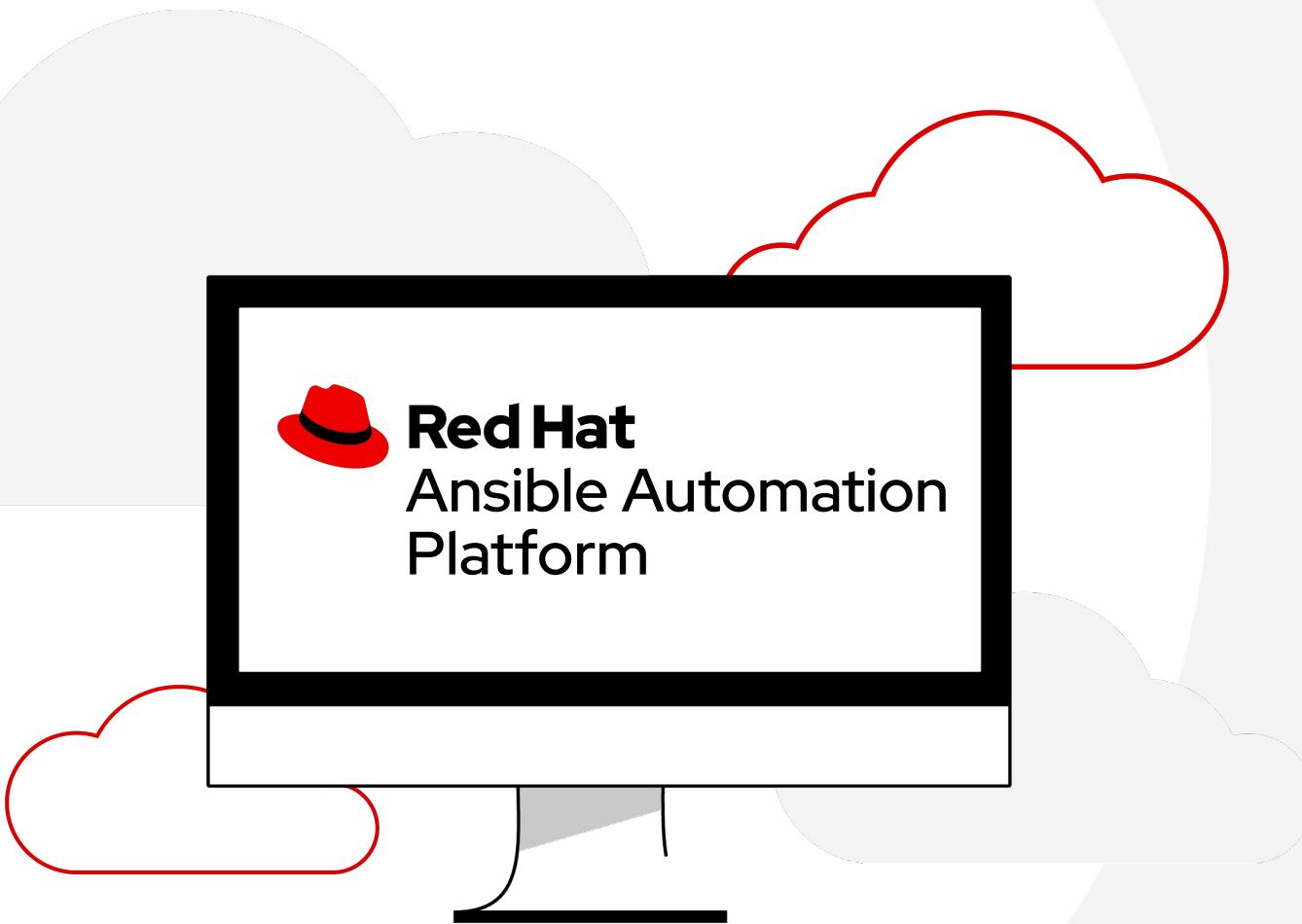




**Red Hat**  
Ansible Automation  
Platform

# Ansible Linux Automation Workshop

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



# What you will learn

- ▶ Intro to Ansible Automation Platform
- ▶ How it Works
- ▶ Understanding modules, tasks, playbooks
- ▶ How to execute Ansible commands
- ▶ Using variables and templates
- ▶ Automation Controller - where it fits in
- ▶ Automation Controller basics
- ▶ Major Automation Controller features - RBAC, workflows



**Red Hat**  
Ansible Automation  
Platform

# Introduction

Topics Covered:

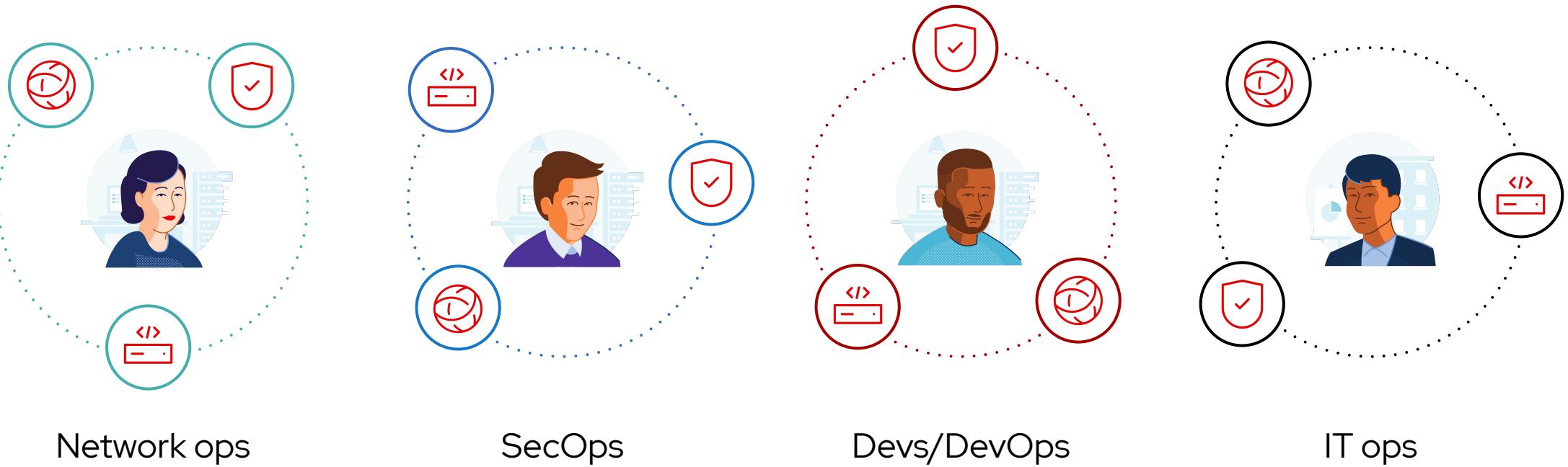
- Why the Ansible Automation Platform?
- What can it do?



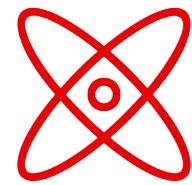
**Automation happens when  
one person meets a problem  
they never want to solve again**

# Many organizations share the same challenge

Too many unintegrated, domain-specific tools

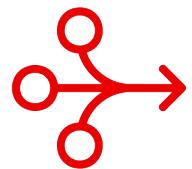


# Why the Ansible Automation Platform?



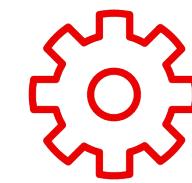
## Powerful

Orchestrate complex processes at enterprise scale.



## Simple

Simplify automation creation and management across multiple domains.



## Agentless

Easily integrate with hybrid environments.

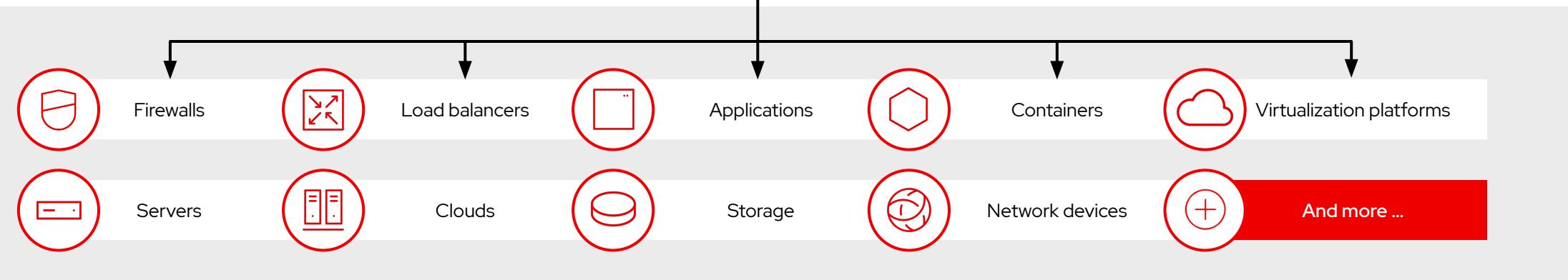
# Automate the deployment and management of automation

Your entire IT footprint

Do this...

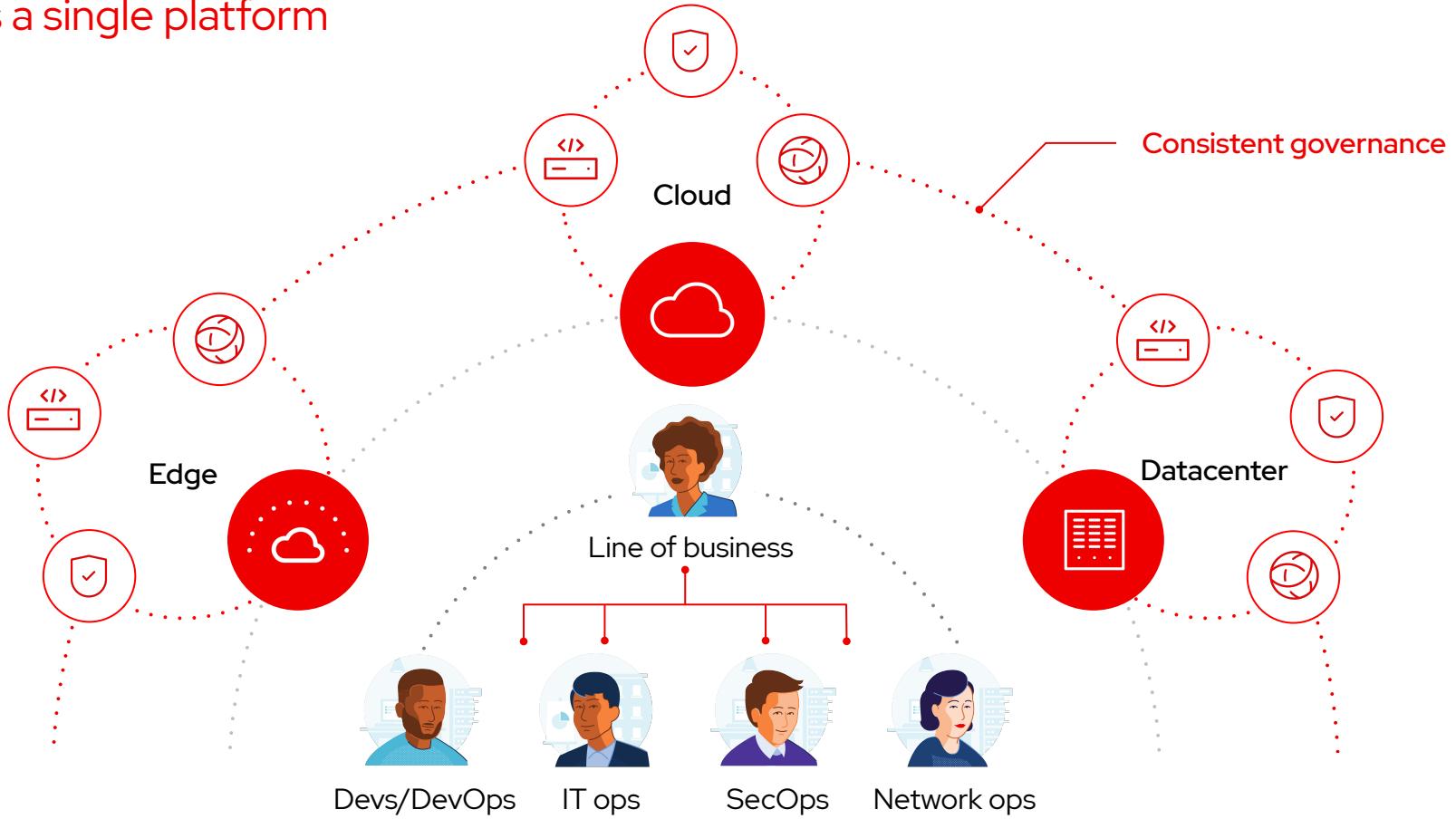
Orchestrate      Manage configurations      Deploy applications      Provision / deprovision      Deliver continuously      Secure and comply

On these...



# Break down silos

Different teams a single platform





## Red Hat Ansible Automation Platform



Content creators



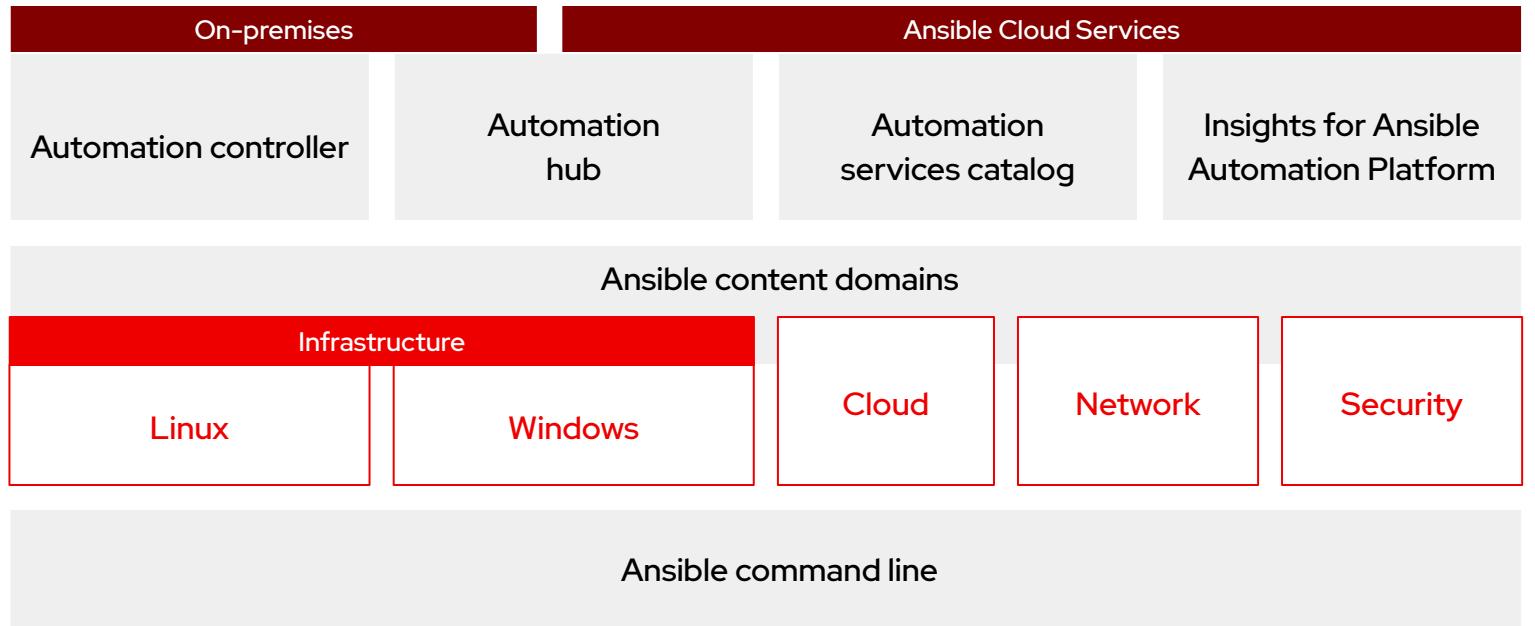
Operators



Domain experts



Users



Fueled by an  
open source community

THE FORRESTER WAVE™  
Infrastructure Automation Platforms  
Q3 2020



## Red Hat named a Leader in The Forrester Wave™

Infrastructure Automation Platforms, Q3 2020

Received highest possible score in the criteria of:



- Deployment functionality
- Product Vision
- Partner Ecosystem
- Supporting products and services
- Community support
- Planned product enhancements

- ▶ “Ansible continues to grow quickly, particularly among enterprises that are automating networks. The solution excels at providing a variety of deployment options and acting as a service broker to a wide array of other automation tools.”
- ▶ “Red Hat’s solution is a good fit for customers that want a holistic automation platform that integrates with a wide array of other vendors’ infrastructure.”

Source:

Gardner, Chris, Glenn O'Donnell, Robert Perdonii, and Diane Lynch. "[The Forrester Wave™: Infrastructure Automation Platforms, Q3 2020](#)." Forrester, 10 Aug. 2020.

DISCLAIMER: The Forrester Wave™ is copyrighted by Forrester Research, Inc. Forrester and Forrester Wave™ are trademarks of Forrester Research, Inc. The Forrester Wave™ is a graphical representation of Forrester's call on a market and is plotted using a detailed spreadsheet with exposed scores, weightings, and comments. Forrester does not endorse any vendor, product, or service depicted in the Forrester Wave™.



**Red Hat**  
Ansible Automation  
Platform

# Section 1

# The Ansible Basics



**Red Hat**  
Ansible Automation  
Platform

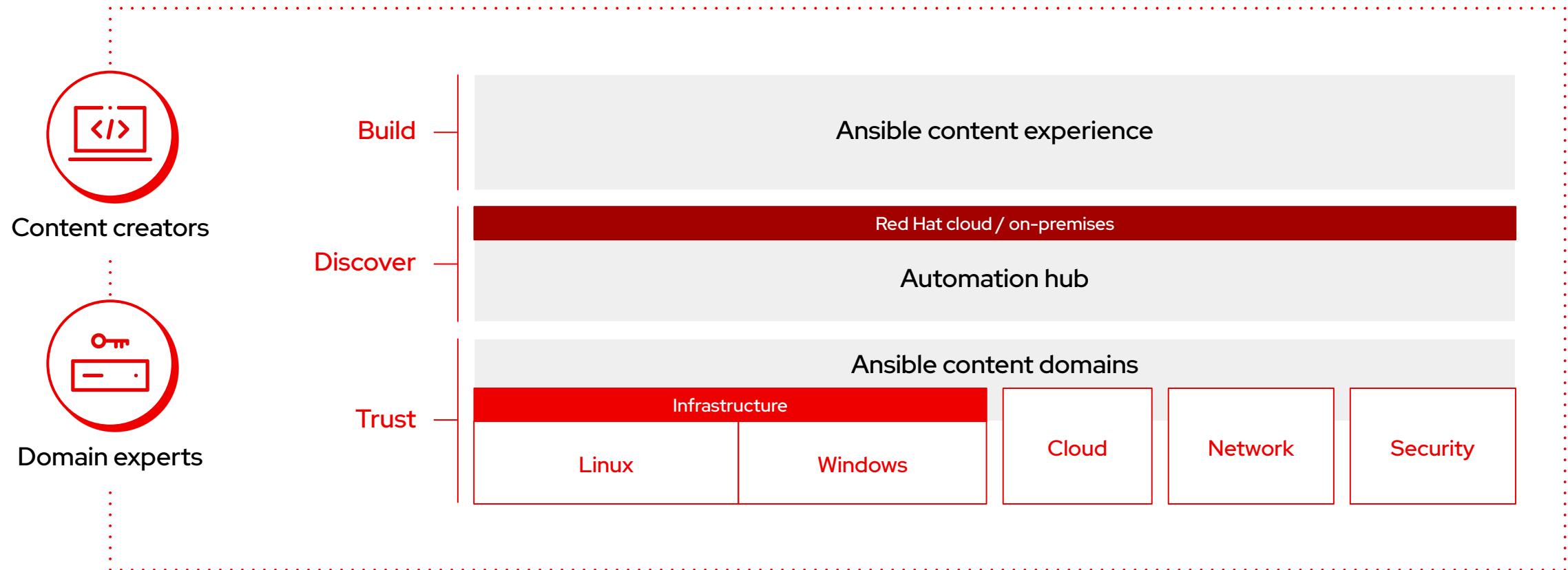
# Exercise 1.1

Topics Covered:

- Understanding the Ansible Infrastructure
- Check the prerequisites

# Create

## The automation lifecycle





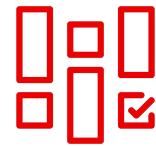
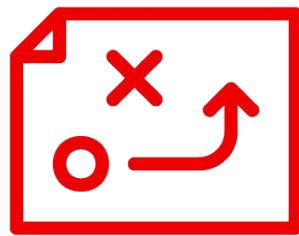
```
---
- name: install and start apache
  hosts: web
  become: yes

  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
```

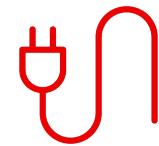
# What makes up an Ansible playbook?



Plays



Modules



Plugins

# Ansible plays

What am I automating?



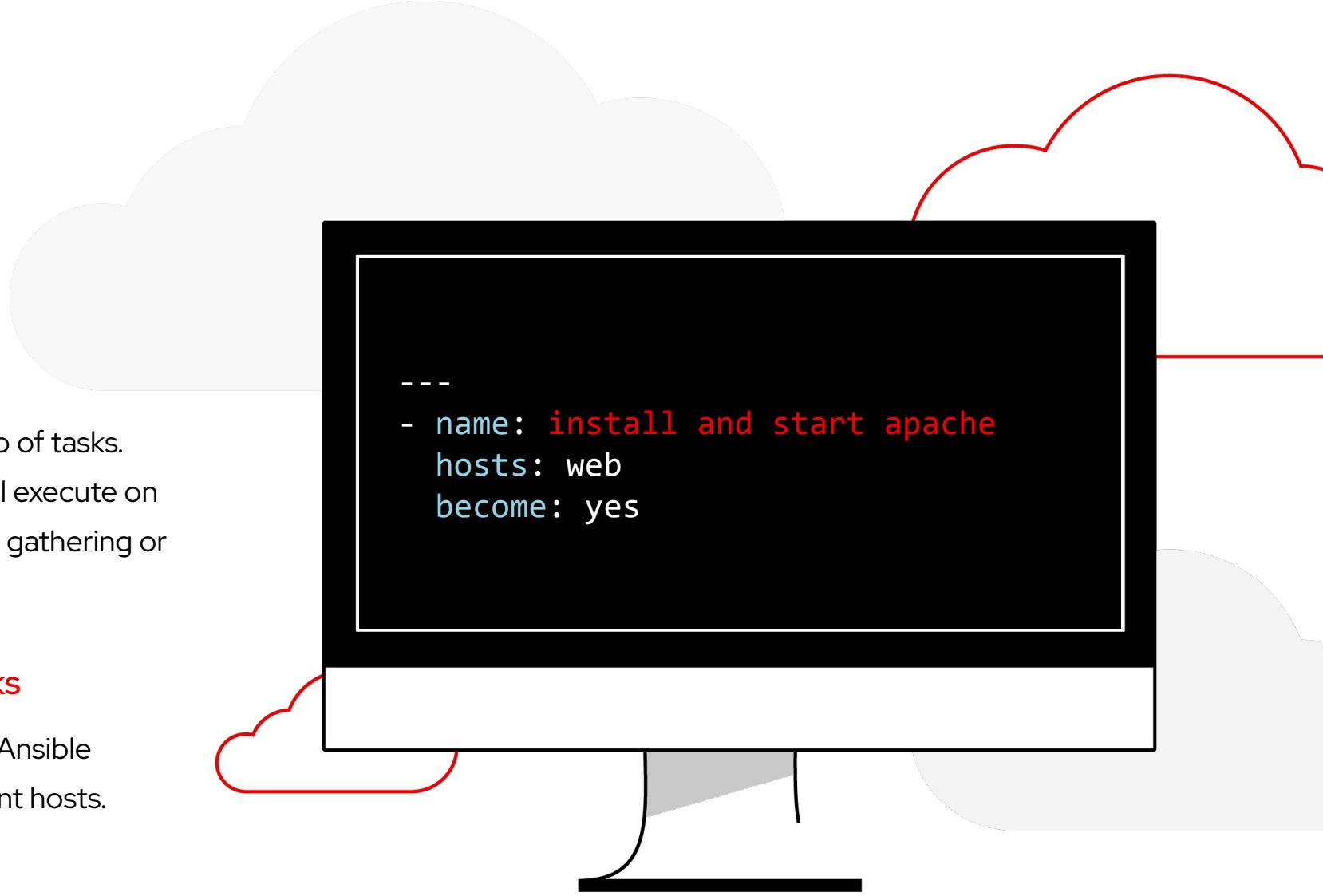
## What are they?

Top level specification for a group of tasks.  
Will tell that play which hosts it will execute on  
and control behavior such as fact gathering or  
privilege level.



## Building blocks for playbooks

Multiple plays can exist within an Ansible  
playbook that execute on different hosts.



# Ansible modules

The “tools in the toolkit”



## What are they?

Parametrized components with internal logic,  
representing a single step to be done.  
The modules “do” things in Ansible.



## Language

Usually Python, or Powershell for Windows  
setups. But can be of any language.



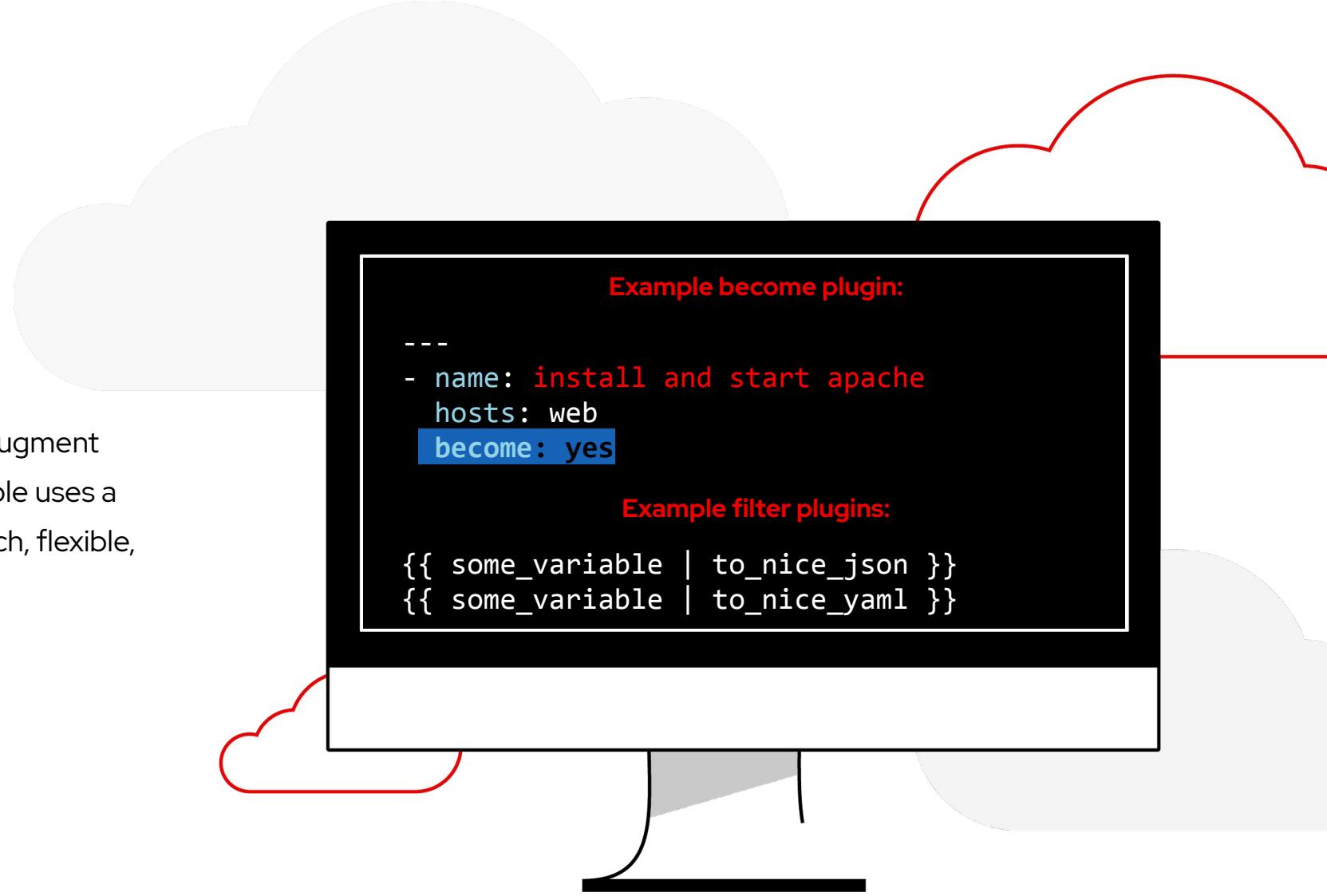
# Ansible plugins

The “extra bits”



## What are they?

Plugins are pieces of code that augment Ansible's core functionality. Ansible uses a plugin architecture to enable a rich, flexible, and expandable feature set.



# Ansible Inventory

The systems that a playbook runs against



What are they?

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
```

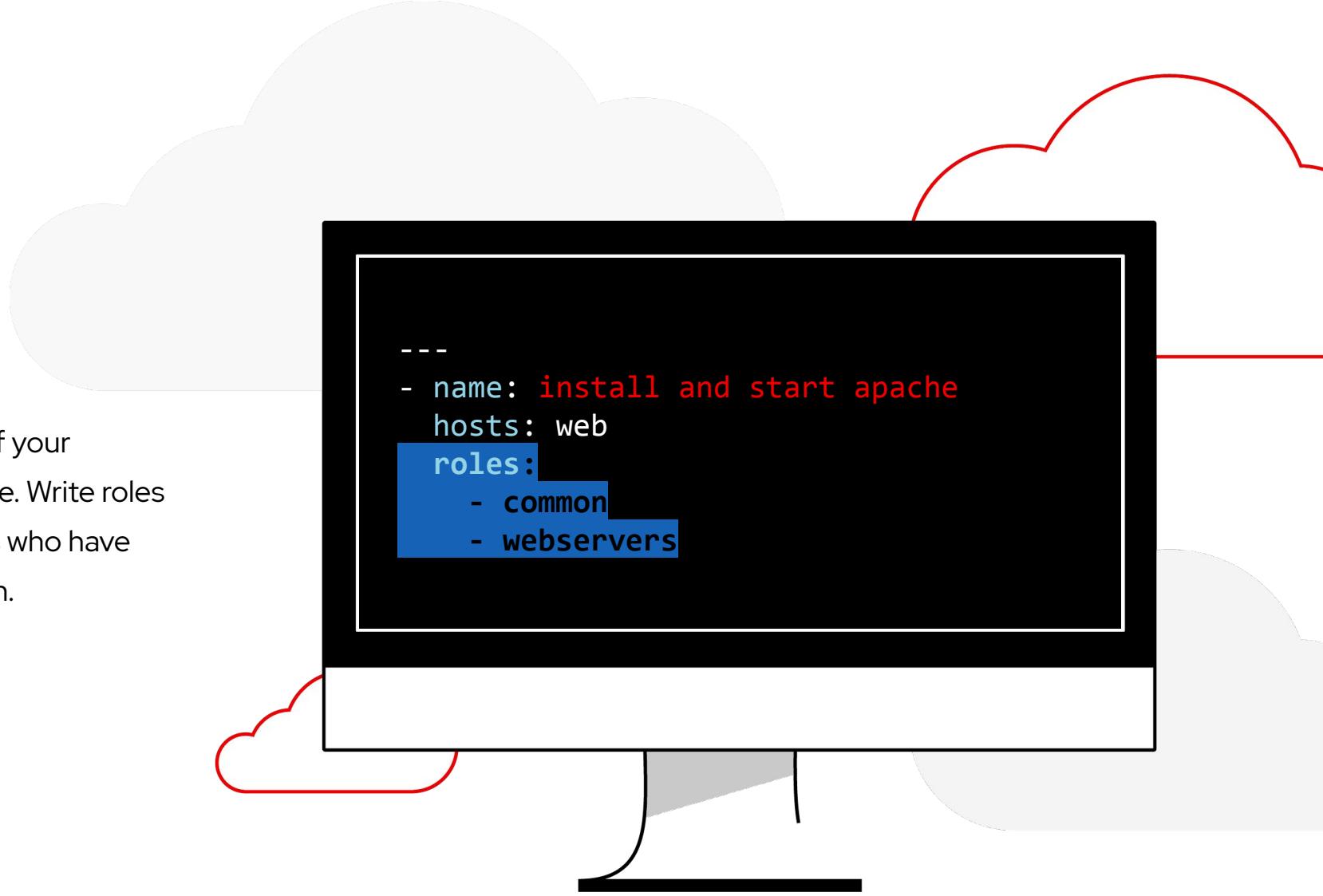
# Ansible roles

Reusable automation actions



## What are they?

Group your tasks and variables of your automation in a reusable structure. Write roles once, and share them with others who have similar challenges in front of them.



# Collections

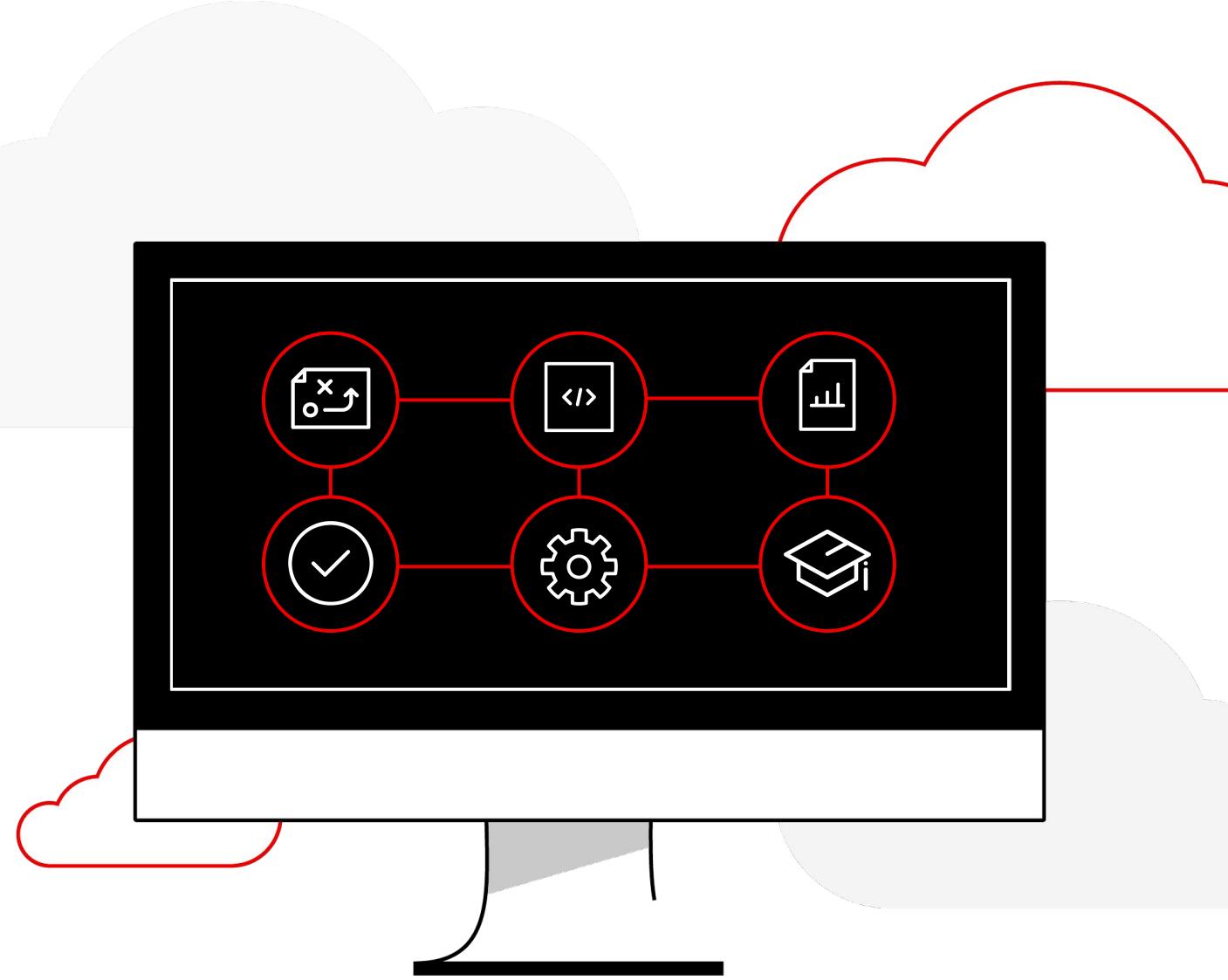
Simplified and consistent content delivery

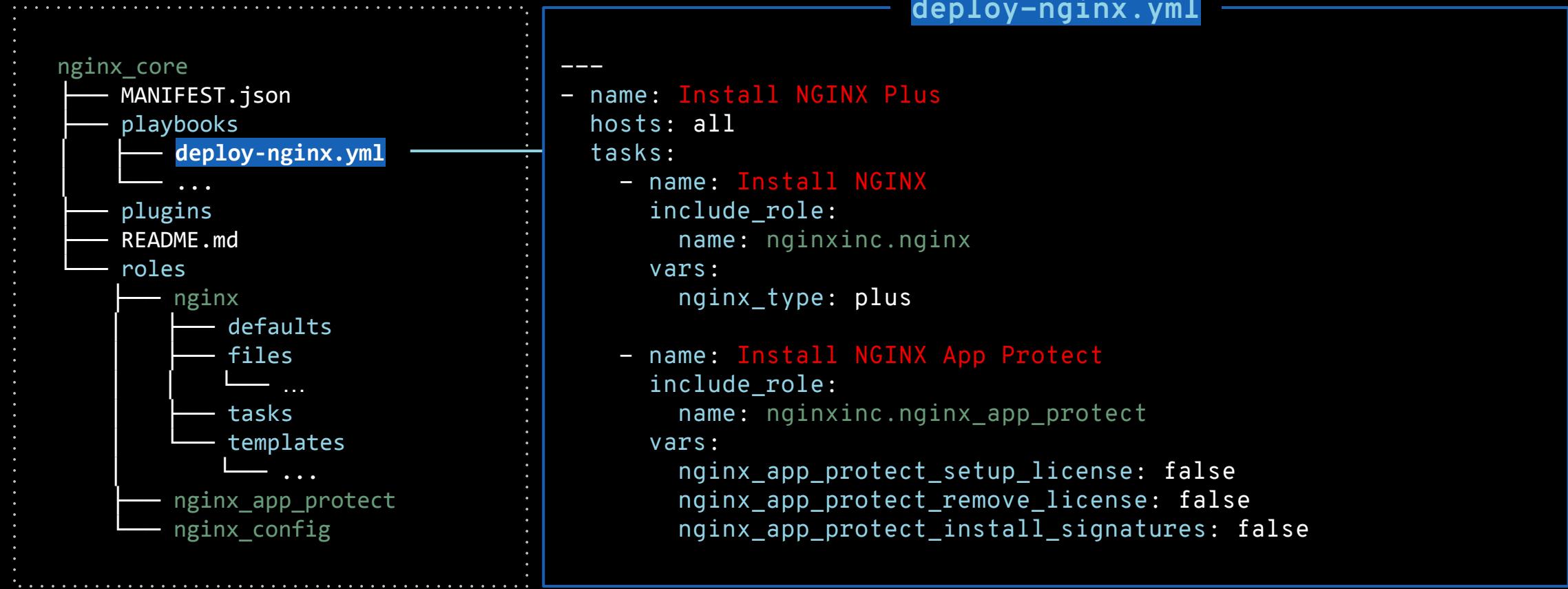


## What are they?

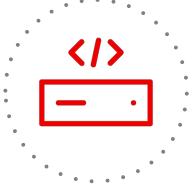
Collections are a data structure containing automation content:

- ▶ Modules
- ▶ Playbooks
- ▶ Roles
- ▶ Plugins
- ▶ Docs
- ▶ Tests





90+  
certified platforms



Infrastructure



Cloud



Network



Security



ARISTA



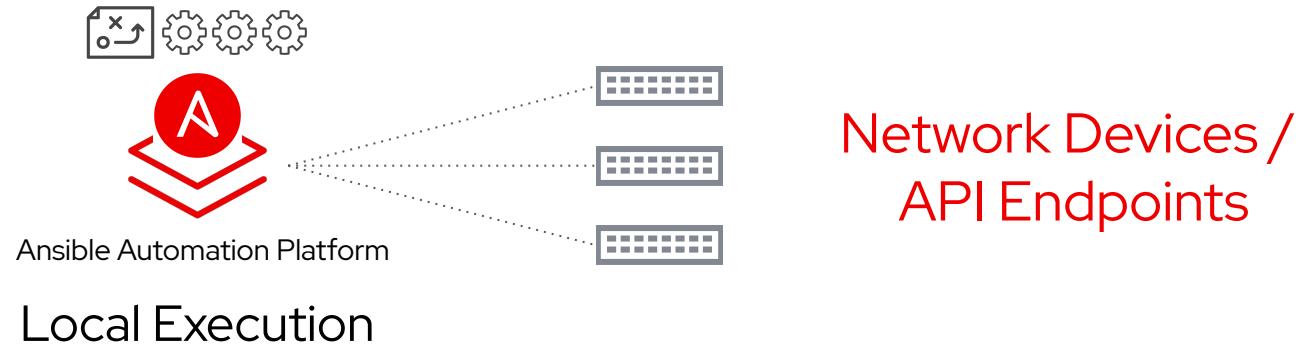
Check Point®  
SOFTWARE TECHNOLOGIES LTD



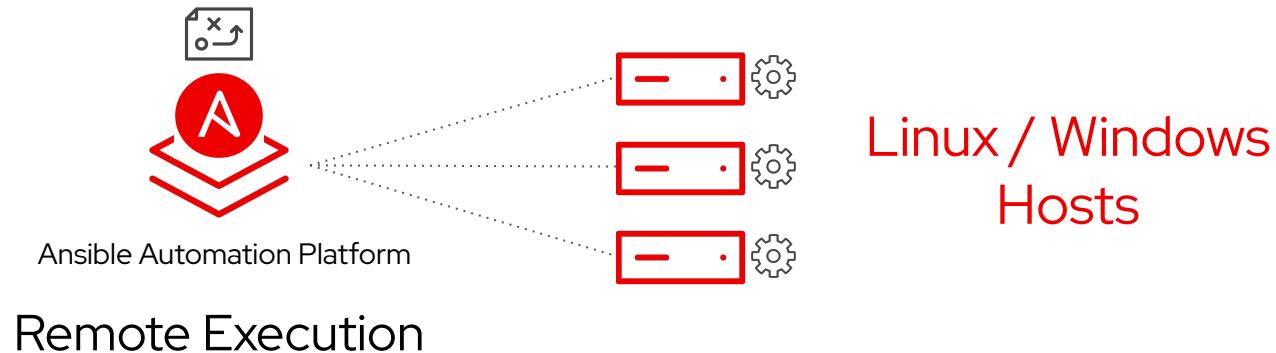
FORTINET®

# How Ansible Automation Works

Module code is  
executed locally on the  
control node



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



# Verify Lab Access

- Follow the steps in 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
- Accessing Ansible docs
- Modules and getting help

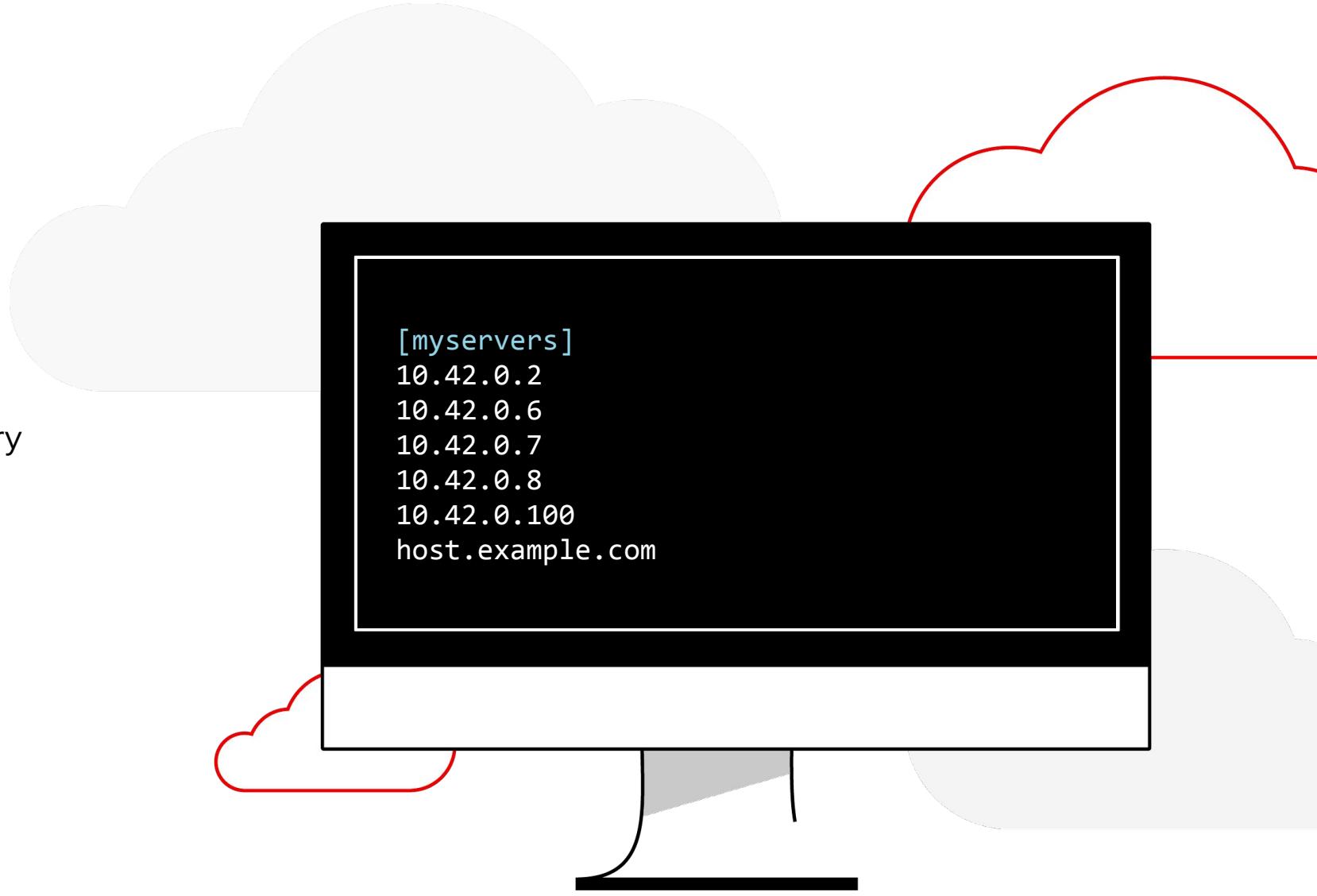
# Inventory

- ▶ 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

# Ansible Inventory

## The Basics

An example of a static Ansible inventory including systems with IP addresses as well as fully qualified domain name (FQDN)



**[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
```

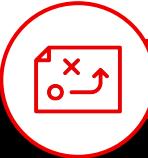


```
[app1srv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[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
```



```
[nashville]
bnaapp01
bnaapp02
```

```
[atlanta]
atlapp03
atlapp04
```

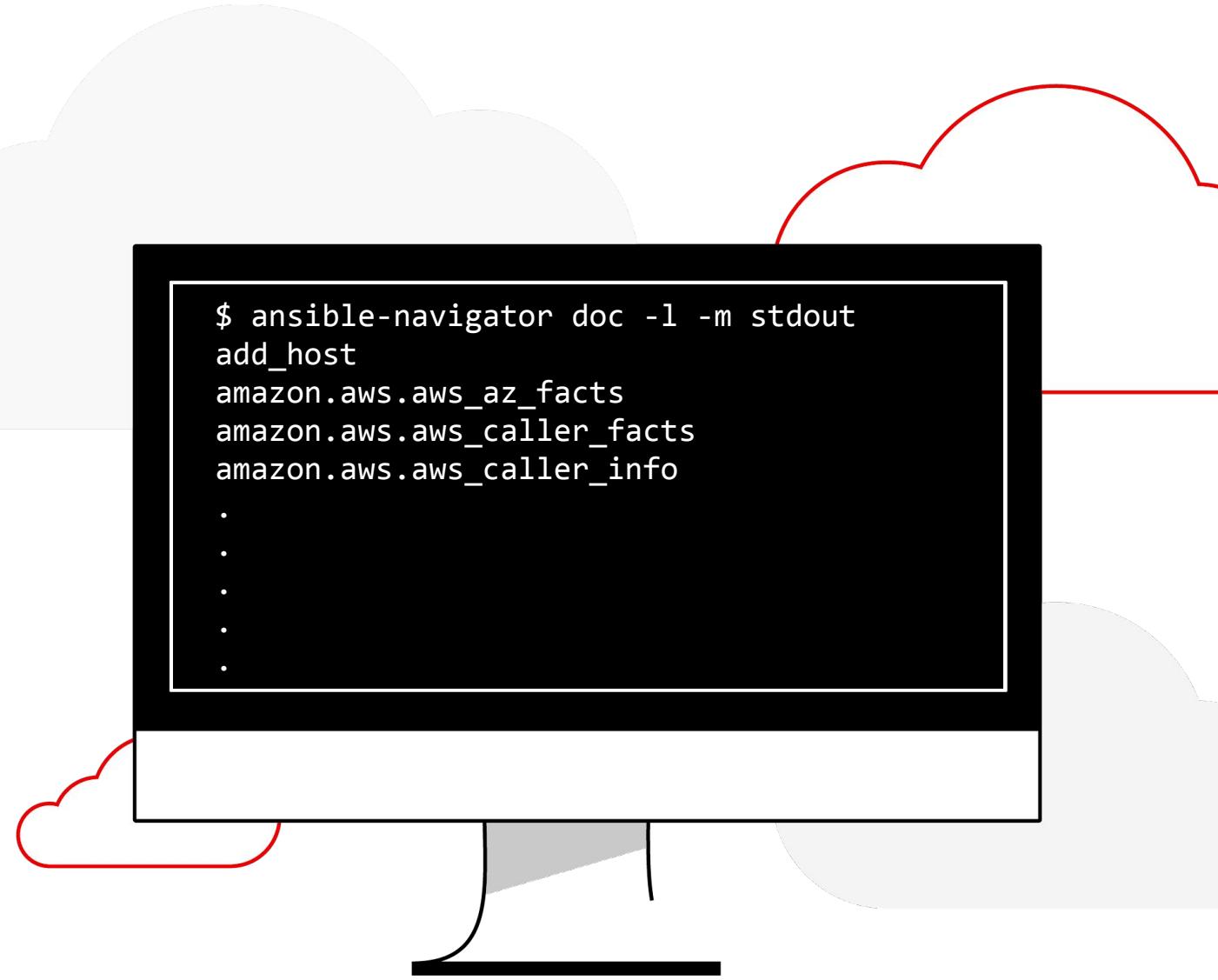
```
[south:children]
atlanta
nashville
hsvapp05
```

# Accessing the Ansible docs

With the use of the latest command utility ansible-navigator, one can trigger access to all the modules available to them as well as details on specific modules.

A formal introduction to ansible-navigator and how it can be used to run playbooks in the following exercise.

```
$ ansible-navigator doc -l -m stdout  
add_host  
amazon.aws.aws_az_facts  
amazon.aws.aws_caller_facts  
amazon.aws.aws_caller_info  
. . . . .
```



# Accessing the Ansible docs

Aside from listing a full list of all the modules, you can use ansible-navigator to provide details about a specific module.

In this example, we are getting information about the user module.

```
$ ansible-navigator doc user -m stdout  
> ANSIBLE.BUILTIN.USER  
(/usr/lib/python3.8/site-packages/ansible/m  
odules/user.py)  
  
Manage user accounts and user attributes.  
For Windows targets, use the  
[ansible.windows.win_user] module  
instead.
```



# Red Hat Ansible Automation Platform

## Lab Time

Complete exercise 1.2 now in your lab environment



Red Hat



**Red Hat**  
Ansible Automation  
Platform

# Exercise 1.3

Topics Covered:

- Playbooks basics
- Running a 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
```



# 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
```



# A 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 Playbooks

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**

# A playbook run

## Where it all starts

- ▶ A playbook is interpreted and run against one or multiple hosts - task by task. The order of the tasks defines the execution.
- ▶ In each task, the module does the actual work.



The screenshot shows a terminal window with a red header containing two white circles and a search bar with a magnifying glass icon and a 'KEY' button. The main area displays a log of a playbook execution:

```
SEARCH
-
1 Identity added: /tmp/awx_2896_5sdng5le/artifacts/2896/ssh_key_data (/tmp/awx_2896_5sdng5le/artifacts/2896/ssh_key_data)
2
3 PLAY [install and start apache] *****
4
5 TASK [Gathering Facts] *****
6 ok: [node1]
7 ok: [node3]
8 ok: [node2]
9
10 TASK [httpd package is present] *****
11 changed: [node1]
12 changed: [node2]
13 changed: [node3]
14
15 TASK [latest index.html file is present] *****
16 changed: [node1]
17 changed: [node2]
18 changed: [node3]
19
20 TASK [httpd is started] *****
21 changed: [node1]
22 changed: [node2]
23 changed: [node3]
24
25 PLAY RECAP *****
26 node1 : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
27 node2 : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
28 node3 : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
29
```

# Running an Ansible Playbook

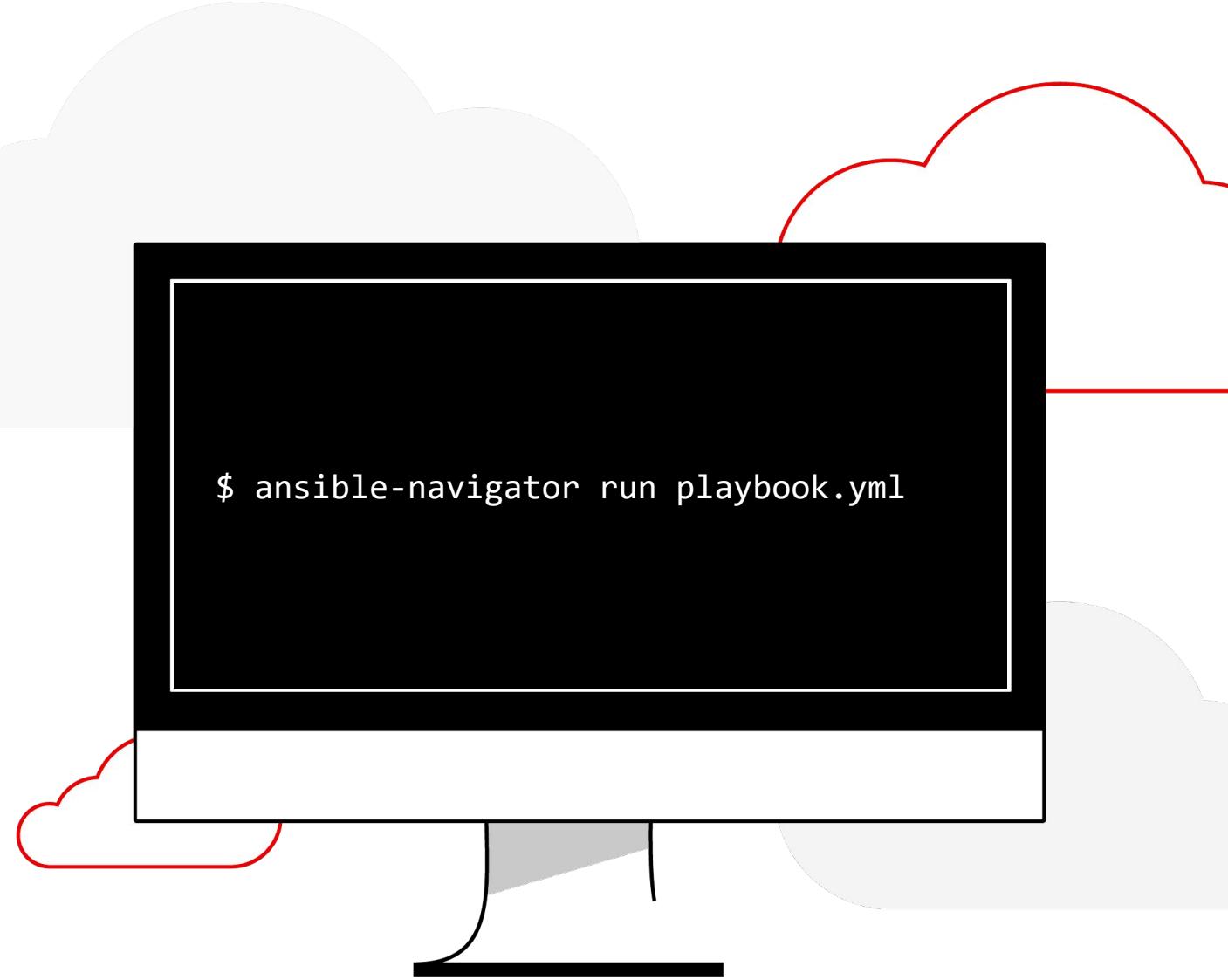
Using the latest `ansible-navigator` command



## What is `ansible-navigator`?

`ansible-navigator` command line utility and text-based user interface (TUI) for running and developing Ansible automation content.

It replaces the previous command used to run playbooks “`ansible-playbook`”.



# ansible-navigator

Bye ansible-playbook, Hello ansible-navigator



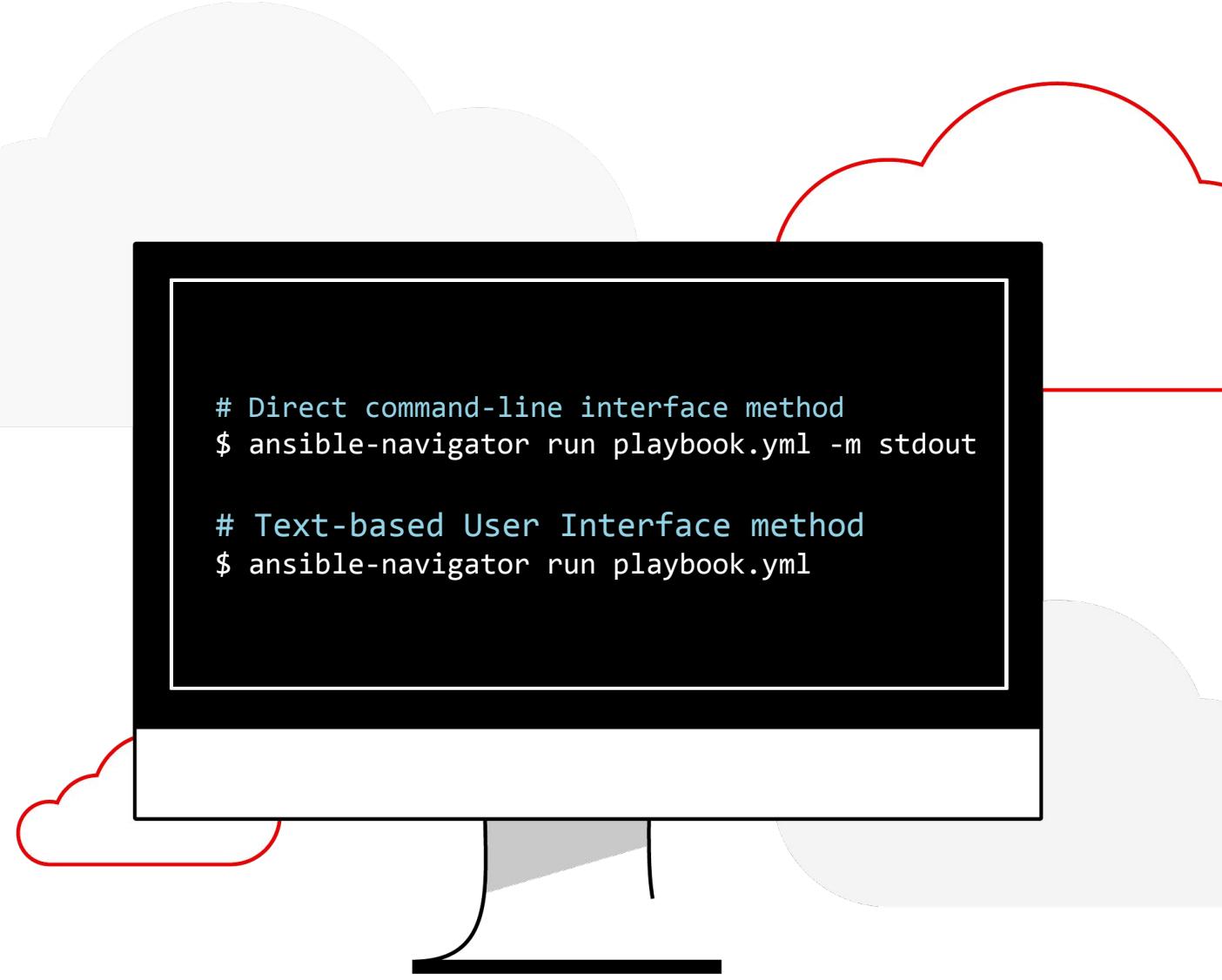
## How do I use ansible-navigator?

As previously mentioned, it replaces the ansible-playbook command.

As such it brings two methods of running playbooks:

- ▶ Direct command-line interface
- ▶ Text-based User Interface (TUI)

```
# Direct command-line interface method  
$ ansible-navigator run playbook.yml -m stdout  
  
# Text-based User Interface method  
$ ansible-navigator run playbook.yml
```



# ansible-navigator

Mapping to previous Ansible commands

<b>ansible command</b>	<b>ansible-navigator command</b>
ansible-config	ansible-navigator config
ansible-doc	ansible-navigator doc
ansible-inventory	ansible-navigator inventory
ansible-playbook	ansible-navigator run

# ansible-navigator

## Common subcommands

Name	Description	CLI Example	Colon command within TUI
collections	Explore available collections	ansible-navigator collections --help	:collections
config	Explore the current ansible configuration	ansible-navigator config --help	:config
doc	Review documentation for a module or plugin	ansible-navigator doc --help	:doc
images	Explore execution environment images	ansible-navigator images --help	:images
inventory	Explore and inventory	ansible-navigator inventory --help	:inventory
replay	Explore a previous run using a playbook artifact	ansible-navigator replay --help	:replay
run	Run a playbook	ansible-navigator run --help	:run
welcome	Start at the welcome page	ansible-navigator welcome --help	:welcome





# 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?



```
---
- 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 }}"
```



```
---
- 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

# Ansible Facts

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





```
---  
- name: facts playbook  
  hosts: localhost  
  
  tasks:  
    - name: Collect all facts of host  
      setup:  
        gather_subset:  
          - 'all'
```

```
$ ansible-navigator run playbook.yml
```



PLAY NAME	OK	CHANGED	UNREACHABLE	FAILED	SKIPPED	IGNORED	IN PROGRESS	TASK COUNT	PROGRESS COMPLETE
0  facts playbook	2	0	0	0	0	0	0	2	COMPLETE



RESULT	HOST	NUMBER	CHANGED	TASK	TASK ACTION	DURATION
0  OK	localhost	0	False	Gathering Facts	gather_facts	1s
1  OK	localhost	1	False	Collect all facts of host	setup	1s



PLAY [facts playbook:1]

\*\*\*\*\*

TASK [Collect all facts of host]

\*\*\*\*\*

OK: [localhost]

.

.

```

12 |   ansible_facts:
13 |   ansible_all_ipv4_addresses:
14 |     - 10.0.2.100
15 |   ansible_all_ipv6_addresses:
16 |     - fe80::1caa:f0ff:fe15:23c4

```



# 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

# Conditionals via VARS

Example of using a variable labeled *my\_mood* and using it as a conditional on a particular task.

```
vars:  
  my_mood: happy  
  
tasks:  
  - name: task, based on my_mood var  
    debug:  
      msg: "Yay! I am {{ my_mood }}!"  
    when: my_mood == "happy"
```



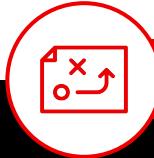
```
---  
- name: variable playbook test  
  hosts: localhost  
  
  vars:  
    my_mood: happy  
  
  tasks:  
    - name: task, based on my_mood var  
      debug:  
        msg: "Yay! I am {{ my_mood }}!"  
      when: my_mood == "happy"
```

## Alternatively

```
- name: task, based on my_mood var  
  debug:  
    msg: "Ask at your own risk. I'm {{ my_mood }}!"  
  when: my_mood == "grumpy"
```



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



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

  tasks:
    - 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
```



```
---  
- name: variable playbook test  
  hosts: localhost  
  
  tasks:  
    - name: Ensure httpd package is present  
      yum:  
        name: httpd  
        state: latest  
      notify: restart_httpd  
  
  handlers:  
    - name: restart_httpd  
      service:  
        name: httpd  
        state: restart
```



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

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

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

```
ok: [web2] unchanged  
ok: [web1]
```

```
TASK [Standardized index.html file] ****
```

```
changed: [web2] changed  
changed: [web1]
```

```
NOTIFIED: [restart_httpd] ***
```

```
changed: [web2]  
changed: [web1]
```

**handler runs 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] ****  
changed: [web2] changed  
changed: [web1]
```

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

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

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

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

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



```
---  
- 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
```



```
---  
- name: Ensure users  
  hosts: node1  
  become: yes  
  
  tasks:  
    - name: Ensure user is 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



**Red Hat**  
Ansible Automation  
Platform

# Exercise 1.6

Topics Covered:

- Templates



```
- 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
```



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

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

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

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

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



# Red Hat Ansible Automation Platform

## Lab Time

Complete exercise 1.6 now in your lab environment



Red Hat



**Red Hat**  
Ansible Automation  
Platform

# Exercise 1.7

Topics Covered:

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

# Role Structure

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

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



# Ansible Galaxy

Sharing  
Content

Community

Roles, and  
more

## v1 - Set config file to use on boot:

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

## v1 - Set config file to use on boot:

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



# **Red Hat** Ansible Automation Platform

## Lab Time

Complete exercise 1.7 now in your lab environment



**Red Hat**



**Red Hat**  
Ansible Automation  
Platform

# Exercise 1.8

Topics Covered:

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



# Red Hat Ansible Automation Platform

## Lab Time

Complete exercise 1.8 now in your lab environment



Red Hat

# Section 2

# Automation Controller



**Red Hat**  
Ansible Automation  
Platform



**Red Hat**  
Ansible Automation  
Platform

# Exercise 2.1

Topics Covered:

- Introduction to Automation Controller



## Red Hat Ansible Automation Platform



Content creators



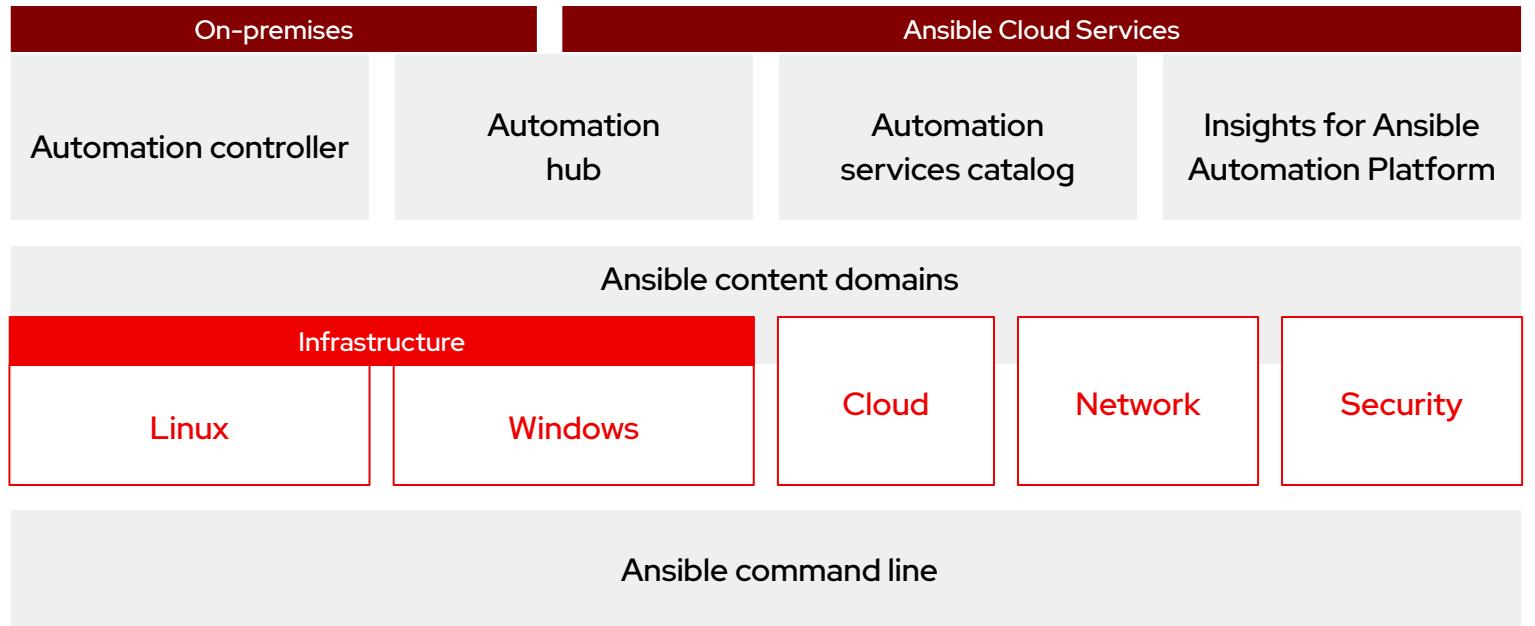
Operators



Domain experts



Users

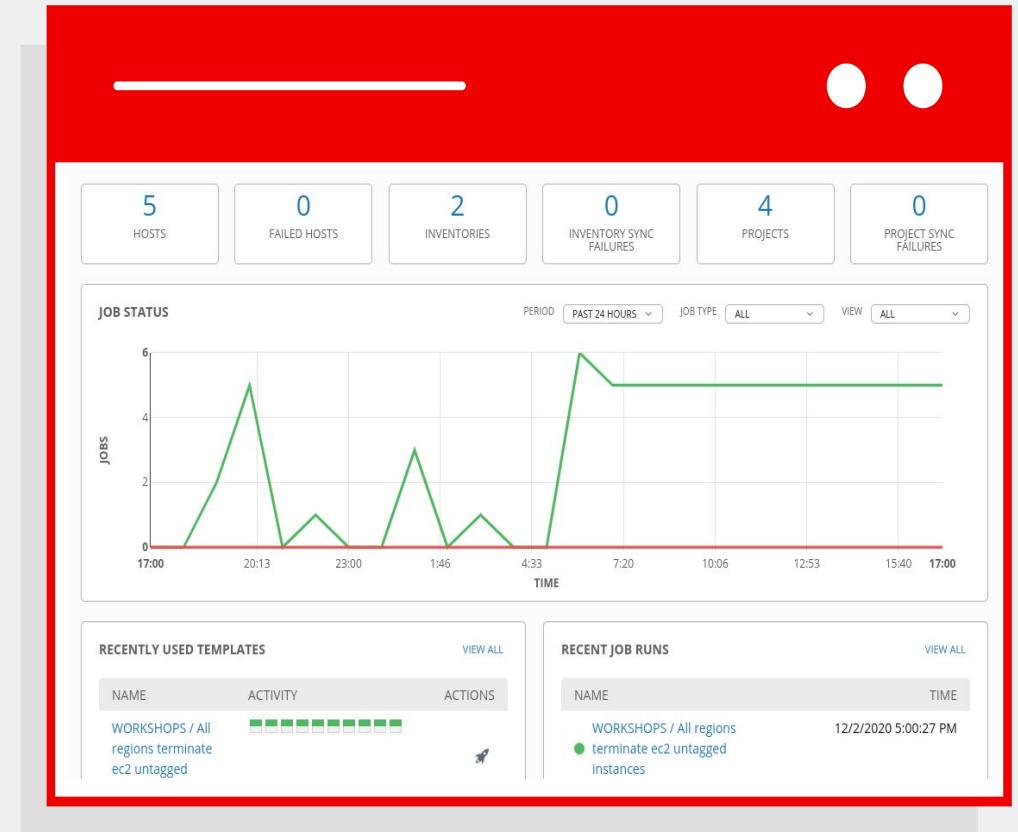


Fueled by an  
open source community

# What is Ansible Automation Controller?

Ansible Automation Controller 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



# Ansible Automation Controller

## 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.

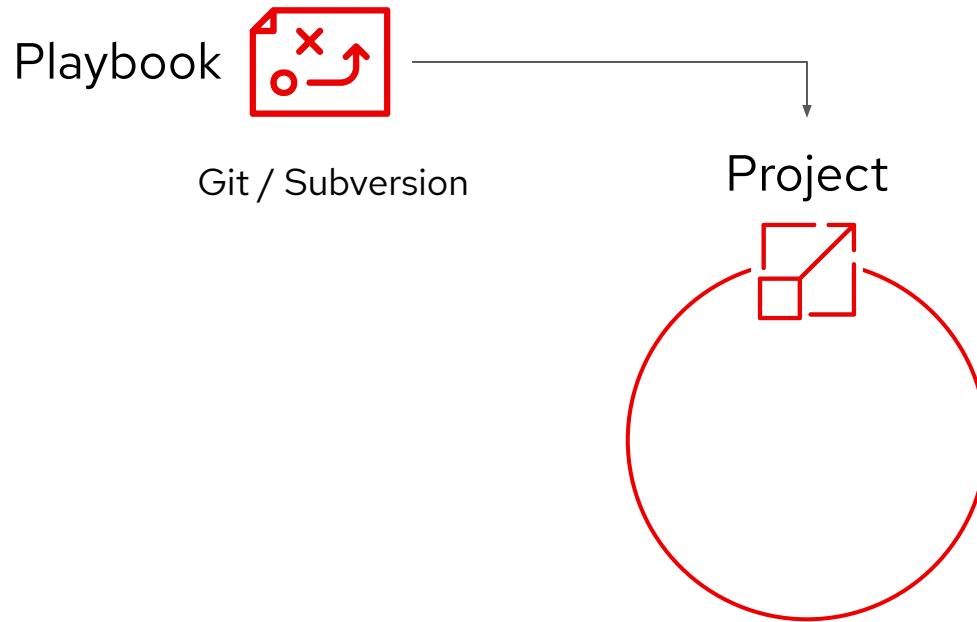
## Centralized logging

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.

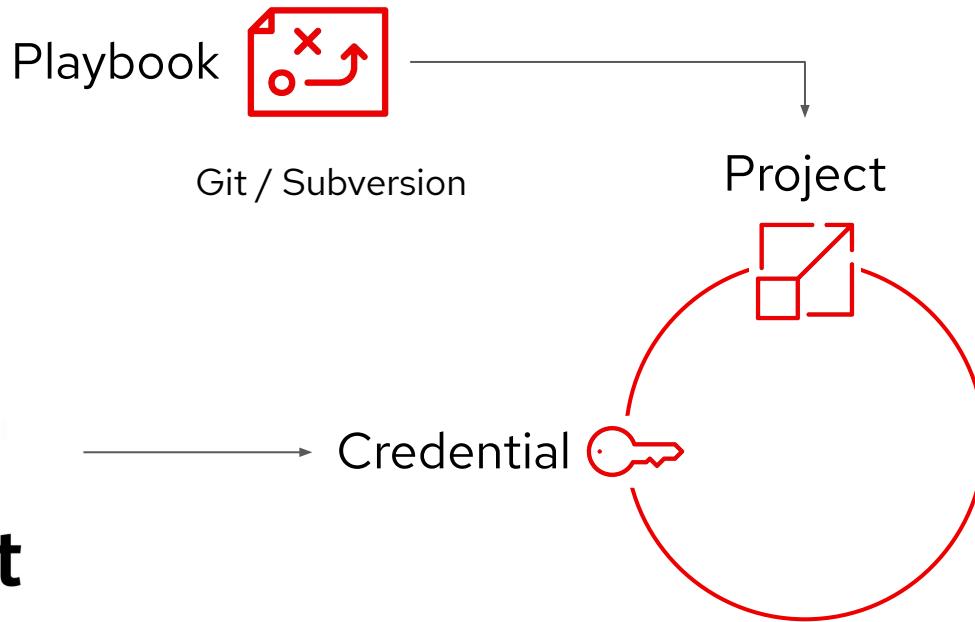
## Workflows

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.

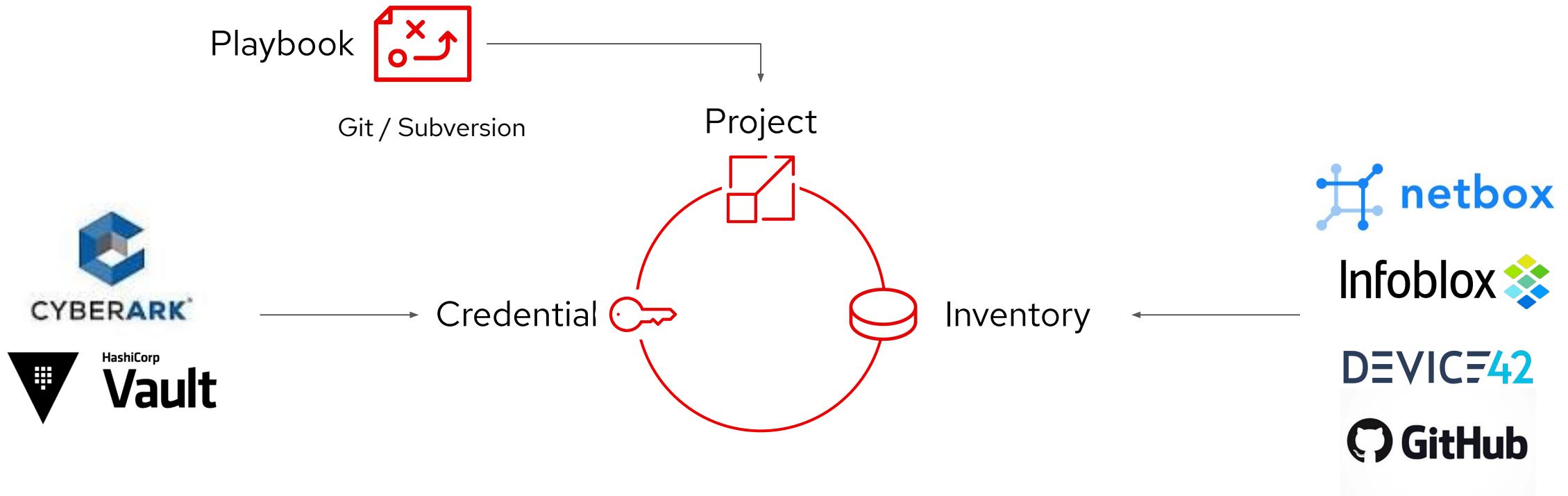
# Anatomy of an Automation Job



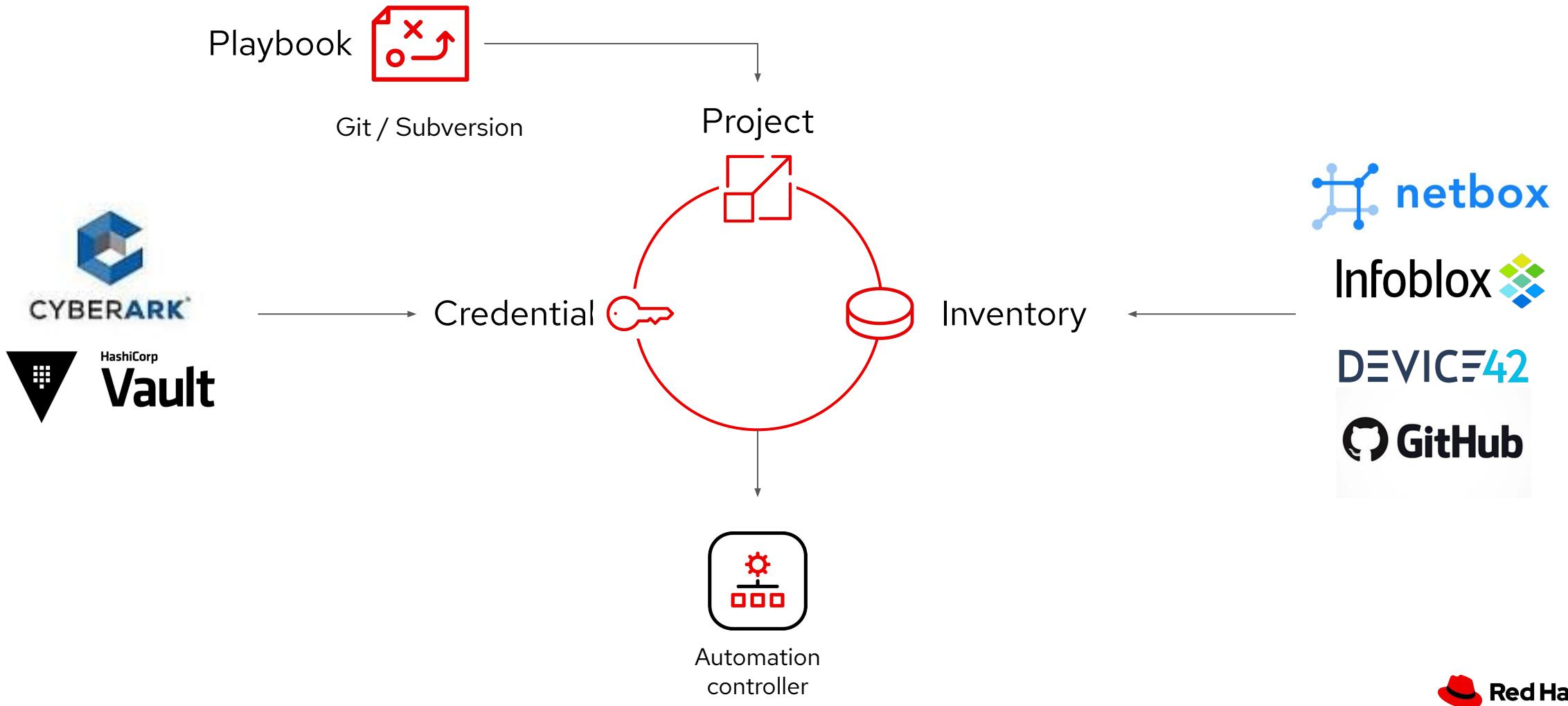
# Anatomy of an Automation Job



# Anatomy of an Automation Job



# Anatomy of an Automation Job





# Red Hat

## Ansible Automation Platform

### Lab Time

Complete exercise 2.1 now in your lab environment



Red Hat



**Red Hat**  
Ansible Automation  
Platform

# Exercise 2.2

Topics Covered:

- Inventories
- Credentials

# Inventory

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

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

The screenshot shows the 'Hosts' tab of an inventory named 'Workshop Inventory'. The interface includes a search bar, an 'Add' button, and a 'Run Command' button. Below the search bar, there is a table listing four hosts: 'ansible-1', 'node1', 'node2', and 'node3'. Each host entry has a checkbox, a name link, and an 'Actions' column with a switch and edit icon. The bottom of the screen shows navigation controls and a page number indicator.

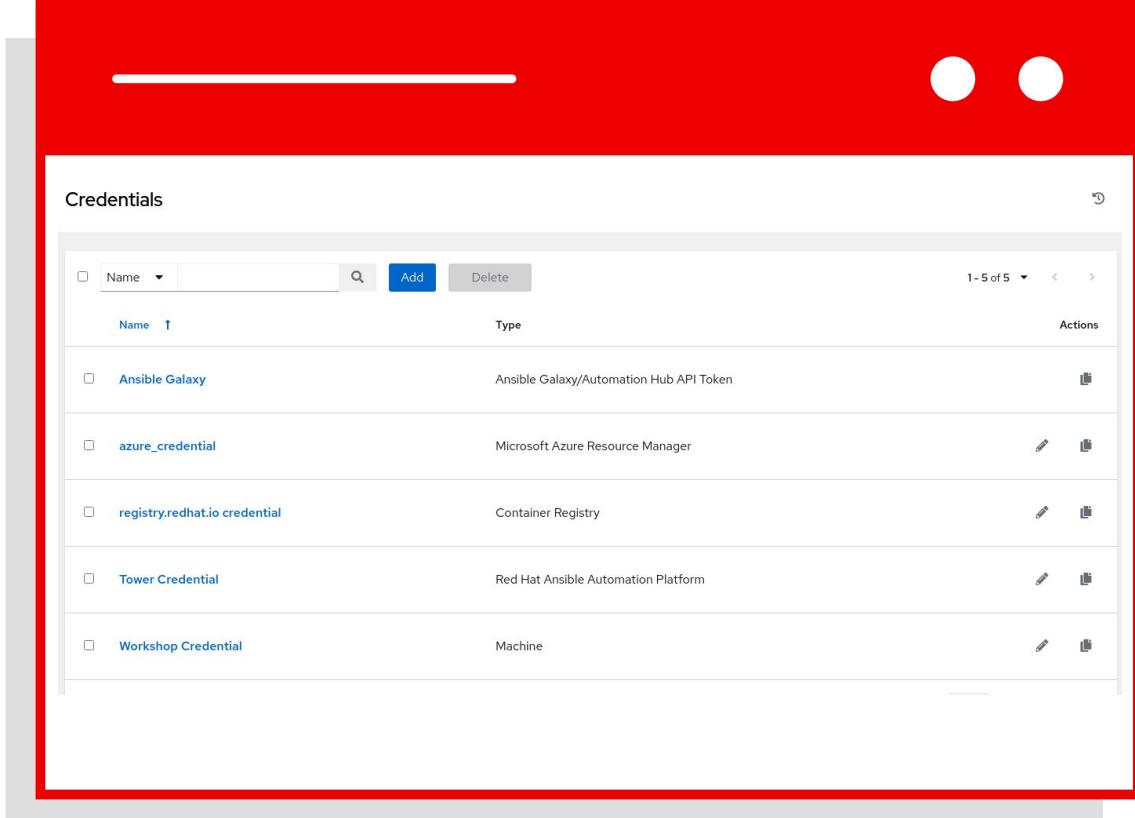
Name	Actions
ansible-1	On
node1	On
node2	On
node3	On

# Credentials

Credentials are utilized by Automation Controller 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.



The screenshot shows a table titled "Credentials" with a red border. The table has columns for "Name", "Type", and "Actions". There are five rows of data:

Name	Type	Actions
Ansbile Galaxy	Ansible Galaxy/Automation Hub API Token	 
azure_credential	Microsoft Azure Resource Manager	 
registry.redhat.io credential	Container Registry	 
Tower Credential	Red Hat Ansible Automation Platform	 
Workshop Credential	Machine	 



# **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

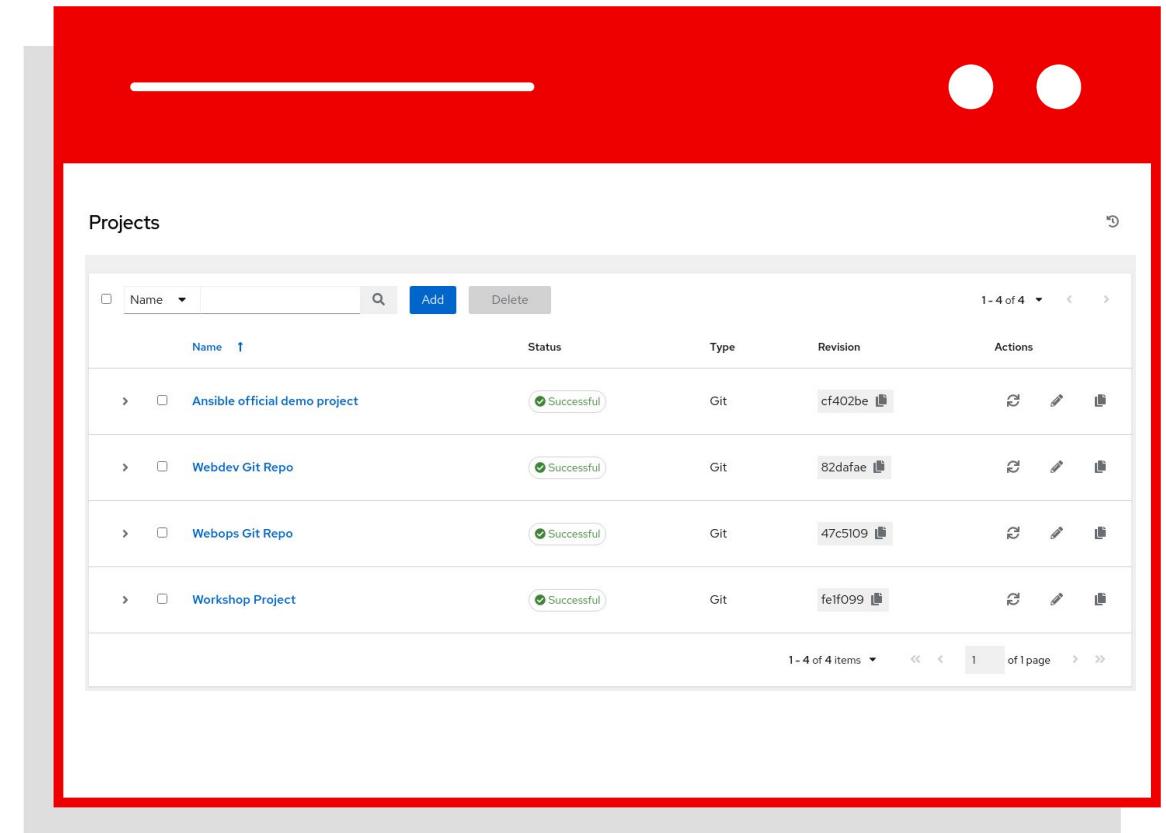
Topics Covered:

- Projects
- Job Templates

# Project

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

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.



The screenshot shows the 'Projects' page in Ansible Tower. The interface has a red header bar with two white circles on the right. Below the header is a search bar with 'Name' and a dropdown, followed by 'Add' and 'Delete' buttons. The main area is titled 'Projects' and contains a table with the following data:

Name	Status	Type	Revision	Actions
Ansible official demo project	Successful	Git	cf402be	 
Webdev Git Repo	Successful	Git	82dafaef	 
Webops Git Repo	Successful	Git	47c5109	 
Workshop Project	Successful	Git	fef099	 

At the bottom, there is a footer with '1 - 4 of 4 items' and navigation arrows. The entire screenshot is framed by a thick red border.

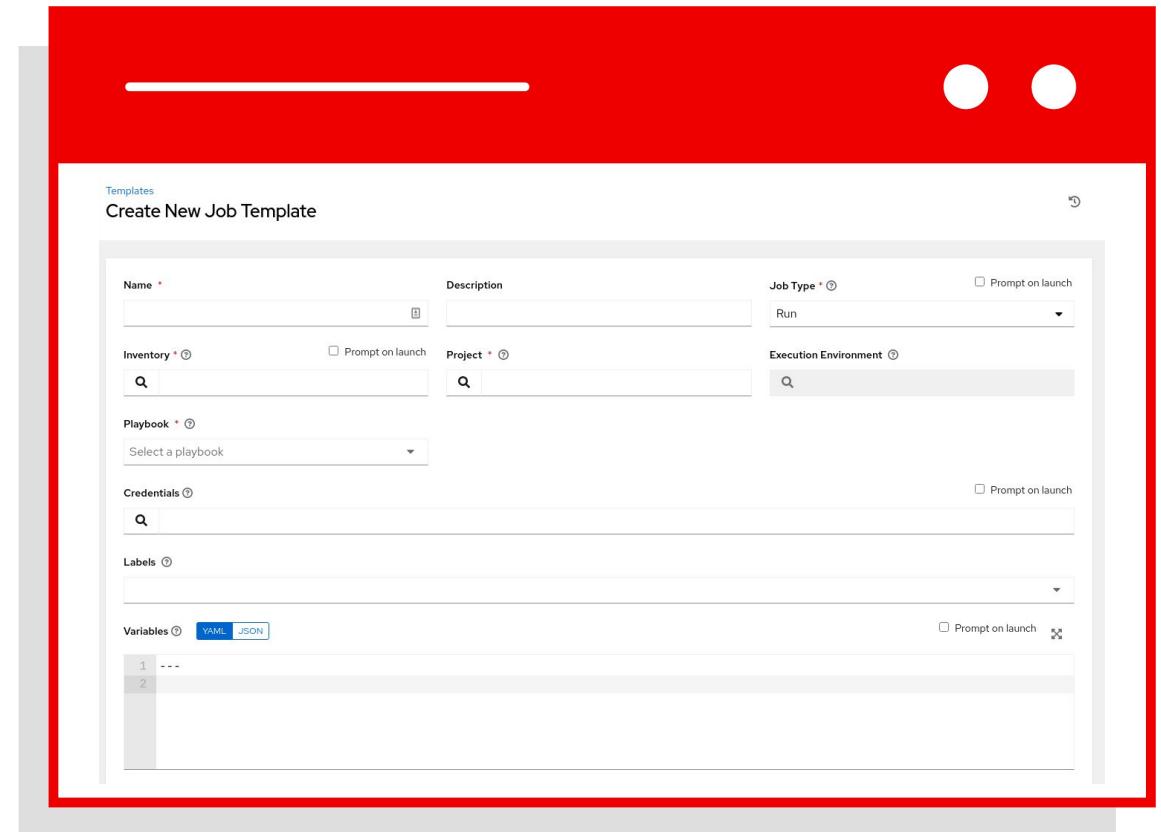
# Job Templates

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

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

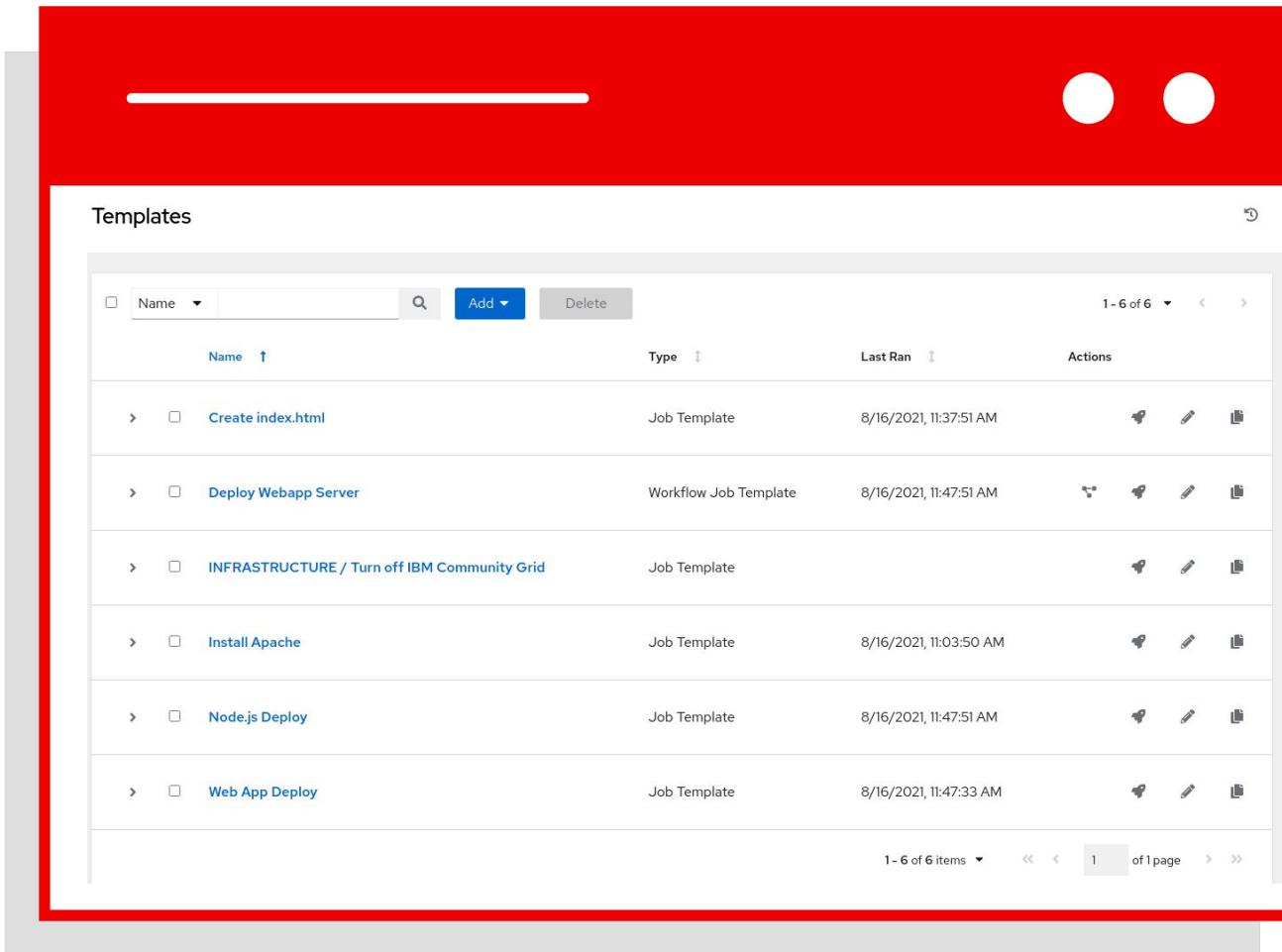
A **Job Template** requires:

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



# Expanding on Job Templates

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



The screenshot shows a list of job templates in a Red Hat interface. A red box highlights the central content area where the table is displayed. The table has columns for Name, Type, Last Ran, and Actions. The data is as follows:

Name	Type	Last Ran	Actions
Create index.html	Job Template	8/16/2021, 11:37:51 AM	Edit, Delete, Preview
Deploy Webapp Server	Workflow Job Template	8/16/2021, 11:47:51 AM	Edit, Delete, Preview
INFRASTRUCTURE / Turn off IBM Community Grid	Job Template		Edit, Delete, Preview
Install Apache	Job Template	8/16/2021, 11:03:50 AM	Edit, Delete, Preview
Node.js Deploy	Job Template	8/16/2021, 11:47:51 AM	Edit, Delete, Preview
Web App Deploy	Job Template	8/16/2021, 11:47:33 AM	Edit, Delete, Preview

# Executing an existing Job Template

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



The screenshot shows a web-based interface for managing job templates. At the top, there's a search bar labeled 'Name' with a dropdown arrow, a magnifying glass icon, and a blue 'Add' button with a downward arrow. To the right of the search bar are two circular icons. Below the header is a table titled 'Templates'. The table has columns for 'Name' (sorted by ascending name), 'Type', 'Last Ran', and 'Actions'. There are six rows in the table, each representing a different job template. The 'Actions' column for each row contains three icons: a gear, a pencil, and a clipboard. A large red rectangular box surrounds the entire table area. A smaller red box highlights the 'Actions' column for the first row.

Name	Type	Last Ran	Actions
Create index.html	Job Template	8/16/2021, 11:37:51 AM	
Deploy Webapp Server	Workflow Job Template	8/16/2021, 11:47:51 AM	
INFRASTRUCTURE / Turn off IBM Community Grid	Job Template		
Install Apache	Job Template	8/16/2021, 11:03:50 AM	
Node.js Deploy	Job Template	8/16/2021, 11:47:51 AM	
Web App Deploy	Job Template	8/16/2021, 11:47:33 AM	

# Creating a new Job Template (1/2)

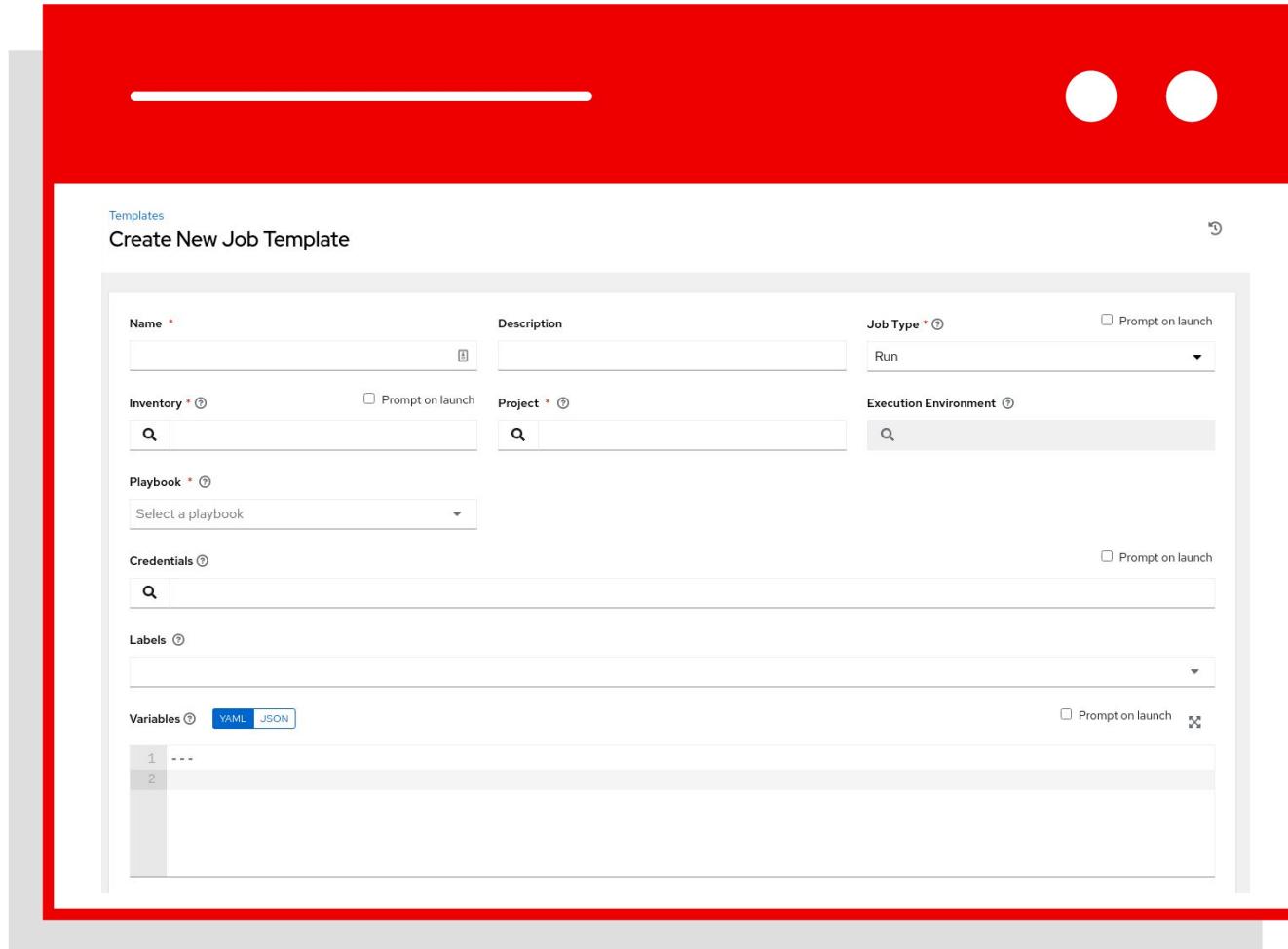
New Job Templates can be created by clicking the **Add button**

The screenshot shows a software interface with a red header bar. In the top left of the header bar, there is a search icon and a blue 'Add' button with a downward arrow, which is highlighted with a red box. In the top right of the header bar, there are two white circles. Below the header is a table titled 'Templates'. The table has columns for 'Name', 'Type', 'Last Ran', and 'Actions'. There are six rows in the table, each representing a template. The first row is highlighted with a red box and contains the text 'Create index.html', 'Job Template', '8/16/2021, 11:37:51 AM', and three icons for edit, delete, and details. The other five rows show similar information for 'Deploy Webapp Server', 'INFRASTRUCTURE / Turn off IBM Community Grid', 'Install Apache', 'Node.js Deploy', and 'Web App Deploy', all categorized as 'Job Template'.

Name	Type	Last Ran	Actions
Create index.html	Job Template	8/16/2021, 11:37:51 AM	
Deploy Webapp Server	Workflow Job Template	8/16/2021, 11:47:51 AM	
INFRASTRUCTURE / Turn off IBM Community Grid	Job Template		
Install Apache	Job Template	8/16/2021, 11:03:50 AM	
Node.js Deploy	Job Template	8/16/2021, 11:47:51 AM	
Web App Deploy	Job Template	8/16/2021, 11:47:33 AM	

# Creating a new Job Template (2/2)

This **New Job Template** window is where the inventory, project and credential are assigned. The red asterisk **\*** means the field is required.



The screenshot shows the 'Create New Job Template' dialog box. The form fields are as follows:

- Name \***: A text input field.
- Description**: A text input field.
- Job Type \***: A dropdown menu set to "Run".
- Prompt on launch**: A checkbox.
- Inventory \***: A search input field.
- Prompt on launch**: A checkbox.
- Project \***: A search input field.
- Execution Environment**: A search input field.
- Playbook \***: A dropdown menu set to "Select a playbook".
- Credentials**: A search input field.
- Labels**: A search input field.
- Variables**: A section with tabs for "YAML" (selected) and "JSON". It contains two numbered entries: "1 ---" and "2".
- Prompt on launch**: A checkbox.



# Red Hat Ansible Automation Platform

## Lab Time

Complete exercise 2.3 now in your lab environment



Red Hat



**Red Hat**  
Ansible Automation  
Platform

# Exercise 2.4

Topics Covered:

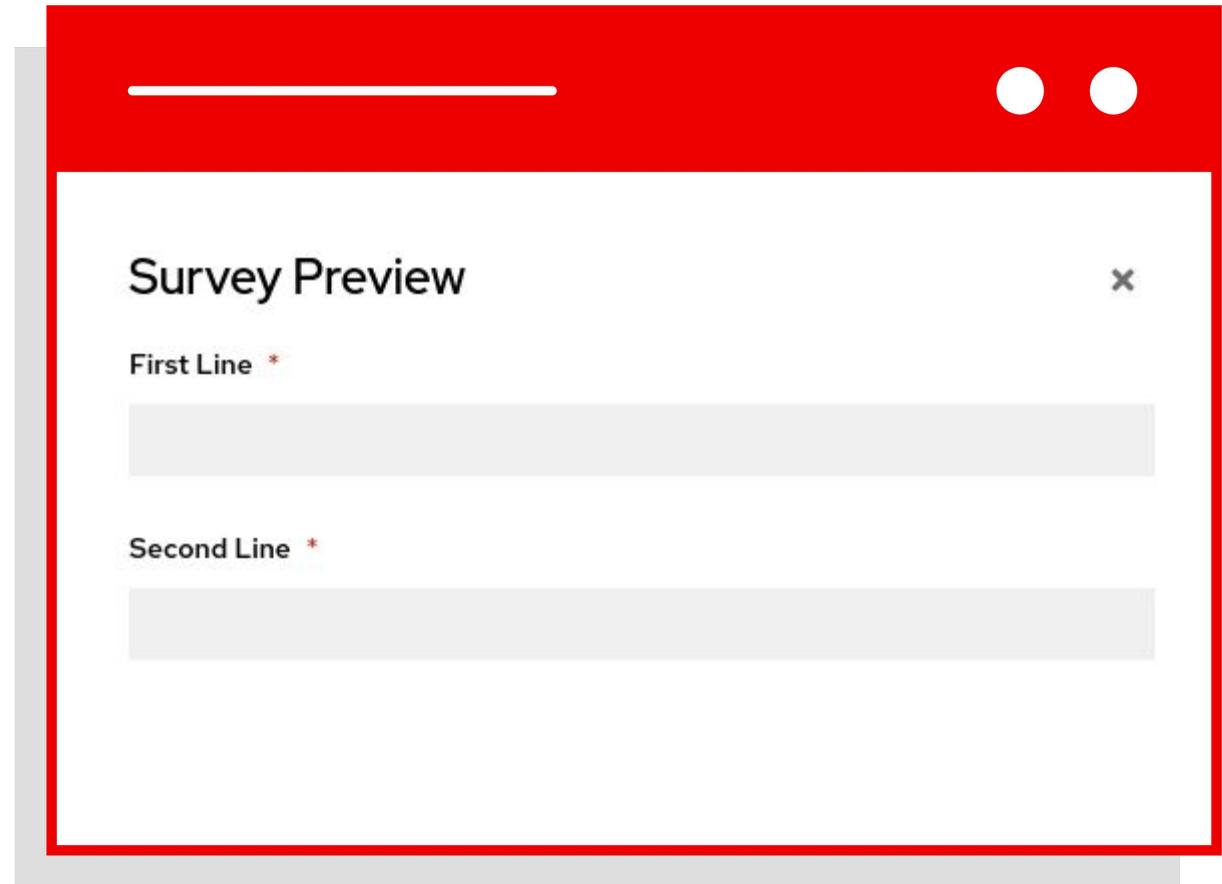
- Surveys

# Surveys

Controller 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 Controller survey is a simple question-and-answer form that allows users to customize their job runs.

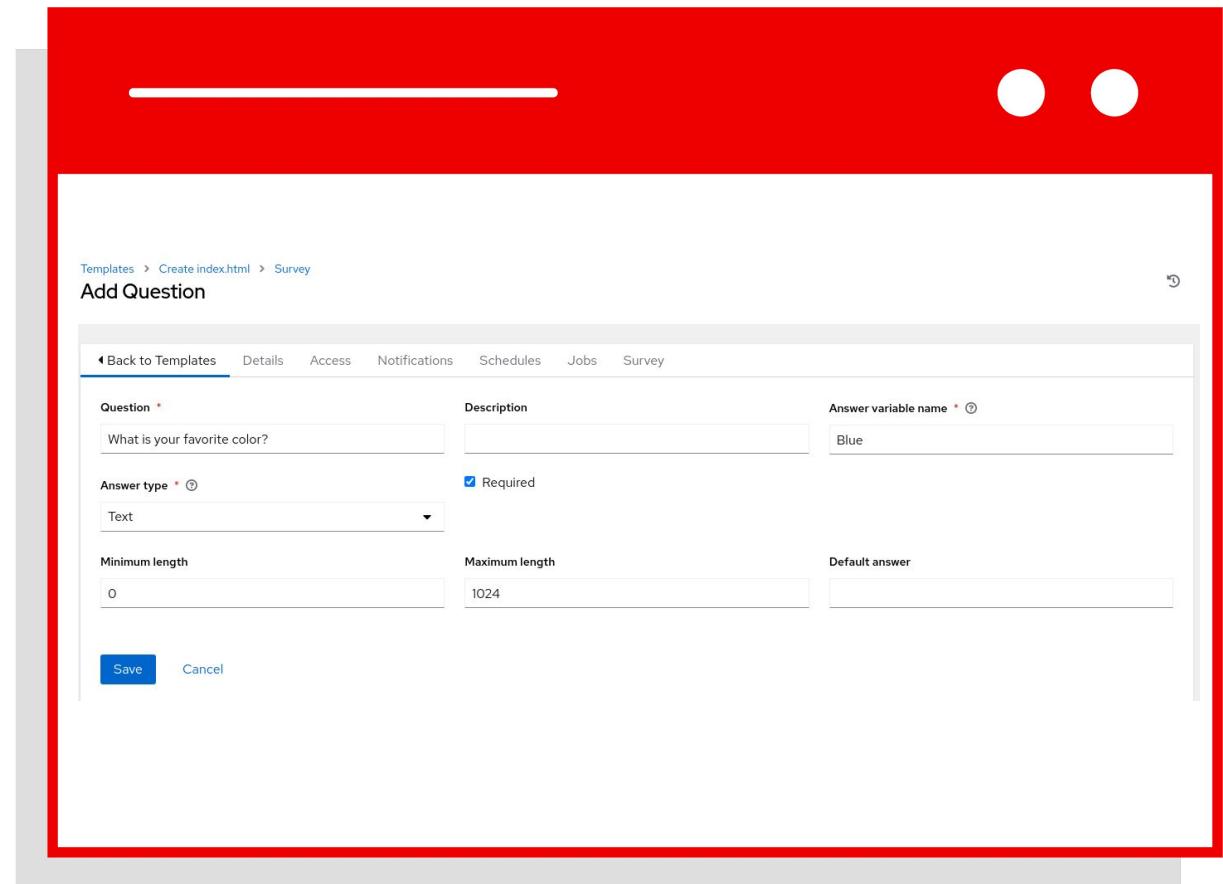
Combine that with Controller's role-based access control, and you can build simple, easy self-service for your users.



# Creating a Survey (1/2)

Once a Job Template is saved, the Survey menu will have an **Add** **Button**

Click the button to open the Add Survey window.



# Creating a Survey (2/2)

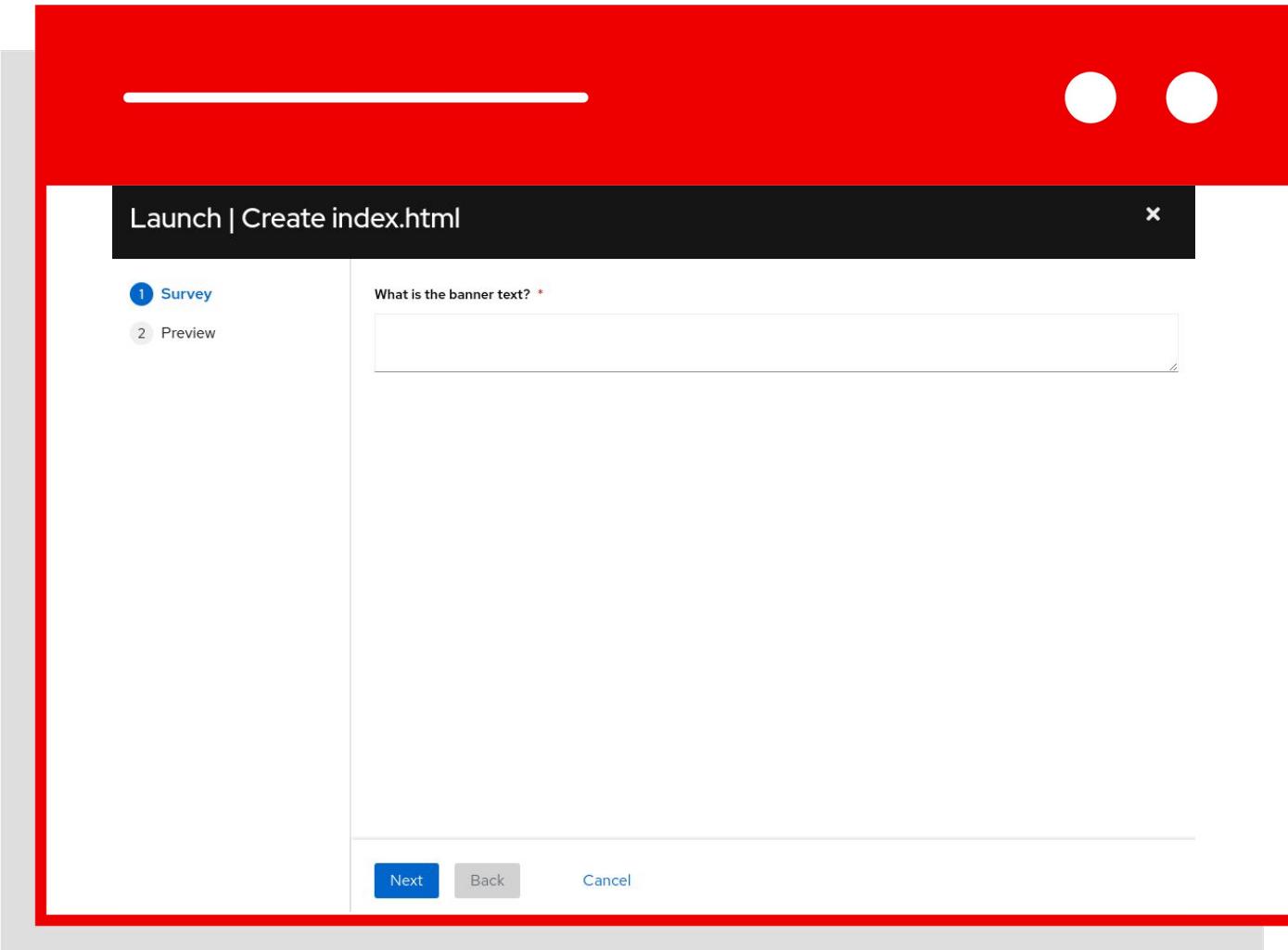
The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.

The screenshot shows the 'Add Question' dialog box within a 'Survey' template. The 'Survey' tab is selected in the top navigation bar. The dialog contains fields for 'Question' (What is the banner text?), 'Description' (empty), 'Answer variable name' (net\_banner), 'Answer type' (Textarea, required), 'Minimum length' (0), 'Maximum length' (1024), and 'Default answer' (empty). A 'Save' button is at the bottom.

The screenshot shows the 'Survey' configuration page for the 'Create index.html' template. The 'Survey' tab is selected. It lists a single survey question: 'What is the banner text?' with a 'Type' of 'textarea'. The 'On' toggle switch is turned 'On'. A 'Preview' button is visible at the bottom.

# 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 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

# Role-based access control

## How to manage access

- ▶ Role-based access control system:

Users can be grouped in teams, and roles can be assigned to the teams.

- ▶ Rights to edit or use can be assigned across all objects.

- ▶ All backed by enterprise authentication if needed.

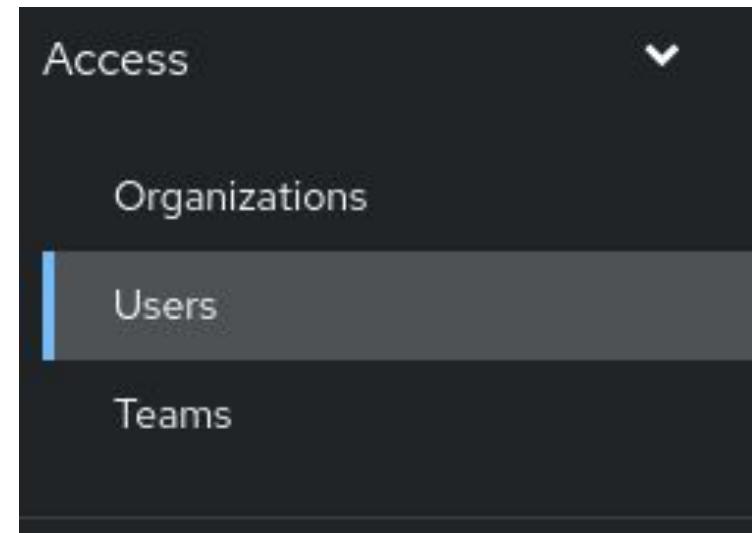
The screenshot shows a user management interface titled "Users". The table has columns: Username, First Name, Last Name, and Role. There are three items listed:

Username	First Name	Last Name	Role
admin			System Administrator
student1			System Administrator
wweb	Werner	Web	Normal User

At the bottom, it says "1 - 3 of 3 items" and "1 of 1 page".

# 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 Automation Controller 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 Ansible Automation Platform

## Lab Time

Complete exercise 2.5 now in your lab environment



Red Hat



**Red Hat**  
Ansible Automation  
Platform

# Exercise 2.6

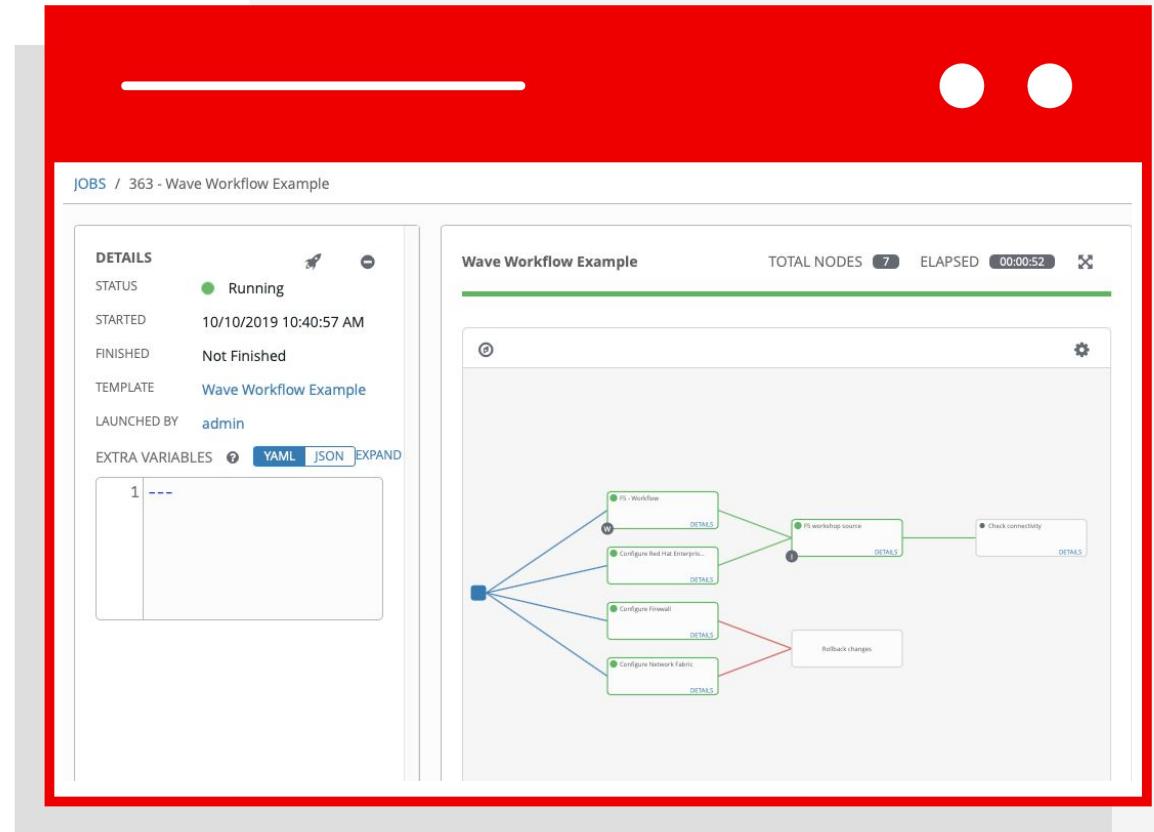
Topics Covered:

- Workflows

# Workflows

Combine automation to create something bigger

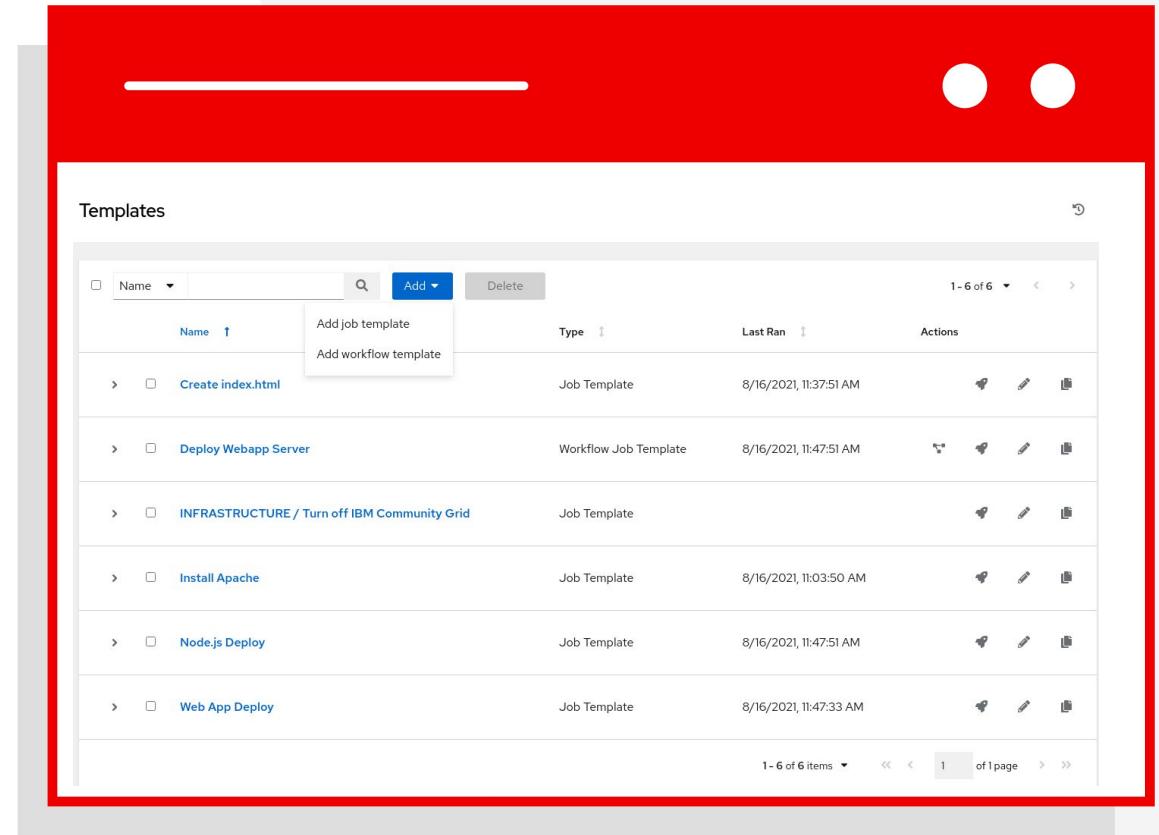
- ▶ Workflows enable the creation of powerful holistic automation, chaining together multiple pieces of automation and events.
- ▶ Simple logic inside these workflows can trigger automation depending on the success or failure of previous steps.



# Adding a New Template

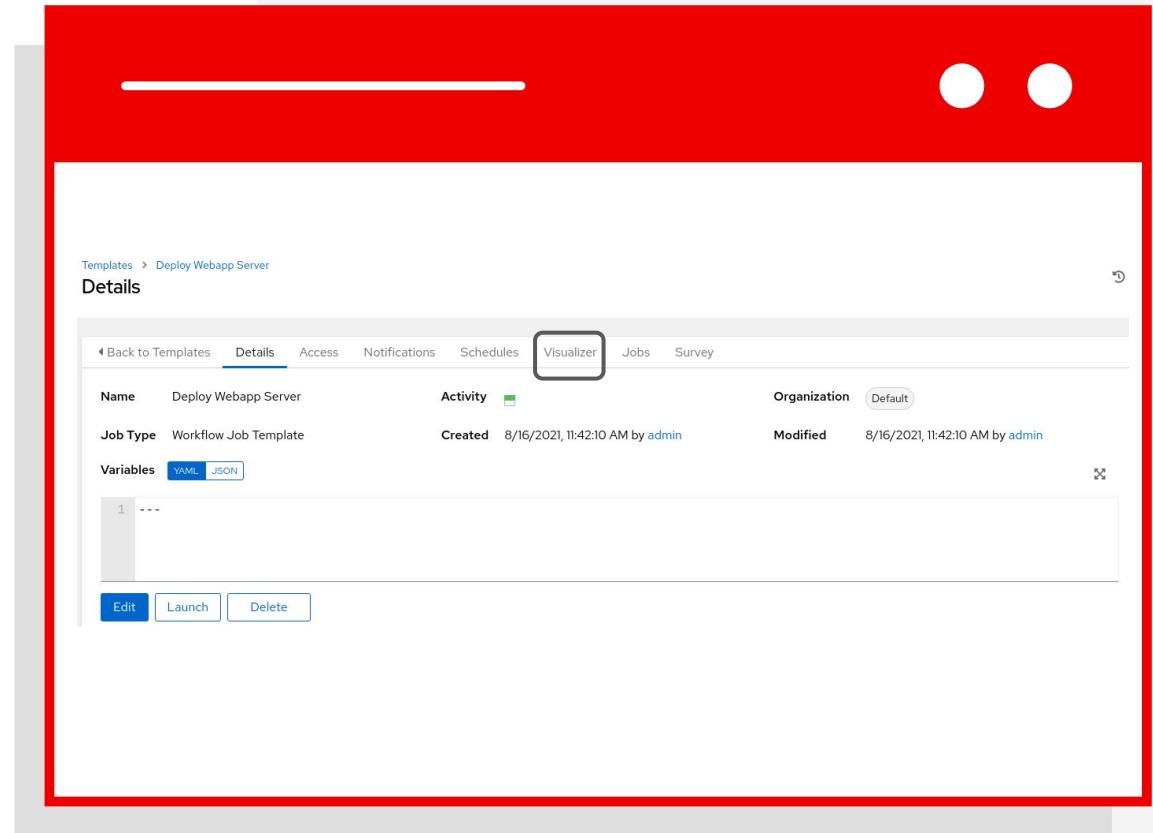
- ▶ To add a new **Workflow** click on the **Add** button.

This time select the **Add workflow template**



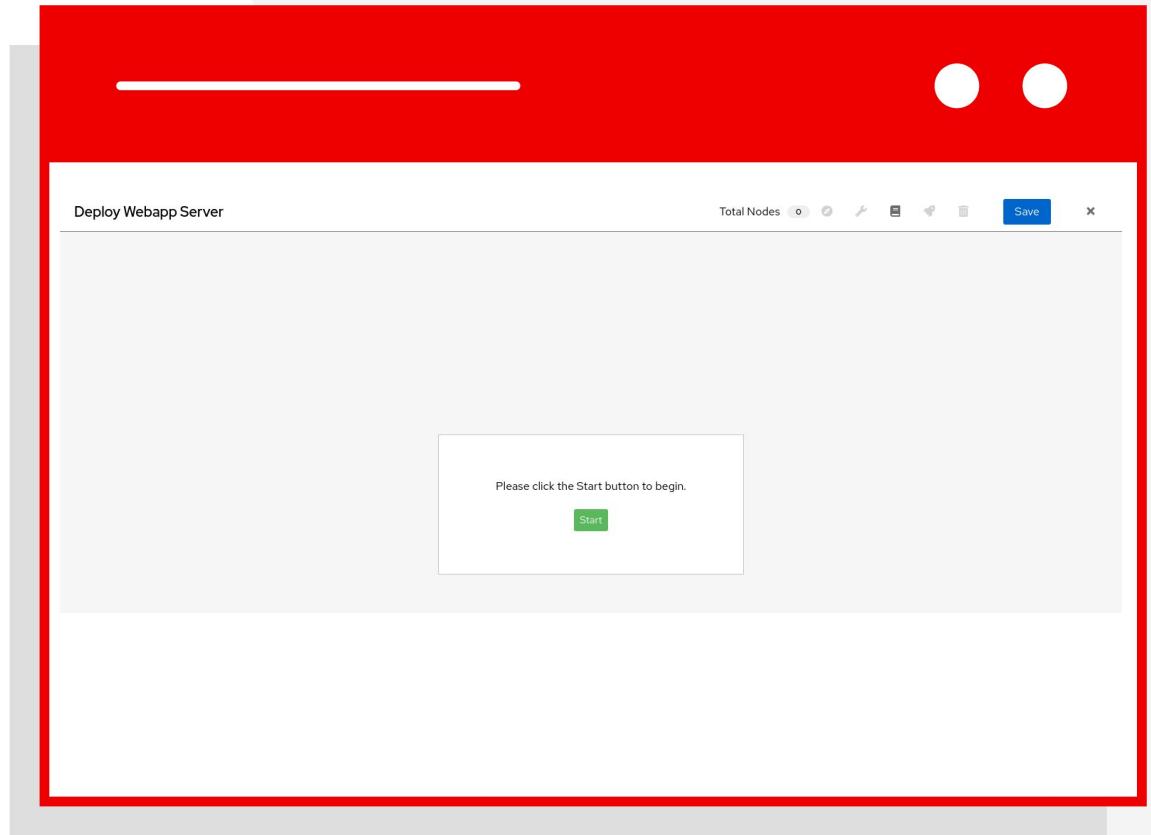
# Creating the Workflow

- ▶ Fill out the required parameters and click **Save**.  
As soon as the Workflow Template is saved the Workflow Visualizer will open.



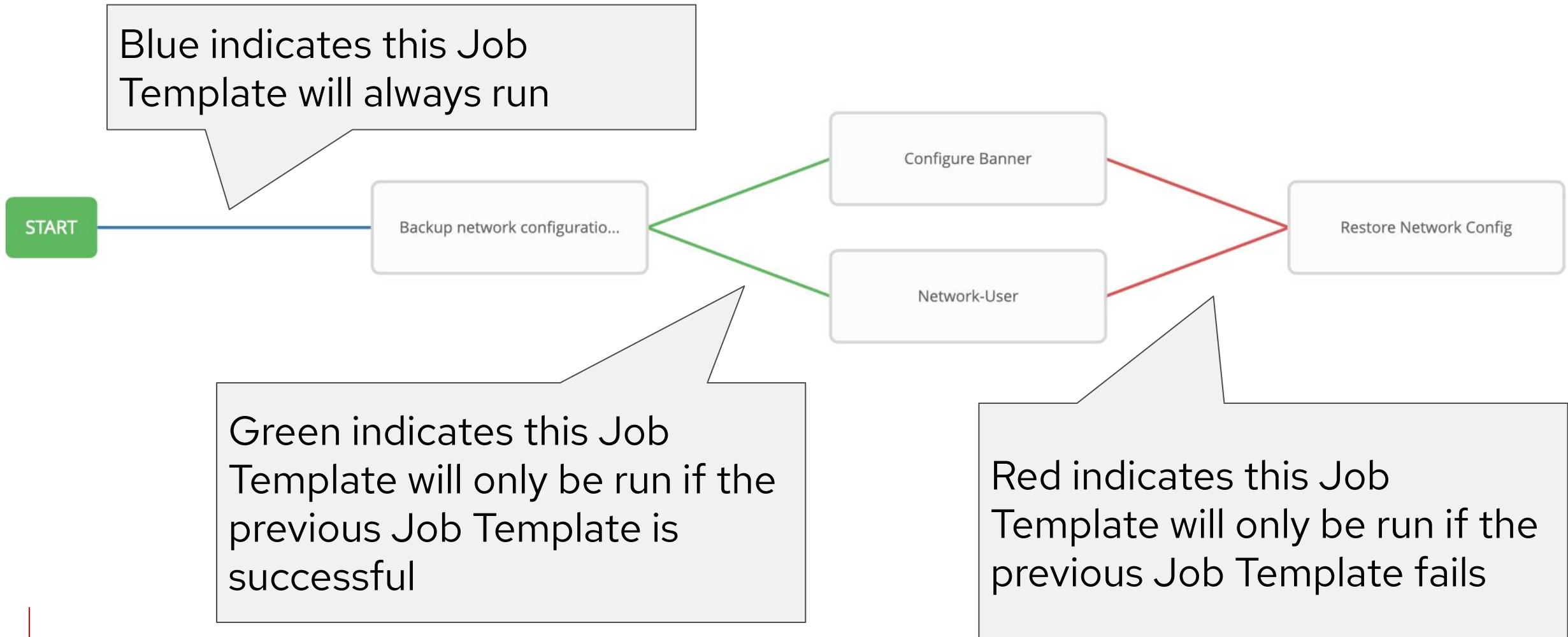
# Workflow Visualizer

- ▶ The Workflow Visualizer will start as a blank canvas.
- ▶ Click the green Start button to start building the workflow.



# 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**

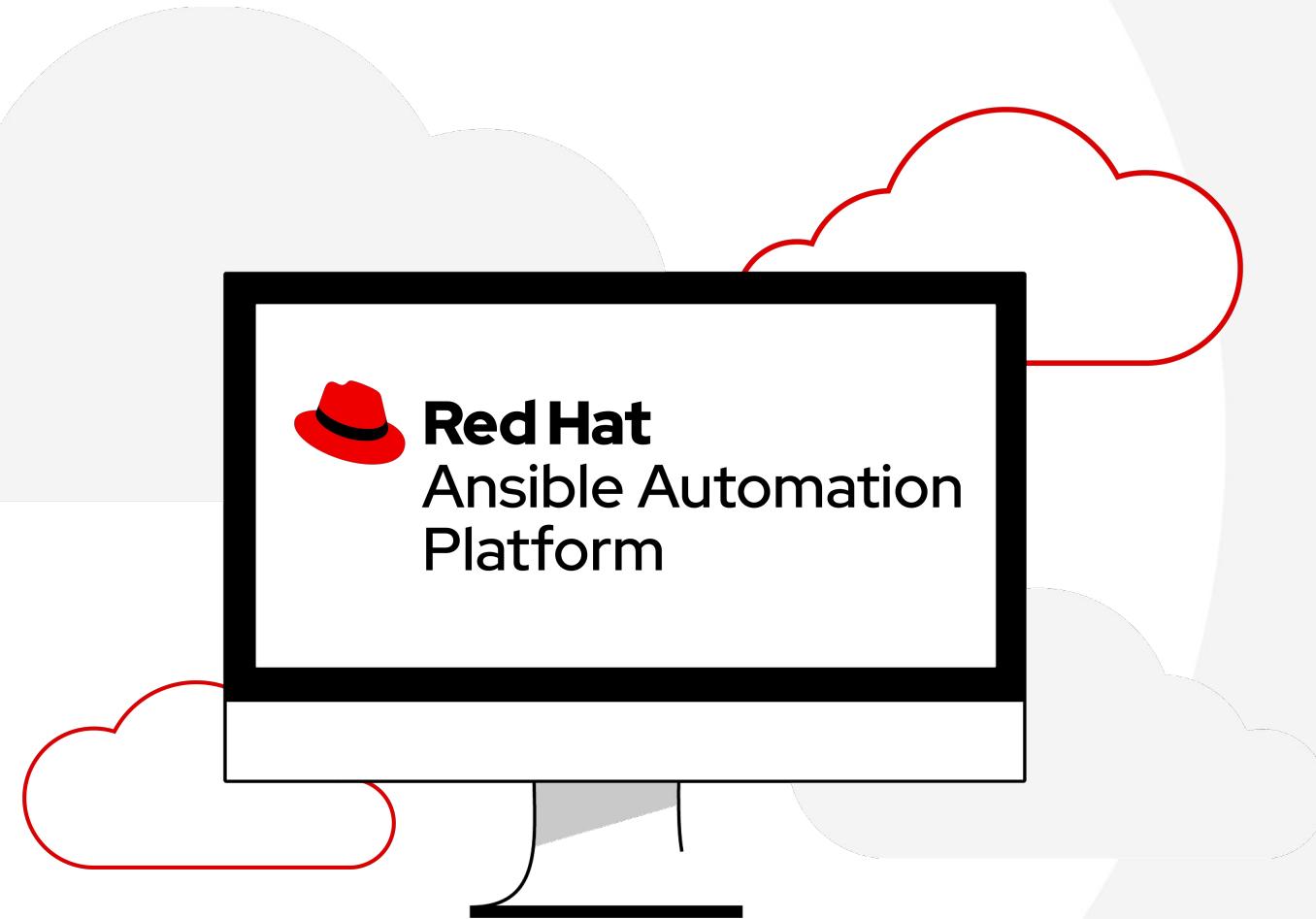
## Ansible Automation Platform

### Lab Time

Complete exercise 2.7 now in your lab environment



**Red Hat**



# Where to go next

## Learn more

- ▶ [Workshops](#)
- ▶ [Documents](#)
- ▶ [Youtube](#)
- ▶ [Twitter](#)

## Get started

- ▶ [Evals](#)
- ▶ [cloud.redhat.com](#)

## Get serious

- ▶ [Red Hat Automation Adoption Journey](#)
- ▶ [Red Hat Training](#)
- ▶ [Red Hat Consulting](#)

# Thank you



[linkedin.com/company/red-hat](https://linkedin.com/company/red-hat)



[youtube.com/AnsibleAutomation](https://youtube.com/AnsibleAutomation)



[facebook.com/ansibleautomation](https://facebook.com/ansibleautomation)



[twitter.com/ansible](https://twitter.com/ansible)



[github.com/ansible](https://github.com/ansible)