

Thanks to:

Kevin "GoKEV" Holmes
Phil Avery
Russell Pavlicek
Matt St Onge
Will Tome
Götz Rieger
Benjamin Blasco

Thanks for providing input, helping proofread, error check, and improving this deck continuously.

Red Hat
Ansible Automation
Platform

Ansible Linux Automation Workshop

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

Red Hat

Housekeeping

- Timing
- Breaks
- Takeaways

What you will learn

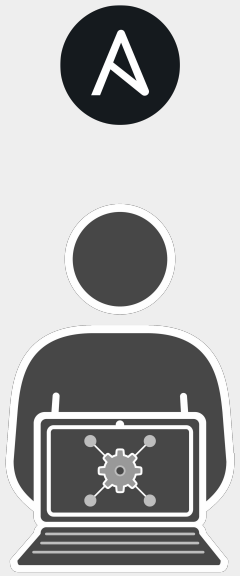
- Introduction to Ansible Automation
- How it works
- Understanding modules, tasks & playbooks
- How to execute Ansible commands
- Using variables & templates
- Tower - where it fits in
- Basic usage of Tower
- Learn major Tower features: RBAC, workflows and so on

Red Hat
Ansible Automation
Platform

Introduction

Topics Covered:

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



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

Red Hat

Teams are automating...

Lines Of Business

Network

Security

Operations

Developers

Infrastructure

Ad-hoc Automation is happening in silos

Developers → Ansible used in silo

Security → DIY scripting automation

Infrastructure → Open source config management tool

Network → Proprietary vendor supplied automation

Is organic automation enough?

Why Ansible?

Simple

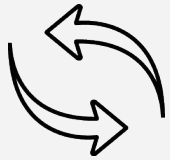
Human readable automation

No special coding skills needed

Tasks executed in order

Usable by every team

Get productive quickly



Powerful

App deployment

Configuration management

Workflow orchestration

Network automation

Orchestrate the app lifecycle

Agentless

Agentless architecture

Uses OpenSSH & WinRM

No agents to exploit or update

Get started immediately

More efficient & more secure

Red Hat

What can I do using Ansible?

Automate the deployment and management of your entire IT footprint.

Do this...

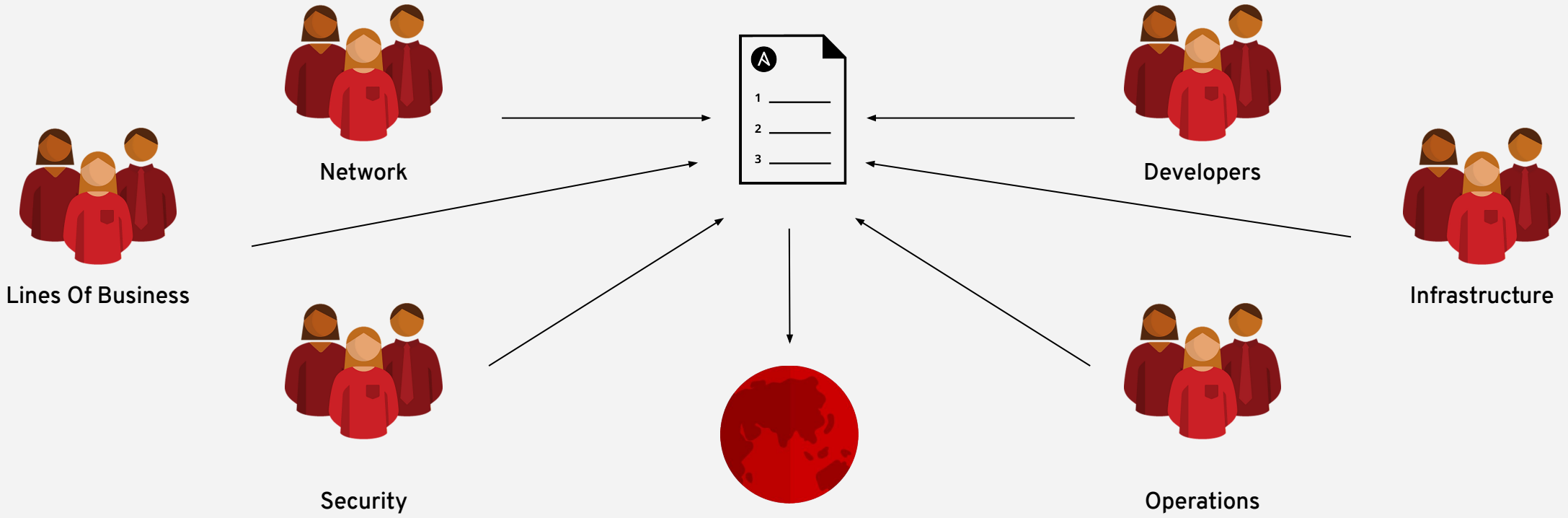
Orchestration	Configuration Management	Application Deployment	Provisioning	Continuous Delivery	Security and Compliance

On these...

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

Red Hat

When automation crosses teams, you need an automation platform



Red Hat Ansible Automation Platform

Network | Lines of business | Security | Operations | Infrastructure | Developers

Engage	**Ansible Hosted Services:** Engage users with an automation focused experience
Scale	**Ansible Tower:** Operate & control at scale
Create	**Ansible Engine:** Universal language of automation

Fueled by an open source community

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
Linux
Windows
+more

Virt & Container

Docker
VMware
RHV
OpenStack
OpenShift
+more

Storage

Netapp
Red Hat Storage
Infinidat
+more

Windows

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

Network

A10
Arista
Aruba
Cumulus
Bigswitch
Cisco
Dell
Extreme
F5
Lenovo
MikroTik
Juniper
OpenSwitch
+more

Security

Checkpoint
Cisco
CyberArk
F5
Fortinet
Juniper
IBM
Palo Alto
Snort
+more

Monitoring

Dynatrace
Datadog
LogicMonitor
New Relic
Sensu
+more

Devops

Jira
GitHub
Vagrant
Jenkins
Slack
+more

Red Hat Ansible Tower

by the numbers:

94% Reduction in recovery time following a security incident

84% Savings by deploying workloads to generic systems appliances using Ansible Tower

67% Reduction in man hours required for customer deliveries

Financial summary:

146%

ROI on Ansible Tower

<3 MONTHS

Payback on Ansible Tower

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

Red Hat

Section 1
Ansible Engine

Red Hat
Ansible Automation
Platform

Red Hat

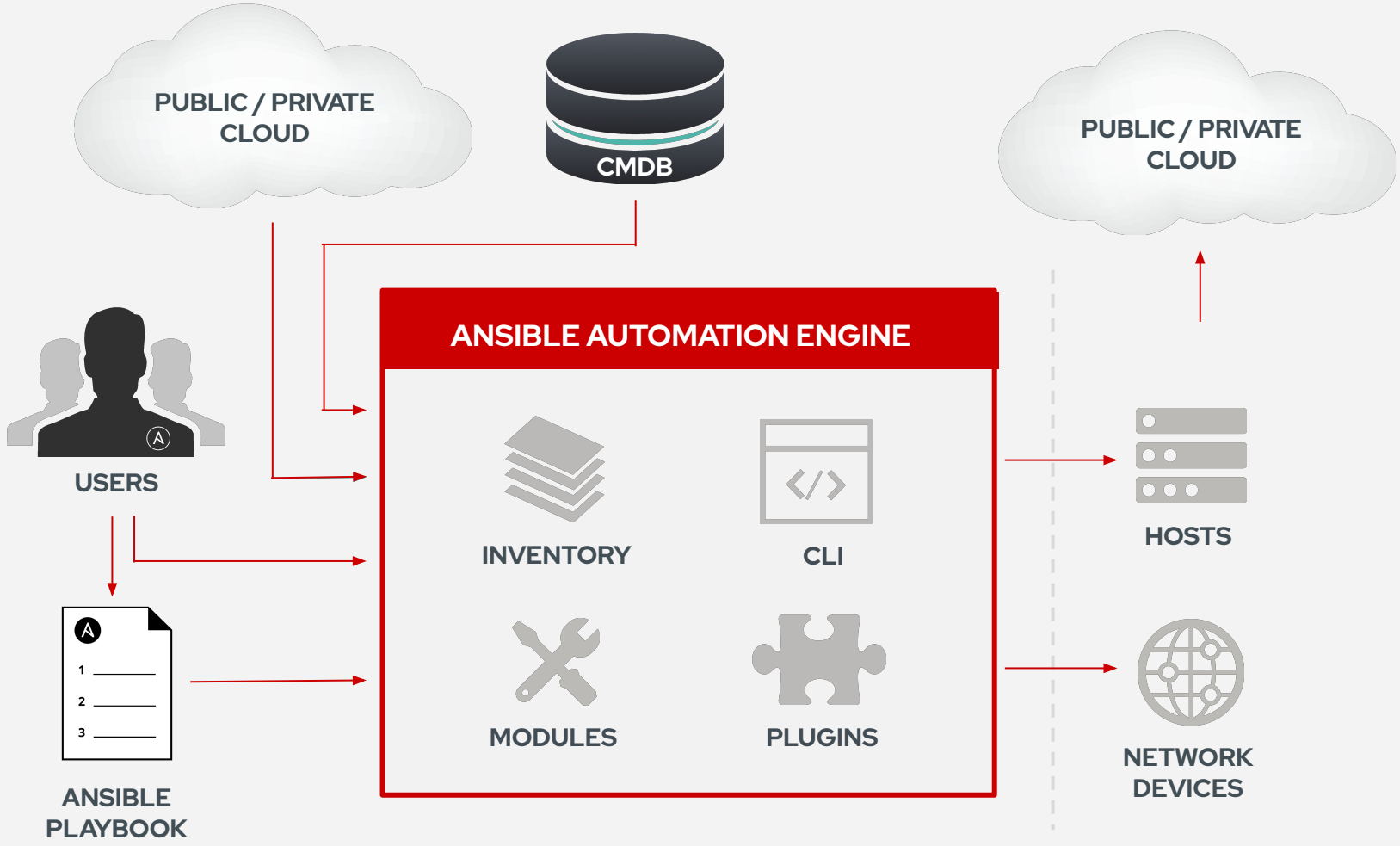
Red Hat
Ansible Automation
Platform

Exercise 1.1

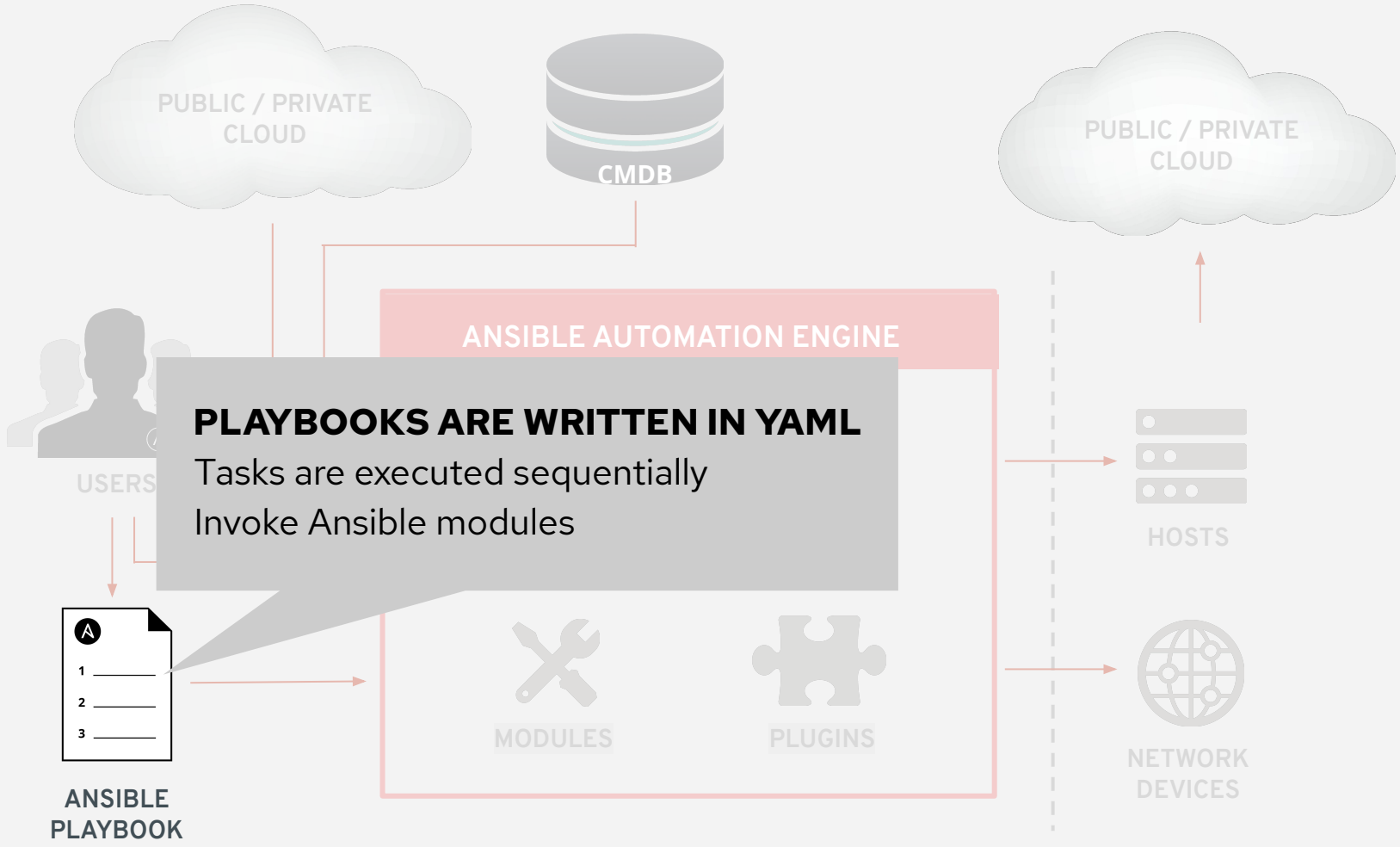
Topics Covered:

- Understanding the Ansible Infrastructure
- Check the prerequisites

Red Hat

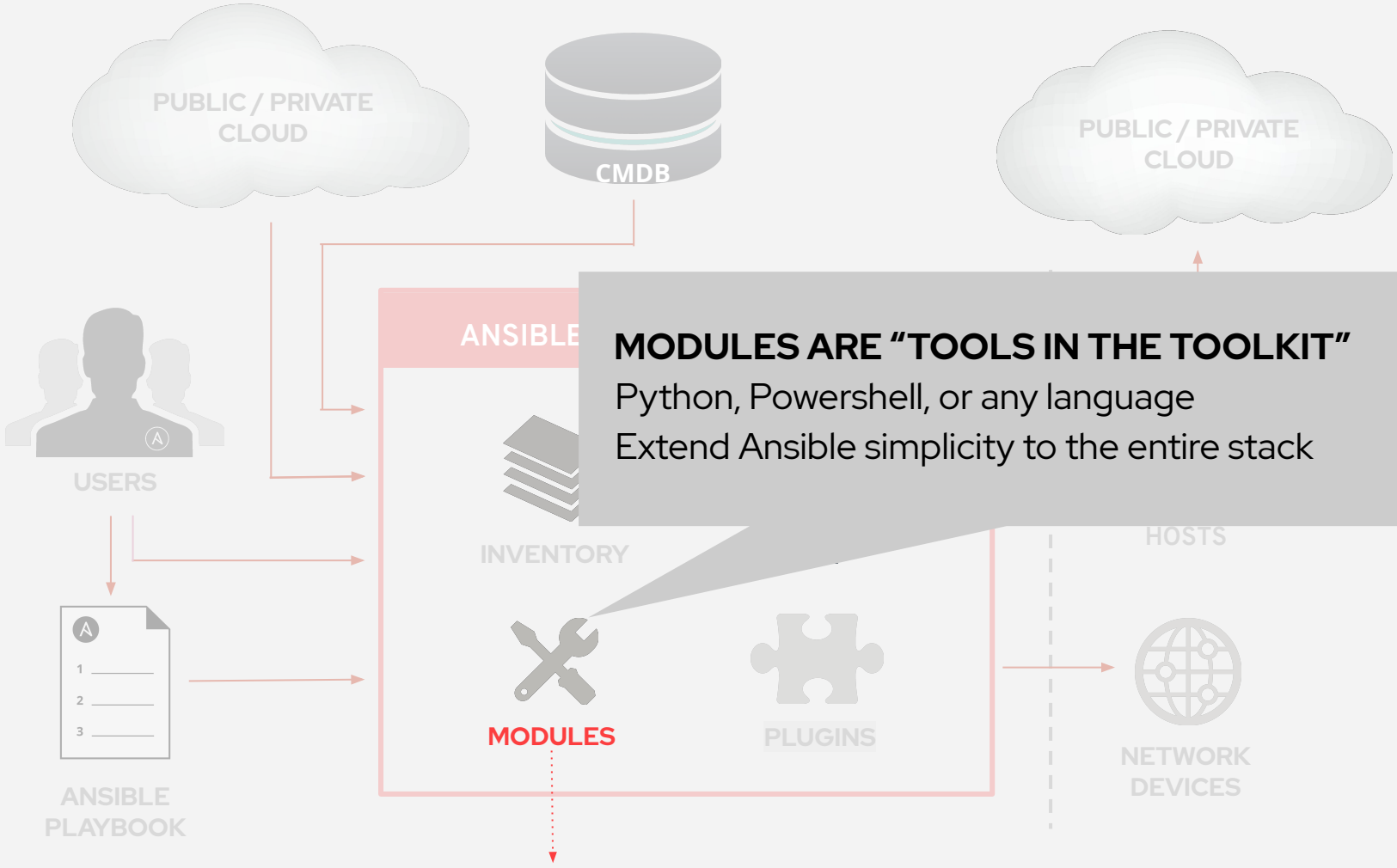

PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
INVENTORY
CLI
HOSTS
MODULES
PLUGINS
NETWORK DEVICES
1
2
3
ANSIBLE PLAYBOOK

Red Hat


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

Red Hat

```
---
- 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
```



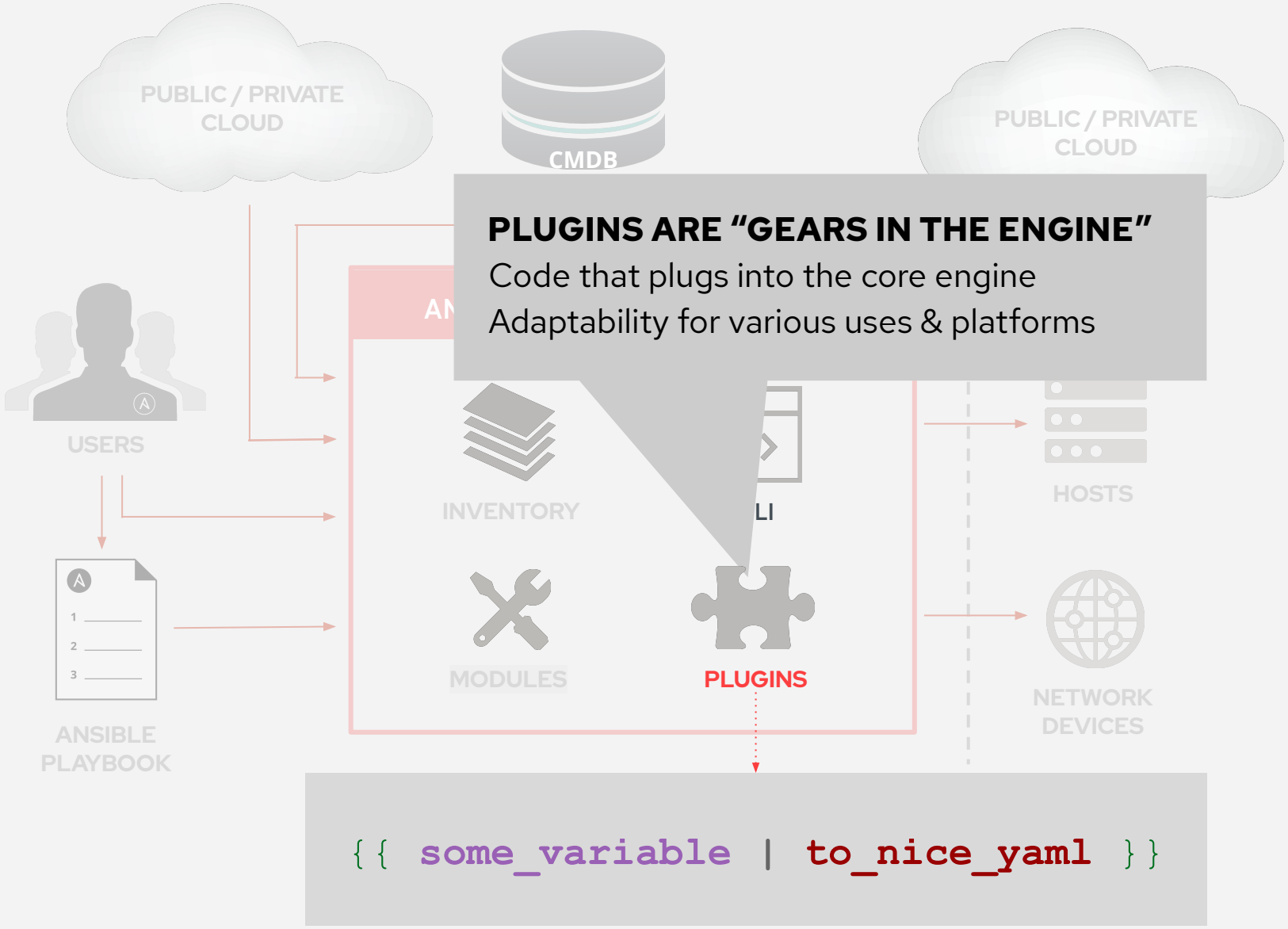
MODULES ARE "TOOLS IN THE TOOLKIT"

Python, Powershell, or any language

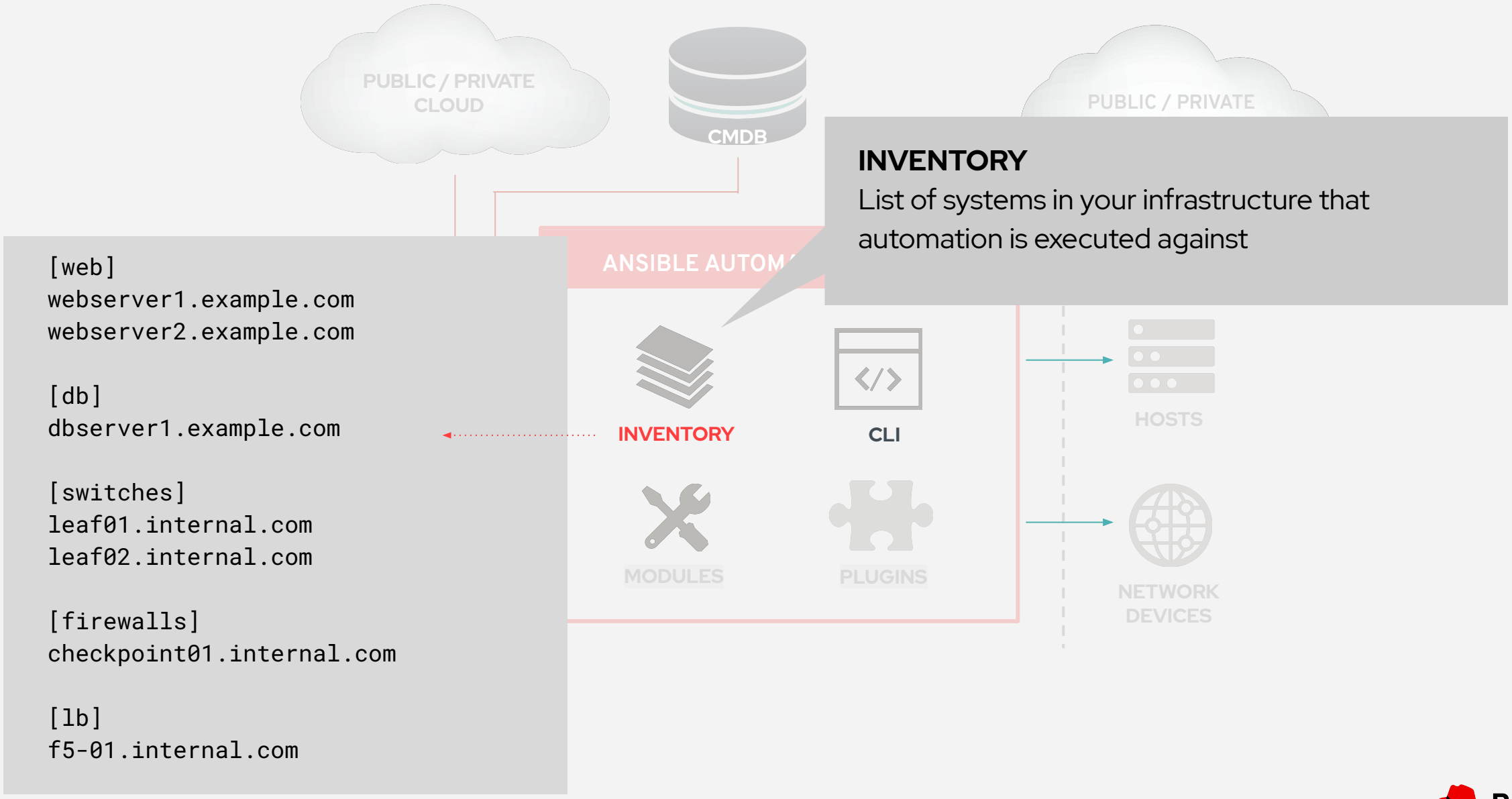
Extend Ansible simplicity to the entire stack

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

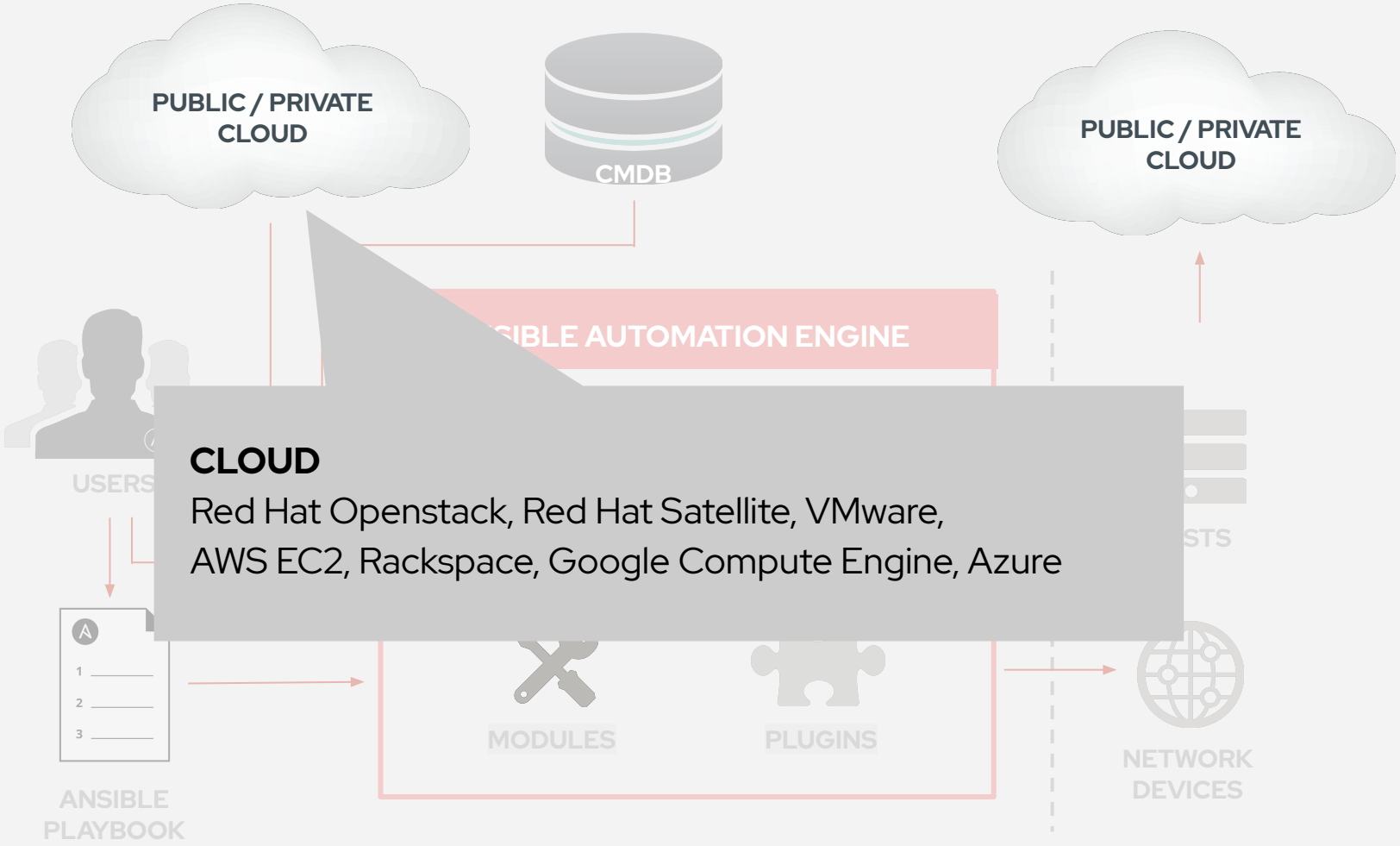
Red Hat


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
PLUGINS ARE “GEARS IN THE ENGINE”
Code that plugs into the core engine
Adaptability for various uses & platforms
USERS
INVENTORY
HOSTS
MODULES
PLUGINS
NETWORK DEVICES
ANSIBLE PLAYBOOK
{{ some_variable | to_nice_yaml }}

Red Hat


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE
INVENTORY
List of systems in your infrastructure that automation is executed against
[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
ANSIBLE AUTOM
INVENTORY
CLI
HOSTS
MODULES
PLUGINS
NETWORK DEVICES

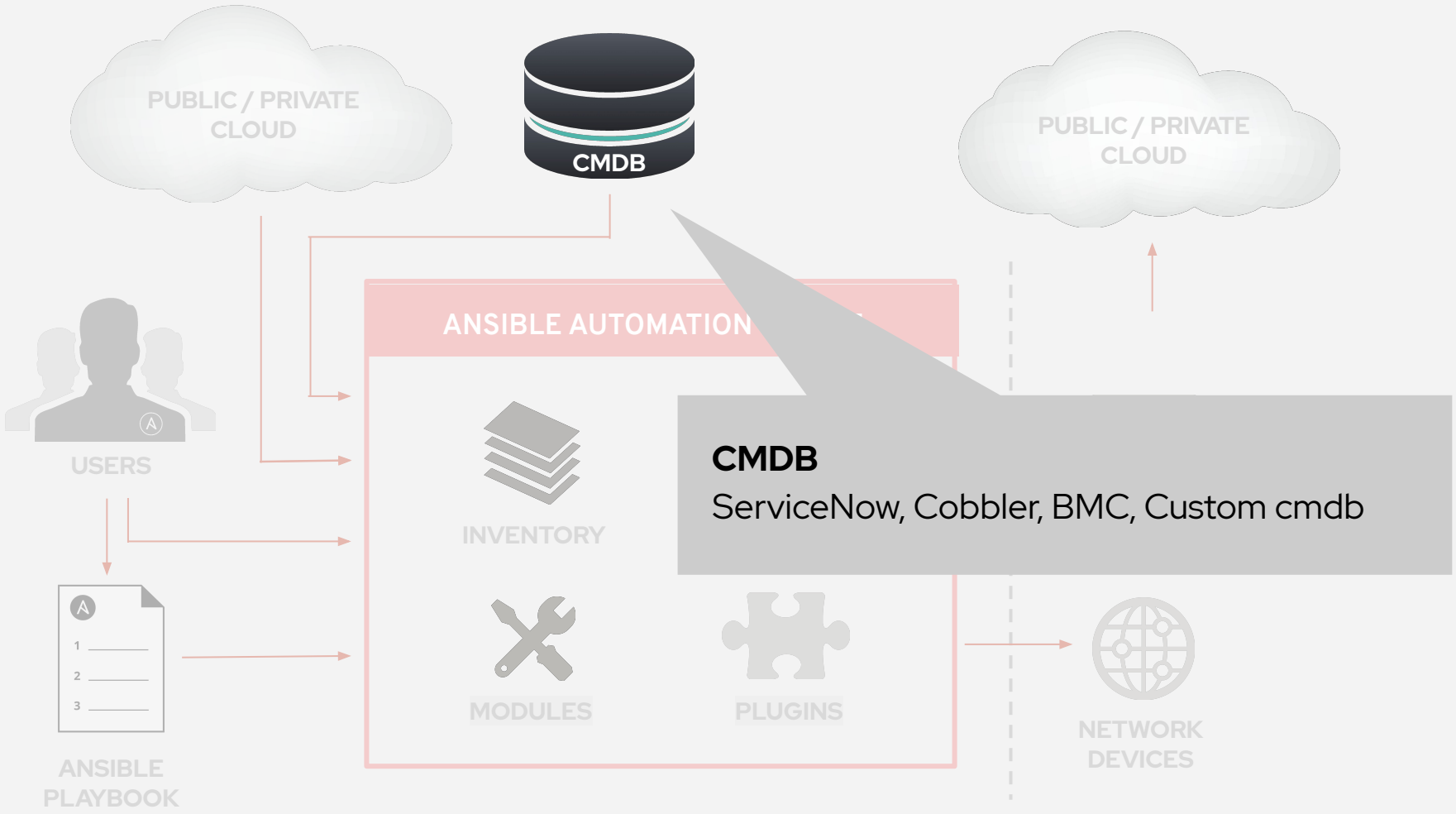
Red Hat



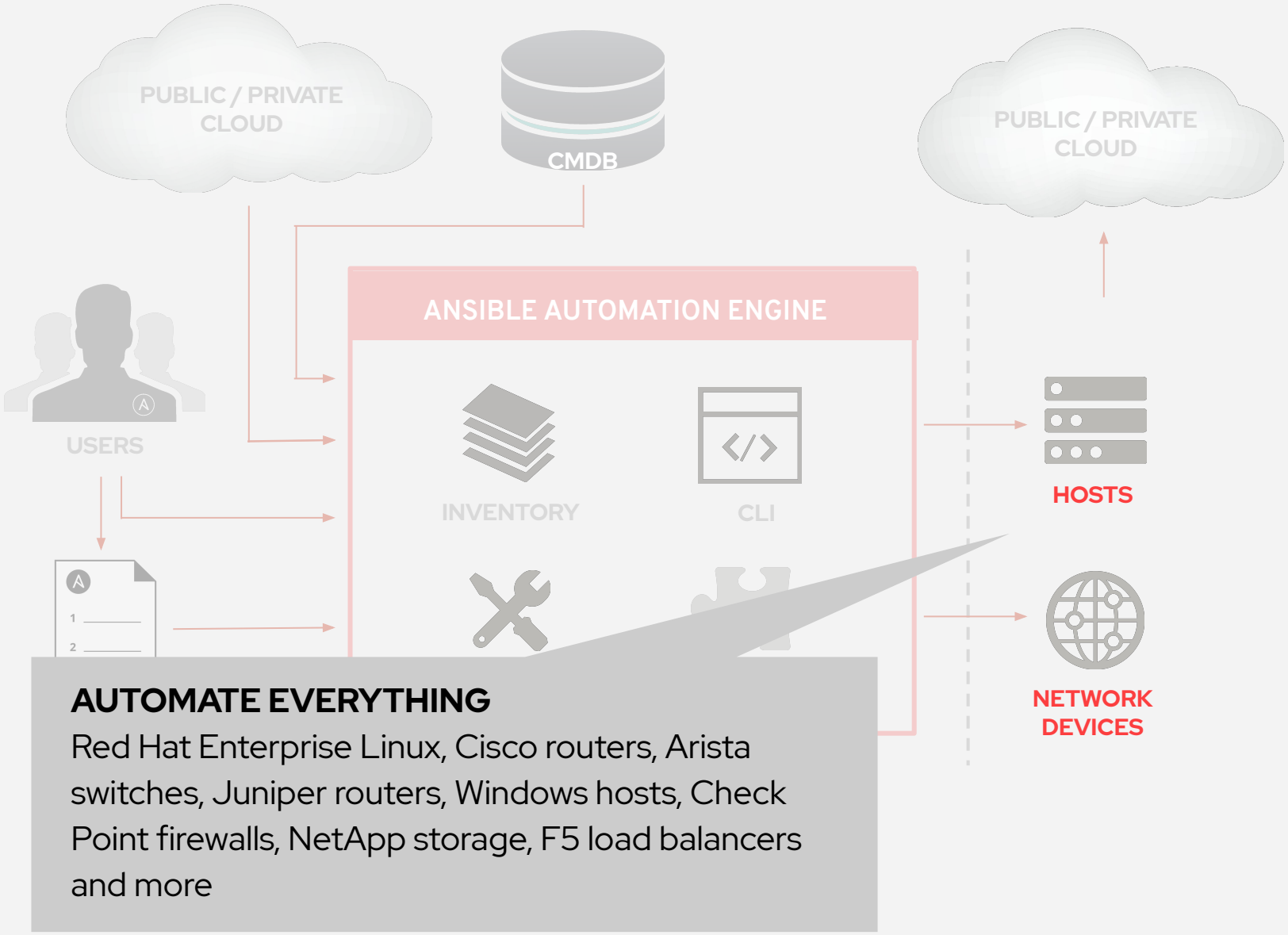
CLOUD

Red Hat Openstack, Red Hat Satellite, VMware,
AWS EC2, Rackspace, Google Compute Engine, Azure

Red Hat


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION
USERS
INVENTORY
CMDB
ServiceNow, Cobbler, BMC, Custom cmdb
MODULES
PLUGINS
NETWORK DEVICES
ANSIBLE PLAYBOOK

Red Hat


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
INVENTORY
CLI
HOSTS
NETWORK DEVICES
AUTOMATE EVERYTHING
Red Hat Enterprise Linux, Cisco routers, Arista switches, Juniper routers, Windows hosts, Check Point firewalls, NetApp storage, F5 load balancers and more

Red Hat

LINUX AUTOMATION

150+

Linux Modules

AUTOMATE EVERYTHING LINUX

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

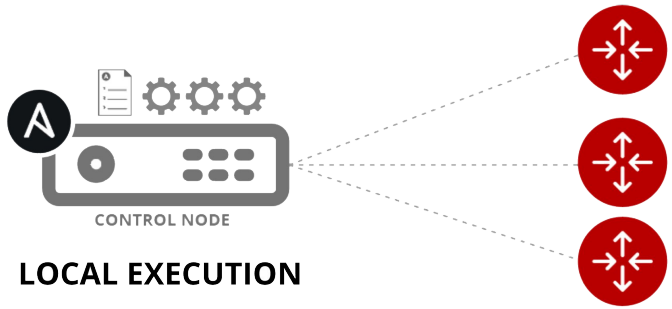
ONLY REQUIREMENTS:
Python 2 (2.6 or later)
***or* Python 3 (3.5 or later)**

ansible.com/get-started

Red Hat

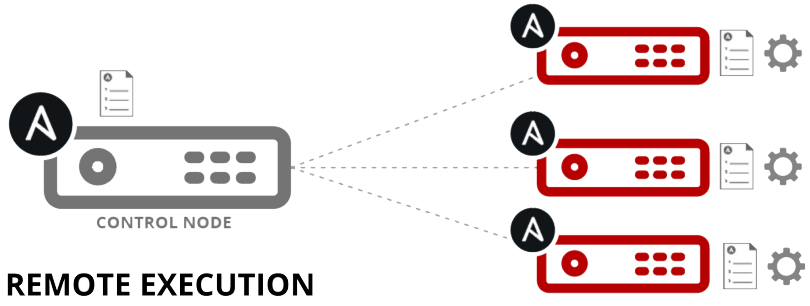
How Ansible Automation works

Module code is executed locally on the control node



NETWORKING DEVICES

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

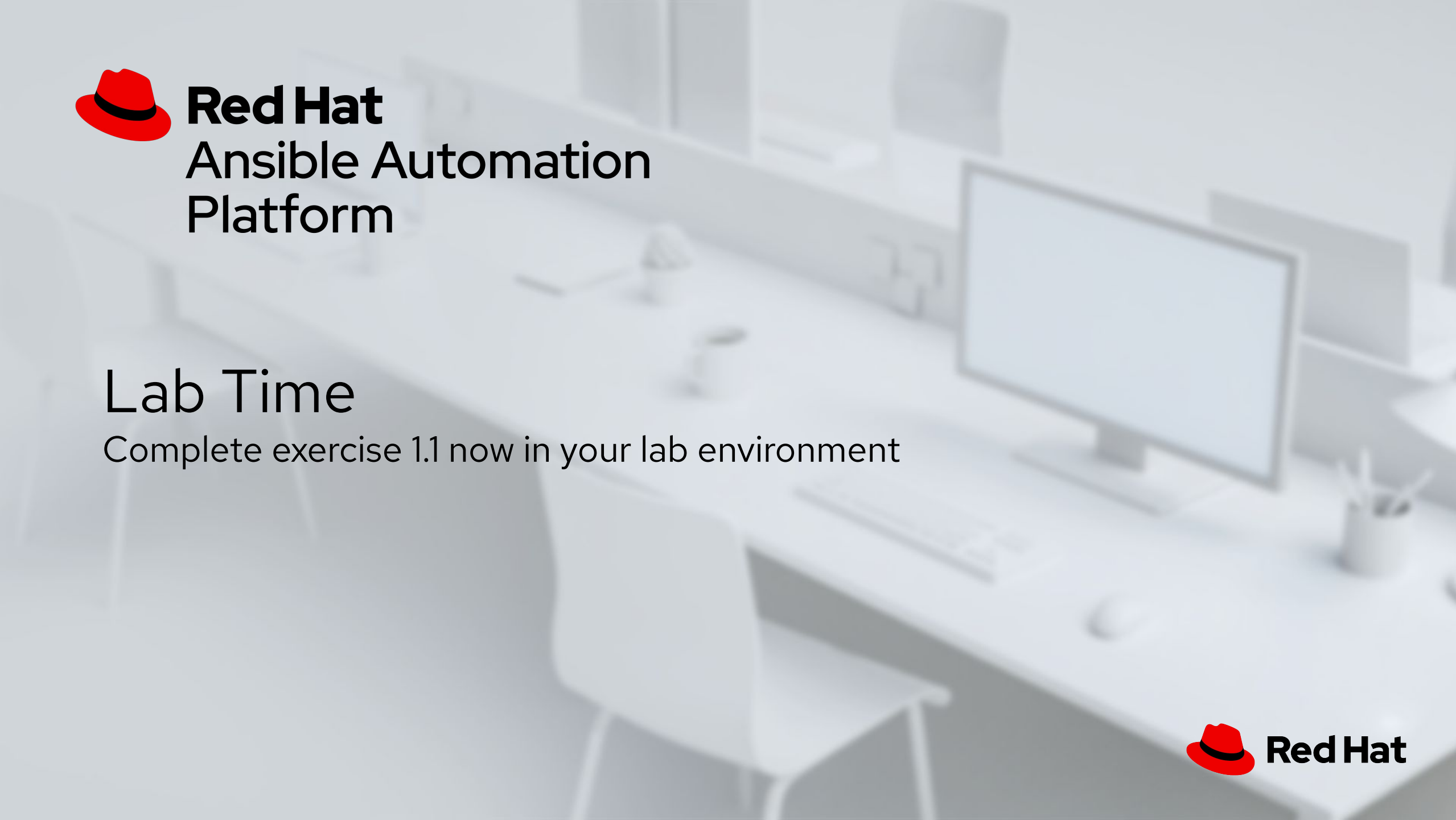


LINUX/WINDOWS HOSTS

Red Hat

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

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 - 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 - 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

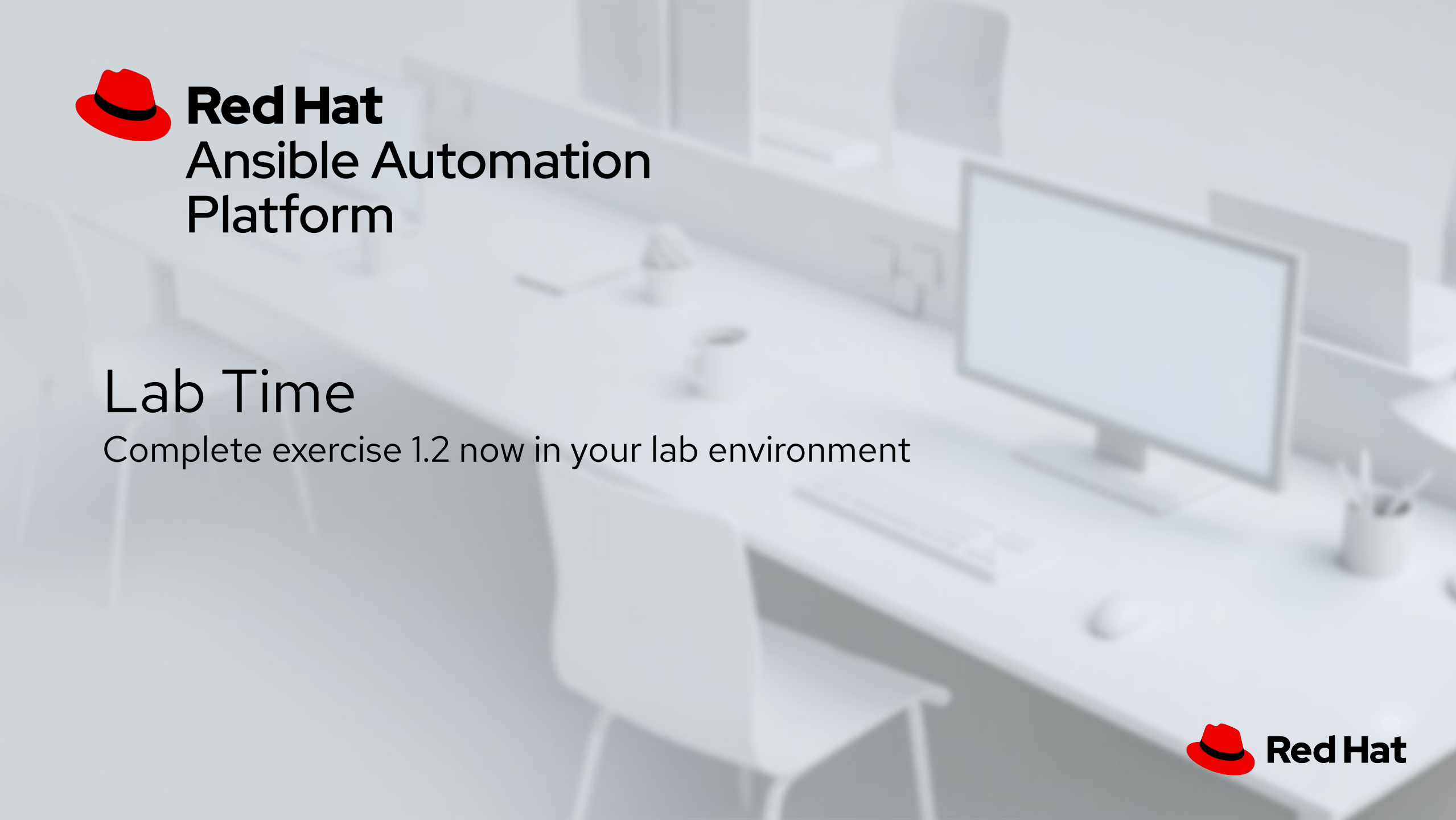
`-b, --become`
Run ad-hoc command with elevated rights such as sudo, the default method

`-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 “web1”
[user@ansible] $ ansible web1 -m setup
```


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

An Ansible Playbook

A play

```
---
- 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
```

An Ansible Playbook

A task

```
---
- 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
```

An Ansible Playbook

module

```
---
- 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
```

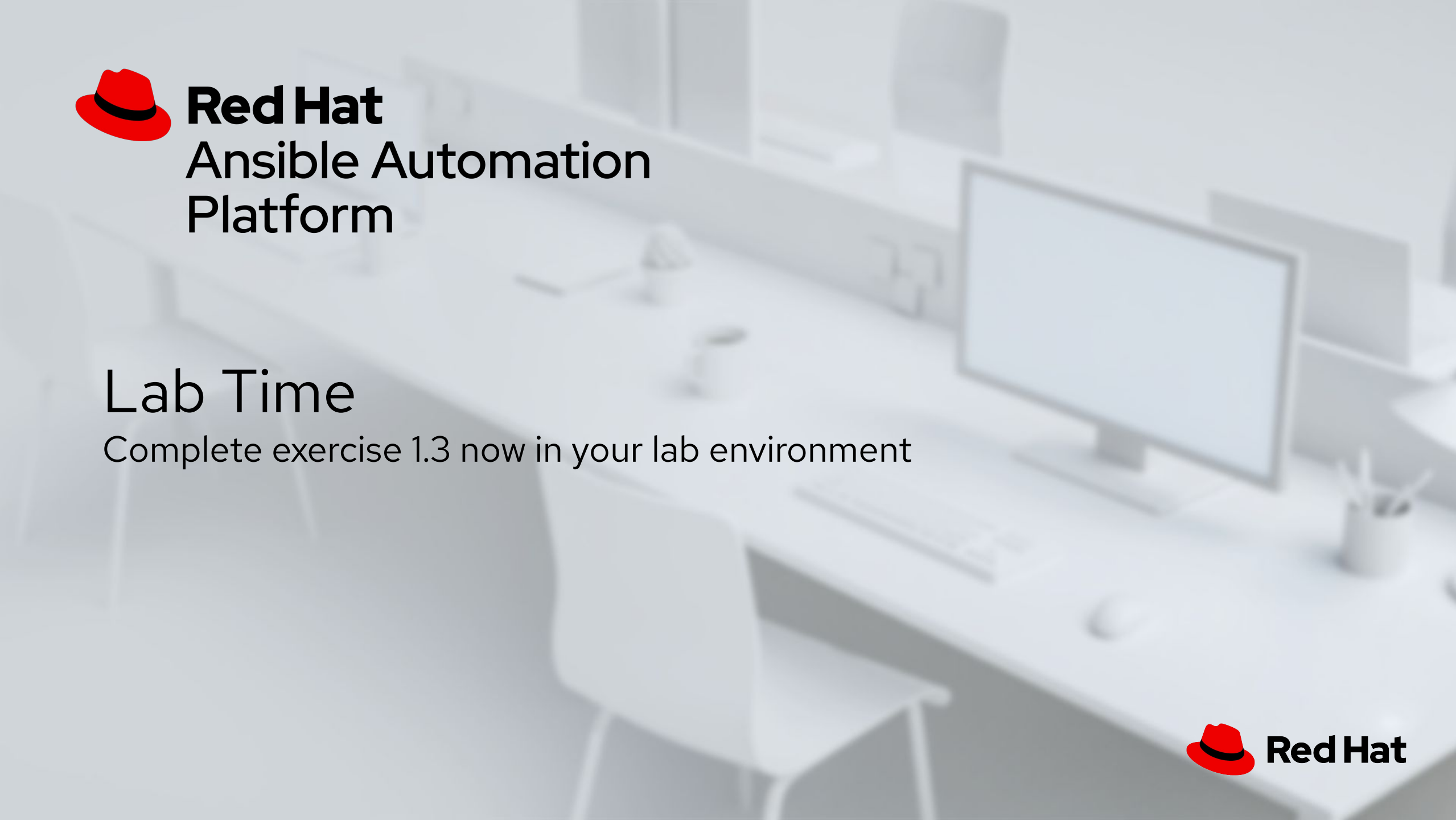
Running an Ansible Playbook:

The most important colors of Ansible

```
A task executed as expected, no change was made.

A task executed as expected, making a change

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
```


Red Hat
Ansible Automation
Platform
Lab Time
Complete exercise 1.3 now in your lab environment
Red Hat

Red Hat
Ansible Automation
Platform

Exercise 1.4

Topics Covered:

- Working with variables
- What are facts?

Red Hat

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

Red Hat

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
│   └── app1srv
│   └── db
│   └── web
├── inventory
└── host_vars
    └ app01
    └ app02
    └ app03
```

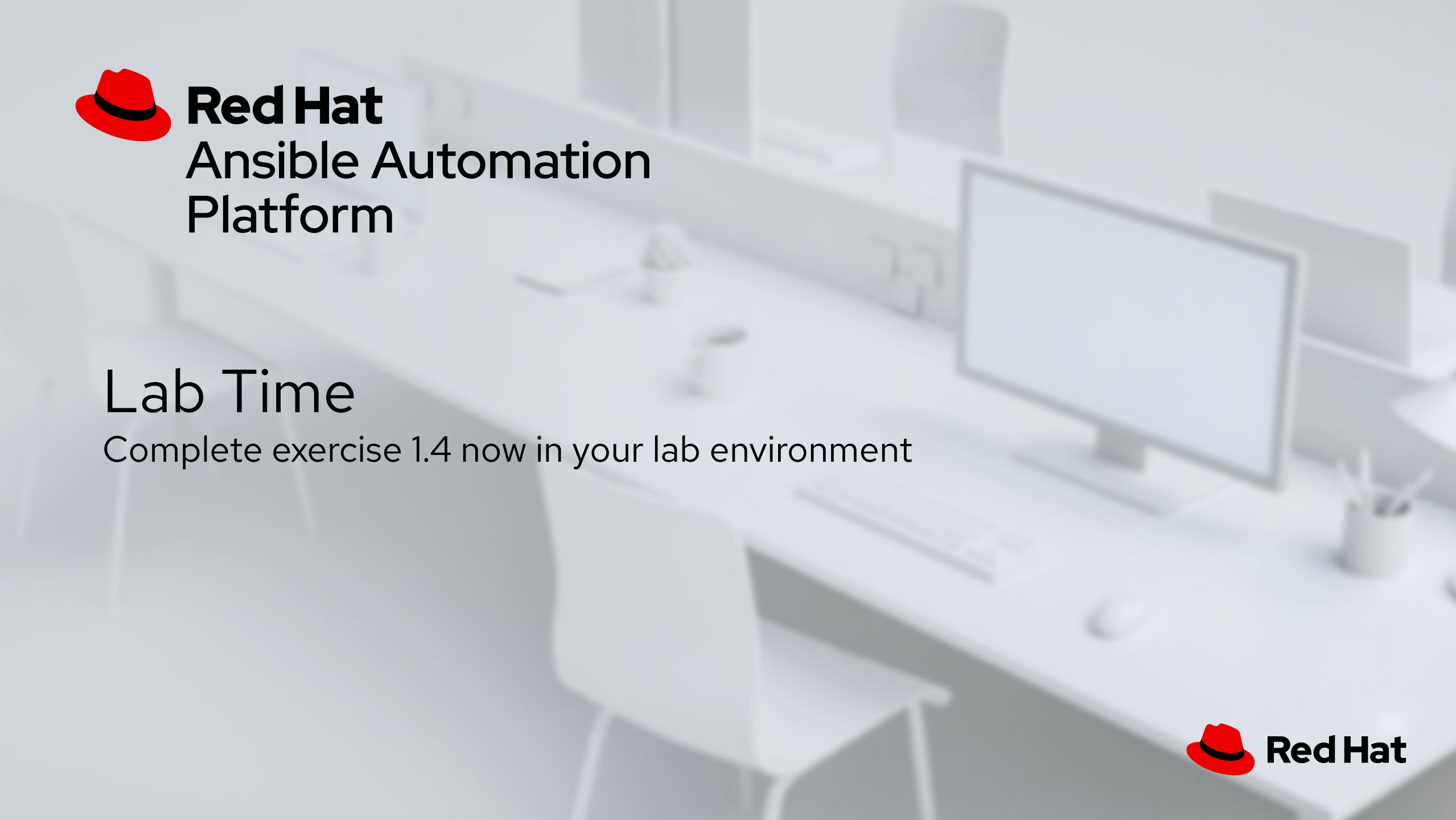
Ansible Inventory - Managing Variables In Files

```
[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

Red Hat
Ansible Automation
Platform

Exercise 1.5

Topics Covered:

- Conditionals
- Handlers
- Loops

Red Hat

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]
ok: [web1]

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

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

unchanged

changed

handler runs once

Red Hat

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]

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

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

changed

changed

handler runs once

Red Hat

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]

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

Red Hat

Variables & Loops

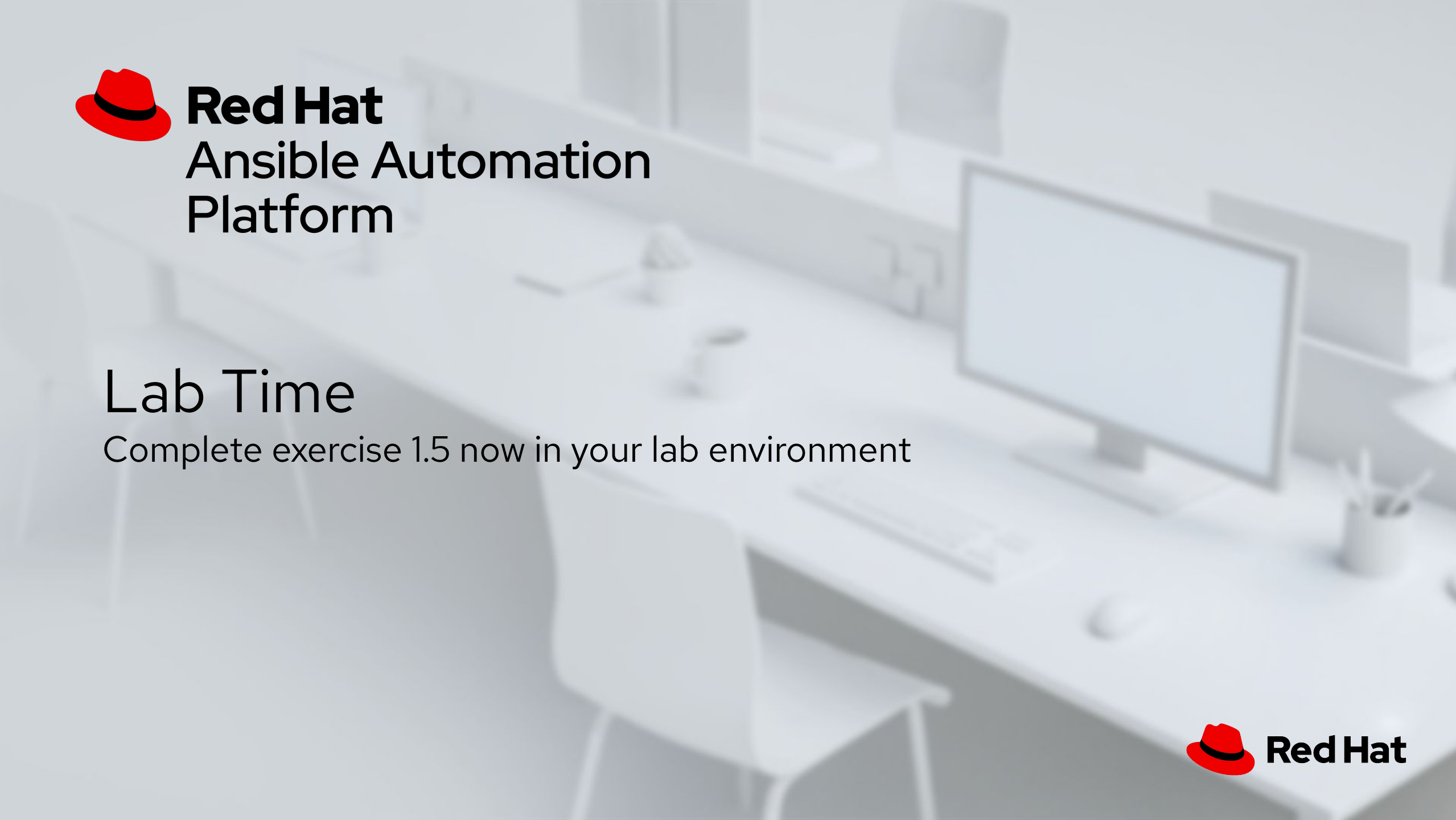
Great opportunity to use a loop

```
---
- 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

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
```

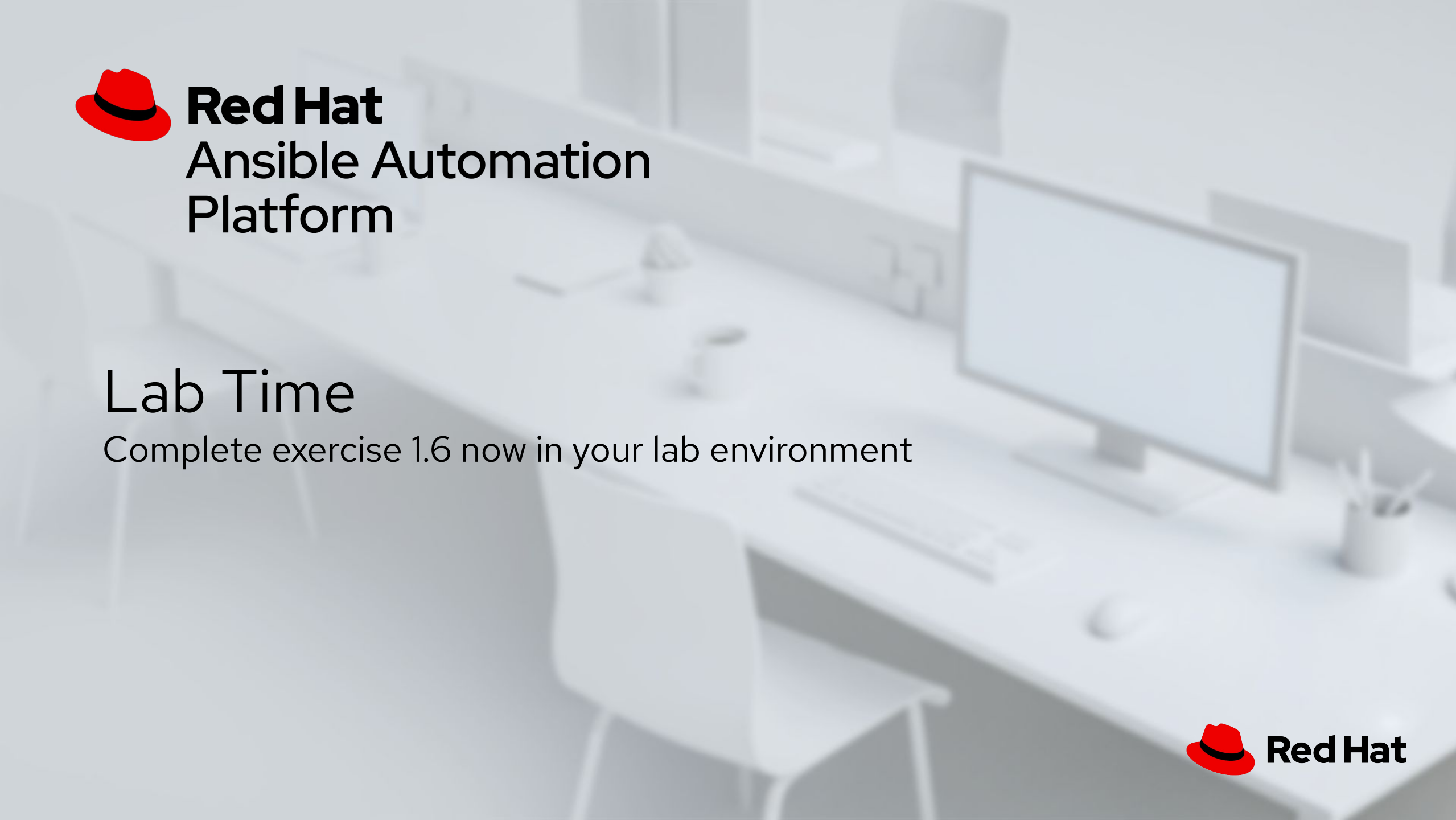
Variables & Templates

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


Red Hat
Ansible Automation
Platform
Lab Time
Complete exercise 1.6 now in your lab environment
Red Hat

Red Hat
Ansible Automation
Platform

Exercise 1.7

Topics Covered:

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

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)

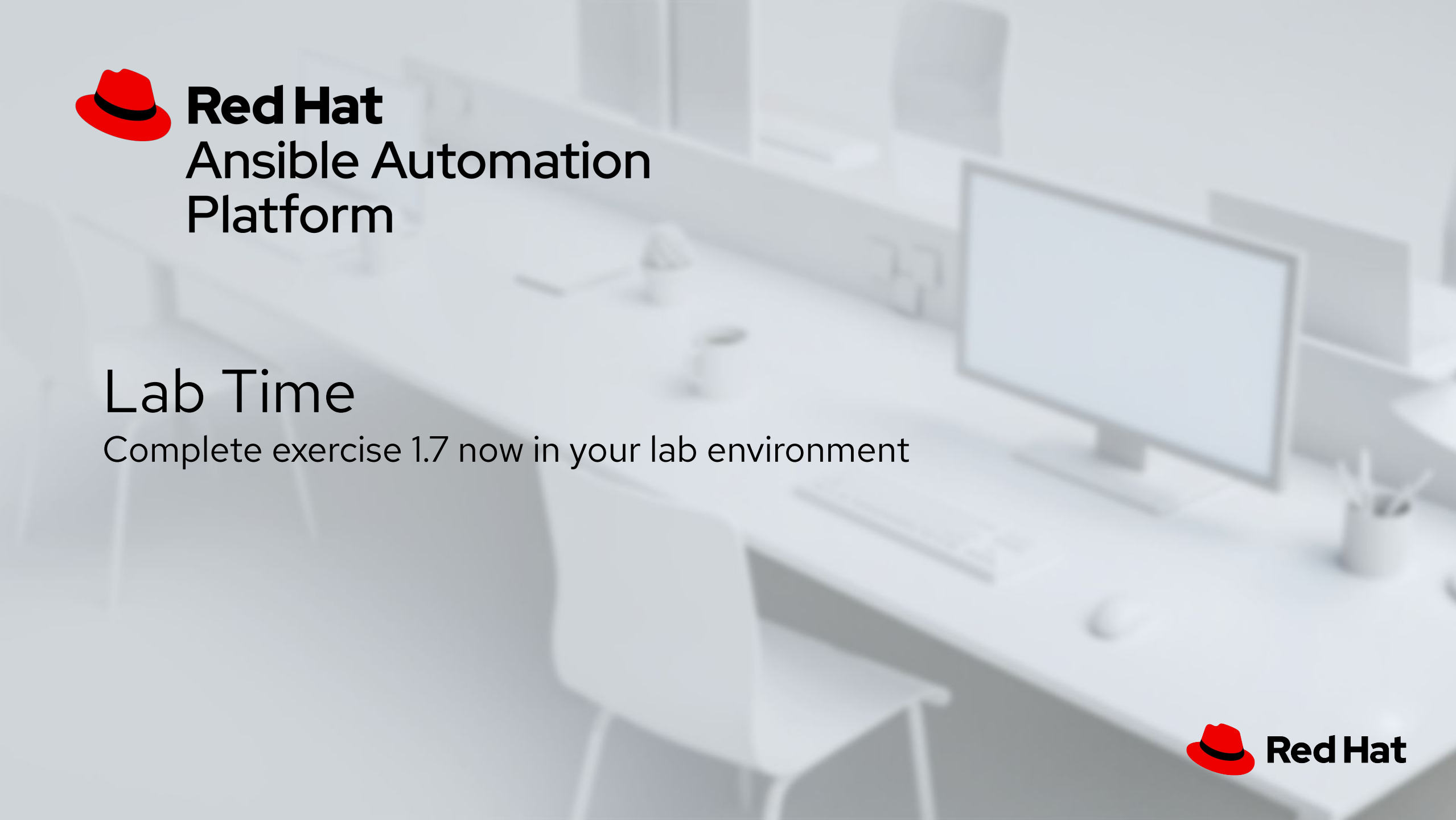
```
user/
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Ansible Galaxy

Sharing Content

Community

Roles, and more


Red Hat
Ansible Automation
Platform
Lab Time
Complete exercise 1.7 now in your lab environment
Red Hat

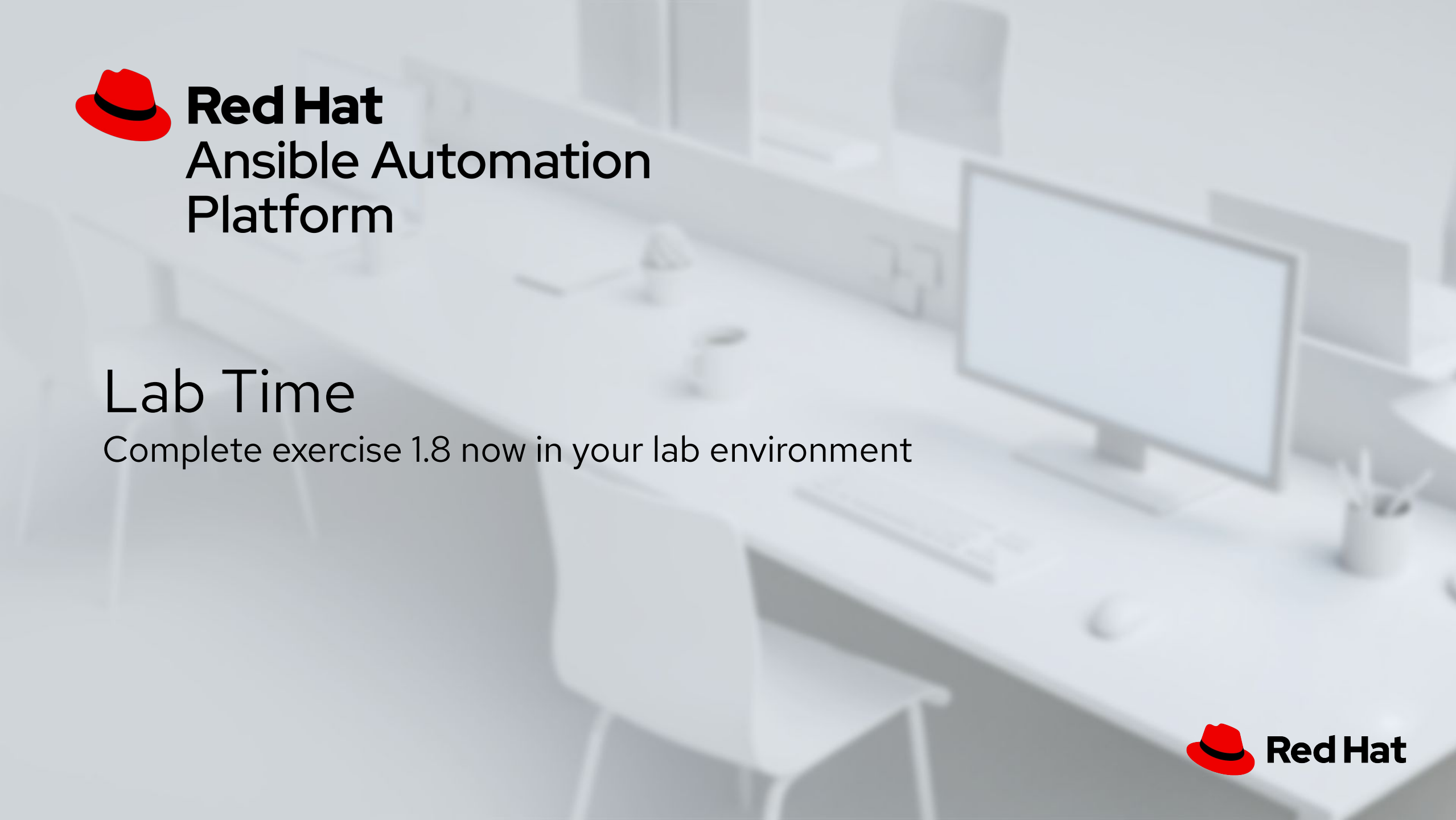
Red Hat
Ansible Automation
Platform

Exercise 1.8

Topics Covered:

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

Red Hat


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

Red Hat

Red Hat
Ansible Automation
Platform

Exercise 2.1

Topics Covered:

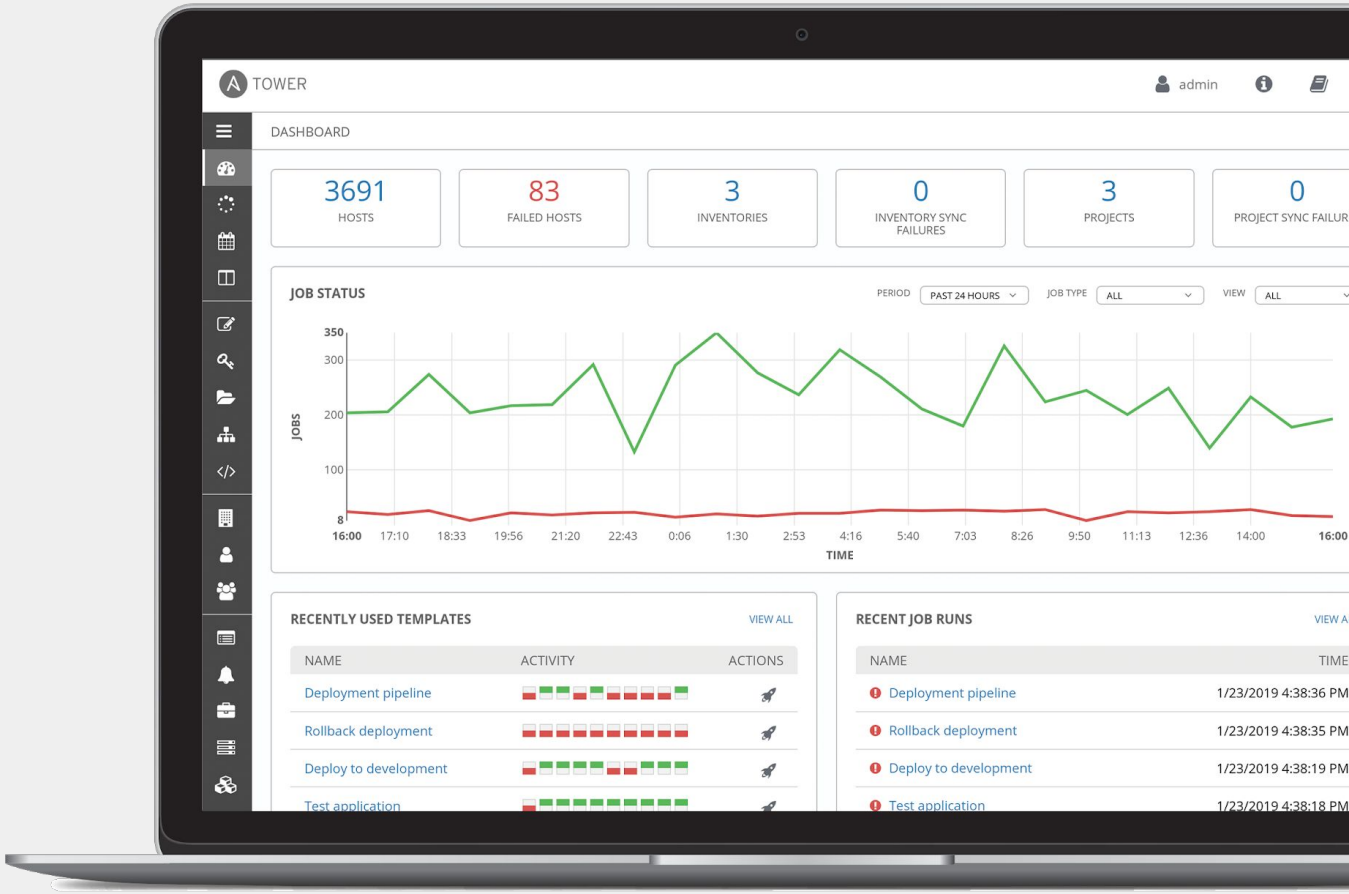
- Introduction to Tower

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

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

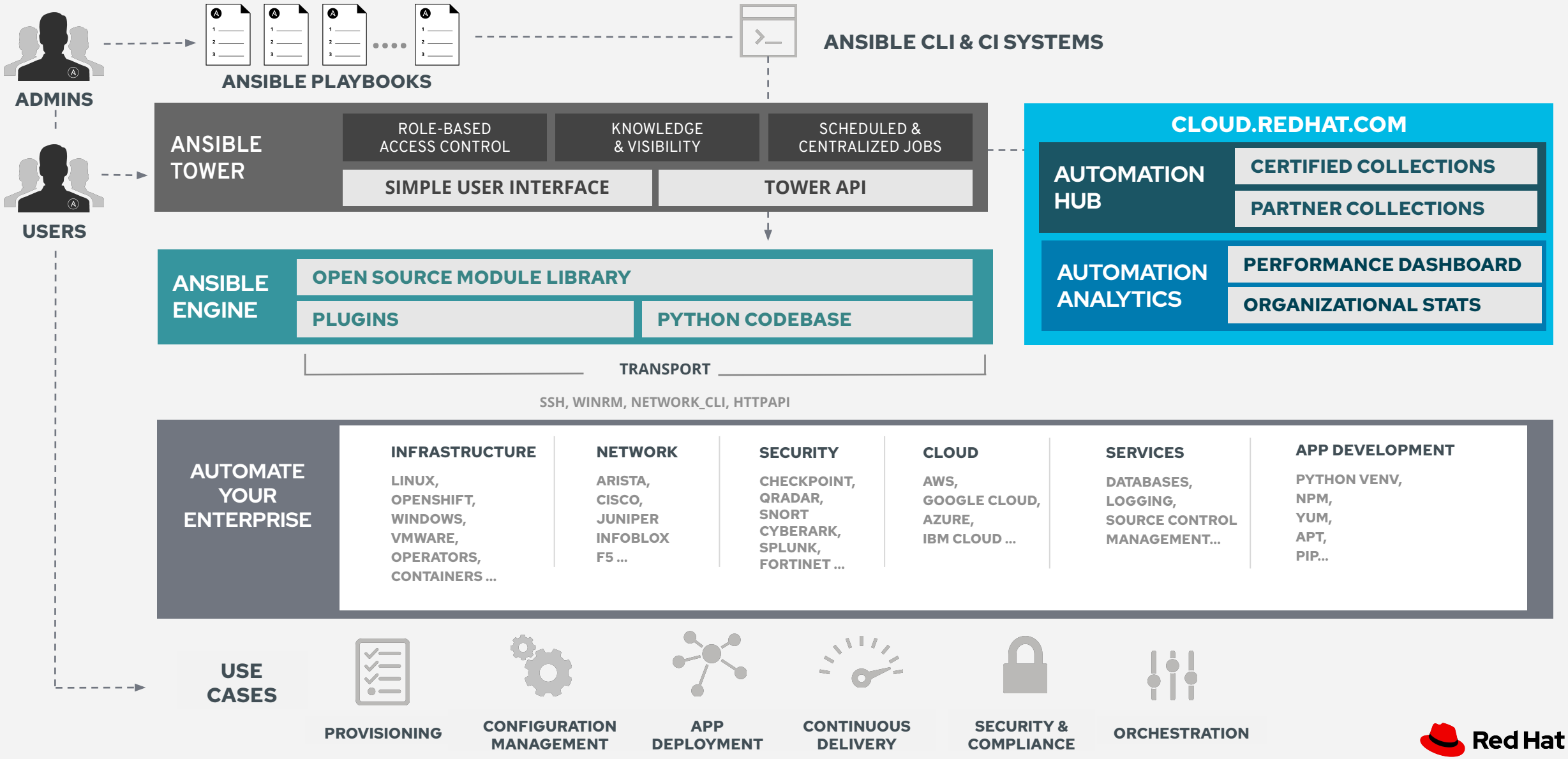
All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Ansible Tower's API.

Workflows

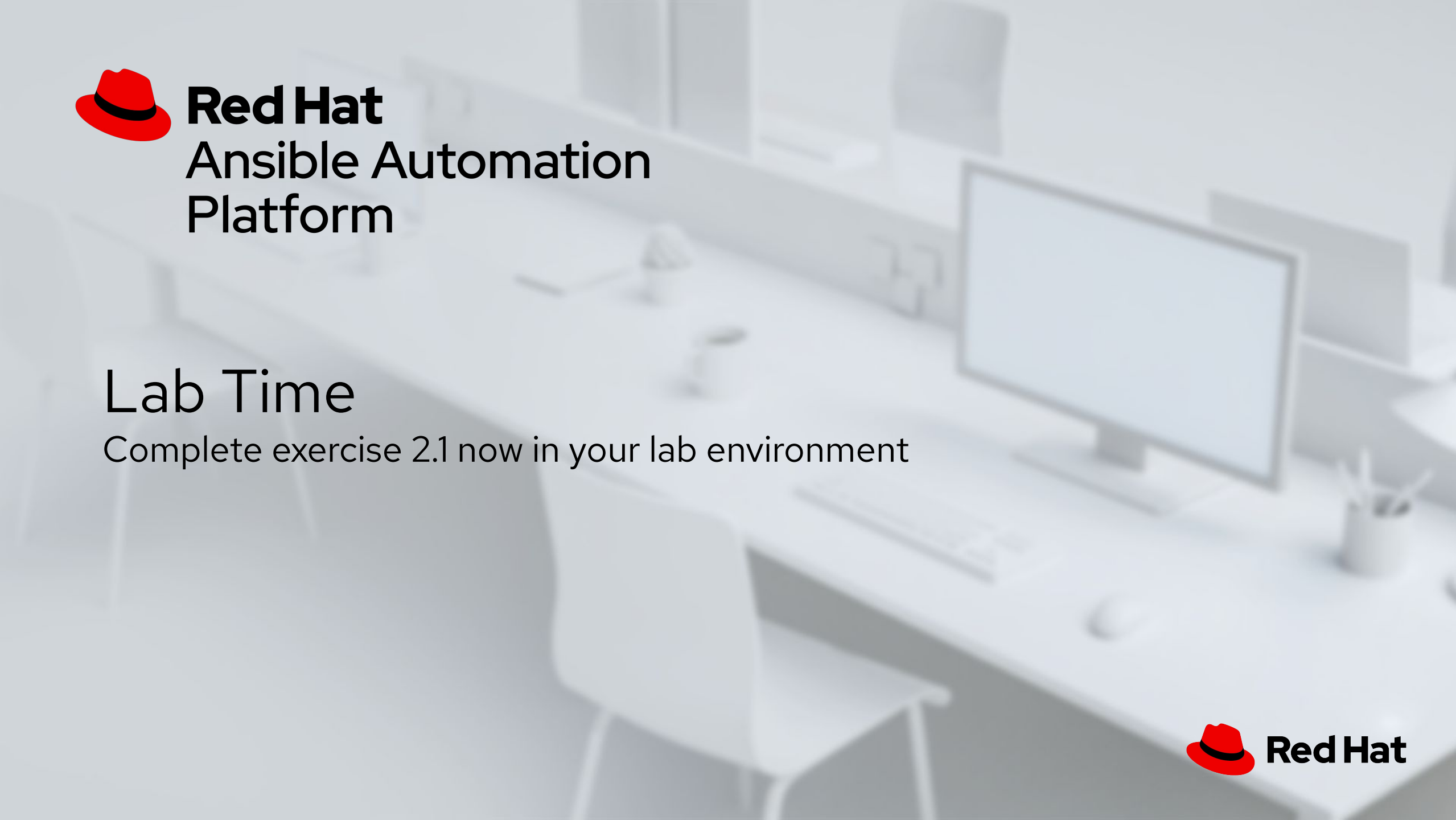
Ansible Tower's multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.

Red Hat

Ansible Automation Platform



Red Hat


Red Hat
Ansible Automation
Platform
Lab Time
Complete exercise 2.1 now in your lab environment
Red Hat

Exercise 2.2

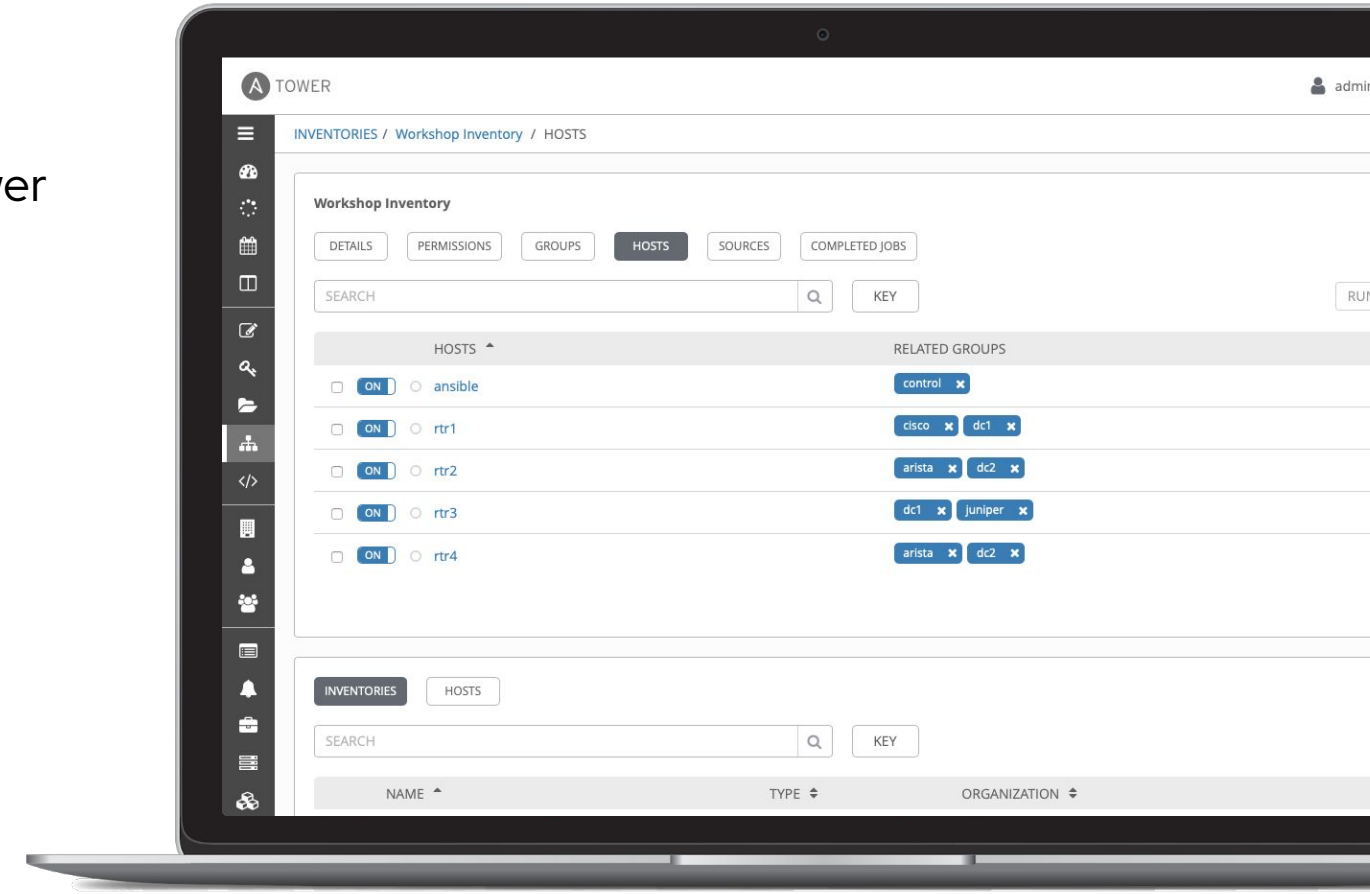
Topics Covered:

- 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



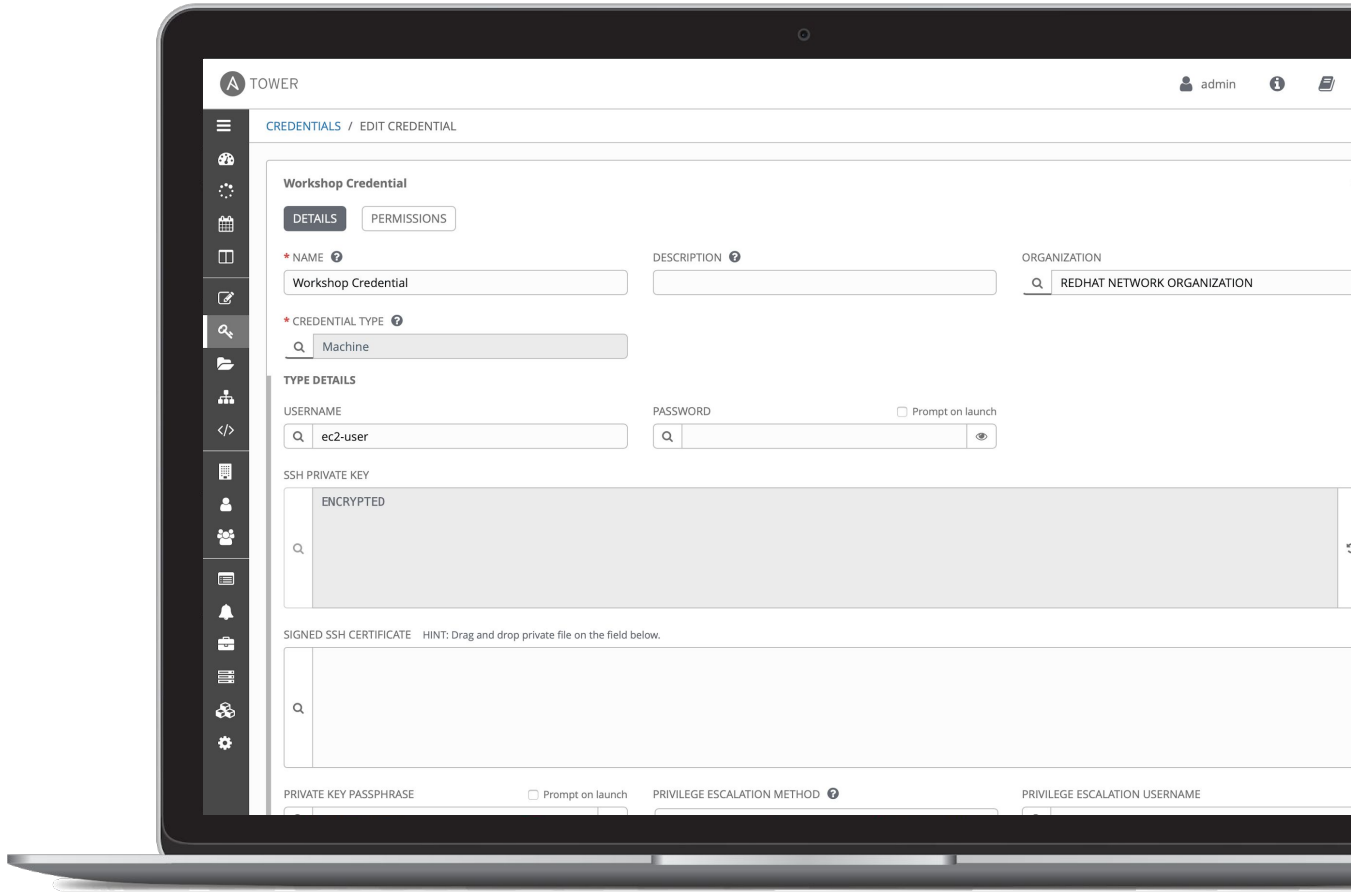
Red Hat

Credentials

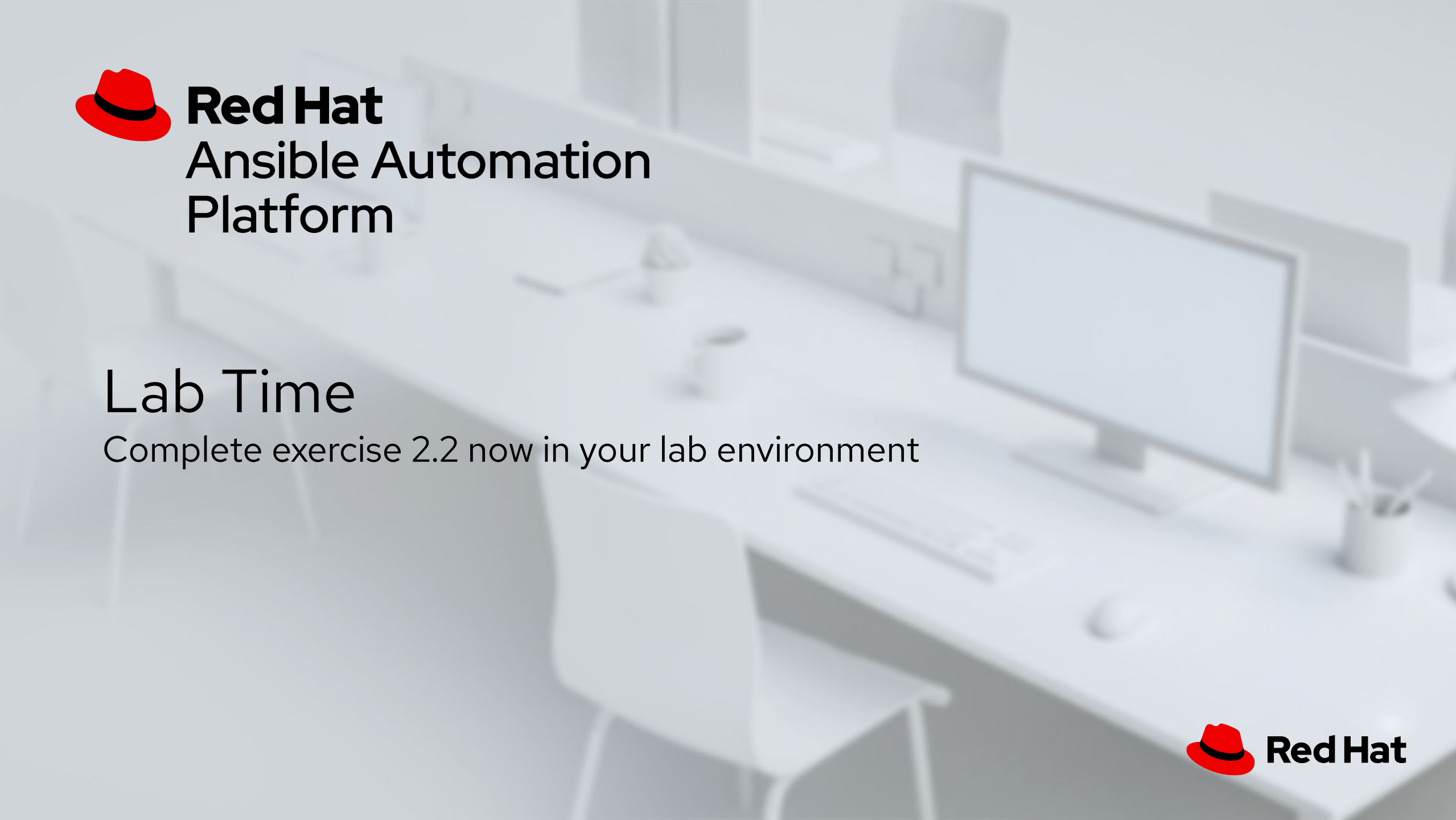
Credentials are utilized by Ansible Tower for authentication with various external resources:

- 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


Red Hat
Ansible Automation
Platform
Lab Time
Complete exercise 2.2 now in your lab environment
Red Hat

Red Hat
Ansible Automation
Platform

Exercise 2.3

Topics Covered:

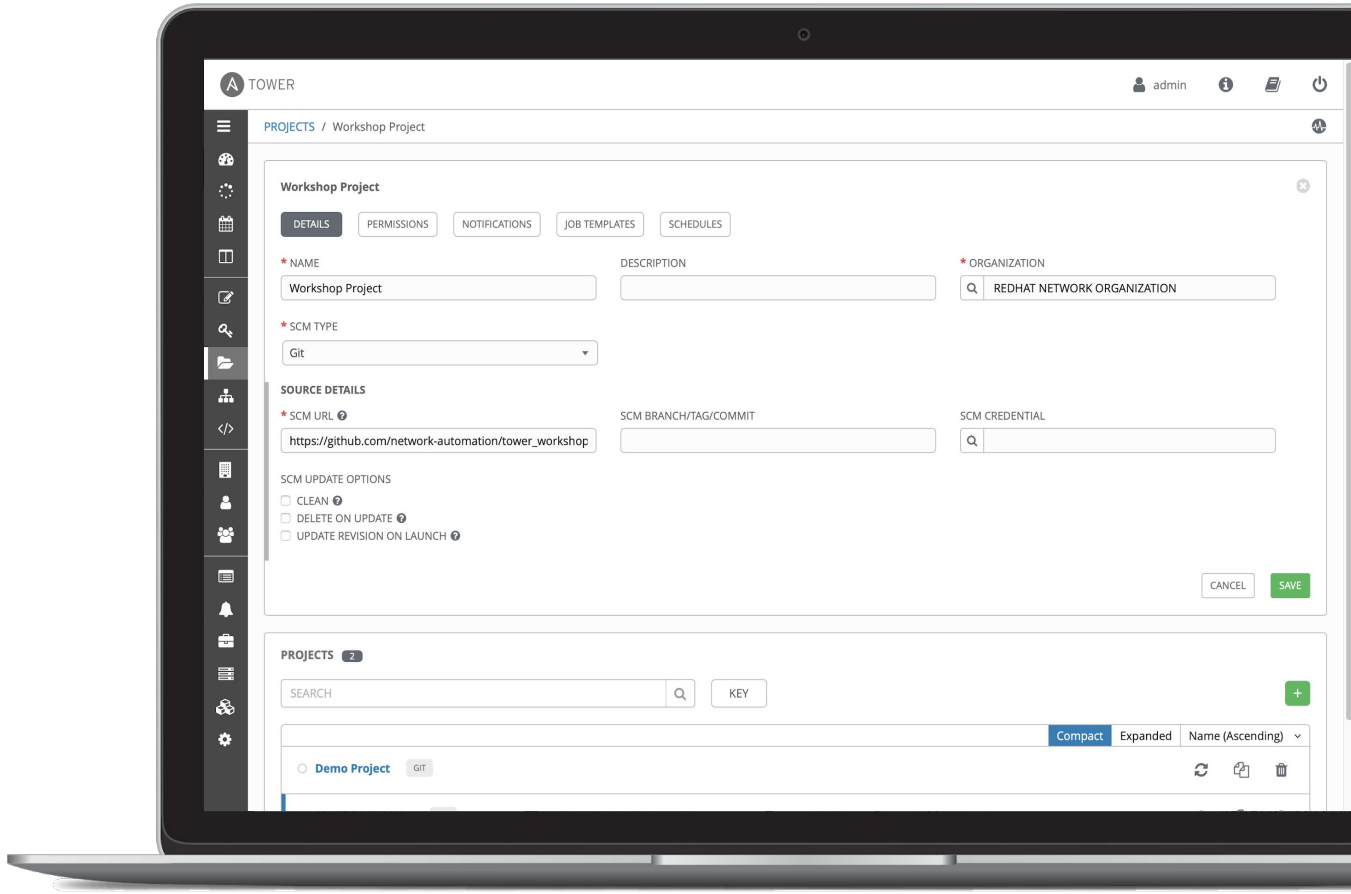
- Projects
- Job Templates

Red Hat

Project

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

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



Red Hat

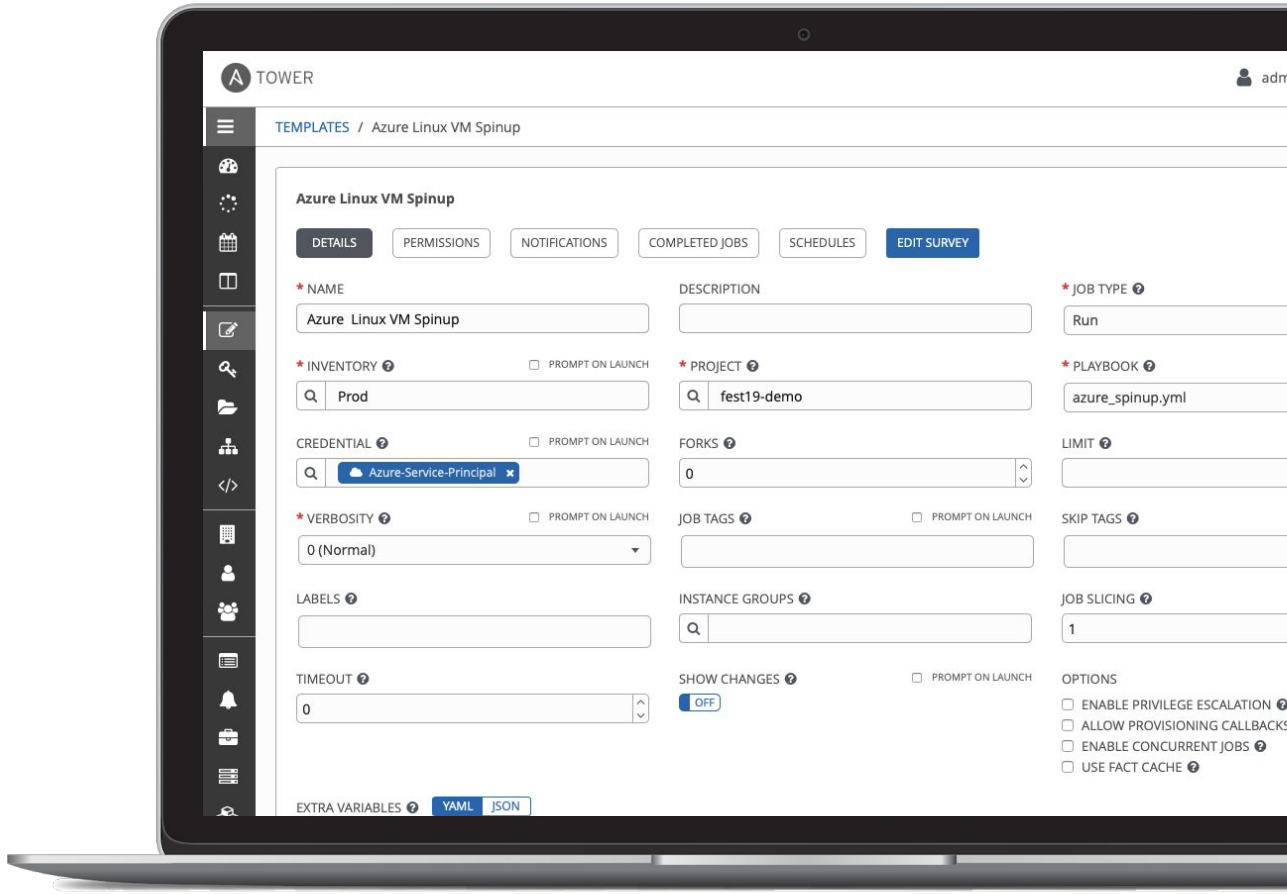
Job Templates

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:

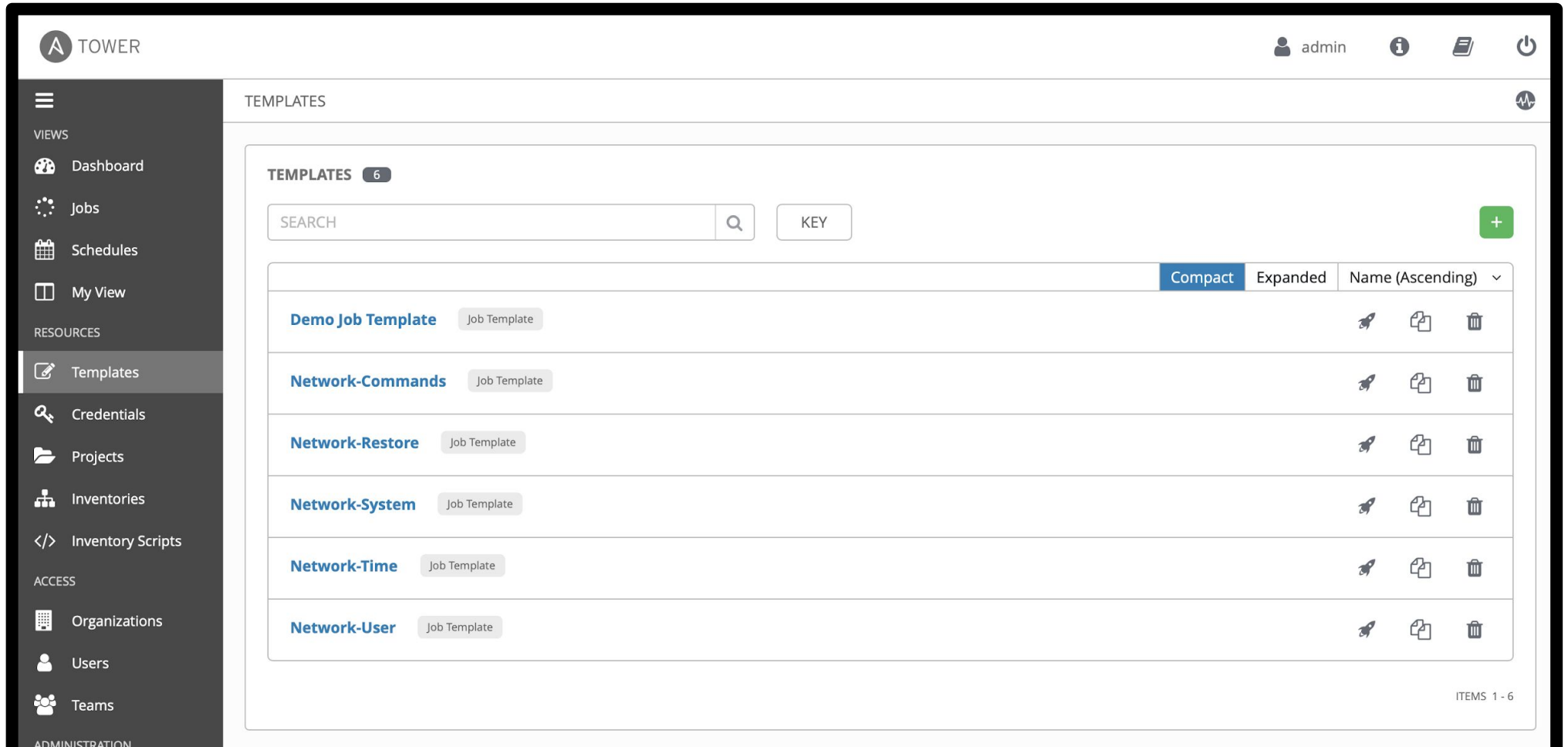
- An **Inventory** to run the job against
- A **Credential** to login to devices.
- A **Project** which contains Ansible Playbooks



Red Hat

Expanding on Job Templates

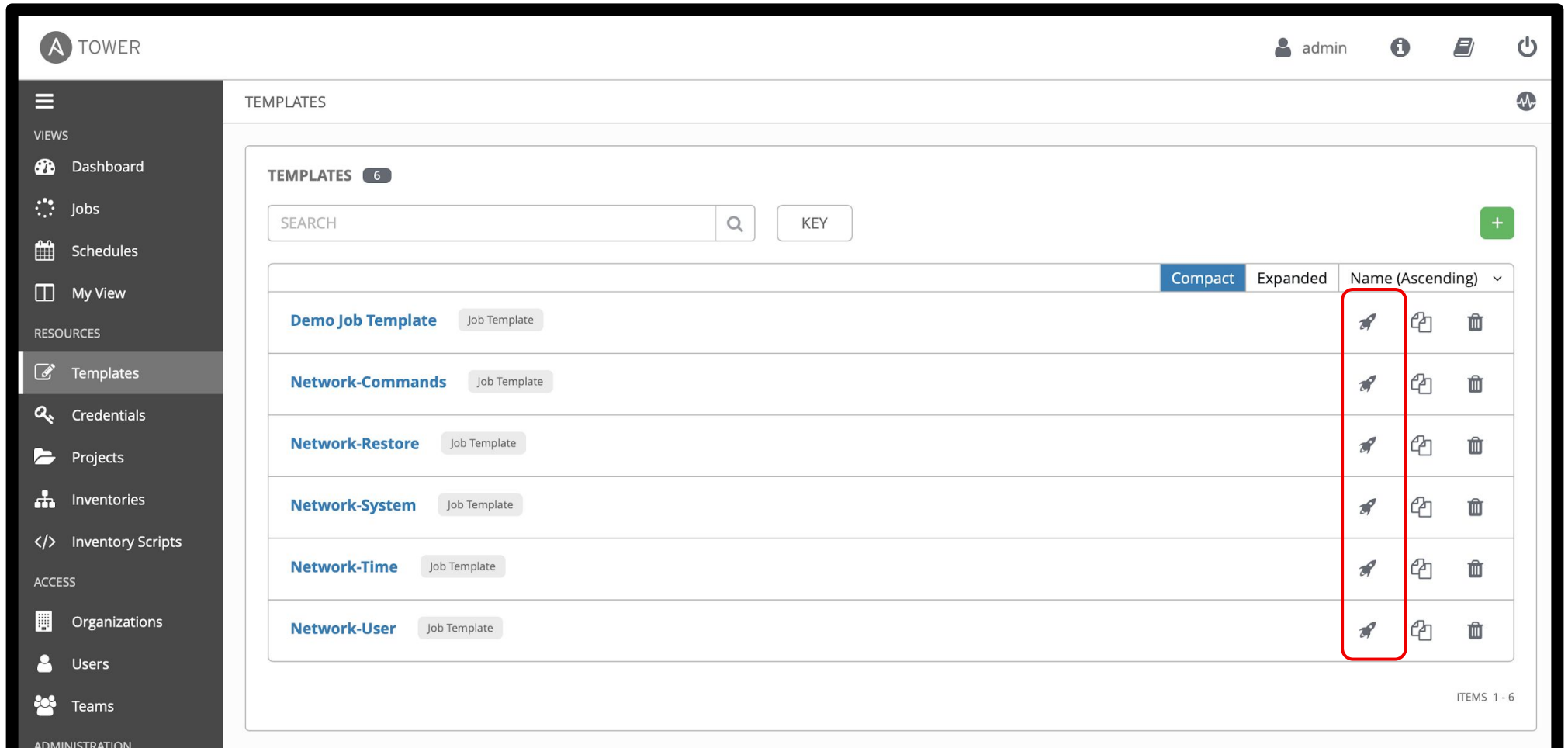
Job Templates can be found and created by clicking the **Templates** button under the *RESOURCES* section on the left menu.



Red Hat

Executing an existing Job Template

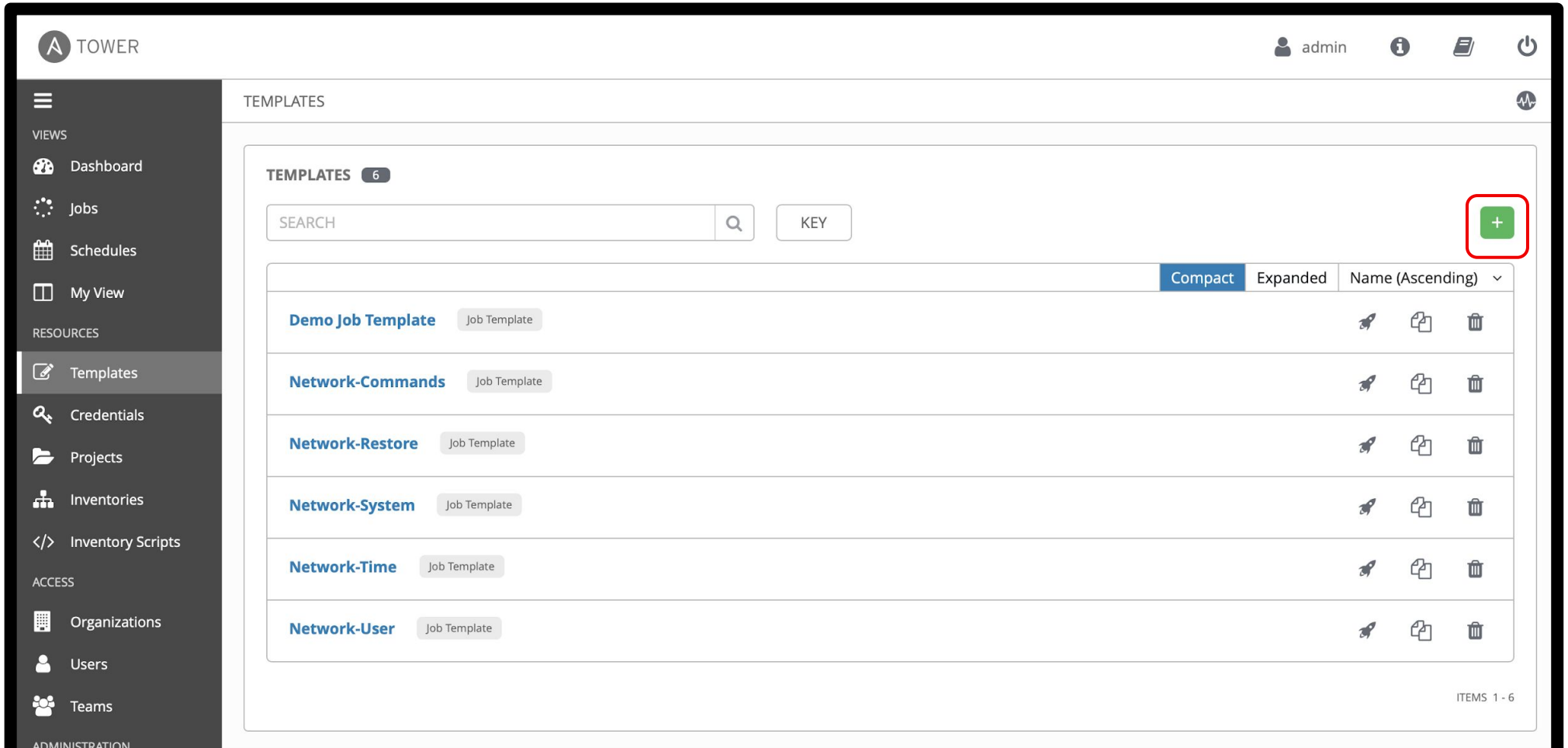
Job Templates can be launched by clicking the **rocketship button** for the corresponding Job Template



Red Hat

Creating a new Job Template (1/2)

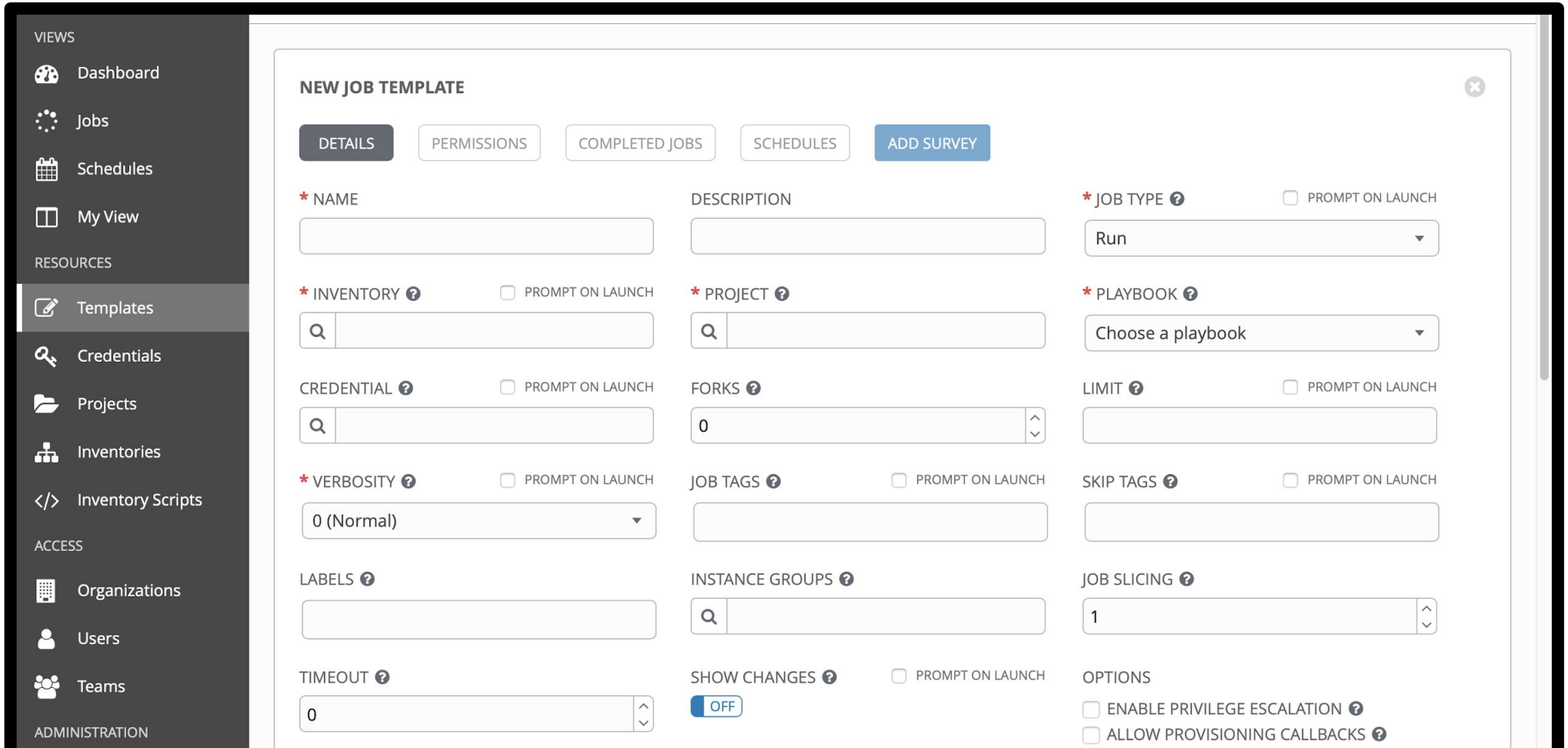
New Job Templates can be created by clicking the **plus button** 



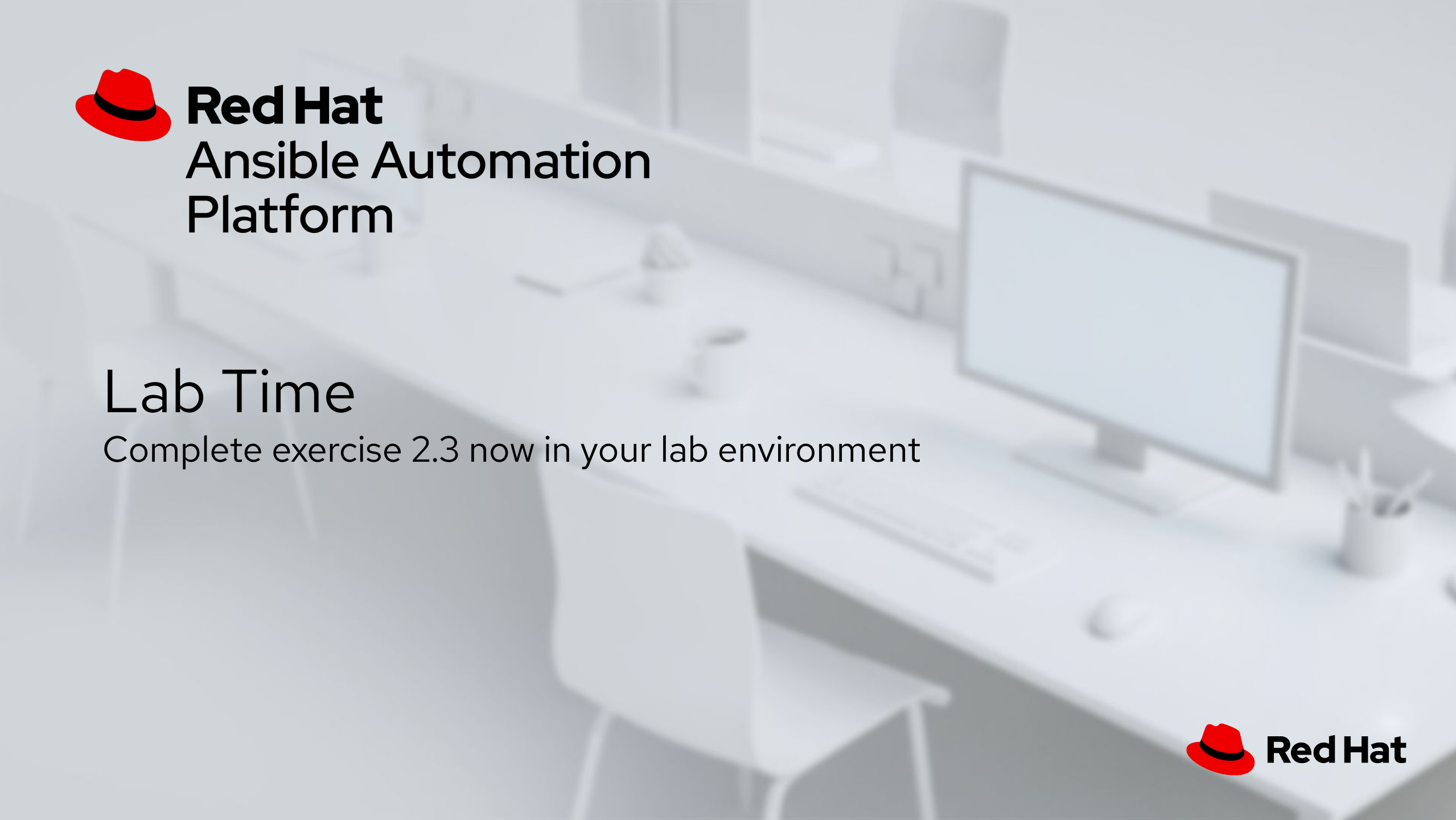
Red Hat

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


Red Hat
Ansible Automation
Platform
Lab Time
Complete exercise 2.3 now in your lab environment
Red Hat

Exercise 2.4

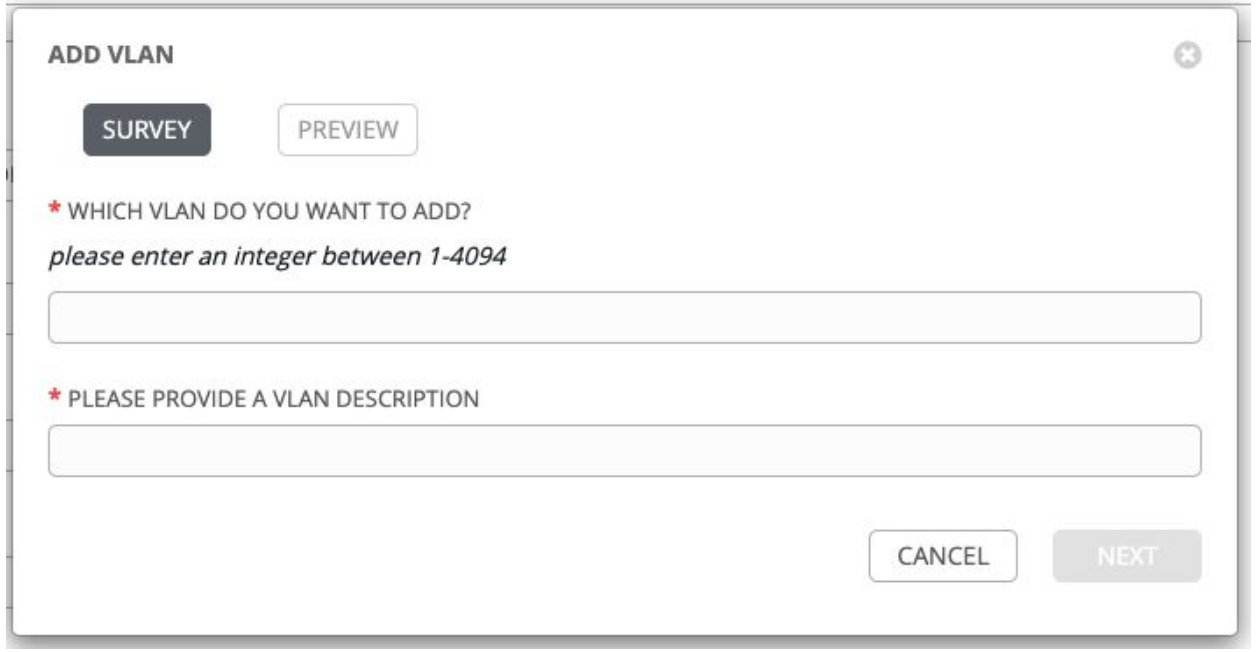
Topics Covered:

- Surveys

Surveys

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

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

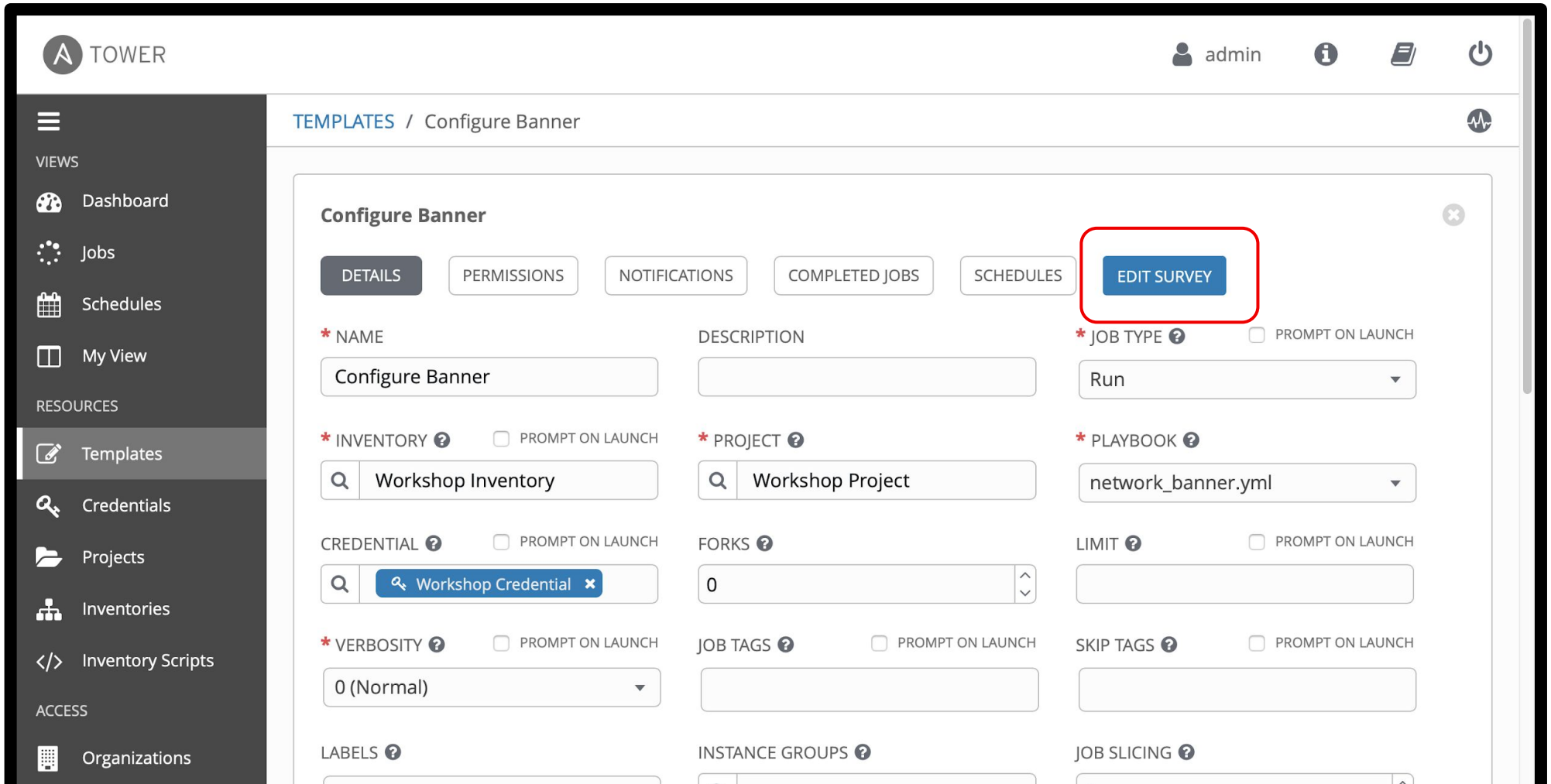


Red Hat

Creating a Survey (1/2)

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

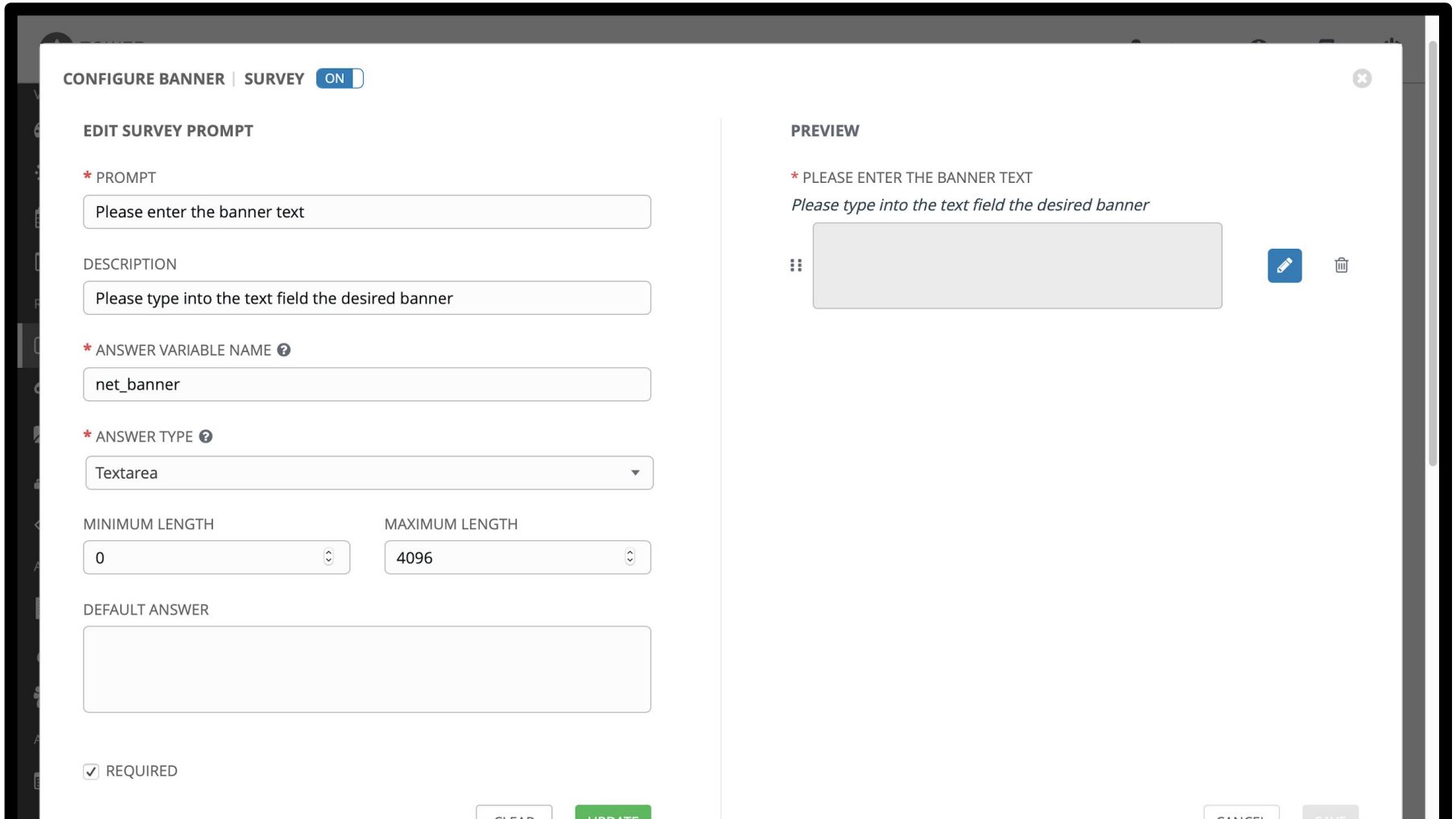
Click the button to open the Add Survey window.



Red Hat

Creating a Survey (2/2)

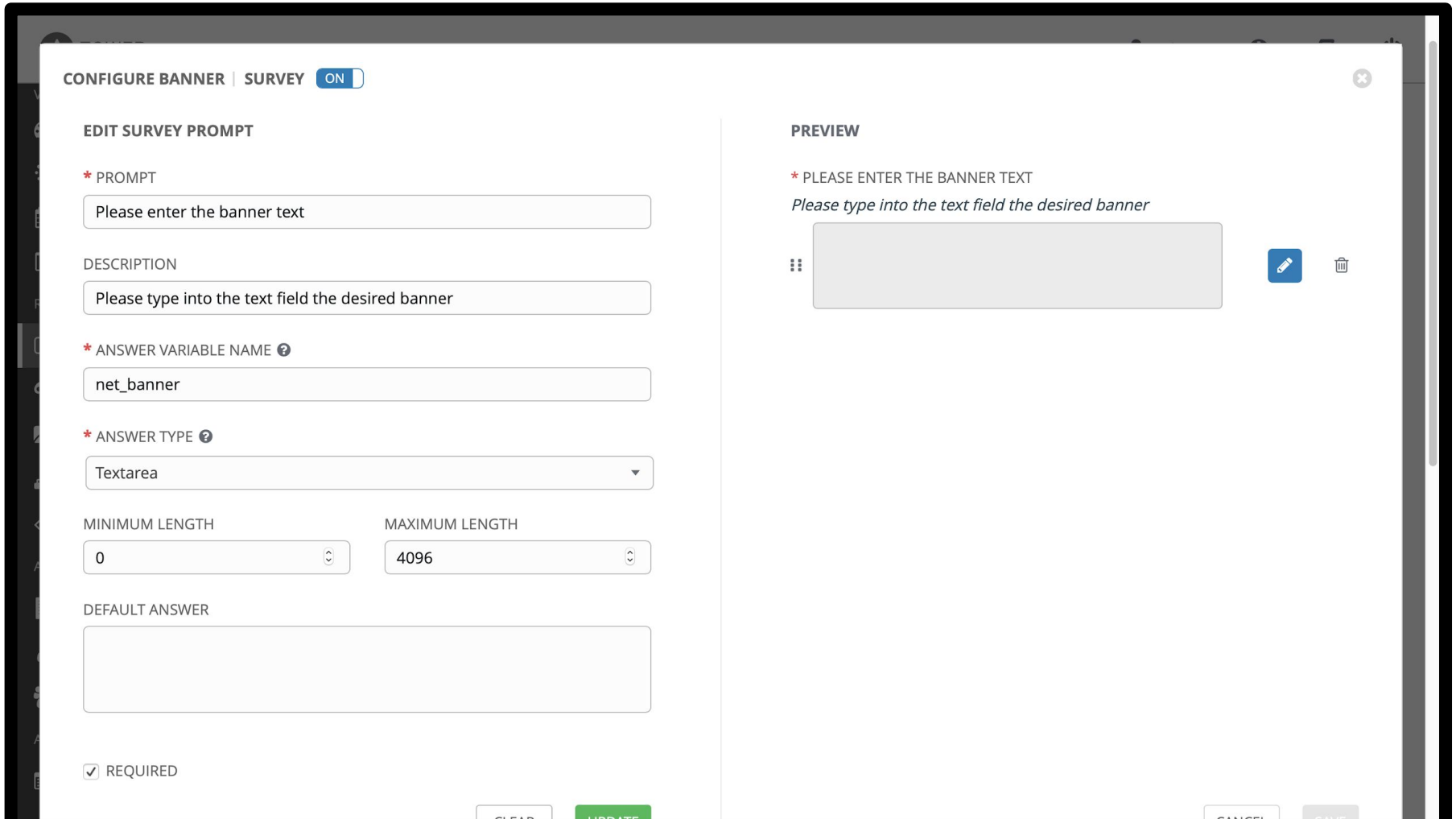
The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.



Red Hat

Creating a Survey (2/2)

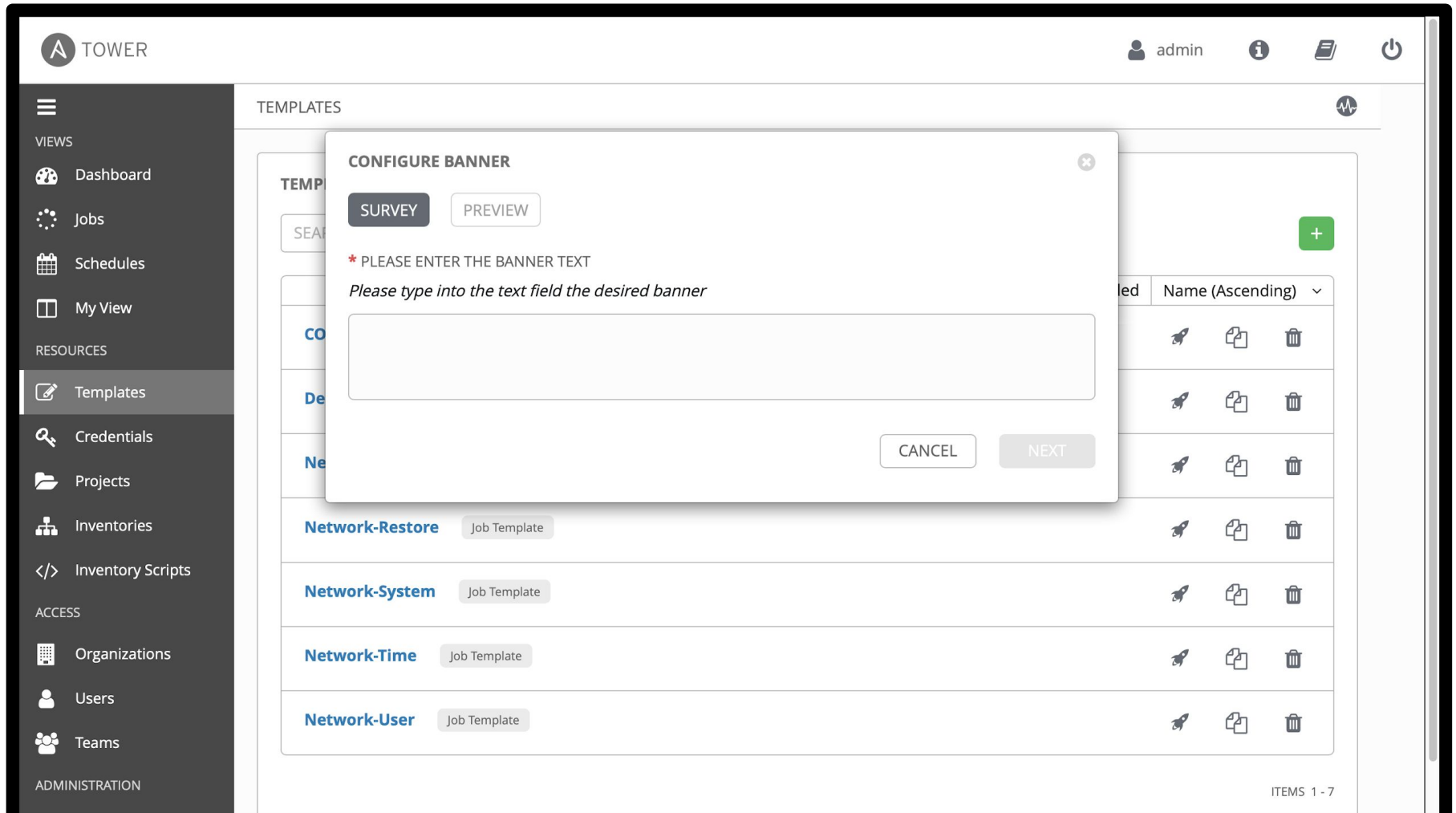
The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.



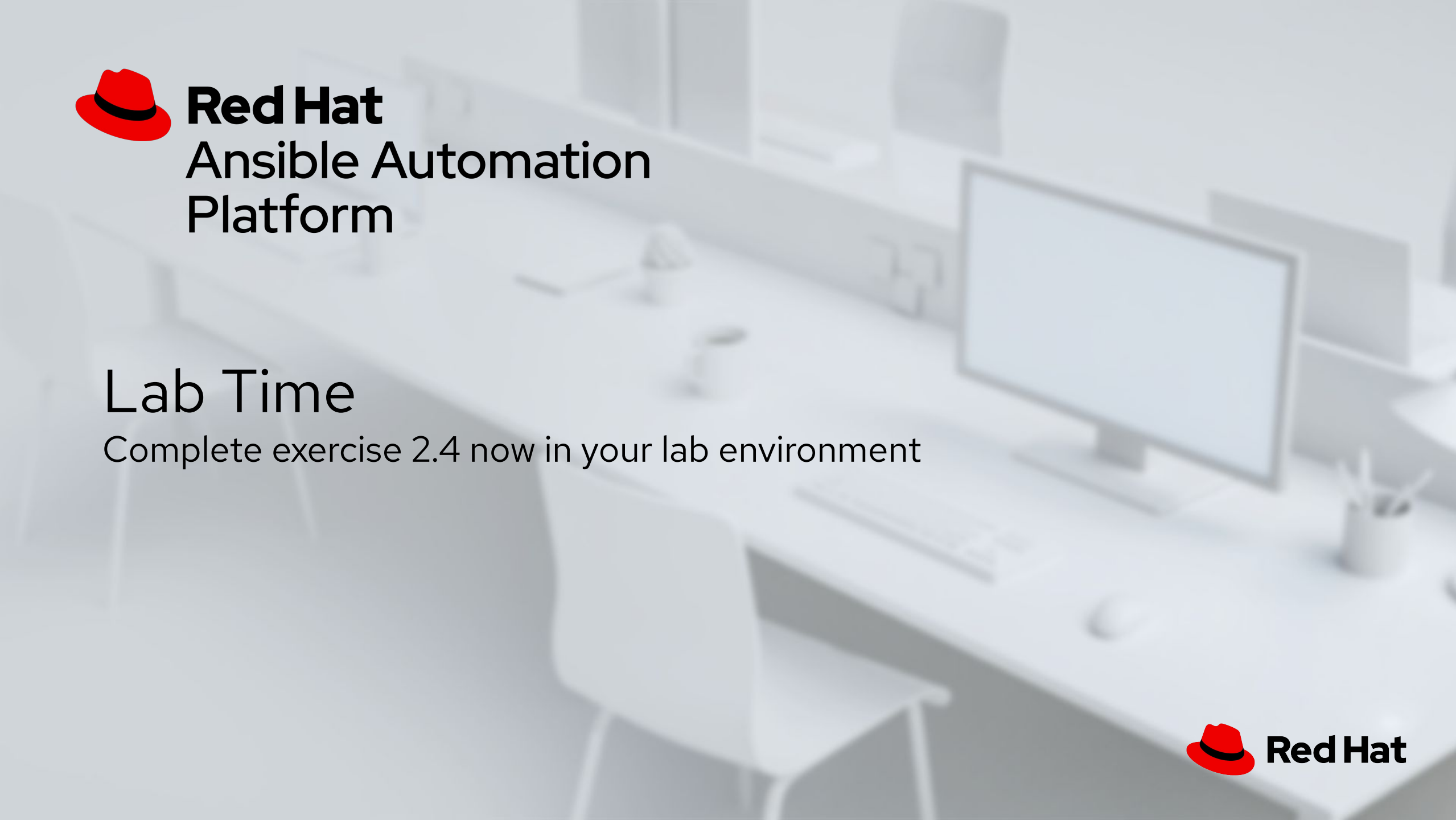
Red Hat

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


Red Hat
Ansible Automation
Platform
Lab Time
Complete exercise 2.4 now in your lab environment
Red Hat

Red Hat
Ansible Automation
Platform

Exercise 2.5

Topics Covered:

- Role based access control

Red Hat

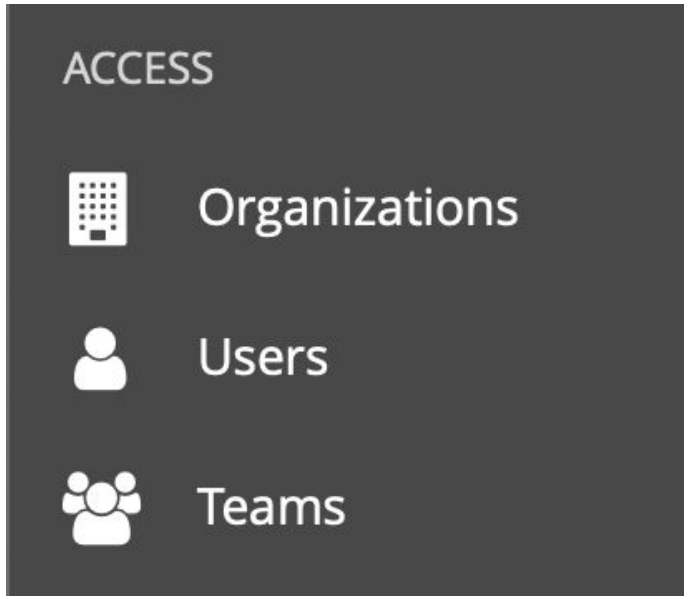
Role Based Access Control (RBAC)

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.

Red Hat

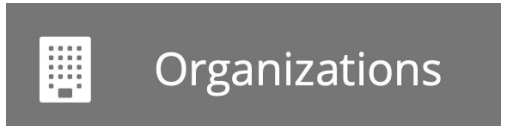
User Management

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

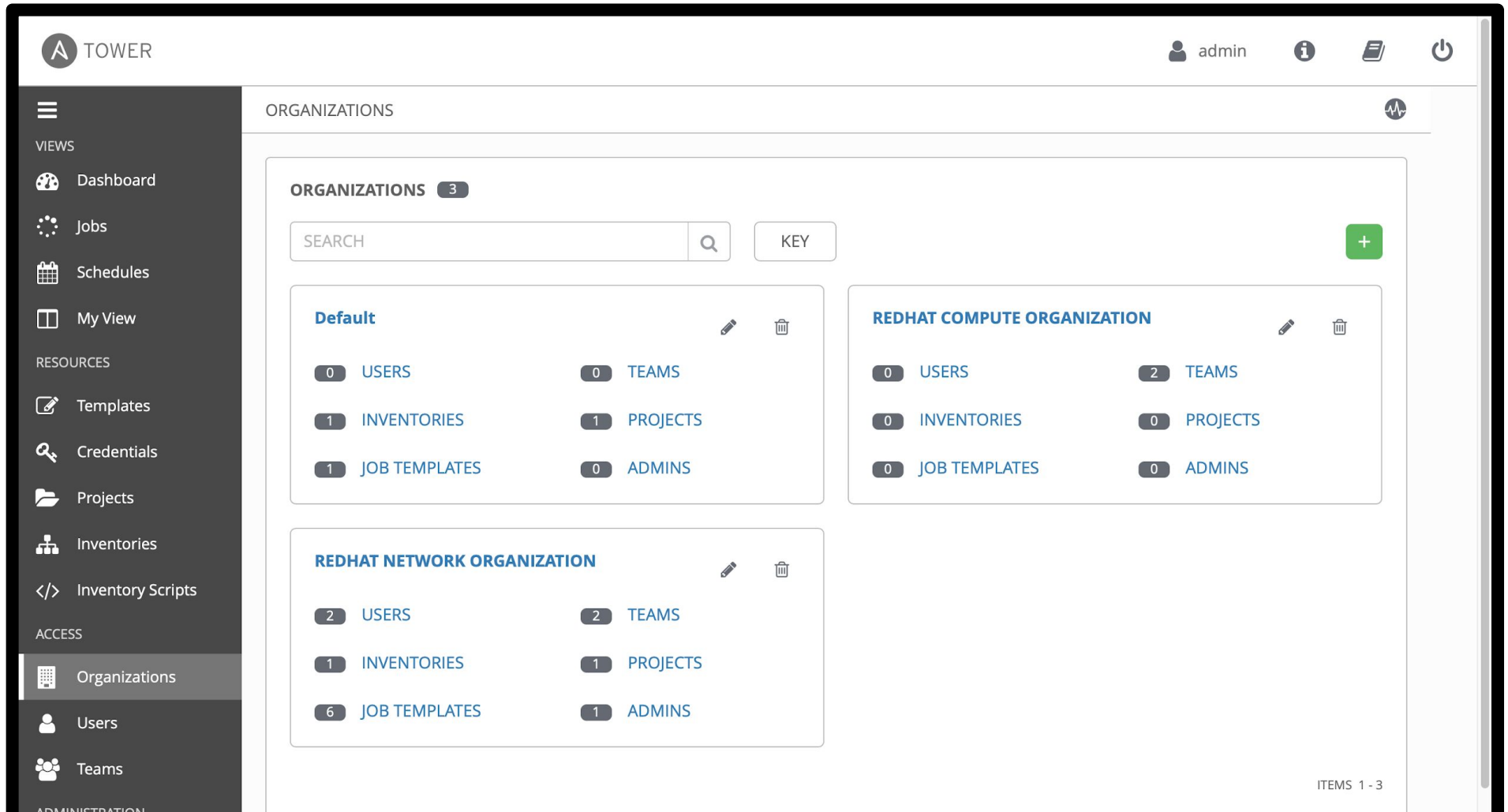


Red Hat

Viewing Organizations

Clicking on the **Organizations** button 

in the left menu will open up the Organizations window

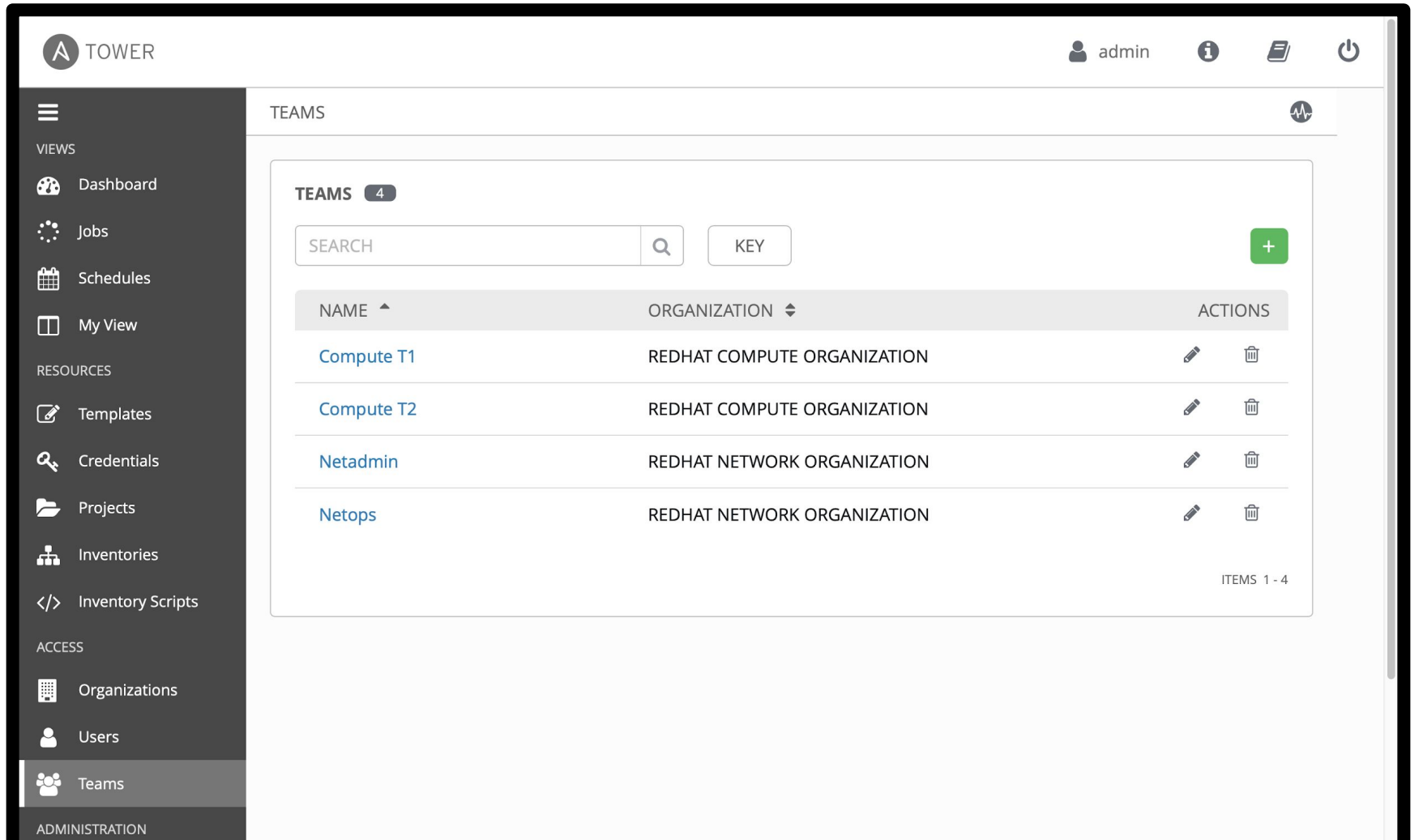


Red Hat

Viewing Teams

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

in the left menu

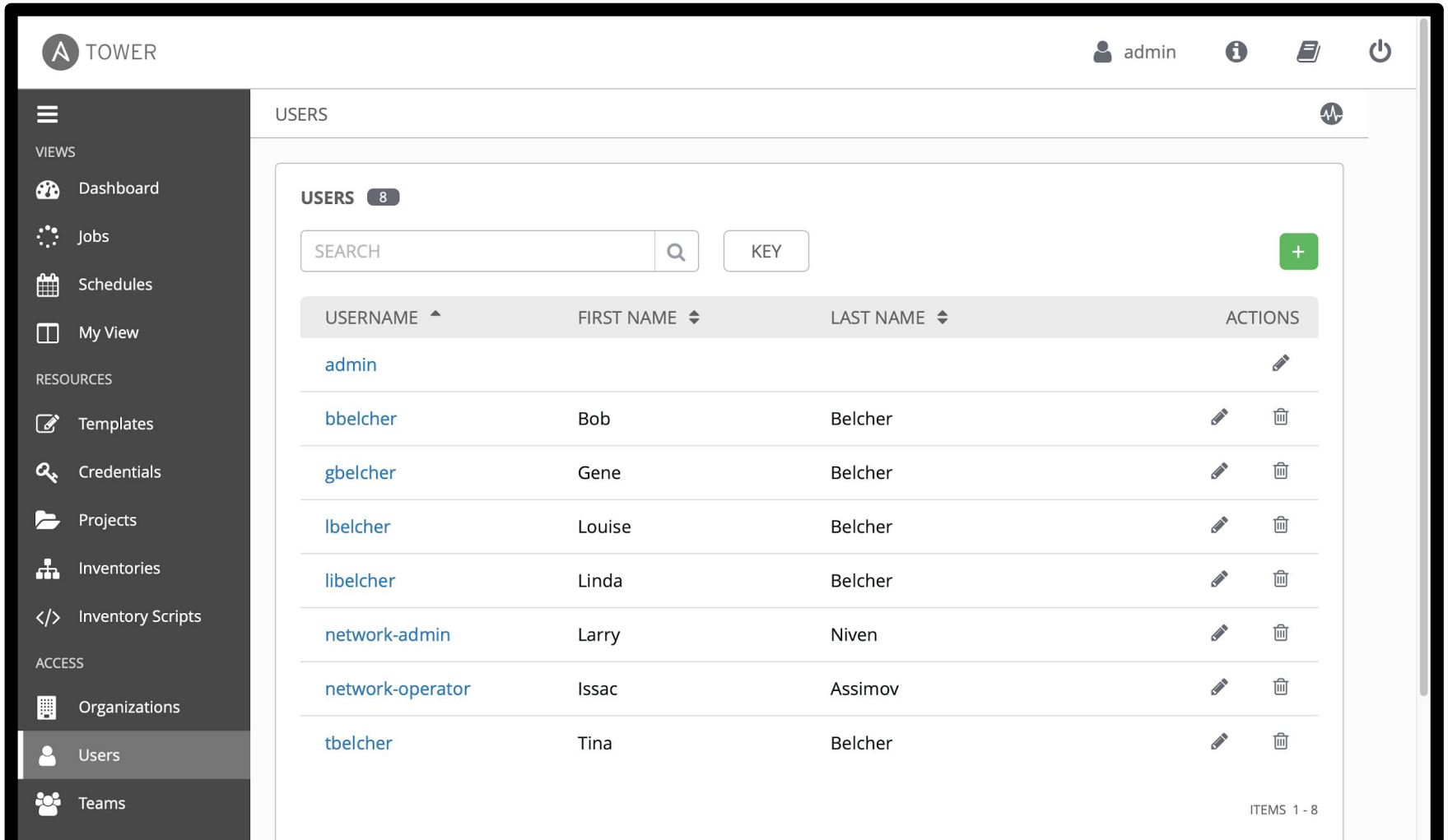


Red Hat

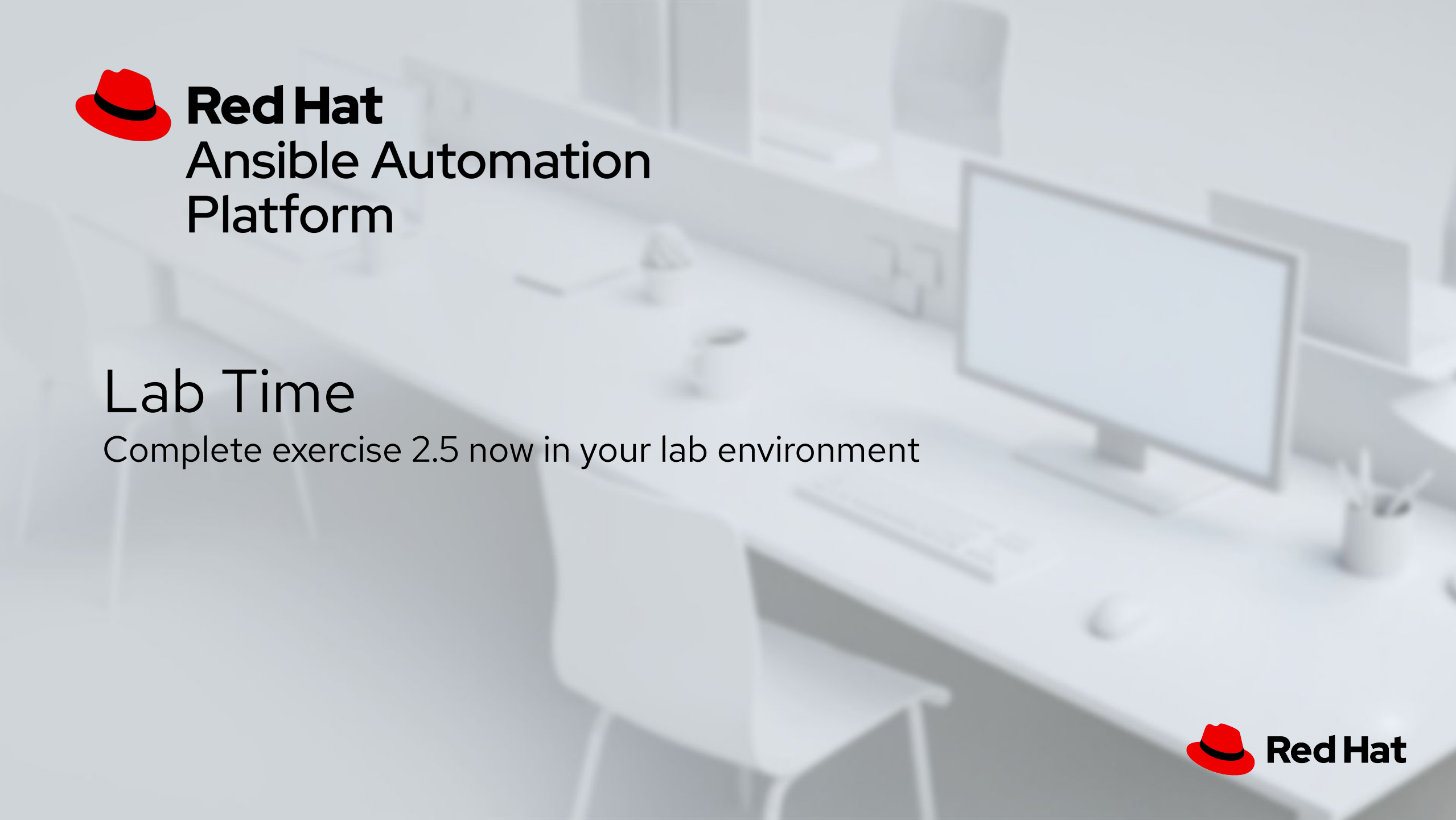
Viewing Users

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

in the left menu



Red Hat


Red Hat
Ansible Automation
Platform
Lab Time
Complete exercise 2.5 now in your lab environment
Red Hat

Exercise 2.6

Topics Covered:

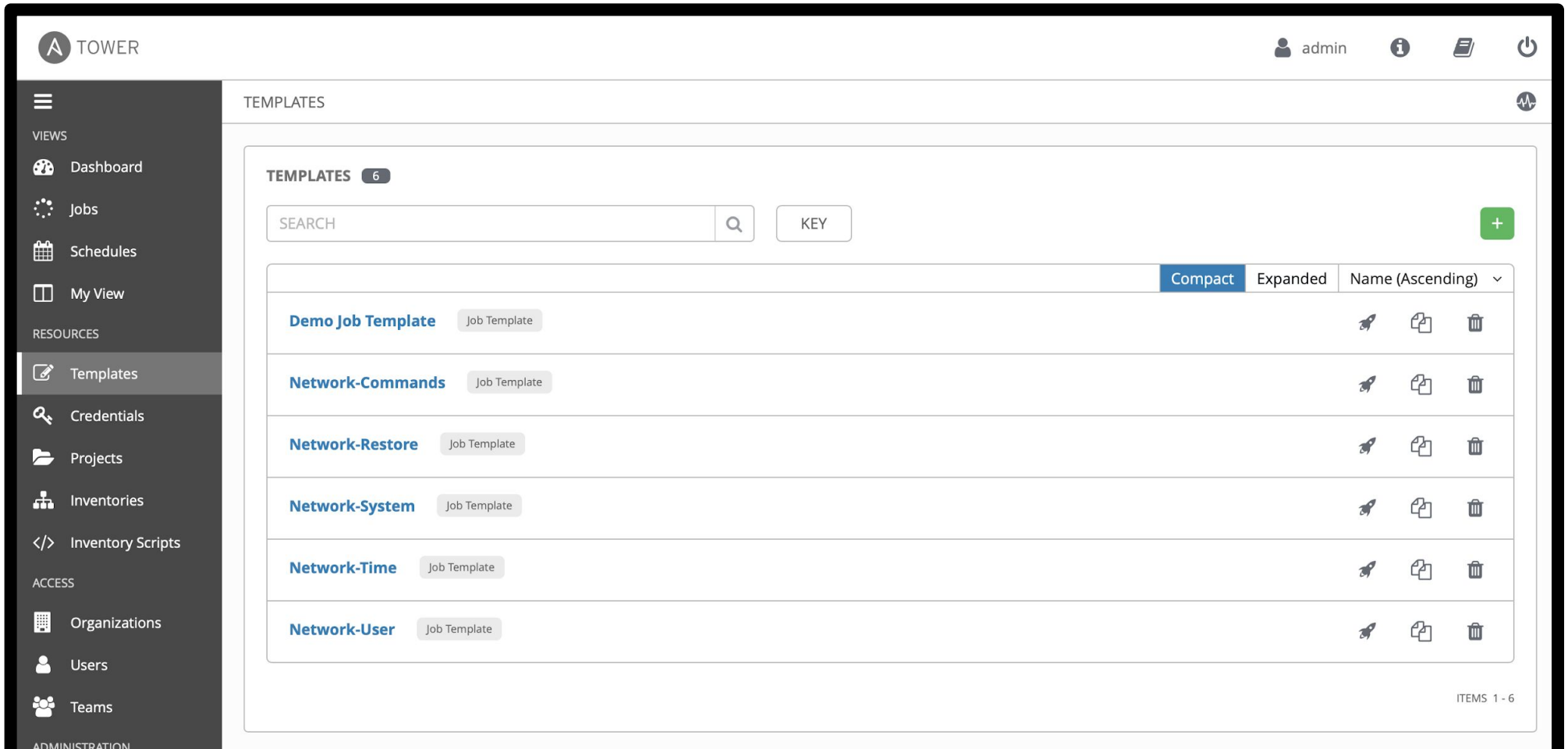
- Workflows

Red Hat
Ansible Automation
Platform

Red Hat

Workflows

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

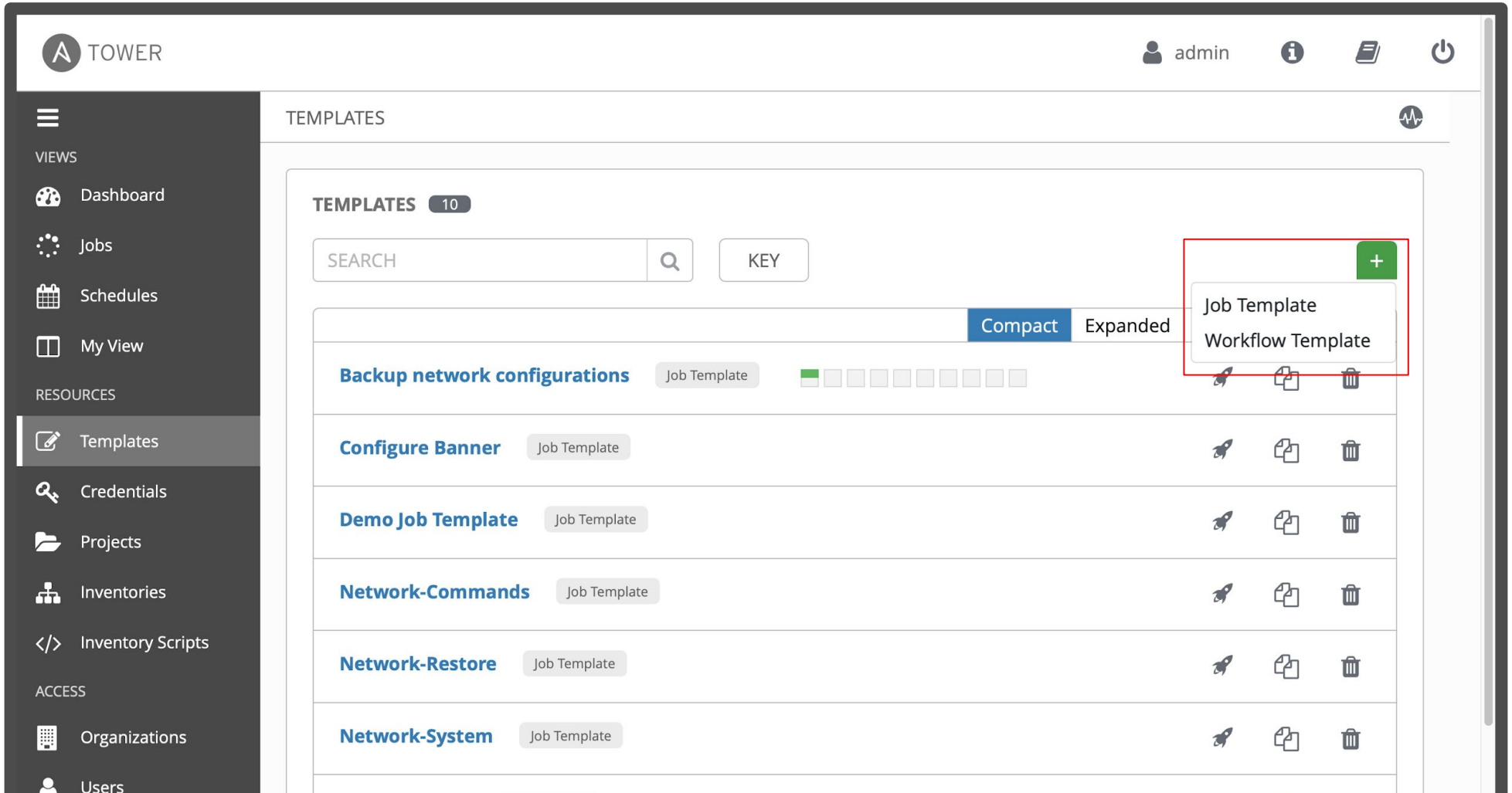


Red Hat

Adding a new Workflow Template

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

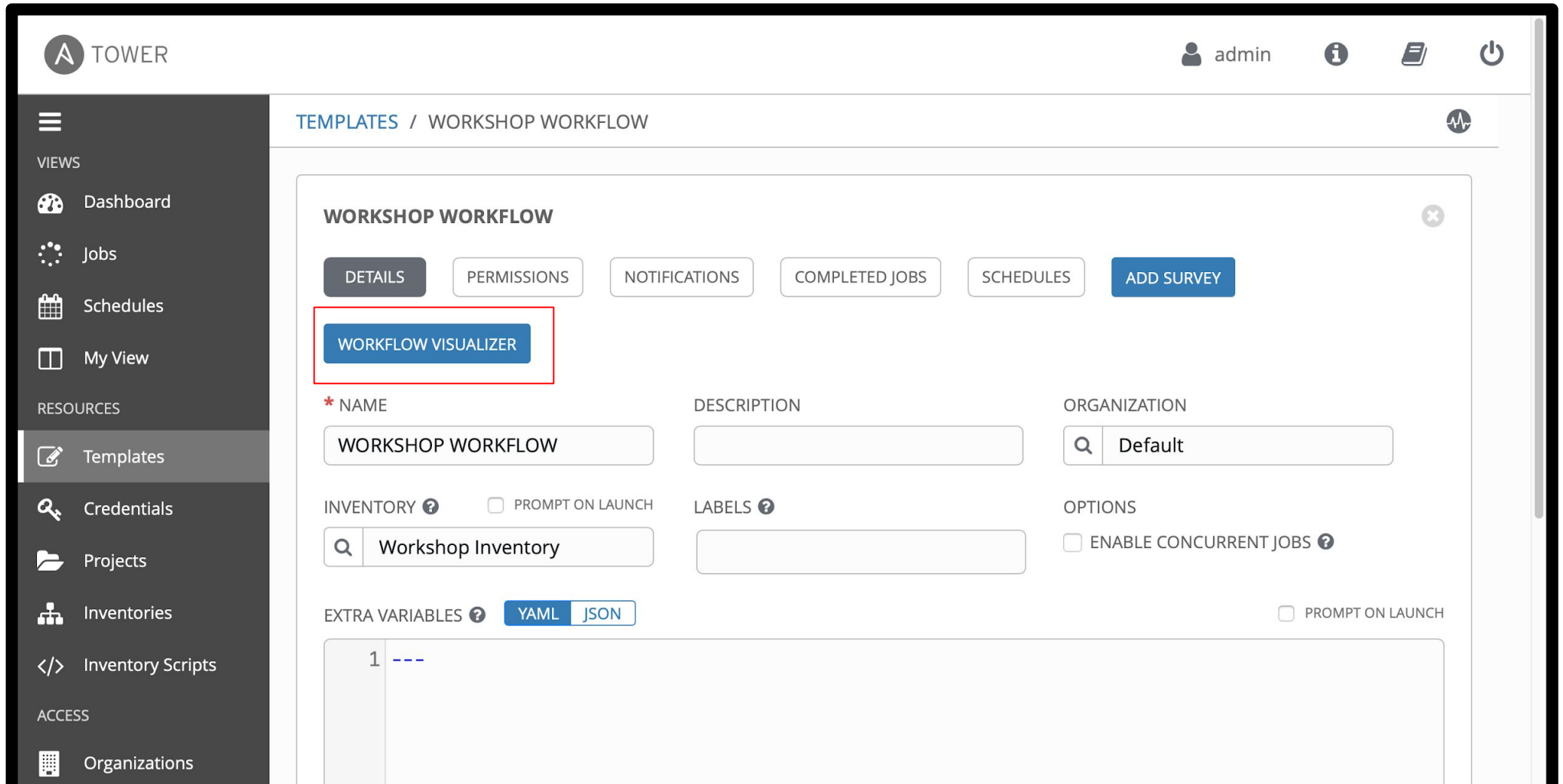
This time select the **Workflow Template**



Red Hat

Creating the Workflow

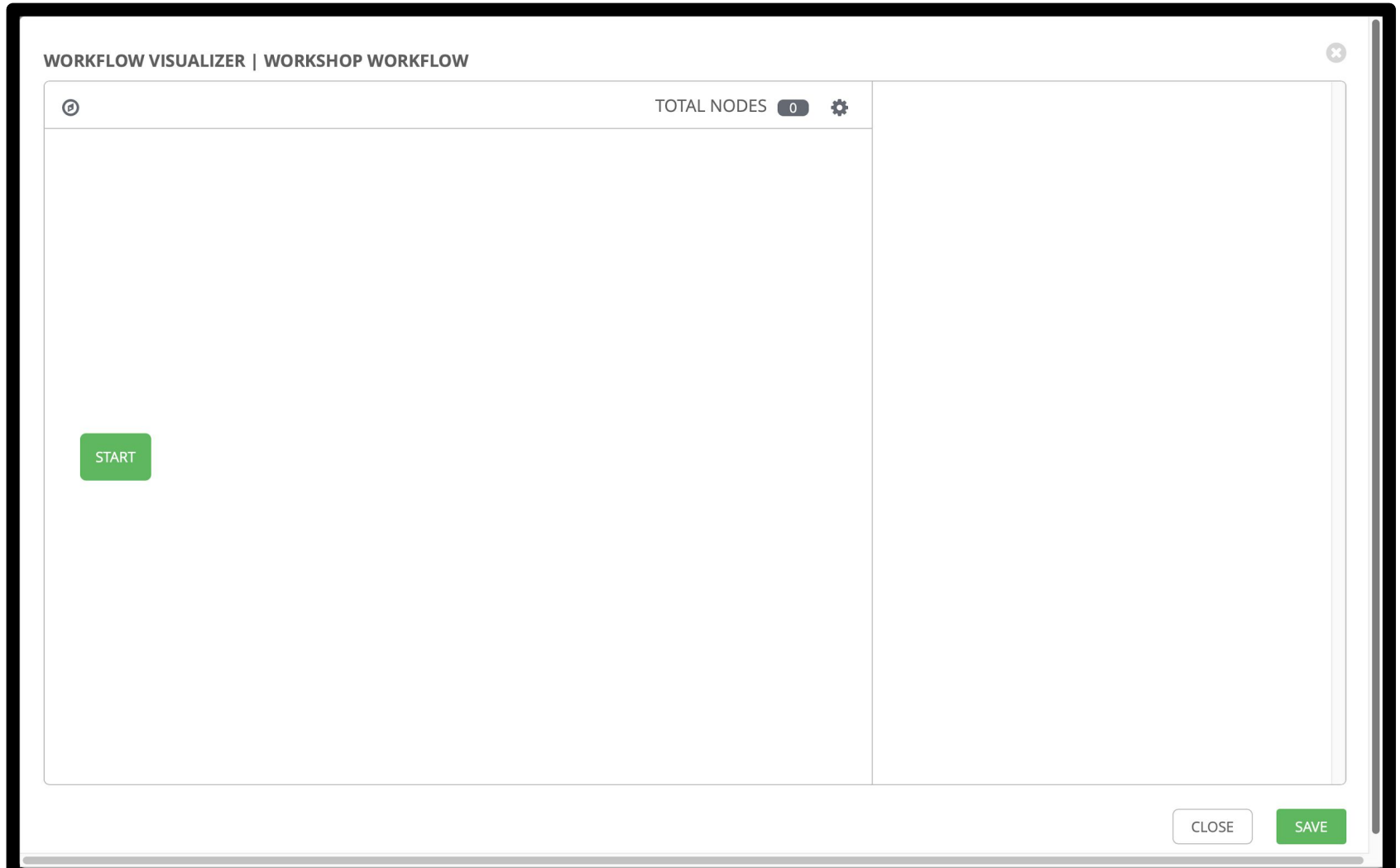
Fill out the required parameters and click **SAVE.** As soon as the Workflow Template is saved the WORKFLOW VISUALIZER will open.



Red Hat

Workflow Visualizer

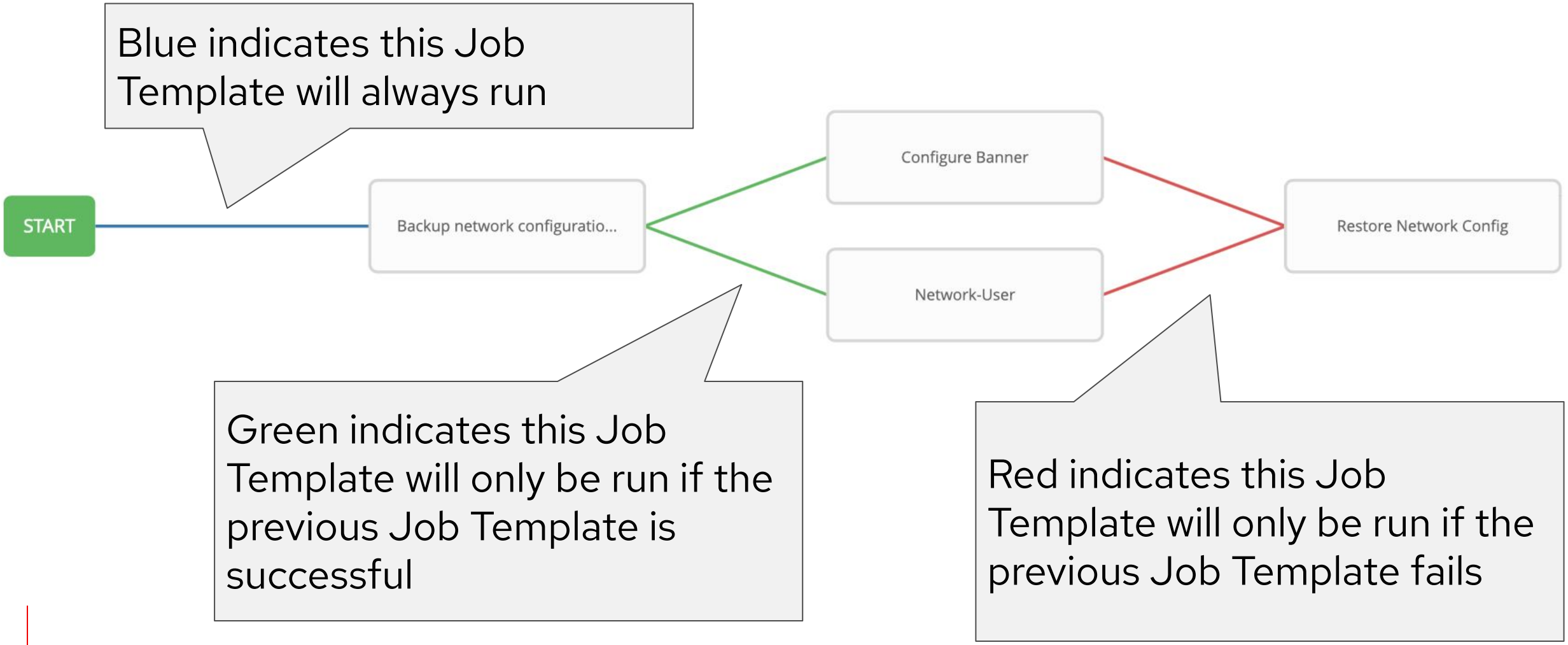
The workflow visualizer will start as a blank canvas.

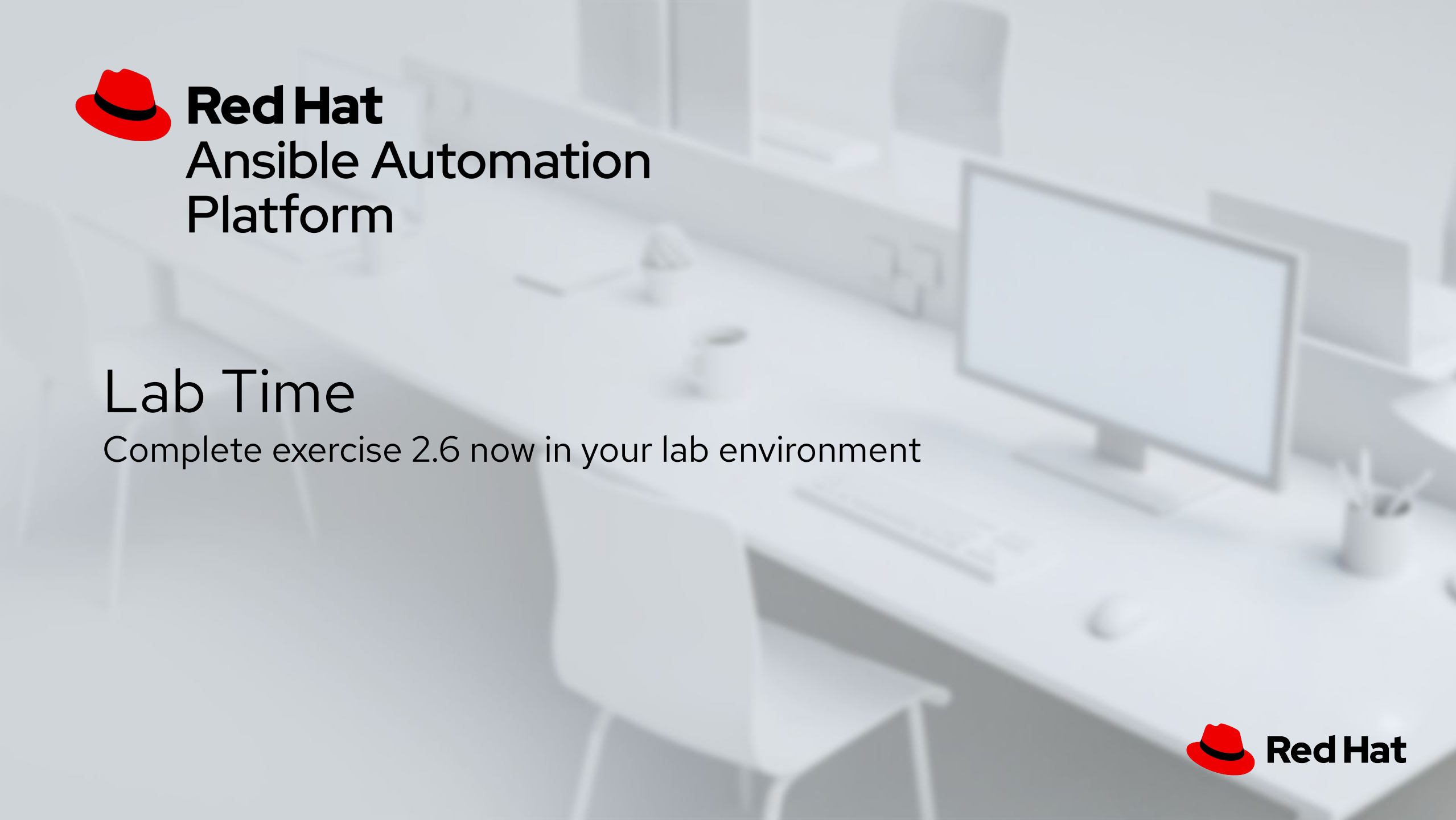


Red Hat

Visualizing a Workflow

Workflows can branch out, or converge in.




Red Hat
Ansible Automation
Platform
Lab Time
Complete exercise 2.6 now in your lab environment
Red Hat

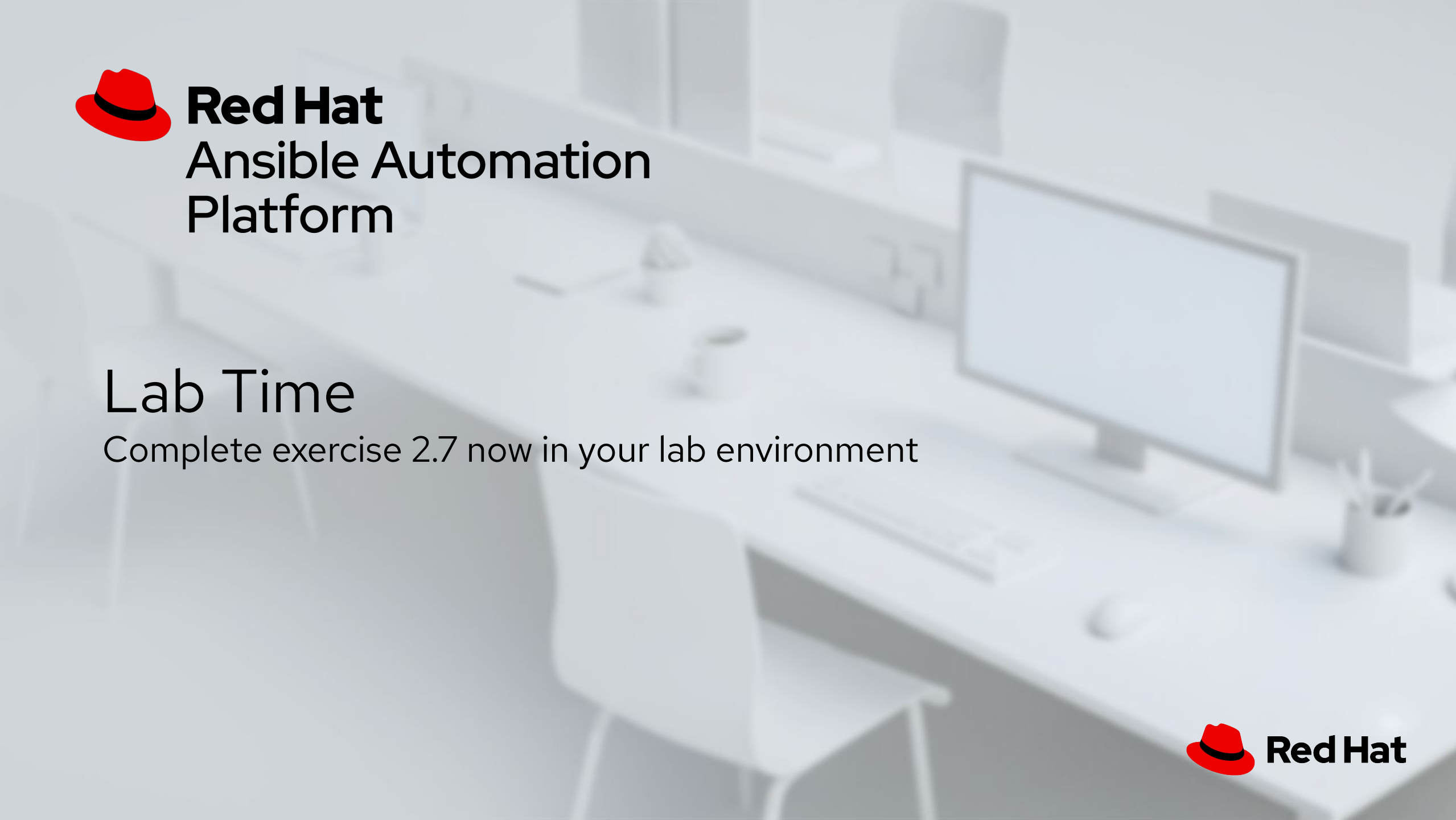
Red Hat
Ansible Automation
Platform

Exercise 2.7

Topics Covered:

- Wrap-up

Red Hat


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

JOIN THE COMMUNITY

ansible.com/community

WORKSHOPS & TRAINING

ansible.com/workshops

Red Hat Training

SHARE YOUR STORY

Follow us @Ansible

Friend us on Facebook

Red Hat

AnsibleFest

October 13-14, 2020 | Virtual Experience

Red Hat

Thank you


linkedin.com/company/red-hat


youtube.com/AnsibleAutomation


facebook.com/ansibleautomation


twitter.com/ansible


github.com/ansible

Red Hat