

Ansible Linux Automation Workshop

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



Red Hat Ansible Automation Platform

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

Introduction

Topics Covered:

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



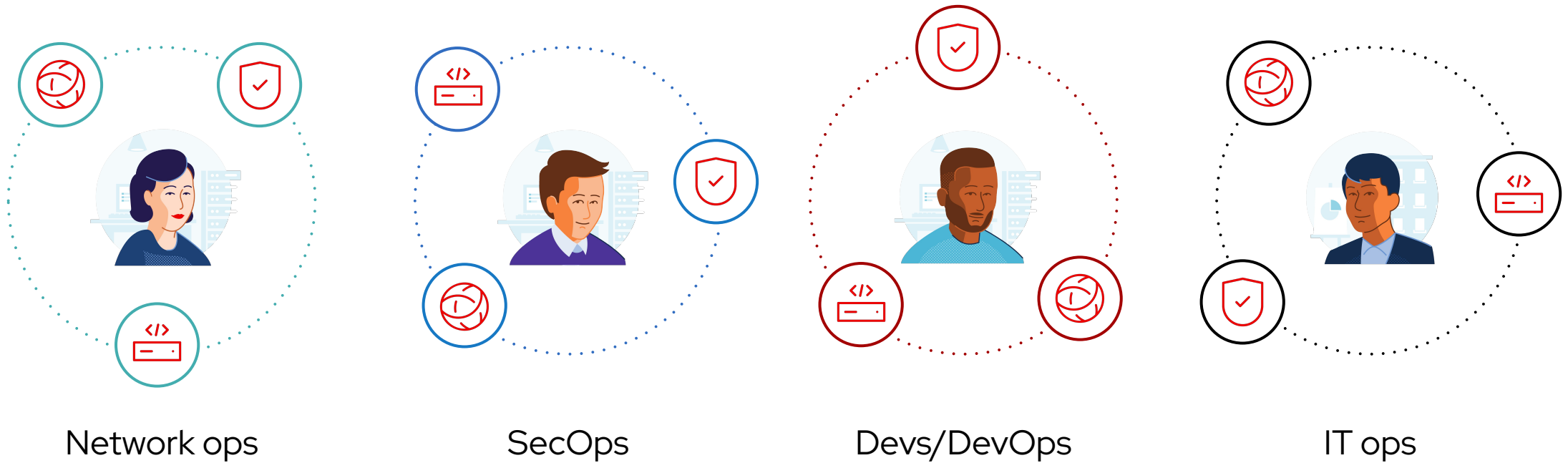
Red Hat
Ansible Automation
Platform



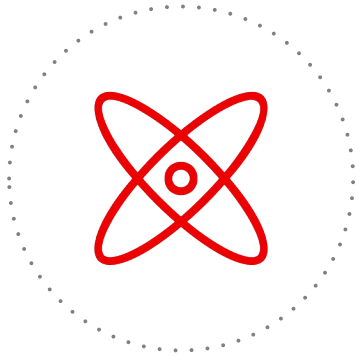
Automation happens when
one person meets a problem
they never want to solve again

Many organizations share the same challenge

Too many unintegrated, domain-specific tools

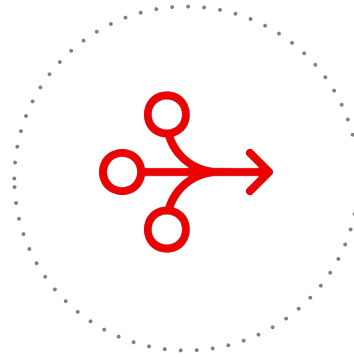


Why the Ansible Automation Platform?



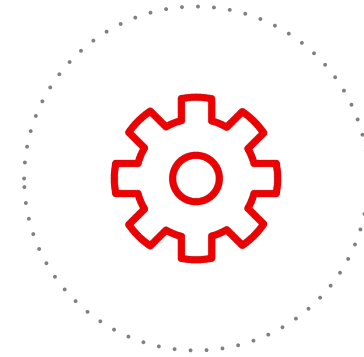
Powerful

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...



Firewalls



Load balancers



Applications



Containers



Virtualization platforms



Servers



Clouds



Storage



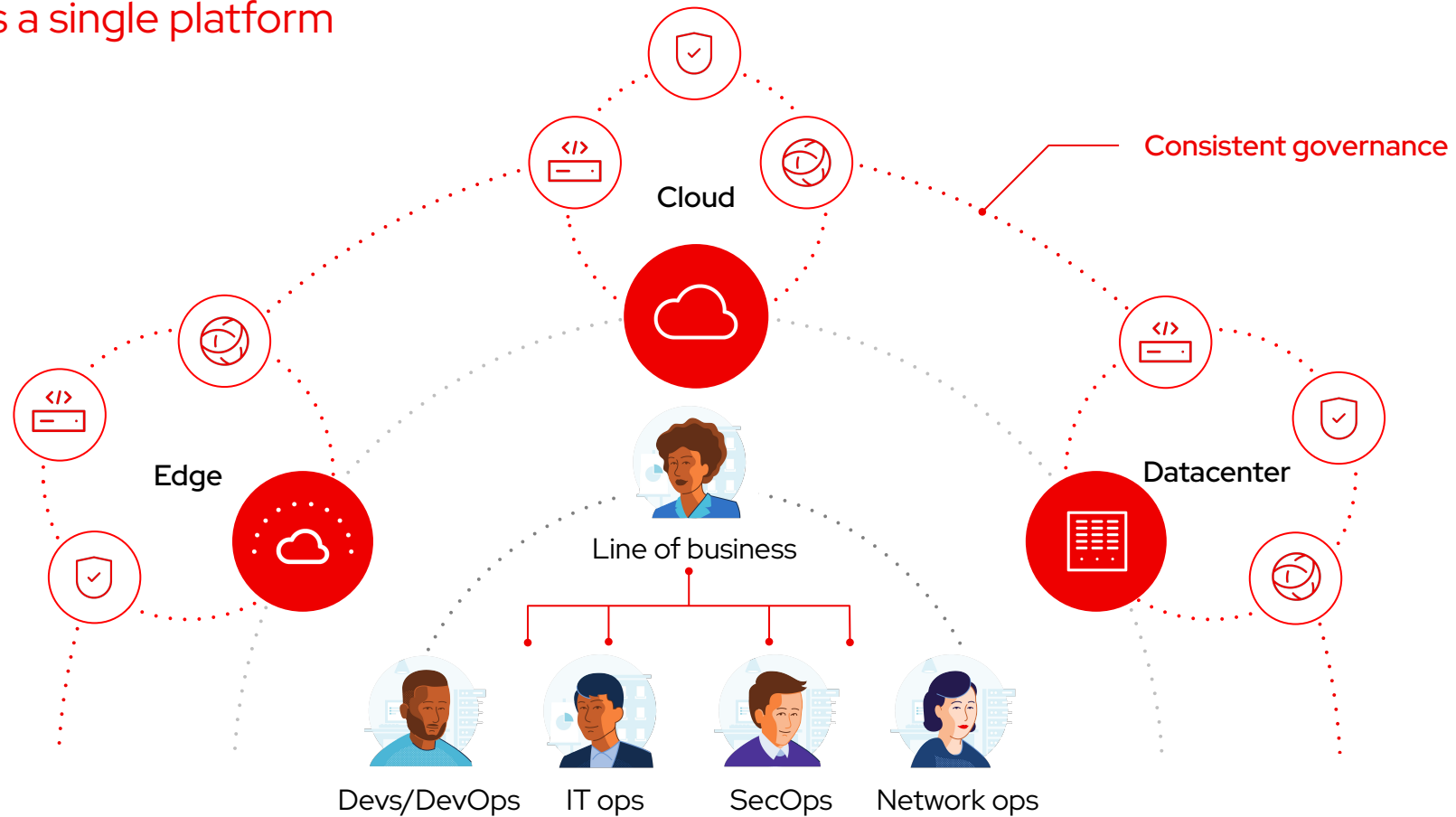
Network devices



And more ...

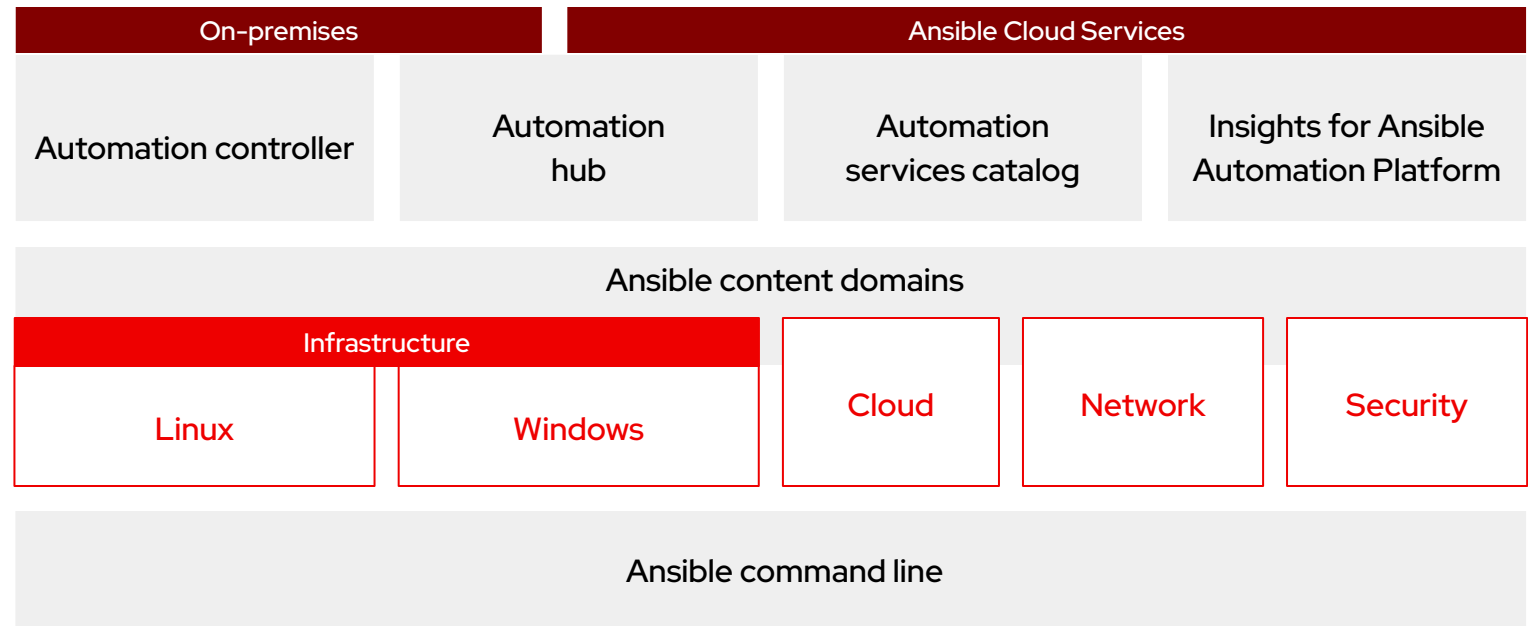
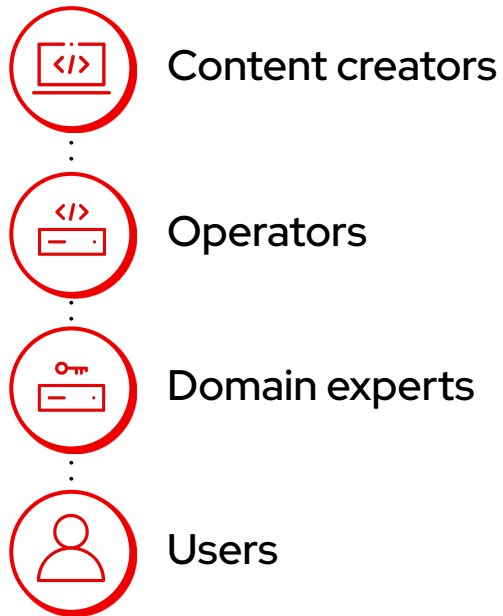
Break down silos

Different teams a single platform



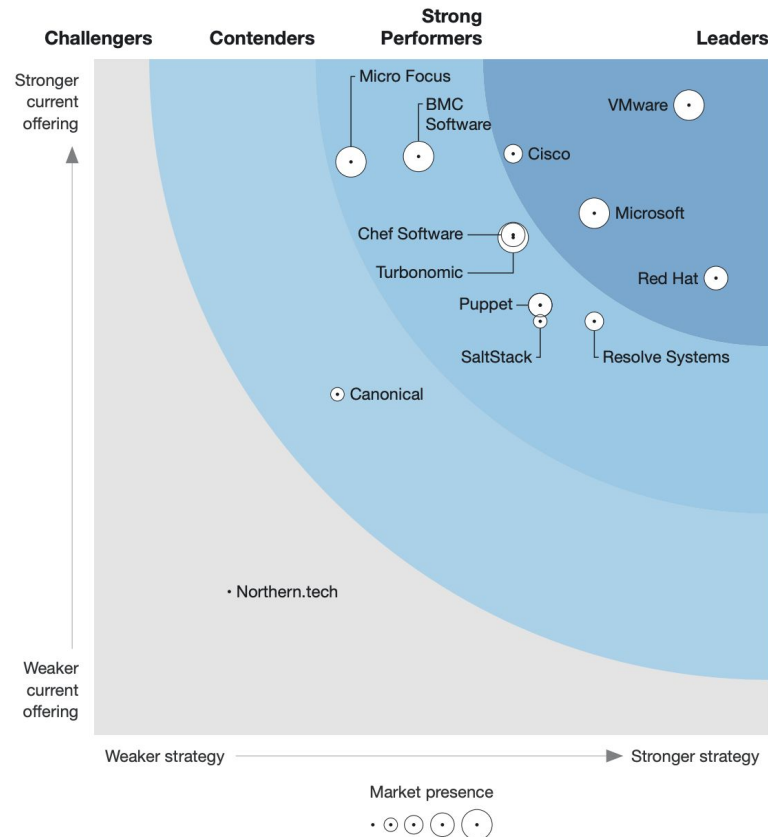
What makes a platform?

Red Hat Ansible Automation Platform



**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™.





Section 1

The Ansible Basics

Exercise 1.1

Topics Covered:

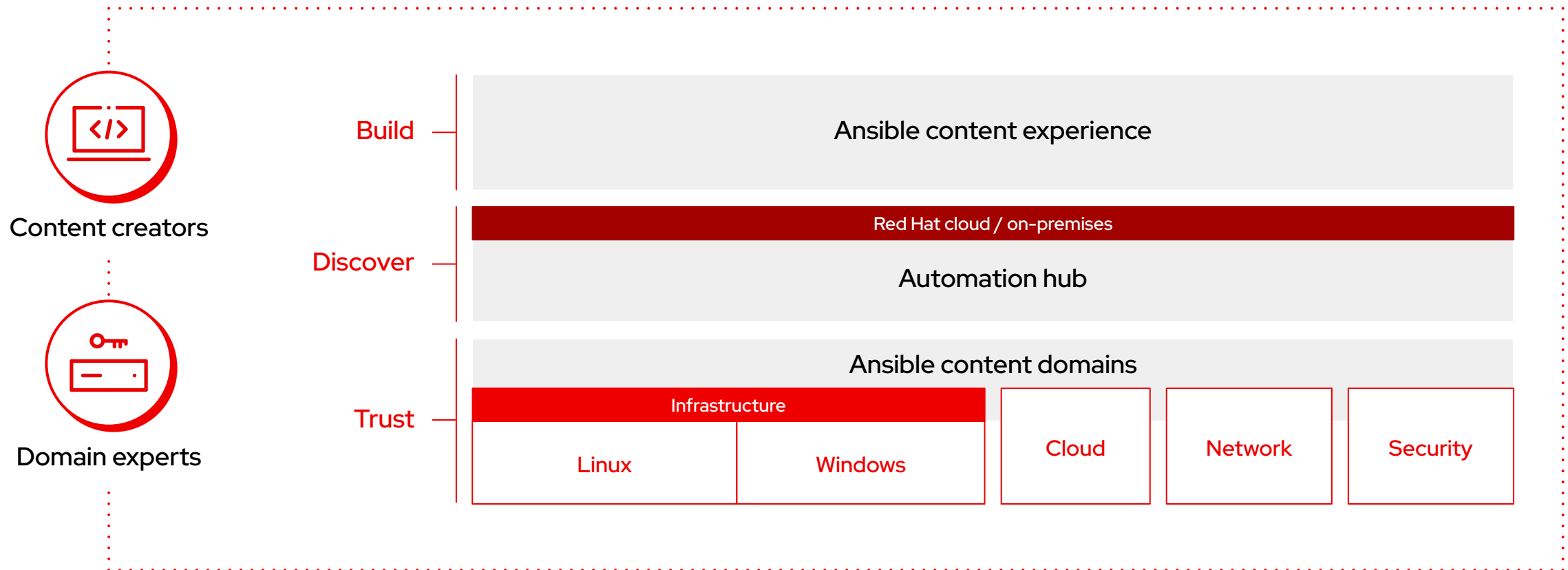
- Understanding the Ansible Infrastructure
- Check the prerequisites



Red Hat
Ansible Automation
Platform

Create

The automation lifecycle





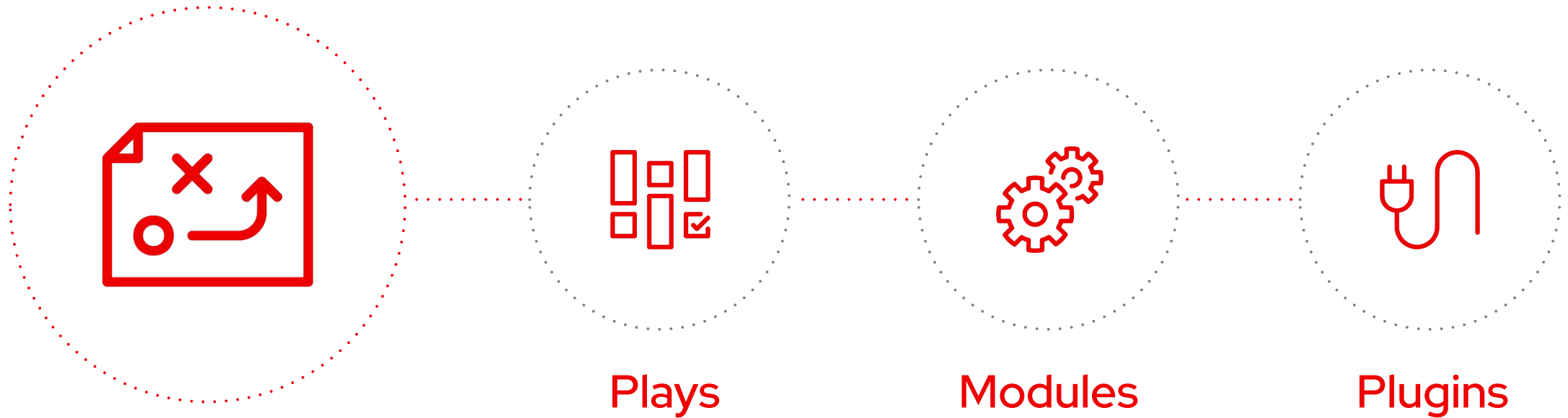
```
---
- name: install and start apache
  hosts: web
  become: yes

  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?



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.

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



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.

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

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.

```
Example become plugin:  
---  
- name: install and start apache  
  hosts: web  
  become: yes  
  
Example filter plugins:  
{{ some_variable | to_nice_json }}  
{{ some_variable | to_nice_yaml }}
```

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.

```
---  
- name: install and start apache  
  hosts: web  
  roles:  
    - common  
    - webservers
```

Collections

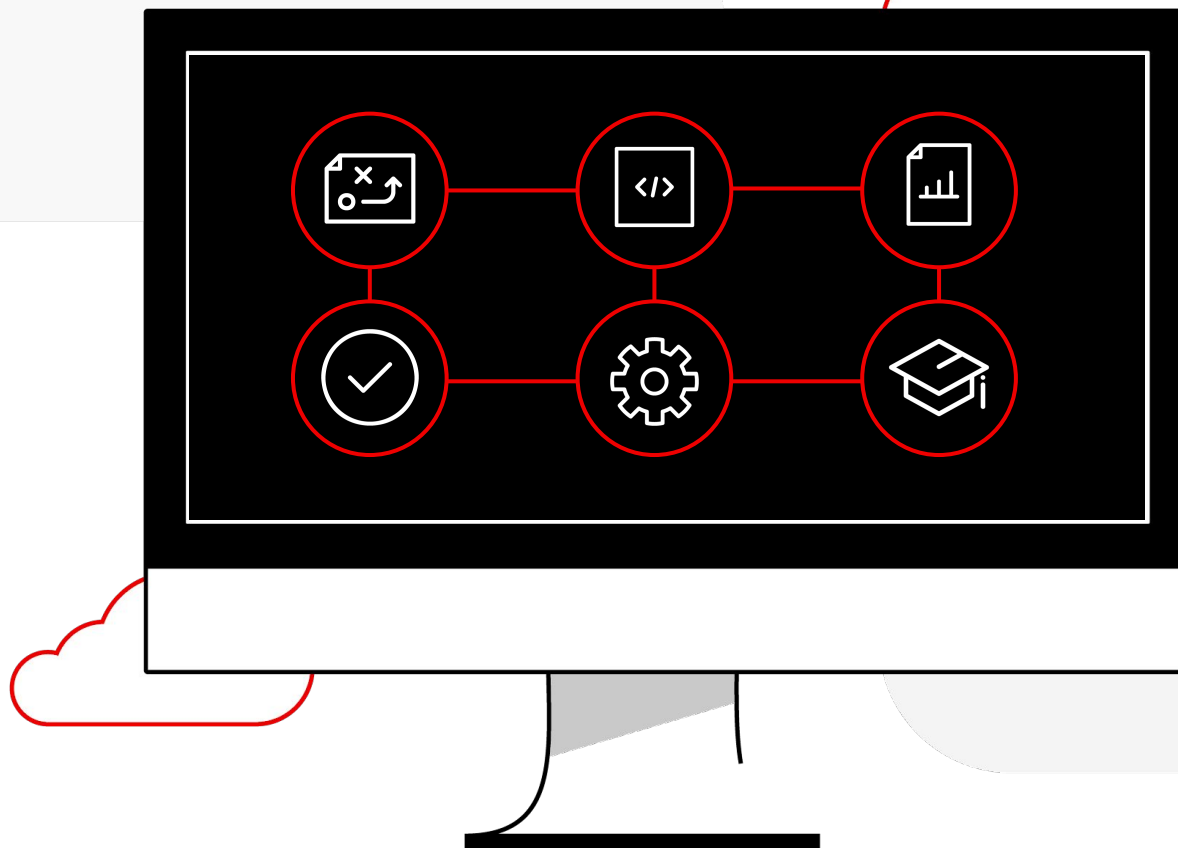
Simplified and consistent content delivery



What are they?

Collections are a data structure containing automation content:

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





```
nginx_core
├── MANIFEST.json
├── playbooks
│   └── deploy-nginx.yml
│       └── ...
├── plugins
├── README.md
├── roles
│   └── nginx
│       ├── defaults
│       ├── files
│       │   └── ...
│       ├── tasks
│       └── templates
│           └── ...
├── nginx_app_protect
└── nginx_config
```

deploy-nginx.yml

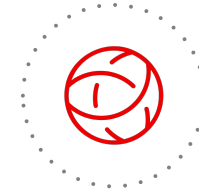
```
---
- name: Install NGINX Plus
  hosts: all
  tasks:
    - name: Install NGINX
      include_role:
        name: nginxinc.nginx
      vars:
        nginx_type: plus
    - name: Install NGINX App Protect
      include_role:
        name: nginxinc.nginx_app_protect
      vars:
        nginx_app_protect_setup_license: false
        nginx_app_protect_remove_license: false
        nginx_app_protect_install_signatures: false
```



Infrastructure



Cloud



Network



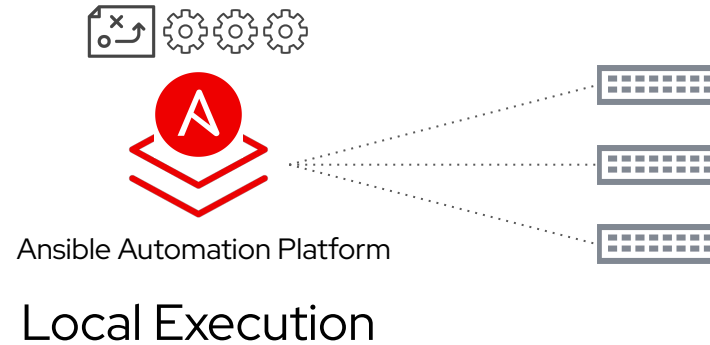
Security

90+
certified platforms

 Red Hat		ARISTA	 Check Point <small>SOFTWARE TECHNOLOGIES LTD</small>
 NetApp™		 CISCO™	 CYBERARK®
 IBM®	 Microsoft	 f5®	FORTINET®

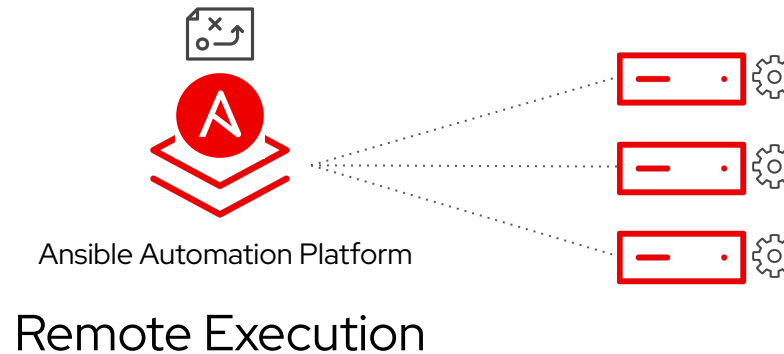
How Ansible Automation Works

Module code is executed locally on the control node



Network Devices /
API Endpoints

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



Linux / Windows
Hosts

Exercise 1.1

- 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



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)



```
[myservers]
10.42.0.2
10.42.0.6
10.42.0.7
10.42.0.8
10.42.0.100
host.example.com
```



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

[web]
node-[1:30]

[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]

[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
```


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/modules/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

Exercise 1.3

Topics Covered:

- Playbooks basics
- Running a playbook



Red Hat
Ansible Automation
Platform



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

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 run playbook.yml
```



ansible-navigator

Bye ansible-playbook, Hello ansible-navigator



How do I use ansible-navigator?

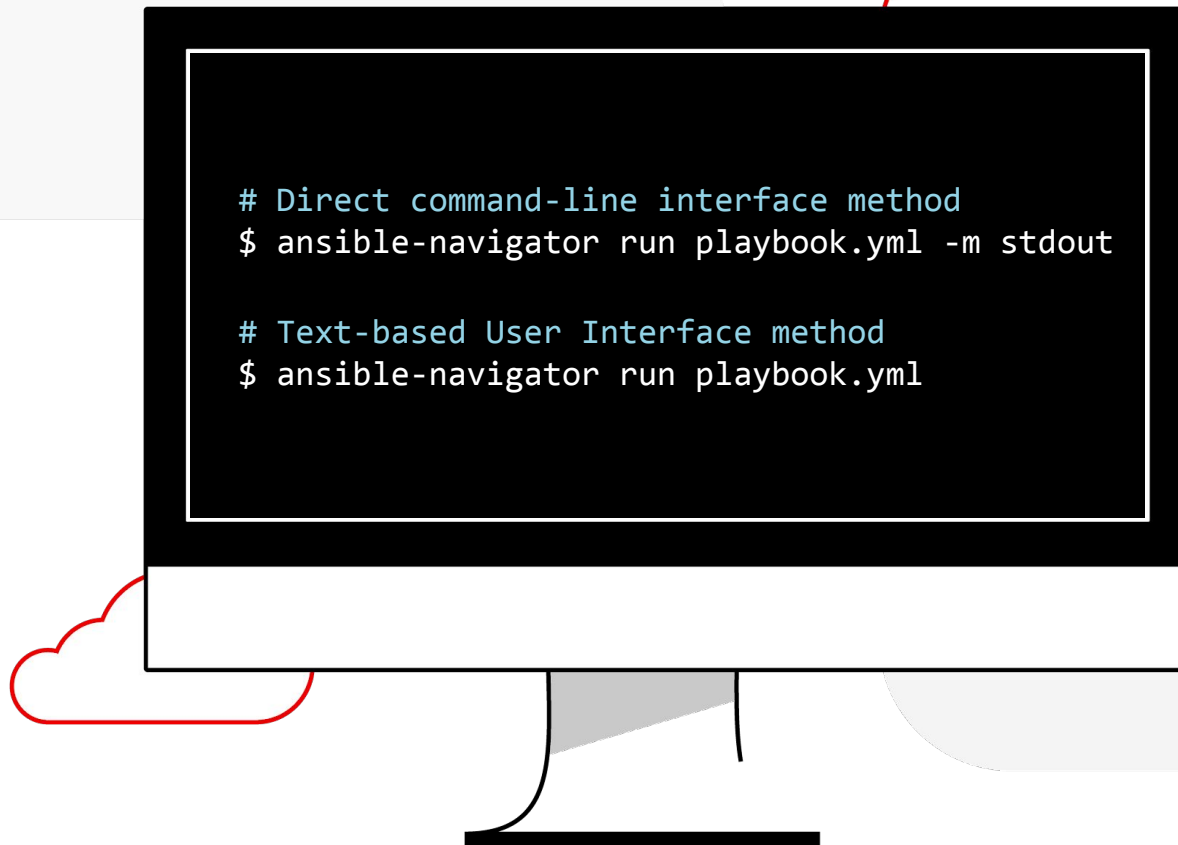
As previously mentioned, it replaces the ansible-playbook command.

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

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

ansible-navigator

Common subcommands

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



Red Hat Ansible Automation Platform

Lab Time

Complete exercise 1.3 now in your lab environment

Exercise 1.4

Topics Covered:

- Working with variables
- What are facts?



Red Hat
Ansible Automation
Platform



```
---
- 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

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



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

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



Ansible Navigator TUI

PLAY NAME	OK	CHANGED	UNREACHABLE	FAILED	SKIPPED	IGNORED	IN PROGRESS	TASK COUNT	PROGRESS
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

Exercise 1.5

Topics Covered:

- Conditionals
- Handlers
- Loops



Red Hat
Ansible Automation
Platform

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: http_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

Exercise 1.6

Topics Covered:

- Templates



Red Hat
Ansible Automation
Platform



```
- 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

Exercise 1.7

Topics Covered:

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



Red Hat
Ansible Automation
Platform

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



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

Section 2

Automation Controller



Red Hat
Ansible Automation
Platform

Exercise 2.1

Topics Covered:

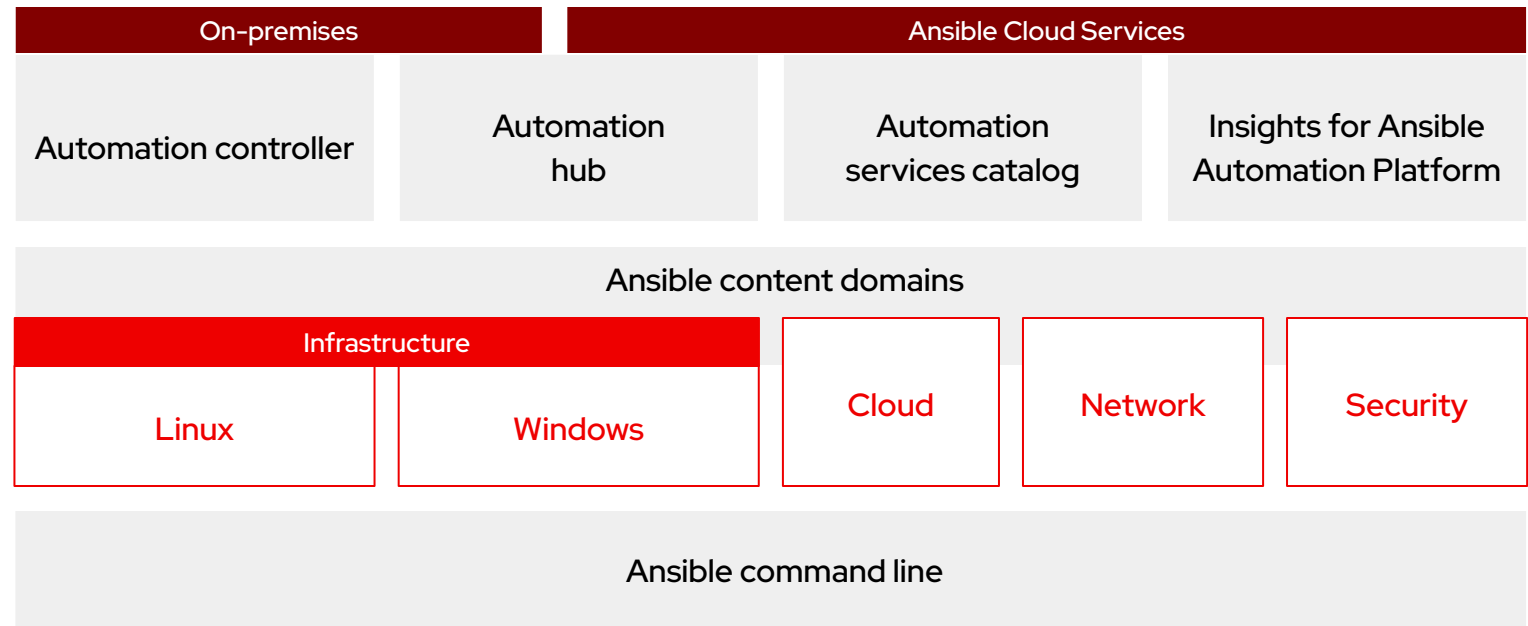
- Introduction to Automation Controller



Red Hat
Ansible Automation
Platform

What makes a platform?

Red Hat Ansible Automation Platform



Fueled by an
open source community

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 controller 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 Automation controllers API.

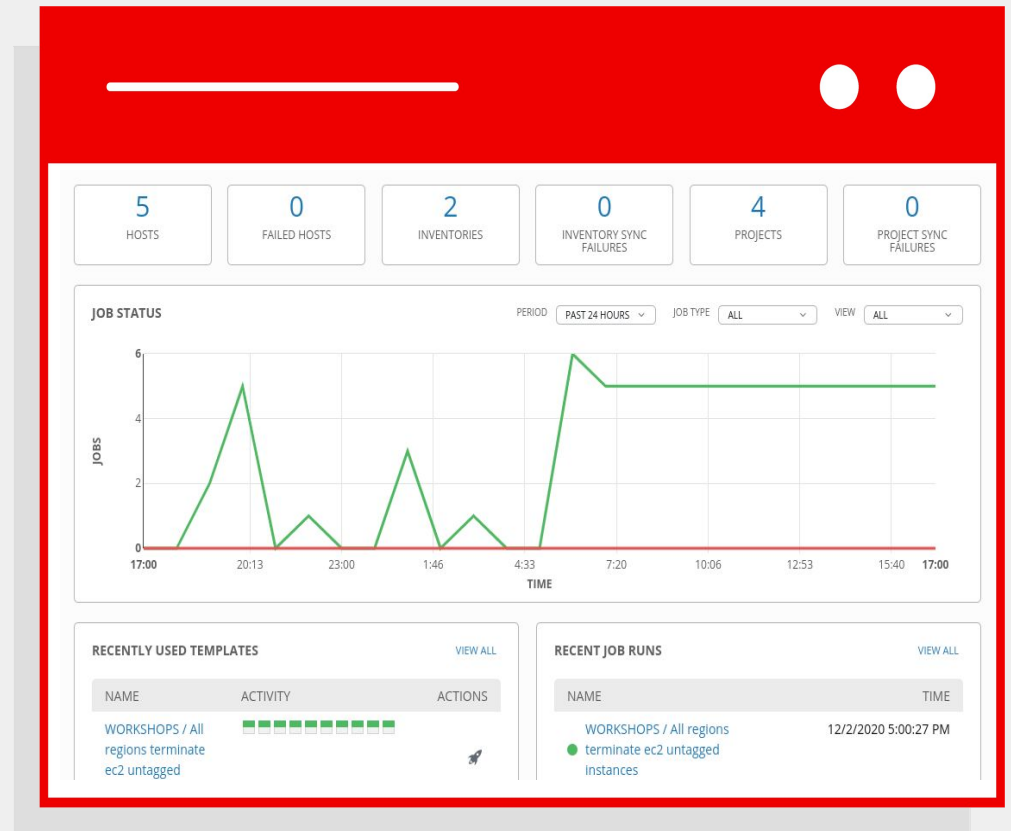
Workflows

Automation controller'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.

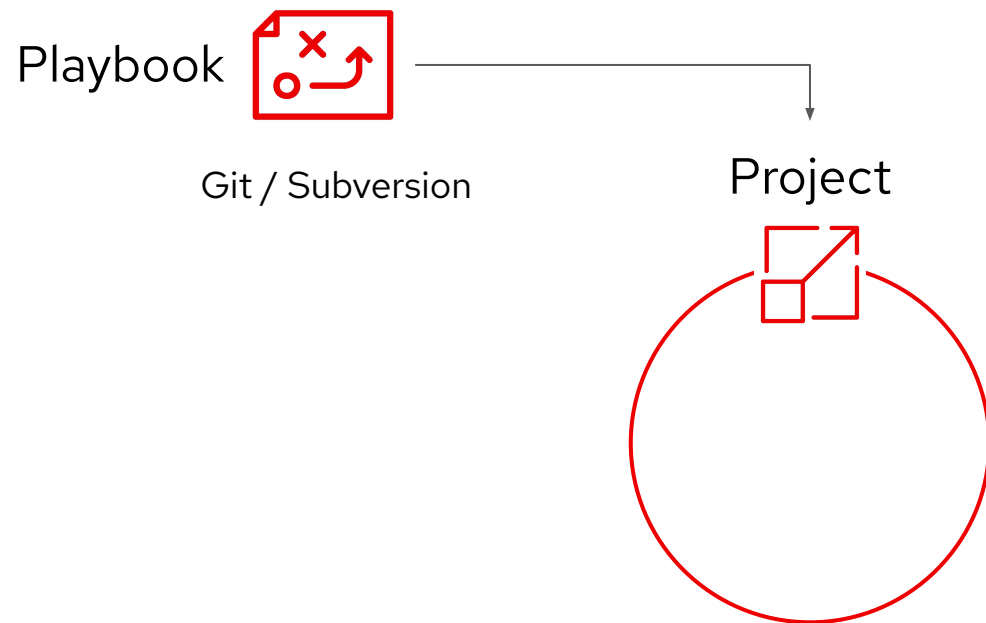
What is Ansible Automation Controller ?

Ansible Automation Controller is a UI and RESTful API allowing you to scale IT automation, manage complex deployments and speed productivity.

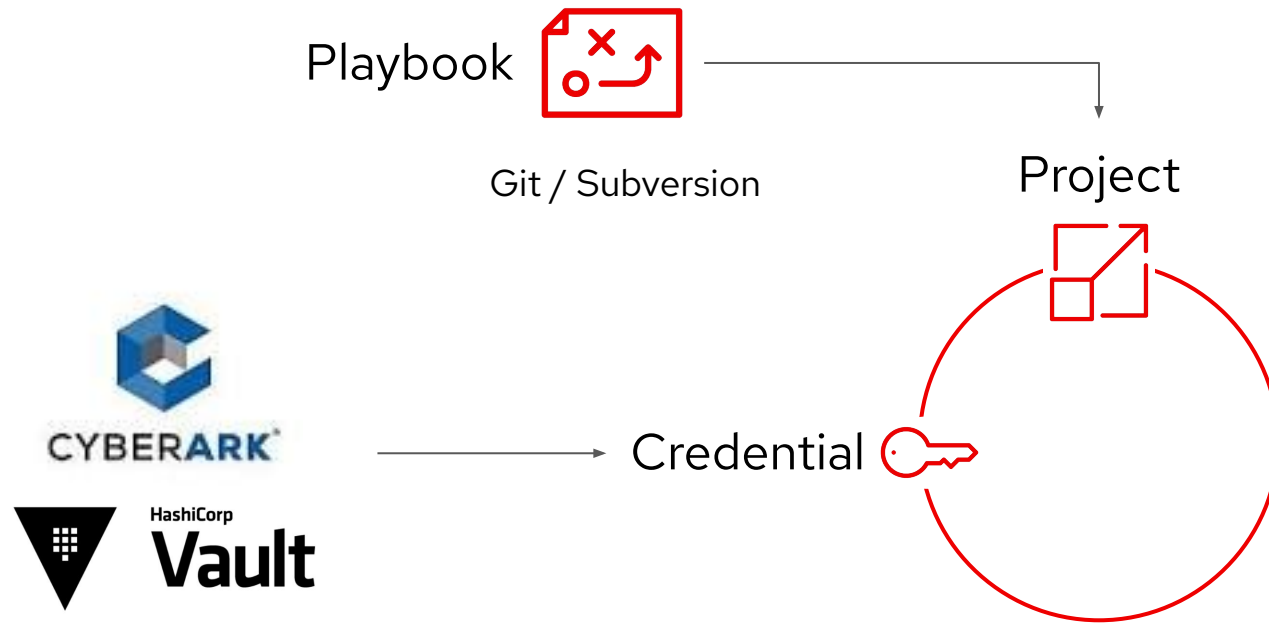
- ▶ Role-based access control
- ▶ Deploy entire applications with push-button deployment access
- ▶ All automations are centrally logged
- ▶ Powerful workflows match your IT processes



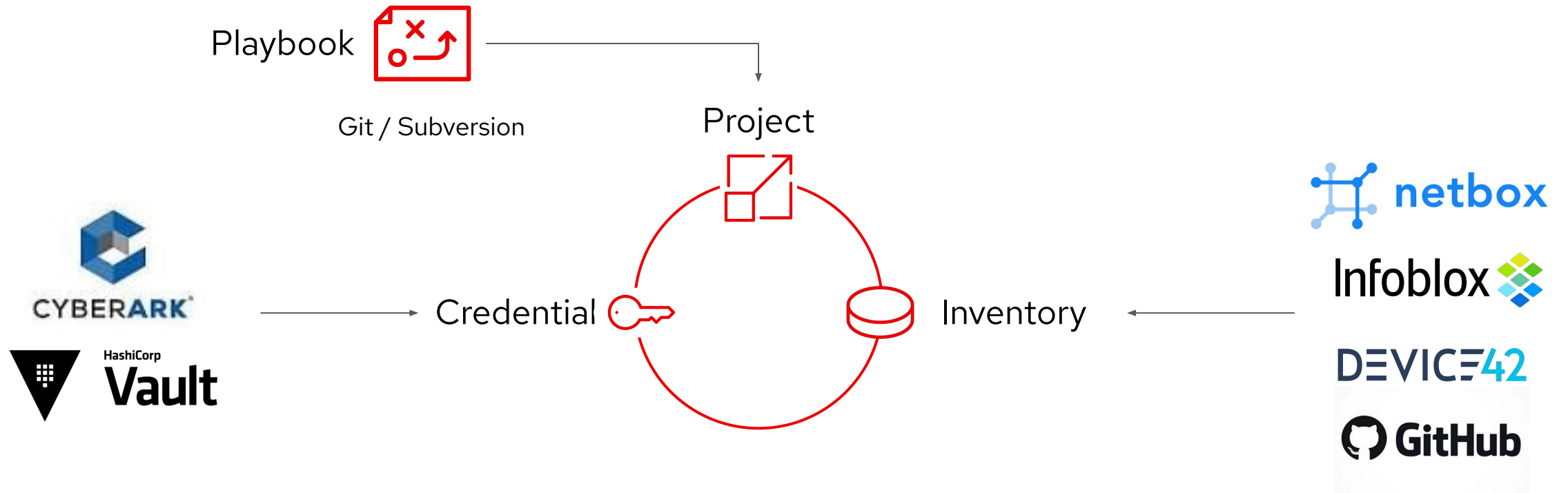
Anatomy of an Automation Job



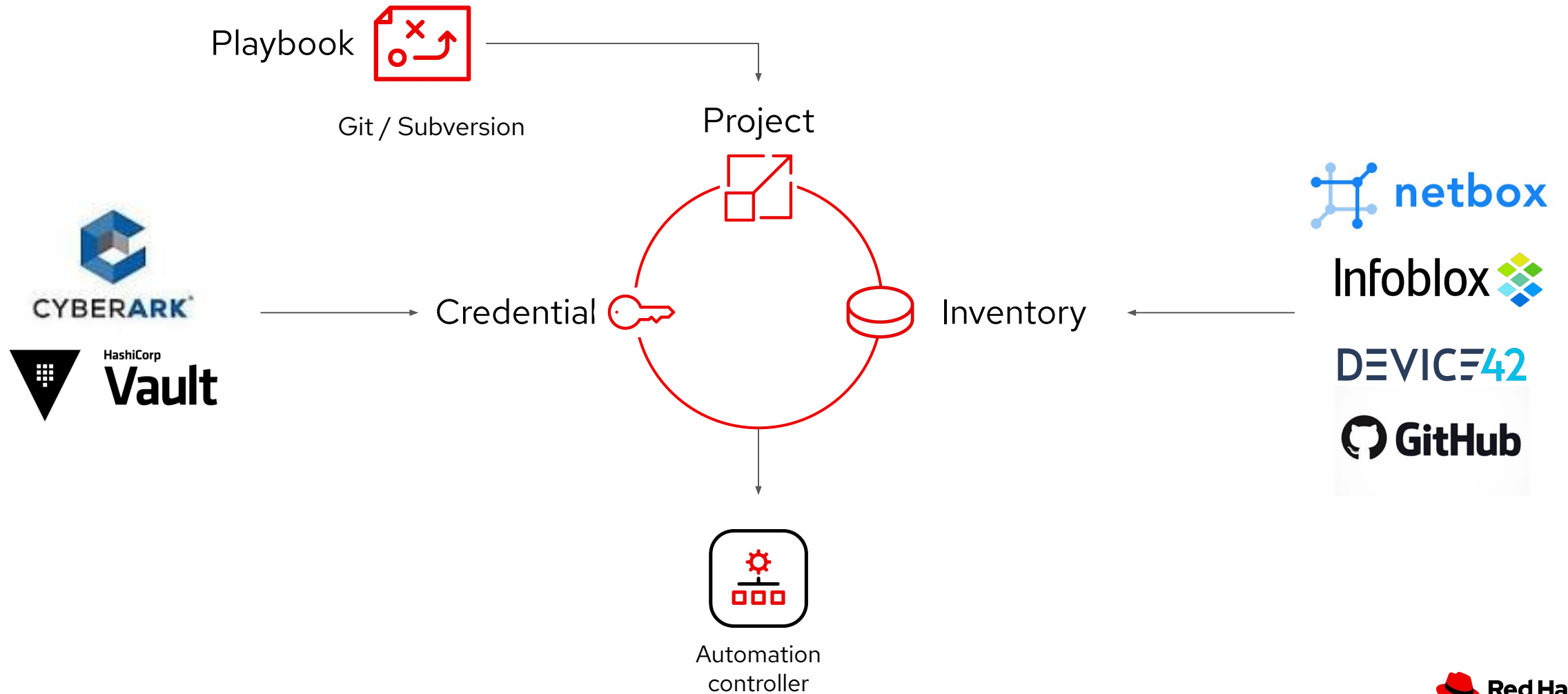
Anatomy of an Automation Job



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

Exercise 2.2

Topics Covered:

- Inventories
- Credentials

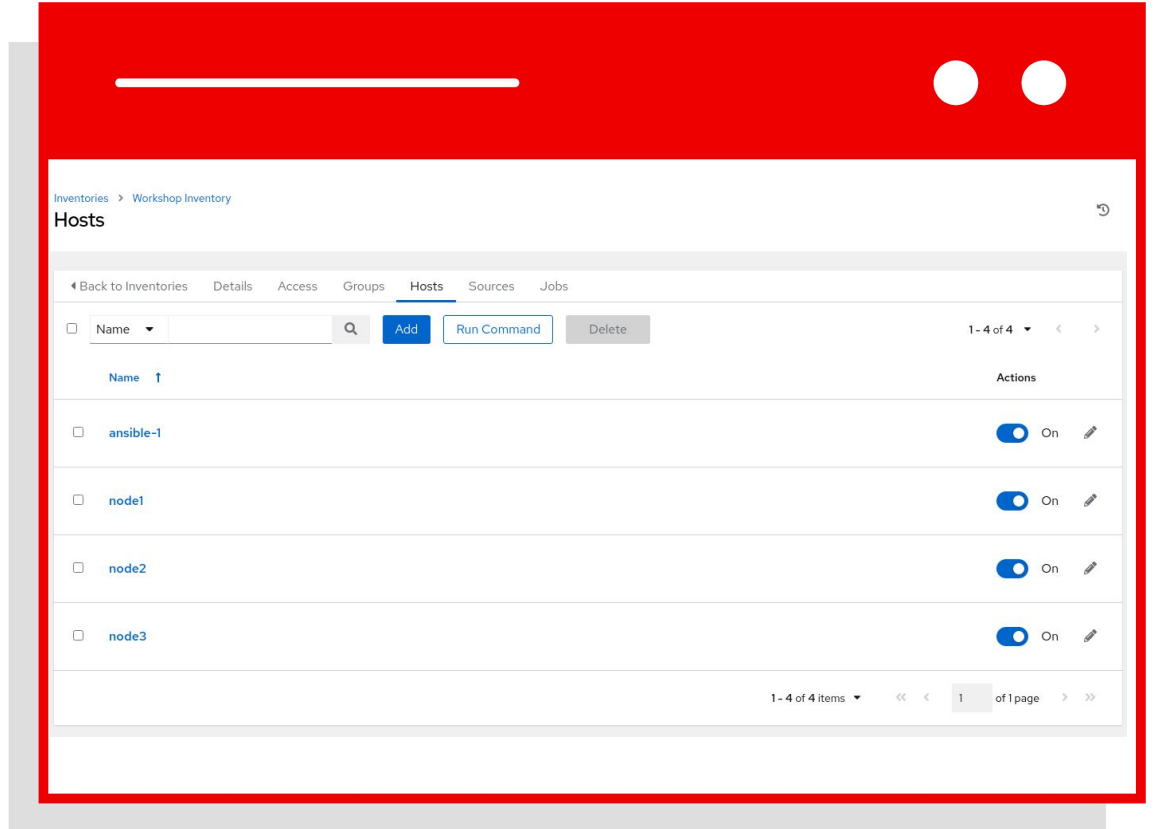


Red Hat
Ansible Automation
Platform

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

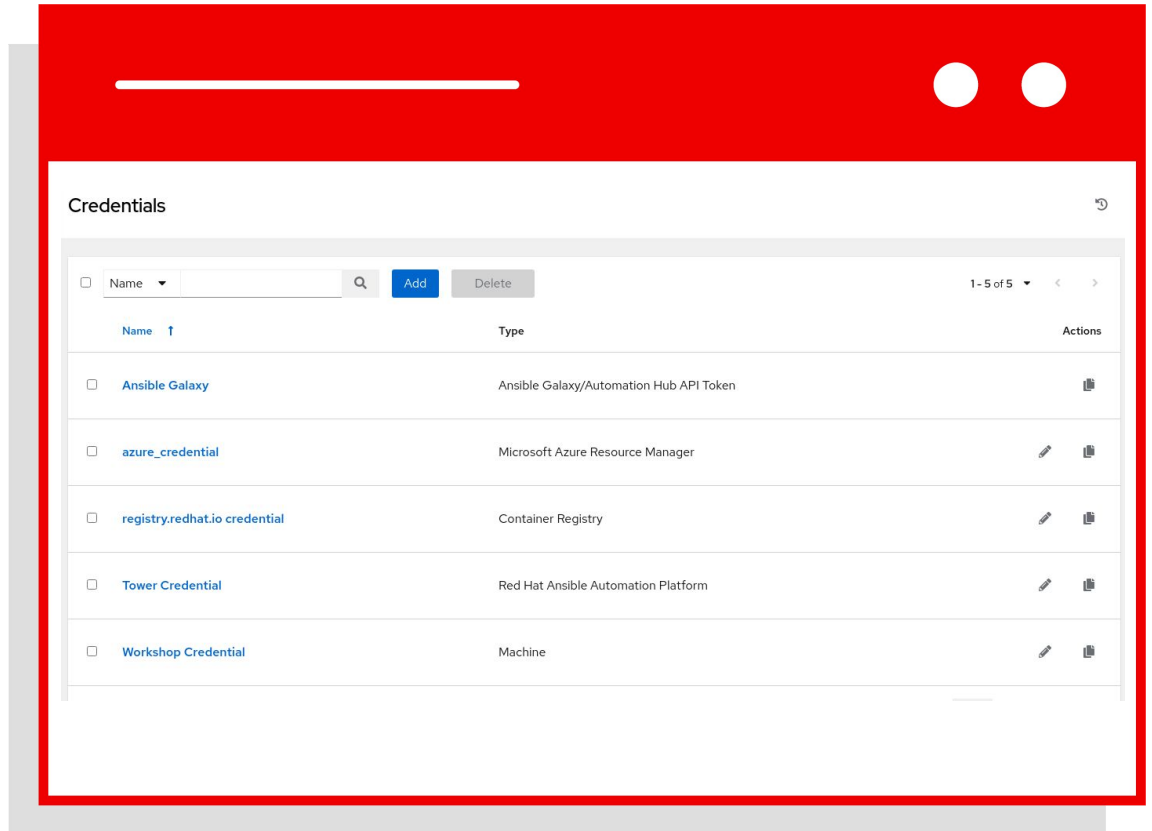


Credentials

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.





Red Hat Ansible Automation Platform

Lab Time

Complete exercise 2.2 now in your lab environment

Exercise 2.3

Topics Covered:

- Projects
- Job Templates

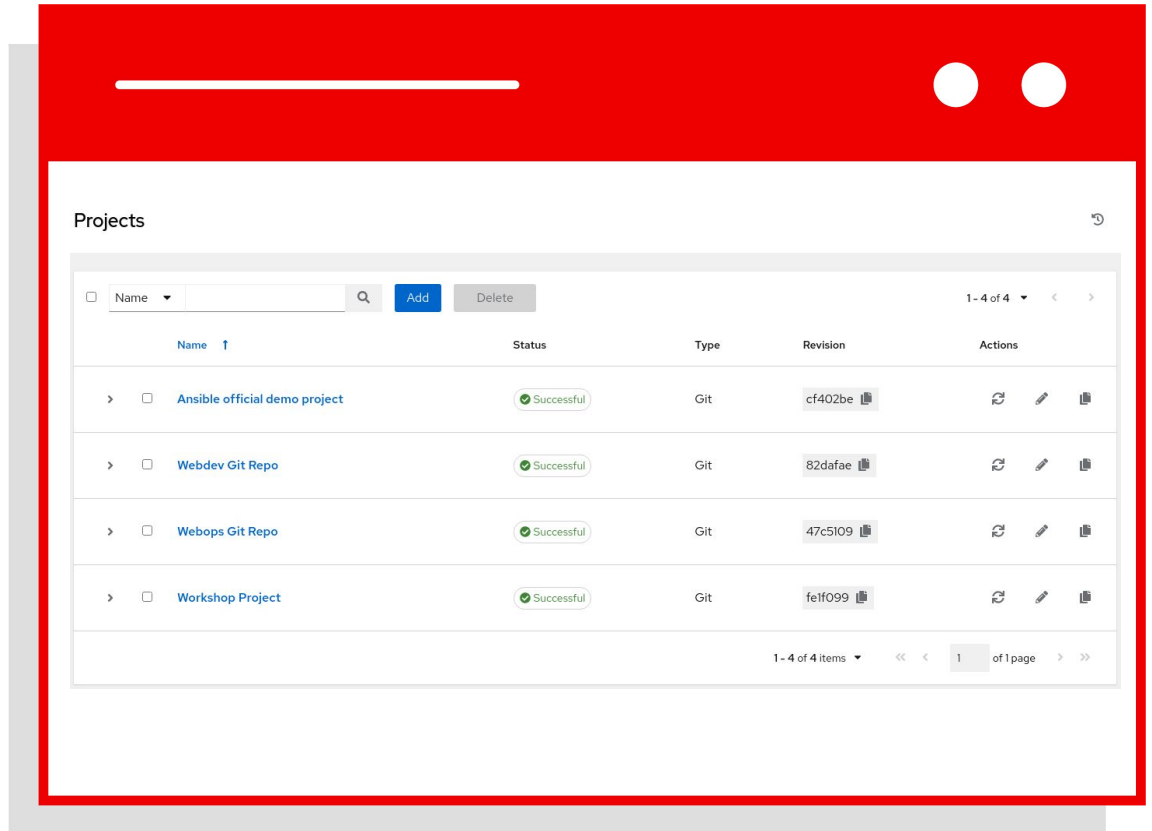


Red Hat
Ansible Automation
Platform

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 Automation controller, including Git and Subversion.



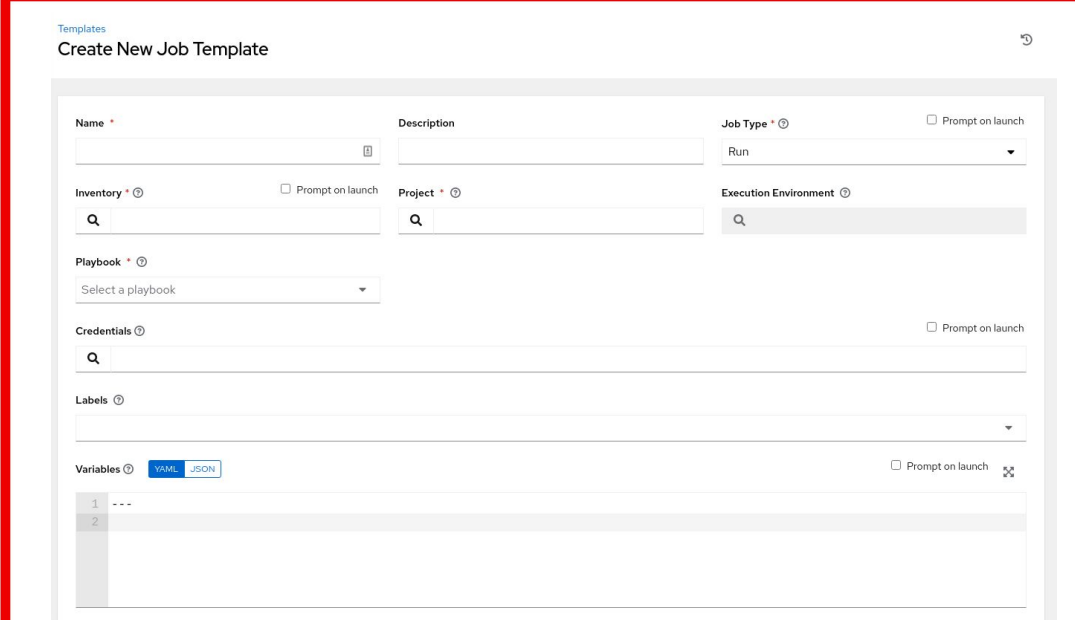
Job Templates

Everything in Automation controller 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



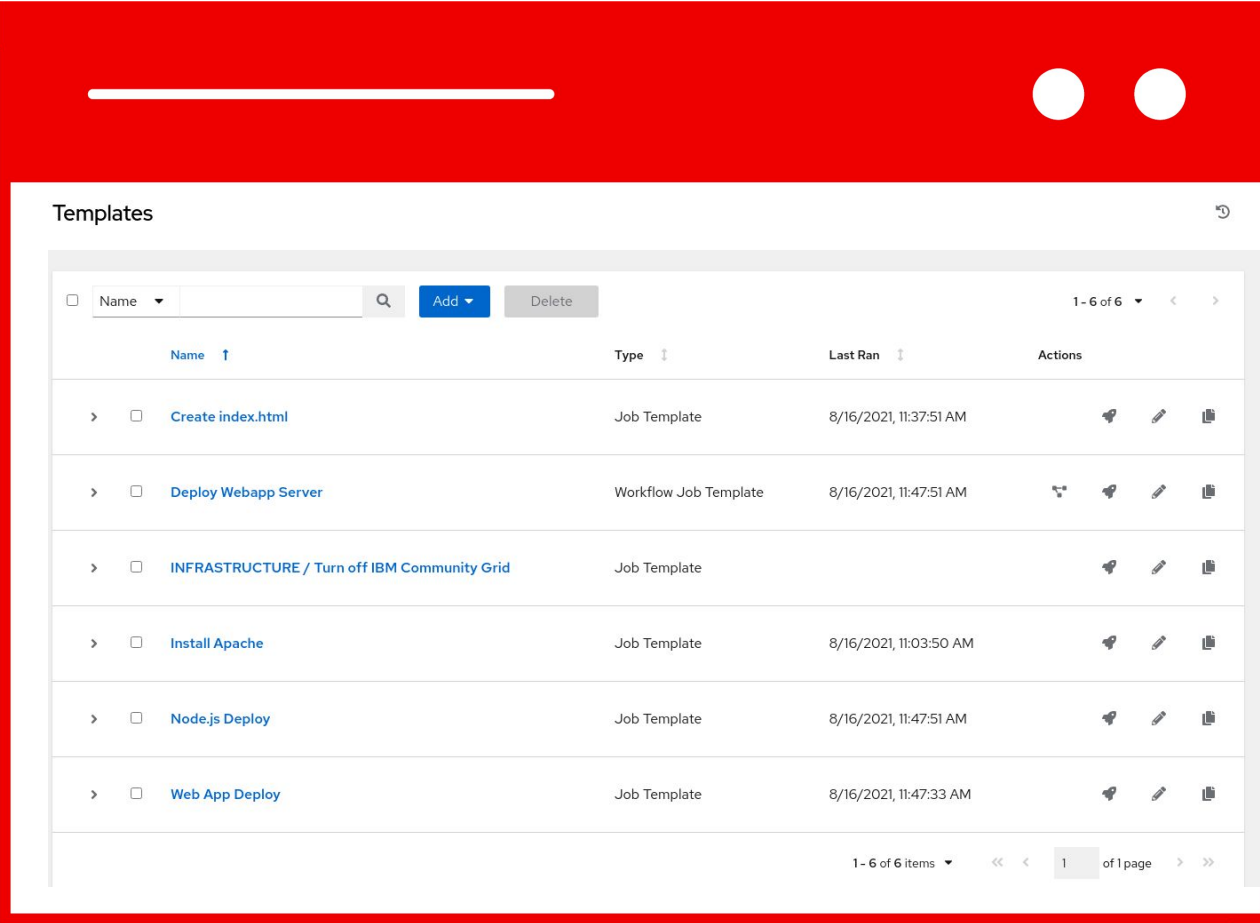
The screenshot shows the 'Create New Job Template' form in the Automation Controller interface. The form is titled 'Create New Job Template' and is located under the 'Templates' section. It contains several input fields and options:

- Name**: A text input field with a search icon.
- Description**: A text input field.
- Job Type**: A dropdown menu with 'Run' selected. There is a 'Prompt on launch' checkbox next to it.
- Inventory**: A search input field with a search icon and a 'Prompt on launch' checkbox.
- Project**: A search input field with a search icon.
- Execution Environment**: A search input field with a search icon.
- Playbook**: A dropdown menu with 'Select a playbook' selected.
- Credentials**: A search input field with a search icon and a 'Prompt on launch' checkbox.
- Labels**: A text input field.
- Variables**: A section with tabs for 'YAML' and 'JSON'. There is a 'Prompt on launch' checkbox and a refresh icon. Below the tabs is a list of variables with a table structure:

1	---
2	

Expanding on Job Templates

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



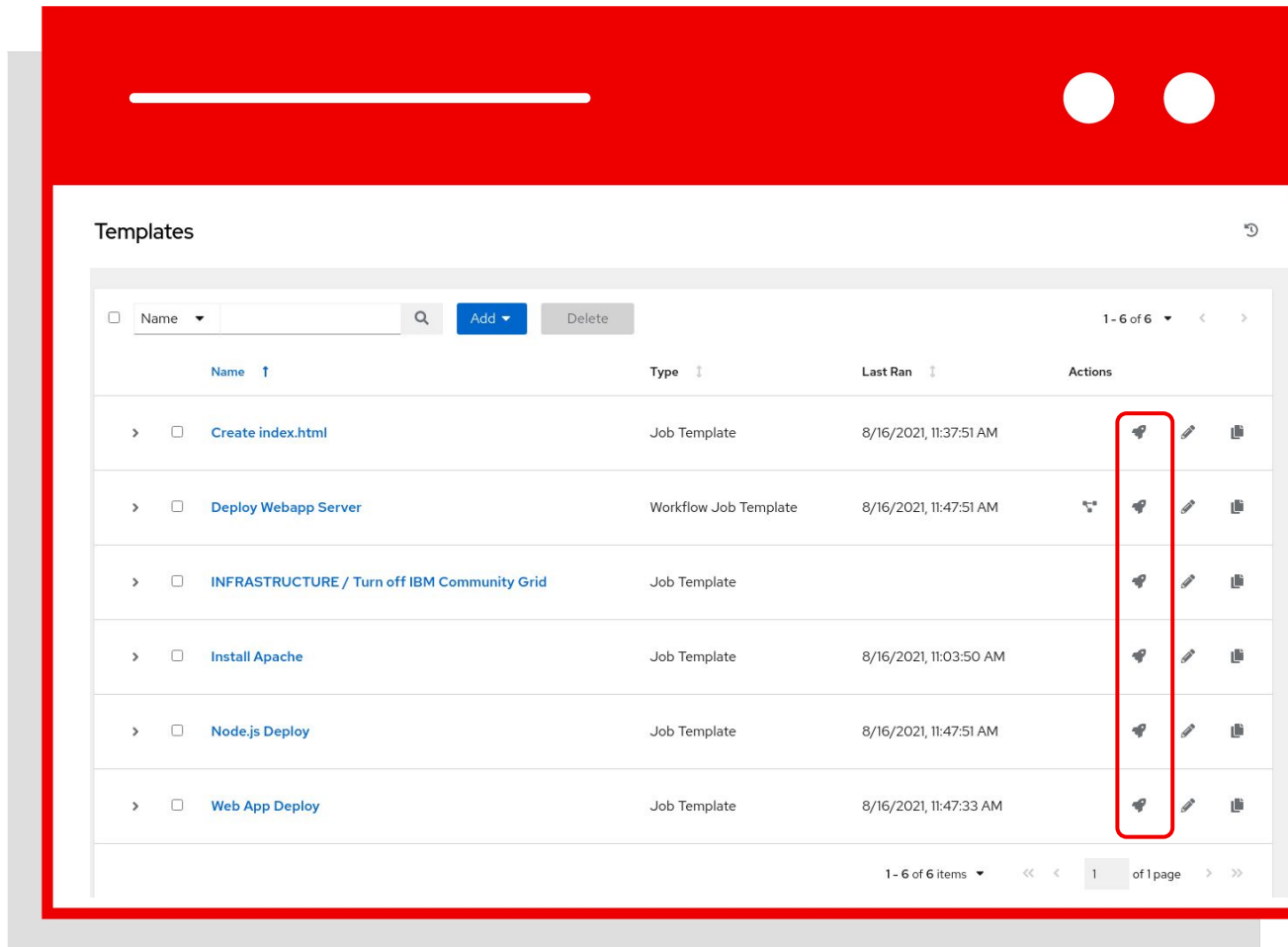
The screenshot displays a web interface for managing templates. At the top, there is a search bar with a magnifying glass icon, an 'Add' button, and a 'Delete' button. Below this is a table with the following columns: Name, Type, Last Ran, and Actions. The table contains six entries, each with a checkbox, a right-pointing chevron, and a link to the template name. The 'Actions' column for each entry contains three icons: a lightning bolt, a pencil, and a document.

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



















At the bottom of the table, there is a pagination control showing '1 - 6 of 6 items' and '1 of 1 page'.

Executing an existing Job Template

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

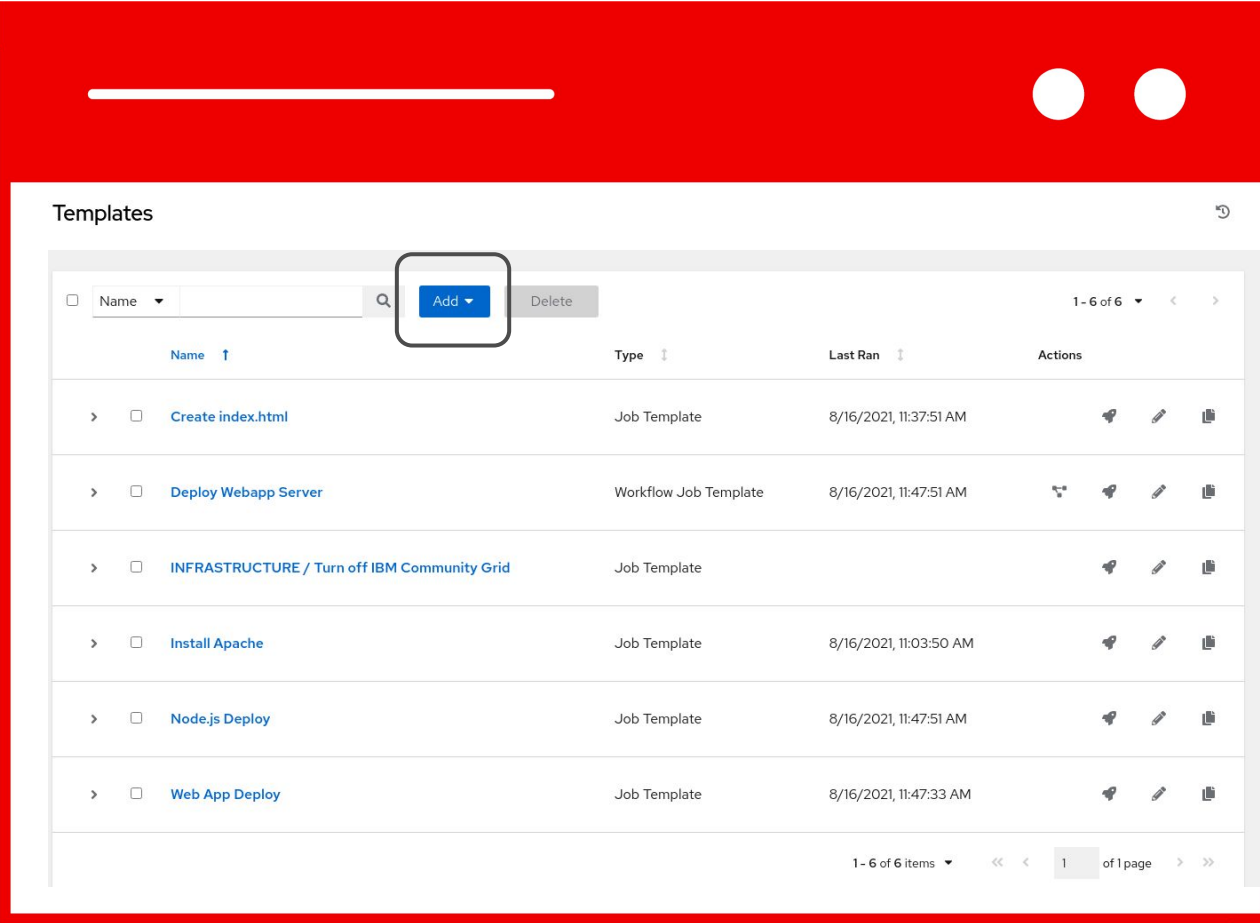


The screenshot displays a web interface for managing templates. At the top, there's a search bar and buttons for 'Add' and 'Delete'. Below is a table with columns for Name, Type, Last Ran, and Actions. A red box highlights the rocketship icon in the Actions column for the first row, 'Create index.html'.

Name	Type	Last Ran	Actions
> <input type="checkbox"/> Create index.html	Job Template	8/16/2021, 11:37:51 AM	  
> <input type="checkbox"/> Deploy Webapp Server	Workflow Job Template	8/16/2021, 11:47:51 AM	  
> <input type="checkbox"/> INFRASTRUCTURE / Turn off IBM Community Grid	Job Template		  
> <input type="checkbox"/> Install Apache	Job Template	8/16/2021, 11:03:50 AM	  
> <input type="checkbox"/> Node.js Deploy	Job Template	8/16/2021, 11:47:51 AM	  
> <input type="checkbox"/> 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 displays a web interface for managing job templates. At the top, there is a search bar with a dropdown menu set to 'Name', a search icon, and a blue 'Add' button with a dropdown arrow, which is highlighted by a red rectangular box. To the right of the search bar is a 'Delete' button. Below the search bar is a table with the following columns: Name, Type, Last Ran, and Actions. The table contains six rows of job templates. At the bottom of the interface, there is a pagination control showing '1 - 6 of 6 items' and '1 of 1 page'.

Name	Type	Last Ran	Actions
> <input type="checkbox"/> Create index.html	Job Template	8/16/2021, 11:37:51 AM	
> <input type="checkbox"/> Deploy Webapp Server	Workflow Job Template	8/16/2021, 11:47:51 AM	
> <input type="checkbox"/> INFRASTRUCTURE / Turn off IBM Community Grid	Job Template		
> <input type="checkbox"/> Install Apache	Job Template	8/16/2021, 11:03:50 AM	
> <input type="checkbox"/> Node.js Deploy	Job Template	8/16/2021, 11:47:51 AM	
> <input type="checkbox"/> 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 .

Templates

Create New Job Template

Name *

Description

Job Type * Prompt on launch

Inventory * Prompt on launch

Project *

Execution Environment

Playbook *

Credentials Prompt on launch

Labels

Variables Prompt on launch

YAML JSON

1 ---

2



Red Hat Ansible Automation Platform

Lab Time

Complete exercise 2.3 now in your lab environment

Exercise 2.4

Topics Covered:

- Surveys

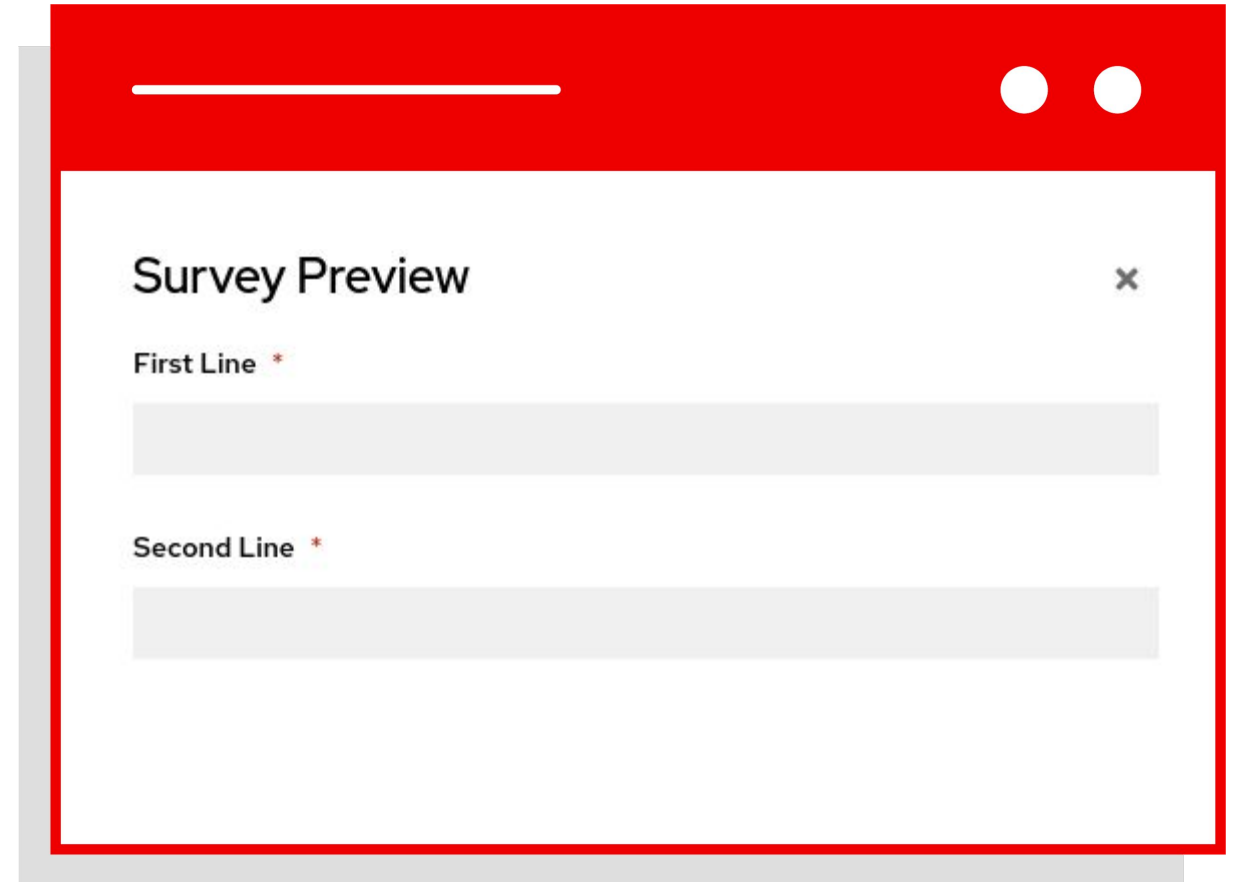


Red Hat
Ansible Automation
Platform

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.



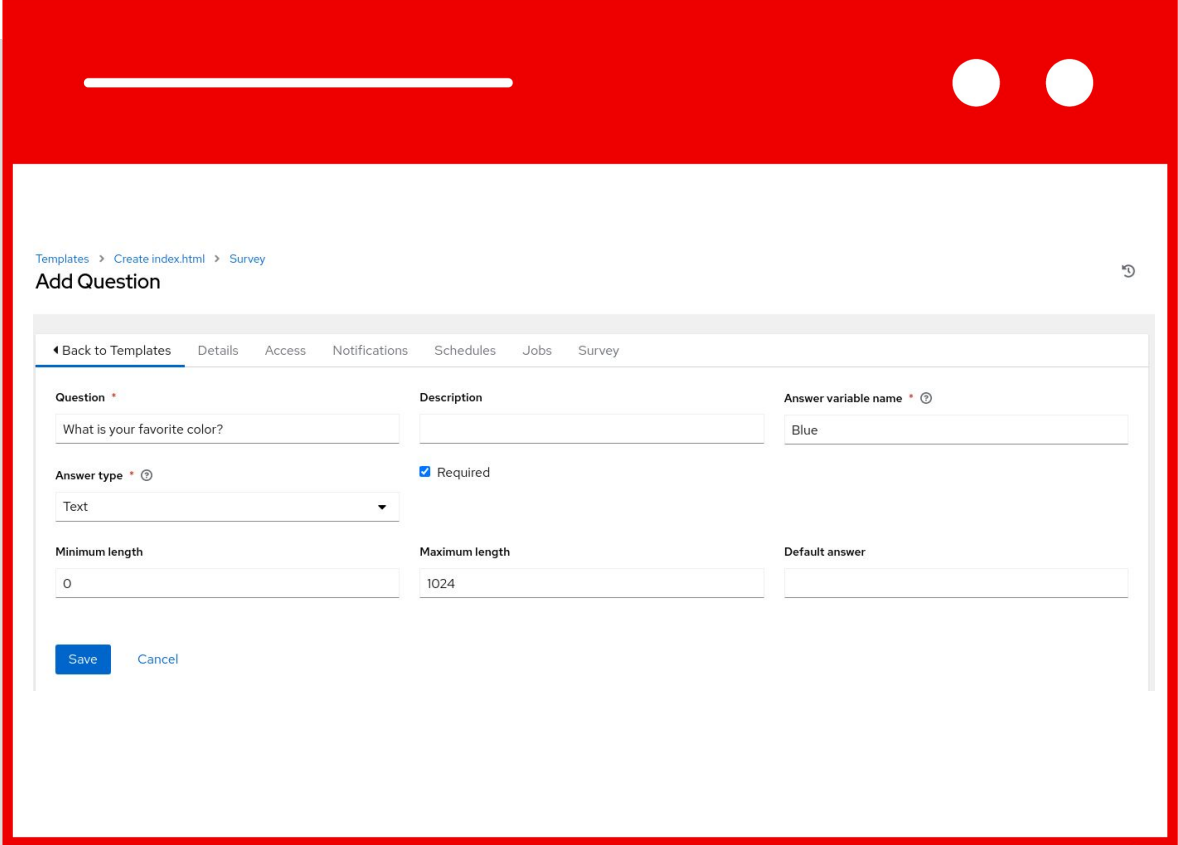
The image shows a screenshot of a web browser window with a red header bar. The browser title bar contains two white circles. The main content area is white and features a 'Survey Preview' window with a close button (an 'x' icon) in the top right corner. The survey form consists of two text input fields. The first field is labeled 'First Line *' and the second field is labeled 'Second Line *'. Both labels have a red asterisk indicating a required field. The input fields are currently empty and have a light gray background.

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.



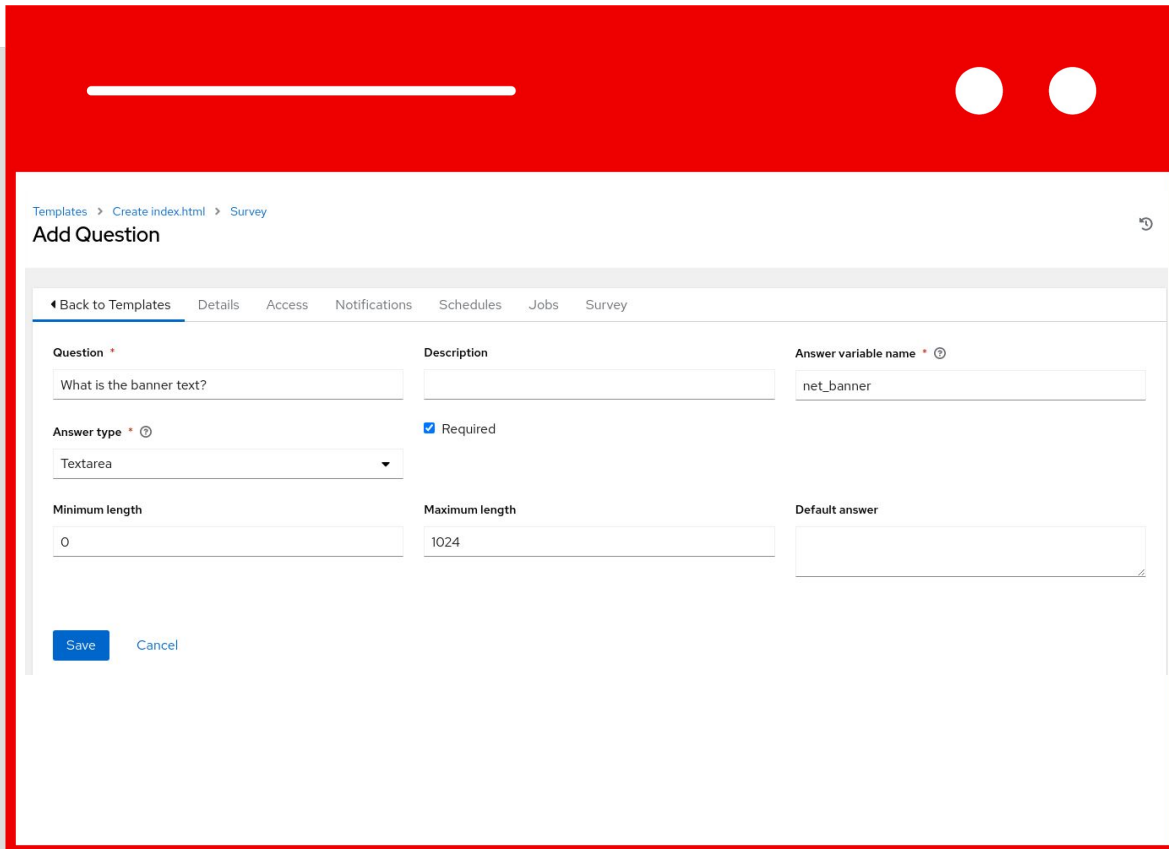
The screenshot shows a web interface for adding a question to a survey. The interface is titled "Add Question" and is part of a "Survey" menu. The form contains the following fields and options:

- Question:** A text input field containing "What is your favorite color?".
- Description:** An empty text input field.
- Answer variable name:** A text input field containing "Blue".
- Answer type:** A dropdown menu set to "Text".
- Required:** A checked checkbox.
- Minimum length:** A text input field containing "0".
- Maximum length:** A text input field containing "1024".
- Default answer:** An empty text input field.

At the bottom of the form, there are two buttons: "Save" (in blue) and "Cancel".

Creating a Survey (2/2)

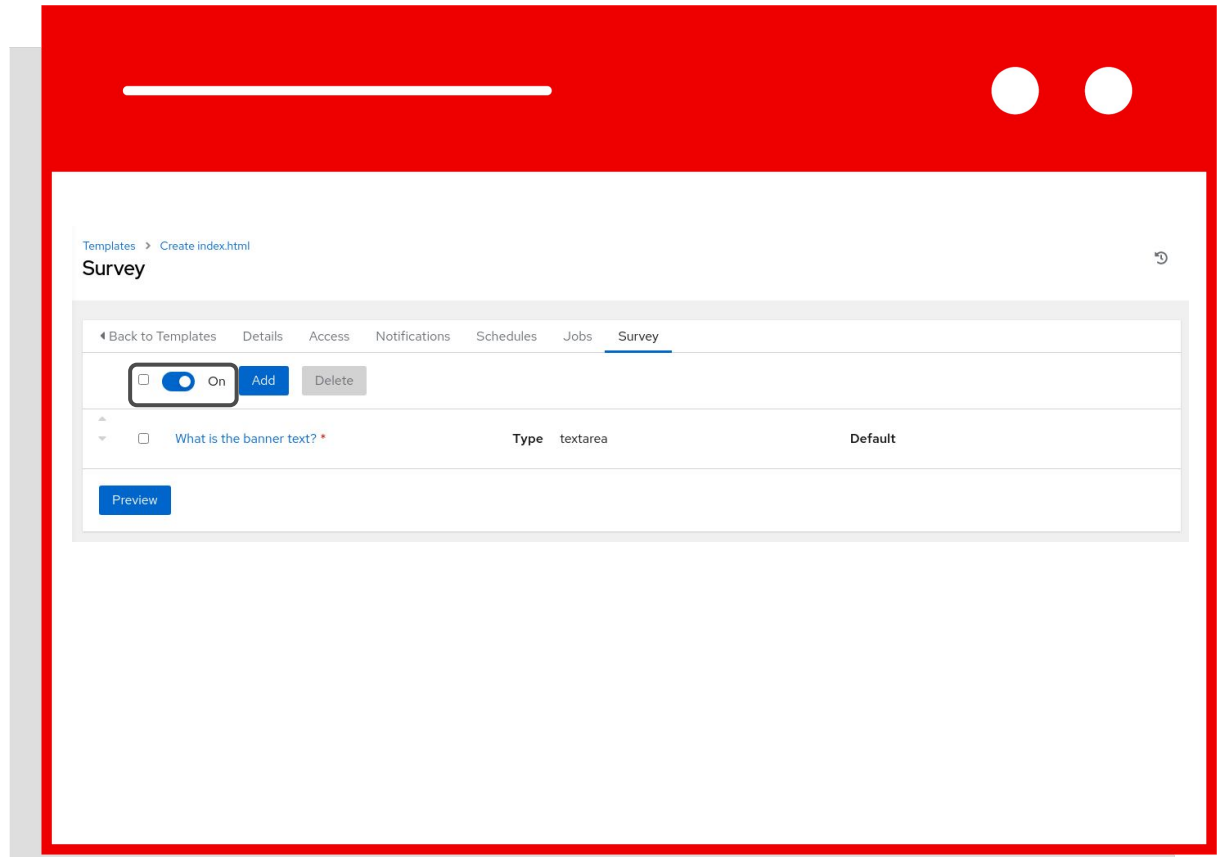
The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.



The screenshot shows the 'Add Question' form in the Ansible Tower interface. The form is titled 'Add Question' and is located under the 'Survey' tab. It contains several input fields and a 'Save' button. The fields are:

- Question ***: A text input field containing 'What is the banner text?'.
- Description**: An empty text input field.
- Answer variable name ***: A text input field containing 'net_banner'.
- Answer type ***: A dropdown menu set to 'Textarea'.
- Required**: A checked checkbox.
- Minimum length**: A text input field containing '0'.
- Maximum length**: A text input field containing '1024'.
- Default answer**: An empty text area.

At the bottom left, there are 'Save' and 'Cancel' buttons.



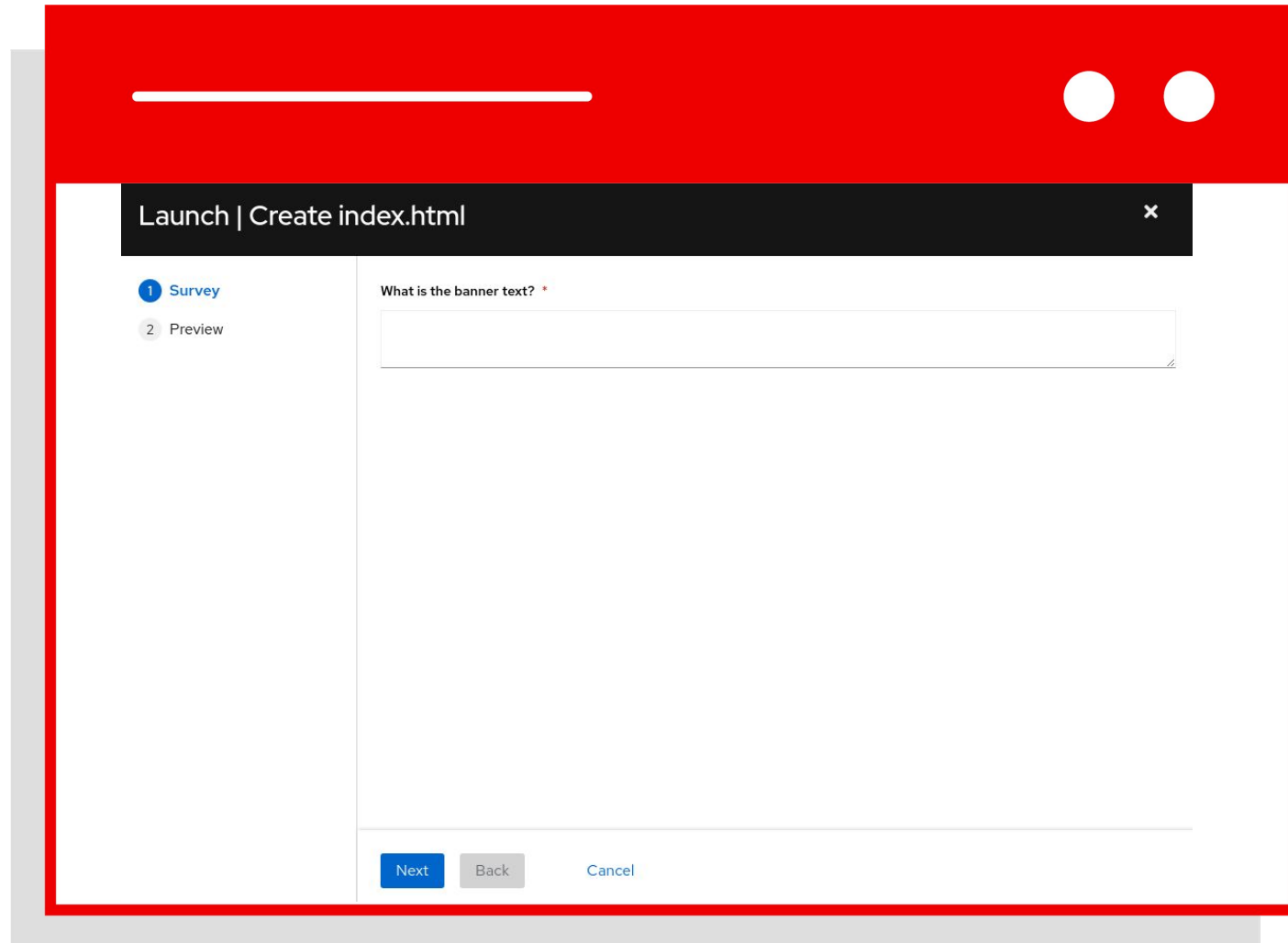
The screenshot shows the 'Survey' configuration page in the Ansible Tower interface. The page is titled 'Survey' and is located under the 'Survey' tab. It contains a list of questions and a 'Preview' button. The list of questions is:

Question	Type	Default
<input type="checkbox"/> <input checked="" type="checkbox"/> On Add Delete		
<input type="checkbox"/> What is the banner text? *	textarea	Default

At the bottom left, there is a 'Preview' button.

Using a Survey

When launching a job, the user will now be prompted with the Survey. The user can be required to fill out the Survey before the Job Template will execute.



The screenshot shows a dialog box with a red title bar and a dark header. The header contains the text "Launch | Create index.html" and a close button (X). The main content area is divided into two sections. On the left, there is a vertical list of steps: "1 Survey" (highlighted with a blue circle) and "2 Preview". On the right, there is a text input field with the label "What is the banner text? *" and a small asterisk indicating a required field. Below the input field, there are three buttons: "Next" (blue), "Back" (grey), and "Cancel" (grey).



Red Hat Ansible Automation Platform

Lab Time

Complete exercise 2.4 now in your lab environment

Exercise 2.5

Topics Covered:

- Role based access control

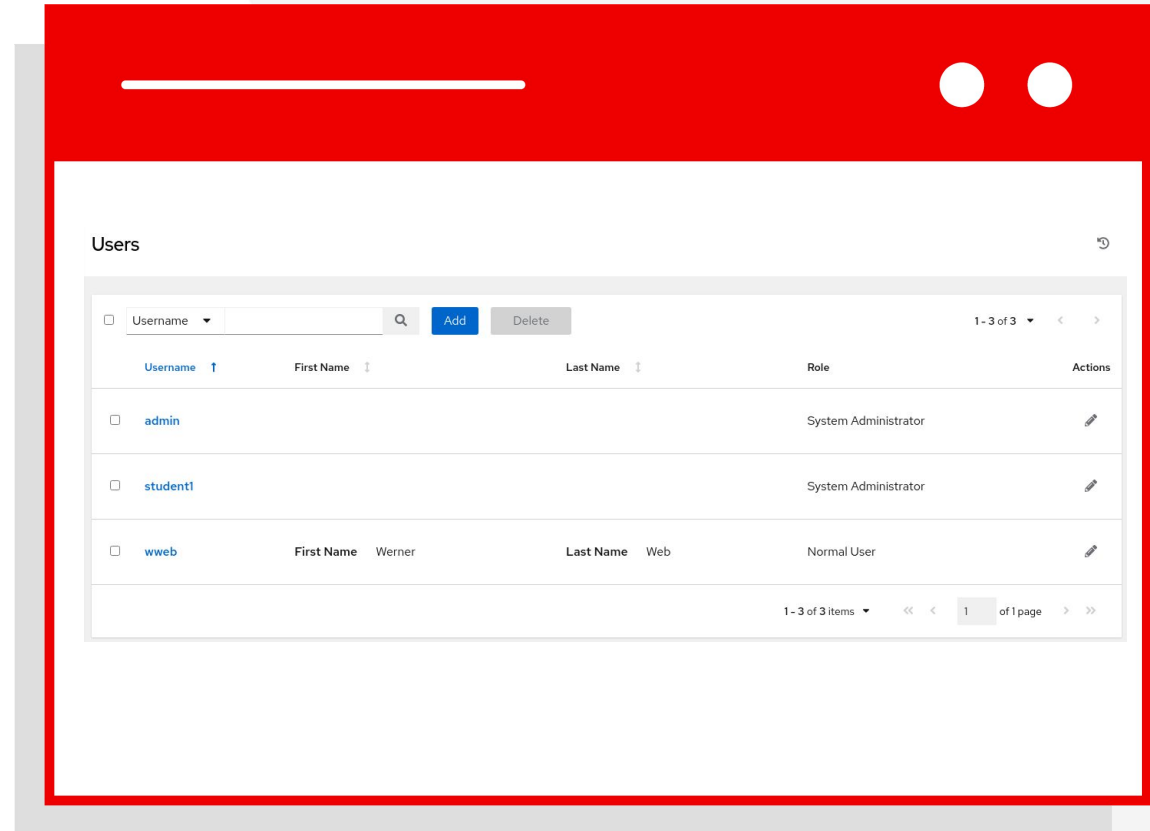


Red Hat
Ansible Automation
Platform

Role-based access control

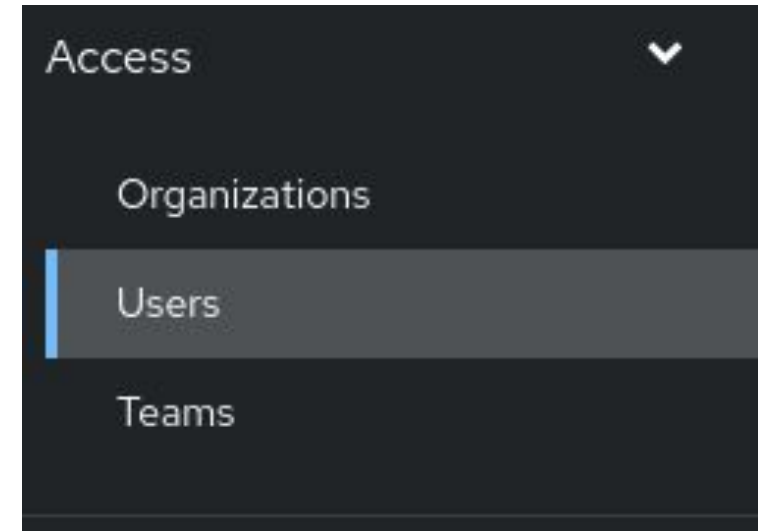
How to manage access

- ▶ 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.



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

Exercise 2.6

Topics Covered:

- Workflows

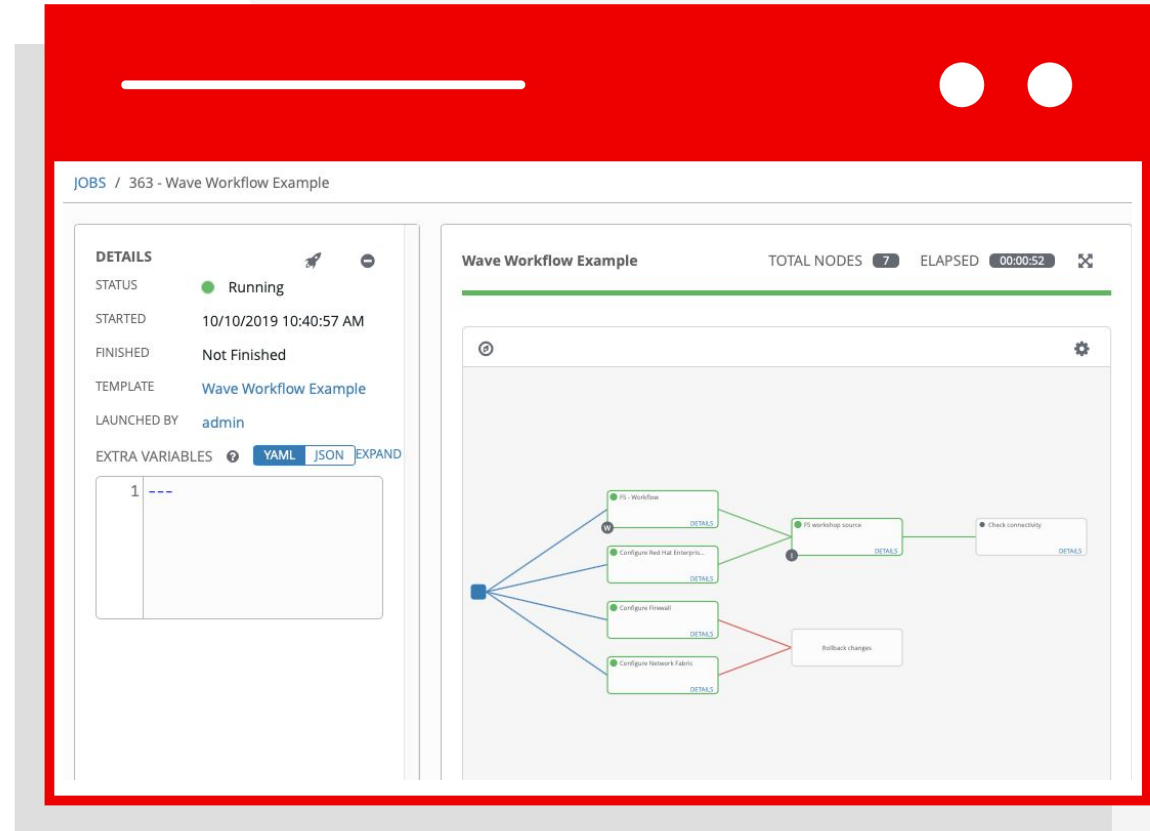


Red Hat
Ansible Automation
Platform

Workflows

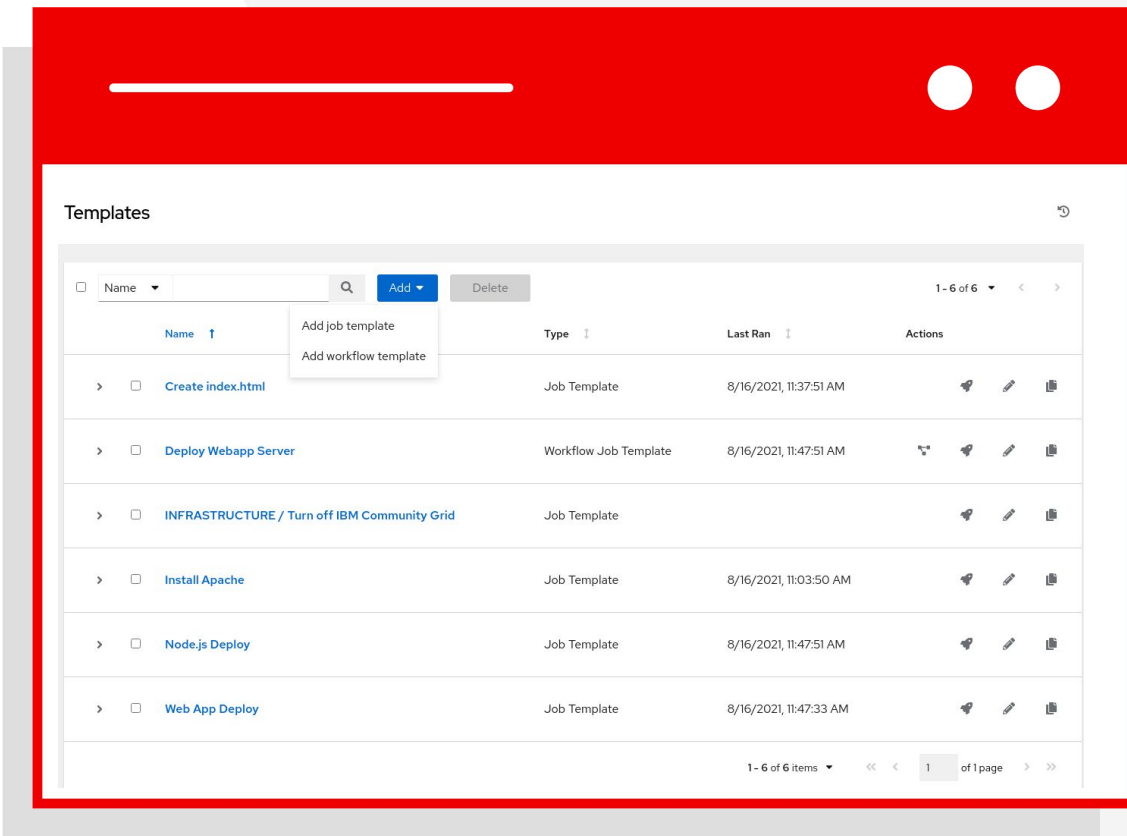
Combine automation to create something bigger

- ▶ 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.



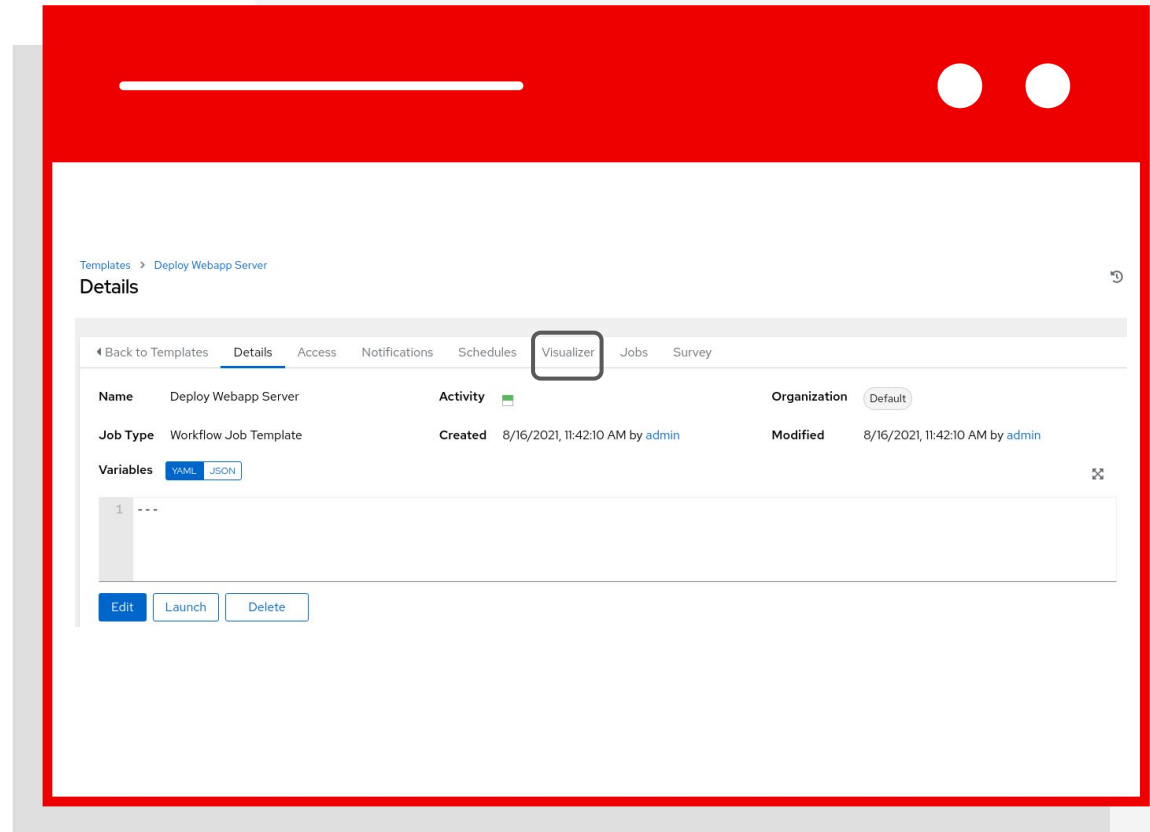
Adding a New Template

- ▶ To add a new **Workflow** click on the **Add** button.
This time select the **Add workflow template**



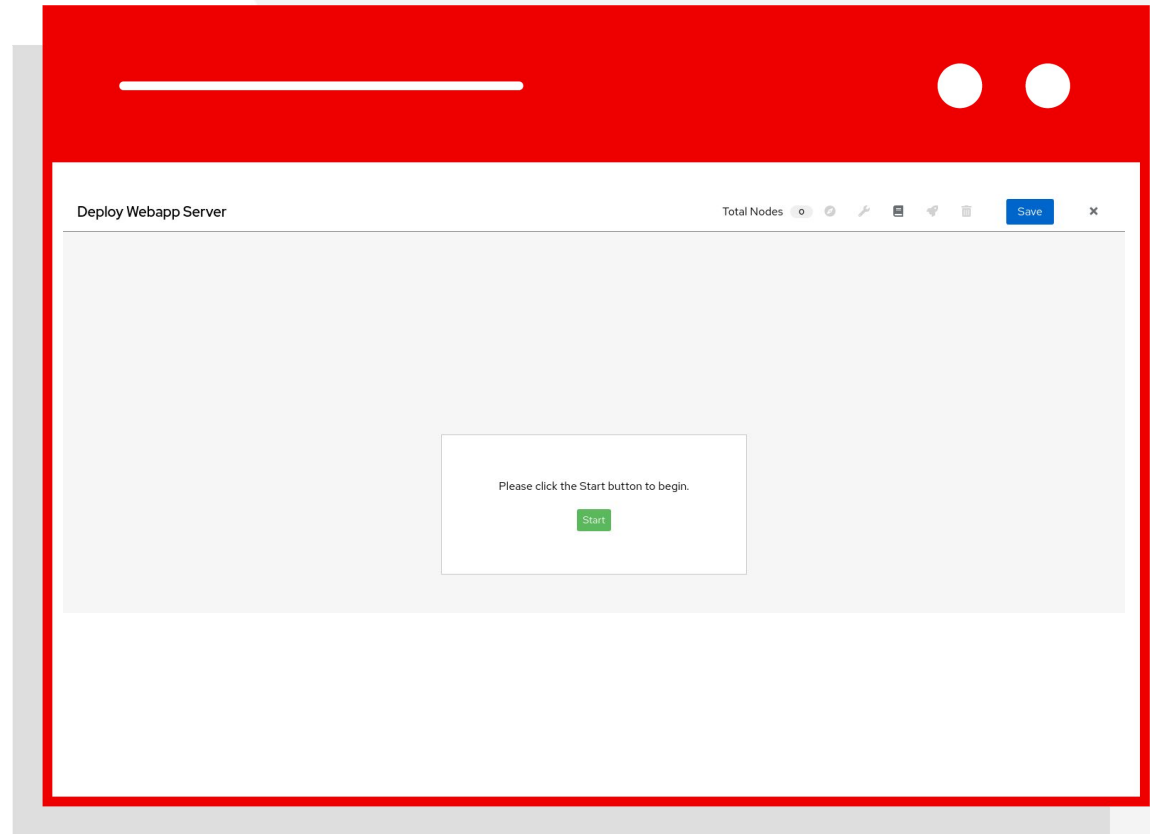
Creating the Workflow

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



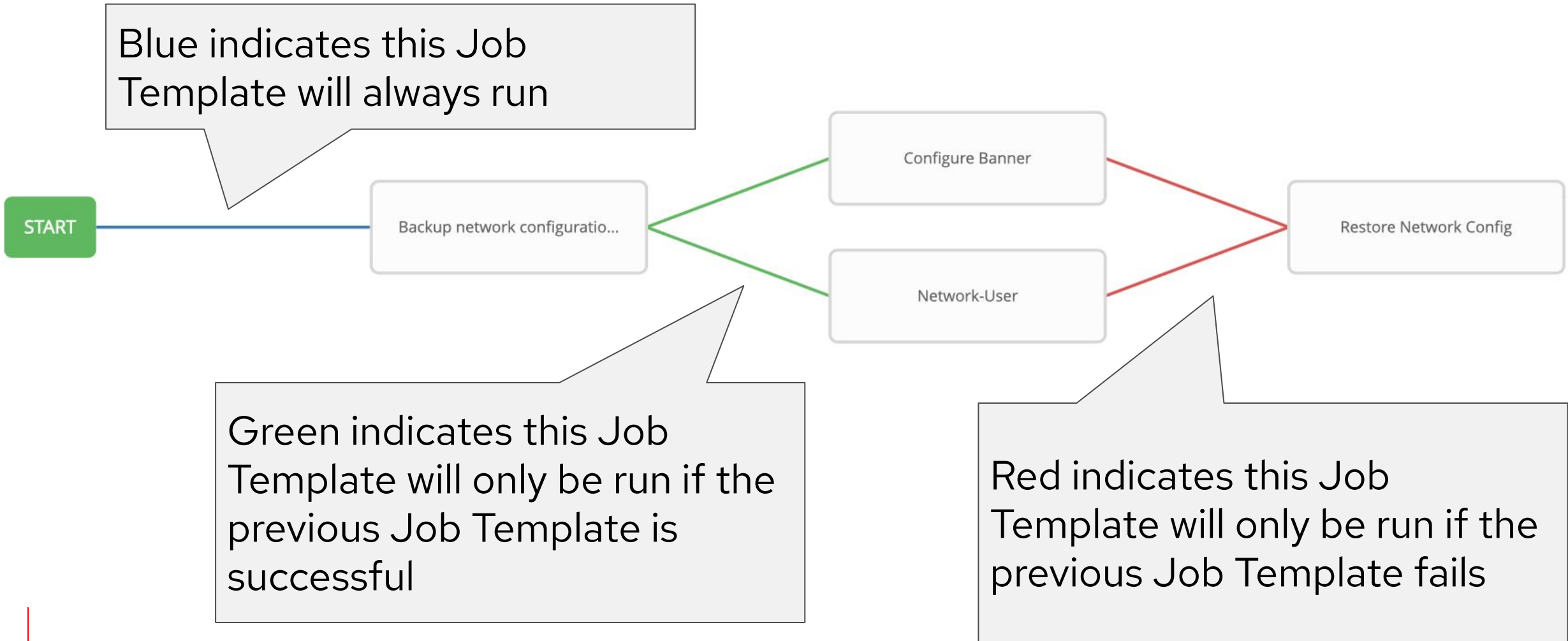
Workflow Visualizer

- ▶ 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

Exercise 2.7

Topics Covered:

- Wrap-up



Red Hat
Ansible Automation
Platform



Red Hat Ansible Automation Platform

Lab Time

Complete exercise 2.7 now in your lab environment



Red Hat
Ansible Automation
Platform

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://www.linkedin.com/company/red-hat)

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

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

 twitter.com/ansible

 github.com/ansible