
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

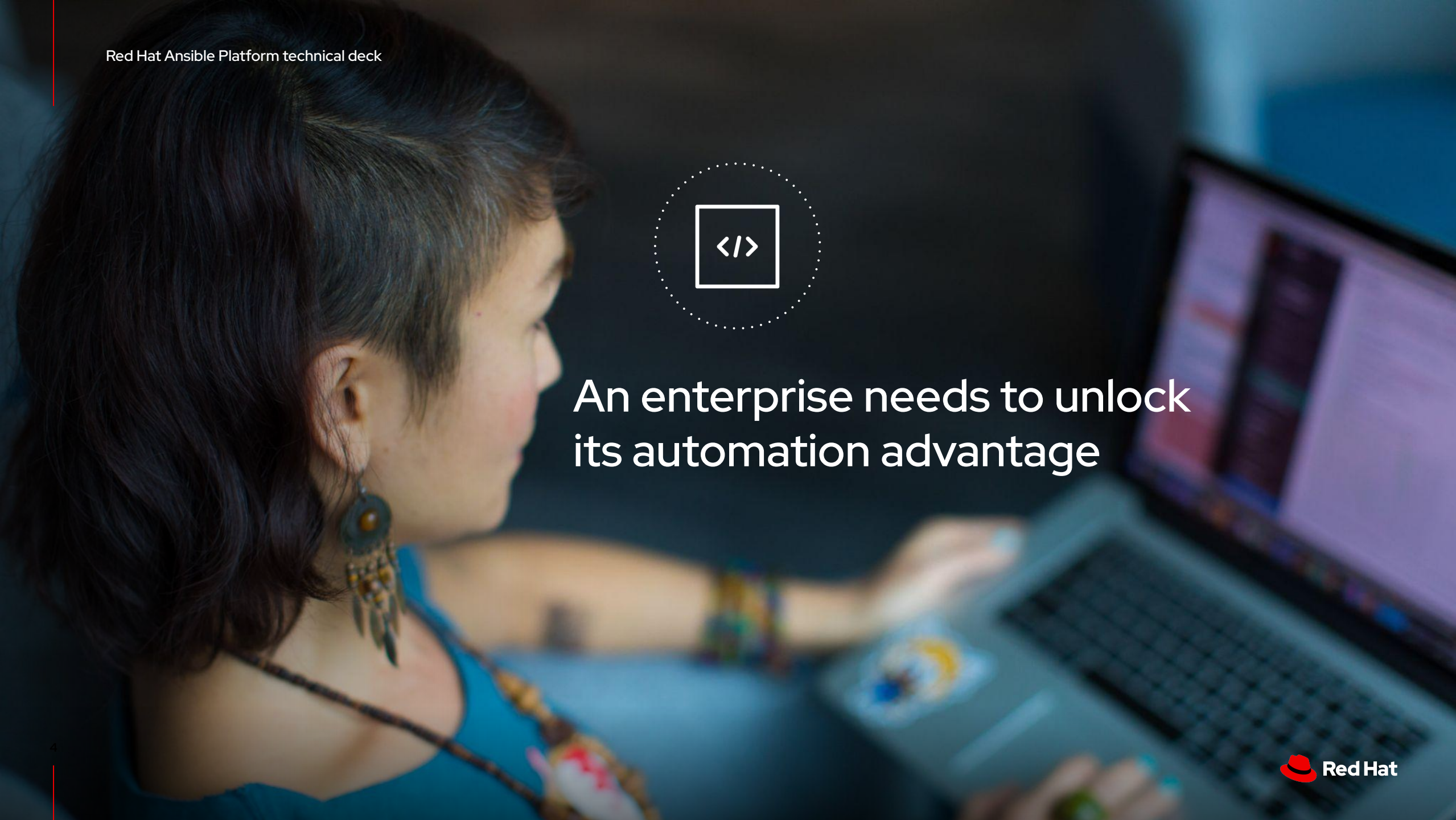
- Intro to Ansible Automation Platform
- How Ansible Works
- Understanding Modules, Tasks, Playbooks
- Leveraging Variables & Templates for Flexibility
- Automation Controller: It’s Role in the Automation Ecosystem
- Automation Controller Basics & Key Concepts
- Core Features of Automation Controller: RBAC, Workflows

Introduction

Topics Covered:

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

Red Hat



An enterprise needs to unlock its automation advantage

Red Hat

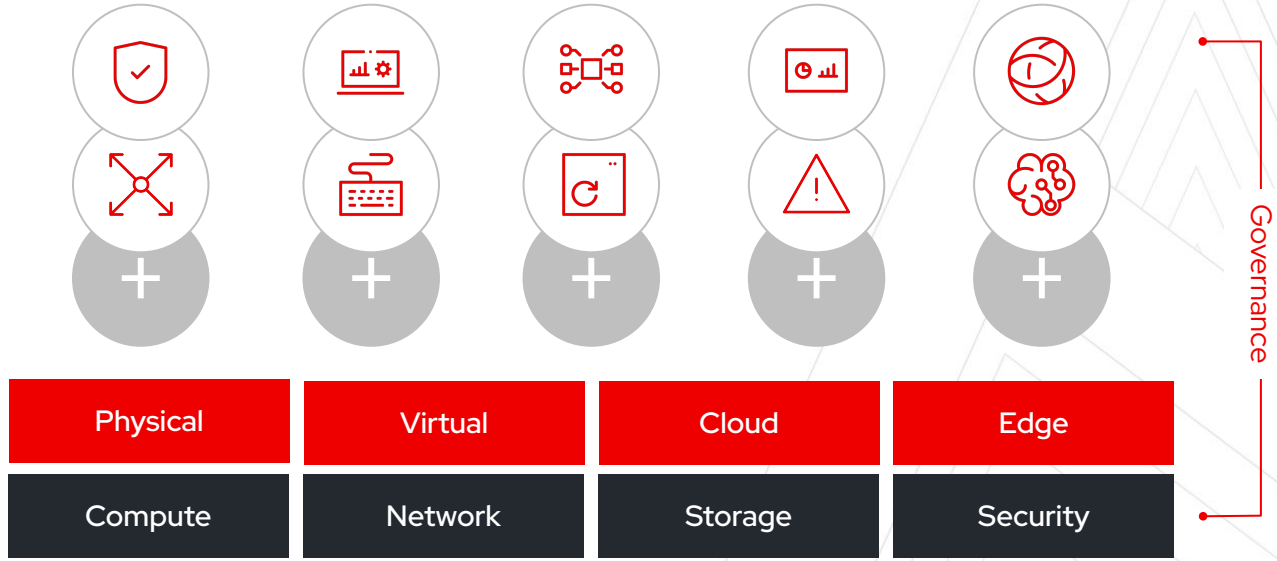
Many organizations share the same challenge.

Differing skills, roles, and responsibilities

Growing number of use cases

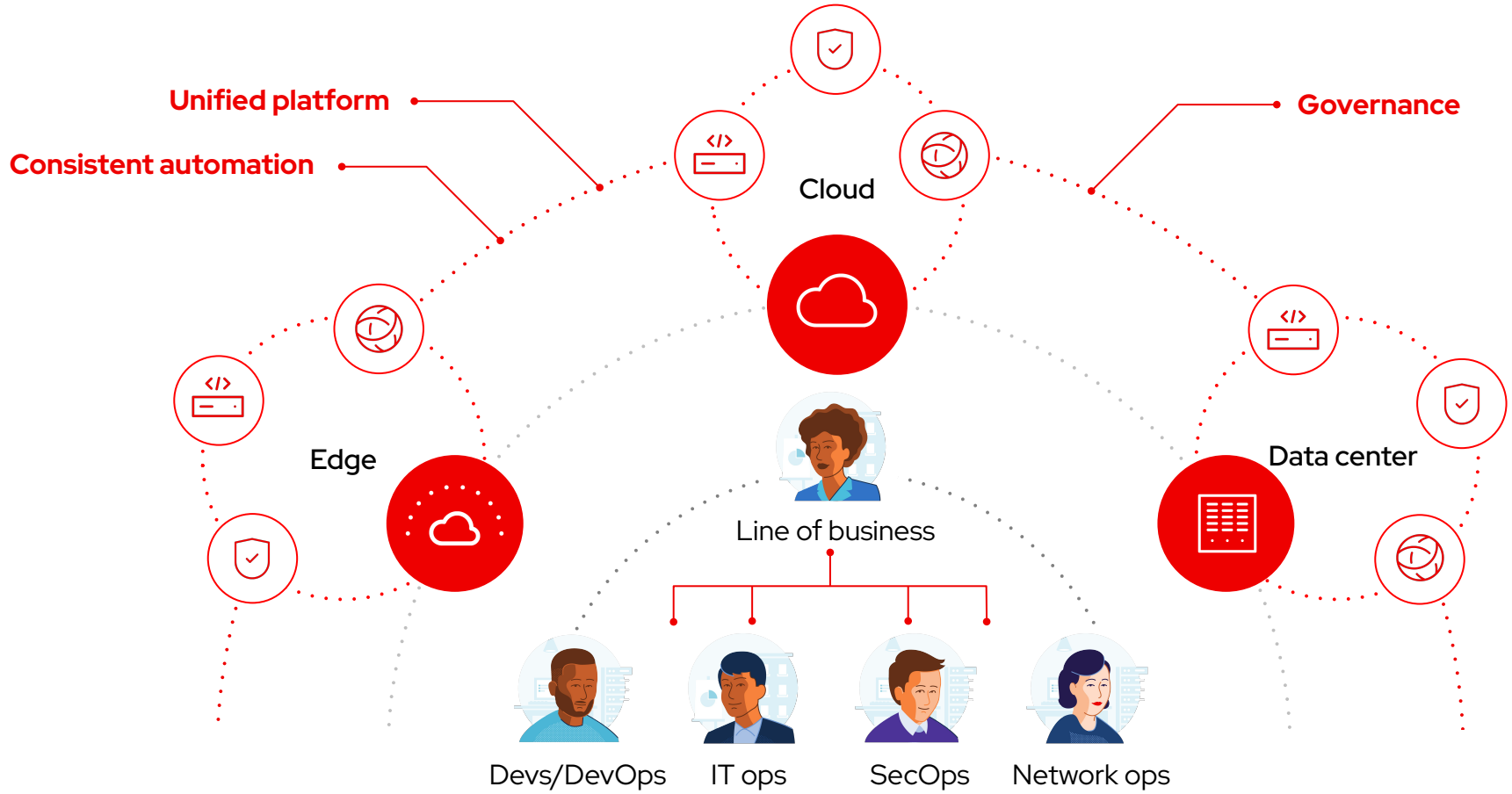
Multiple platforms and locations

Across multiple domains



Red Hat Ansible Automation Platform

The solution? Break down the silos.



Why Ansible Automation Platform?

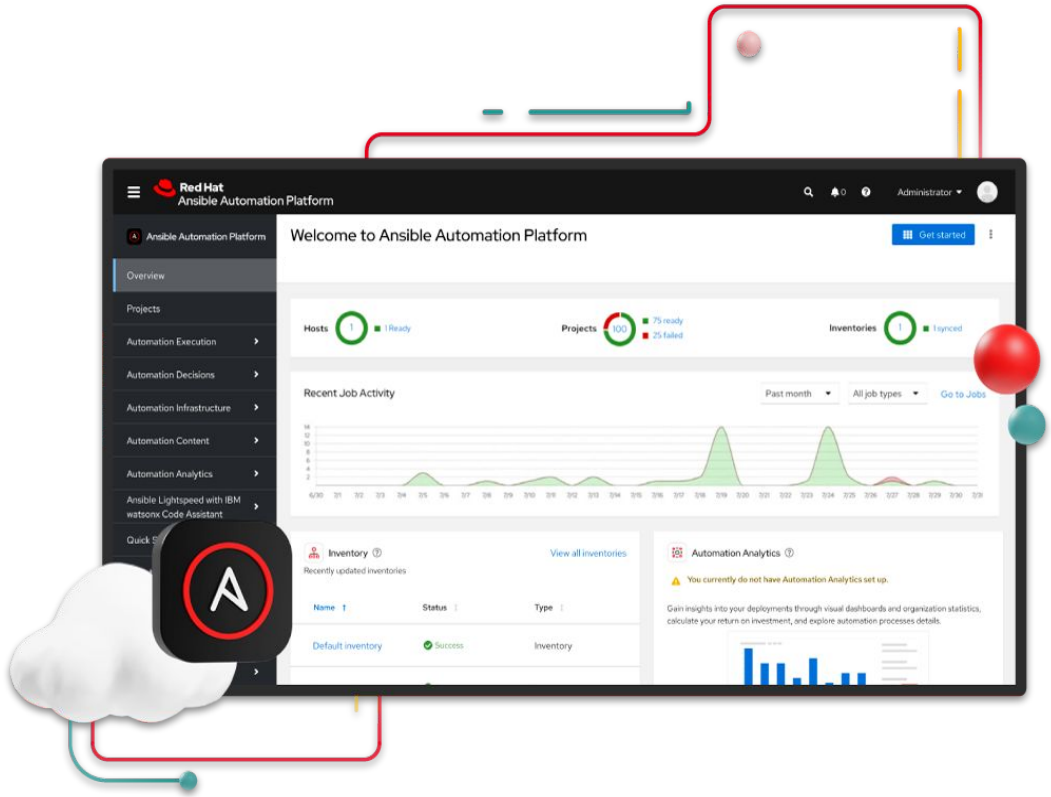
To unlock your automation advantage

Red Hat Ansible is becoming increasingly mission-critical for customers who rely on automation to bridge skills gaps, tame operational complexity across the enterprise, and mitigate costly tool sprawl.

Our goal with Red Hat Ansible Automation Platform 2.5 is to make it easier for every customer to fully unlock the potential of automation to transform IT - and deliver strategic advantages to the business.

This latest release is engineered to help our customers:

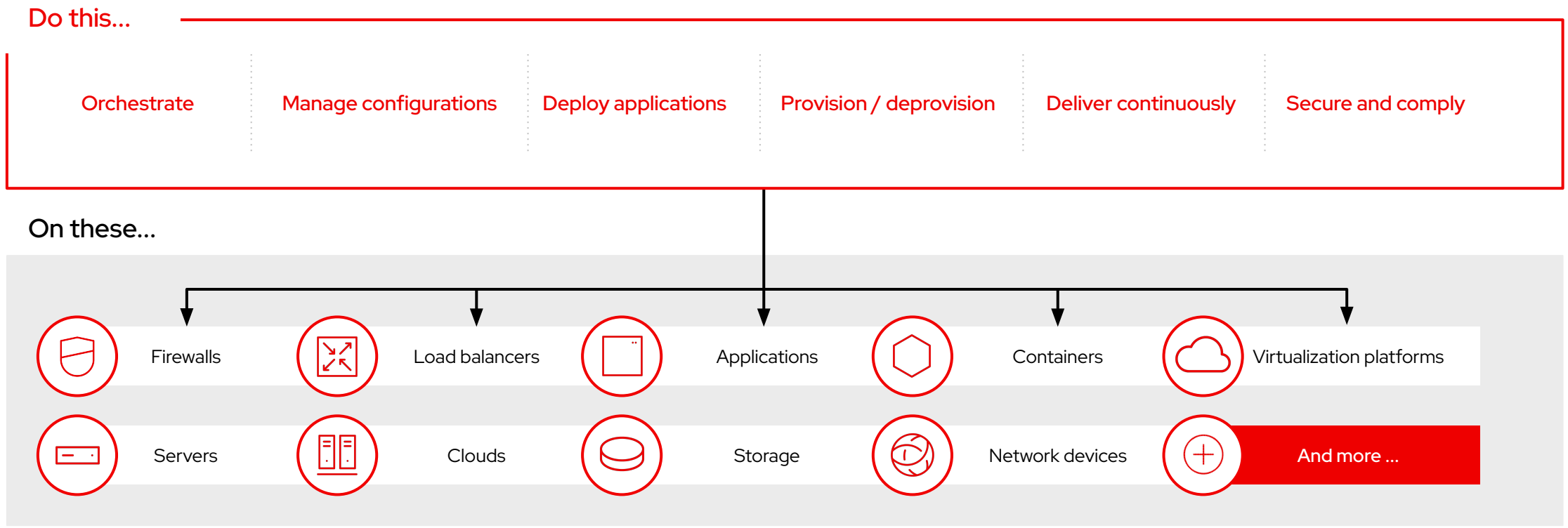
- **Accelerate automation adoption at scale,** with a reimagined automation platform experience, new features for enhanced usability, and tools for more effective collaboration and coordination.
- **Empower automation engineers,** with integrated developer tooling and generative AI capabilities designed to bolster ease and efficiency for a range of skill sets and experience across the entire automation creation lifecycle.
- **Orchestrate across the enterprise,** with event-driven automation capabilities that make intelligence from other tools more actionable, along with a robust ecosystem of integrations that put true end-to-end automation within reach.



Red Hat

Automate the deployment and management of automation

Your entire IT footprint



Red Hat

Supported and certified content you can trust.

170+

Certified and Validated Content Collections

70+

Certified technology partners

Infrastructure | Cloud | Network | Security | Edge

Red Hat Ansible Automation Platform

The flexibility to scale, wherever that may be.

Red Hat
Ansible Automation
Platform

A platform for the entire automation team.

Architecture

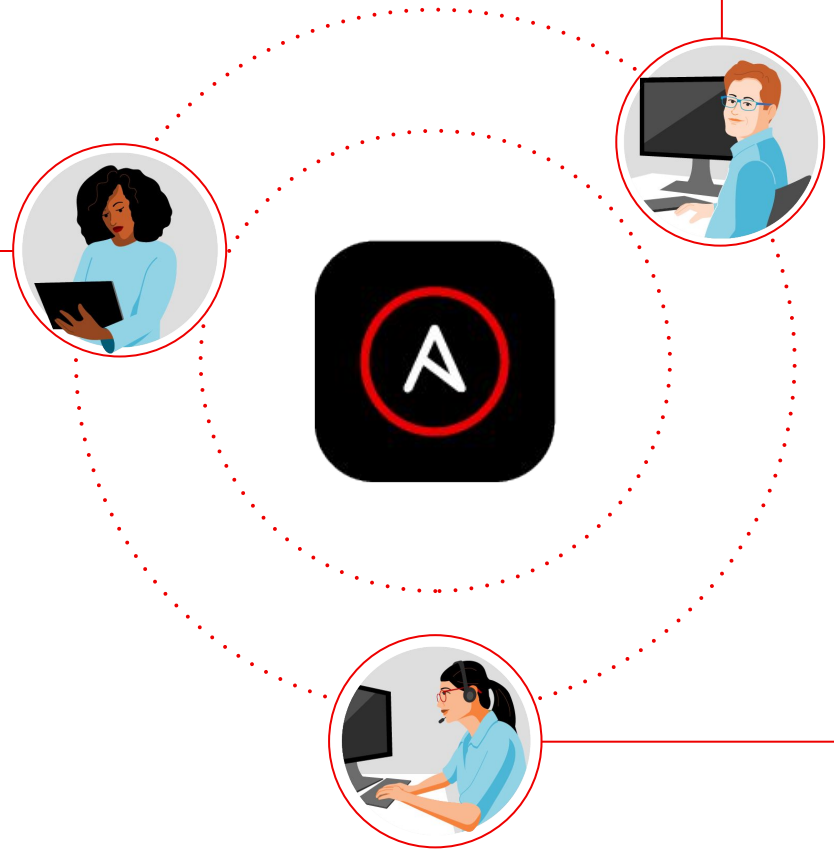
Flexible container-native architecture

Real-time analytics and reporting

Event-Driven Ansible

Containerized Installation

Available on all major cloud providers



Content creation

Ansible developer tools package

Integrated generative AI

Build automation communities

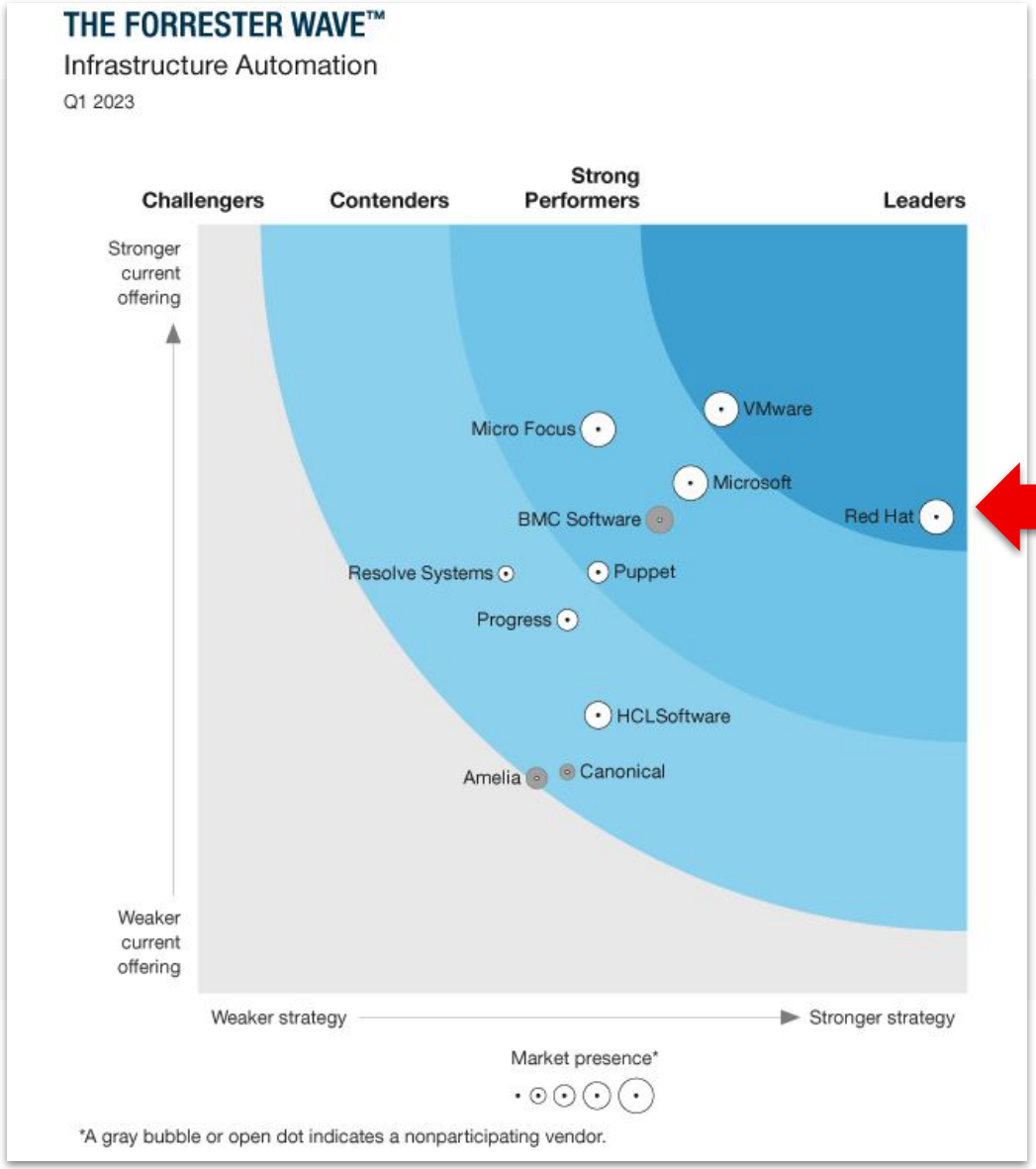
Large ecosystem of certified automation

Operations

Unified UI provides cohesive platform experience

Enterprise features: WebUI, API, role-based access control (RBAC), auditing and workflows for managing at scale

Red Hat is a *leader* in the 2023 Forrester Wave™: Infrastructure Automation



Vendor Profiles

Our analysis uncovered the following strengths and weaknesses of individual vendors.

Leaders

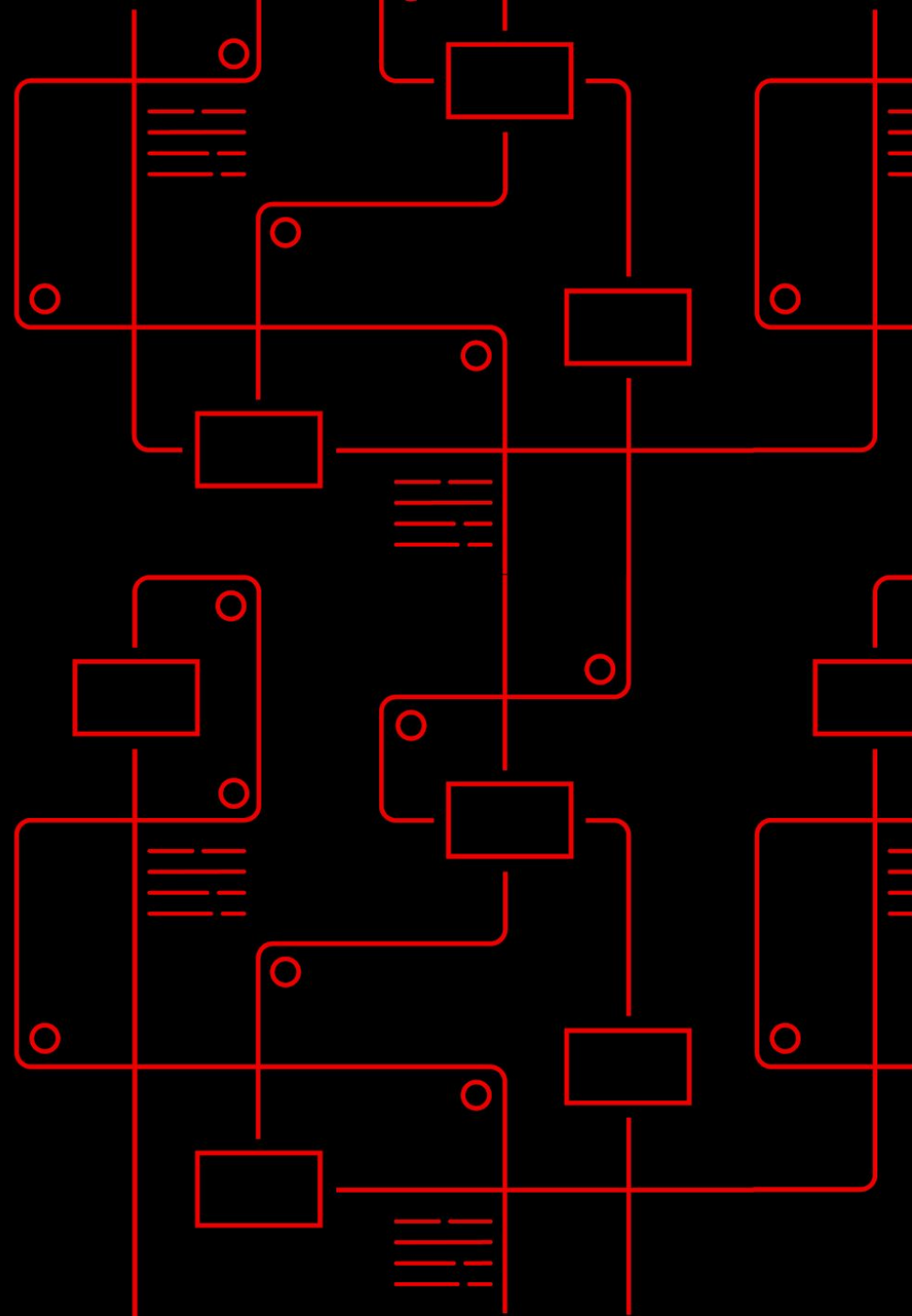
- **Red Hat leverages its strong open source community to power innovation.** Red Hat is well-known for commercializing open source software for enterprises. It adds capabilities to upstream Ansible via its Ansible Automation Platform; this solution includes Automation Hub, Automation Services Catalog, and Insights for Ansible. Red Hat sets the pace of the market by addressing operational challenges, skill gaps, and budgetary pressures. Its strength lies in its community, which has led to solid partnerships and supporting services. Red Hat capitalizes on this ecosystem by adopting and embracing the work of contributors. Key upcoming features include trusted automation supply chain, Event-Driven Ansible, and AI-led automation through Project Wisdom.

  Ansible has strengths in configuration management, integration with configuration management database (CMDB), analytics, and community support. It can clearly handle scale: Large global systems integrators lean on it to deliver managed services. Ansible's minimal support for storage contrasts with its strong server and network capabilities; it also lacks multilayered service blueprints, infrastructure templates, and complex orchestration (handling incidents with automated resolutions or remediation). Reference customers find the upgrade path and process troublesome despite their best efforts. They also want more flexibility and better capabilities for business continuity and disaster recovery. Red Hat is a great fit for firms seeking consolidated automation across many infrastructure technologies and vendors.

Source: Forrester Research. "The Forrester Wave™: Infrastructure automation Q1 2023," 2023.

Red Hat
Ansible Automation Platform

Section 1
The Ansible Basics



Exercise 1.1

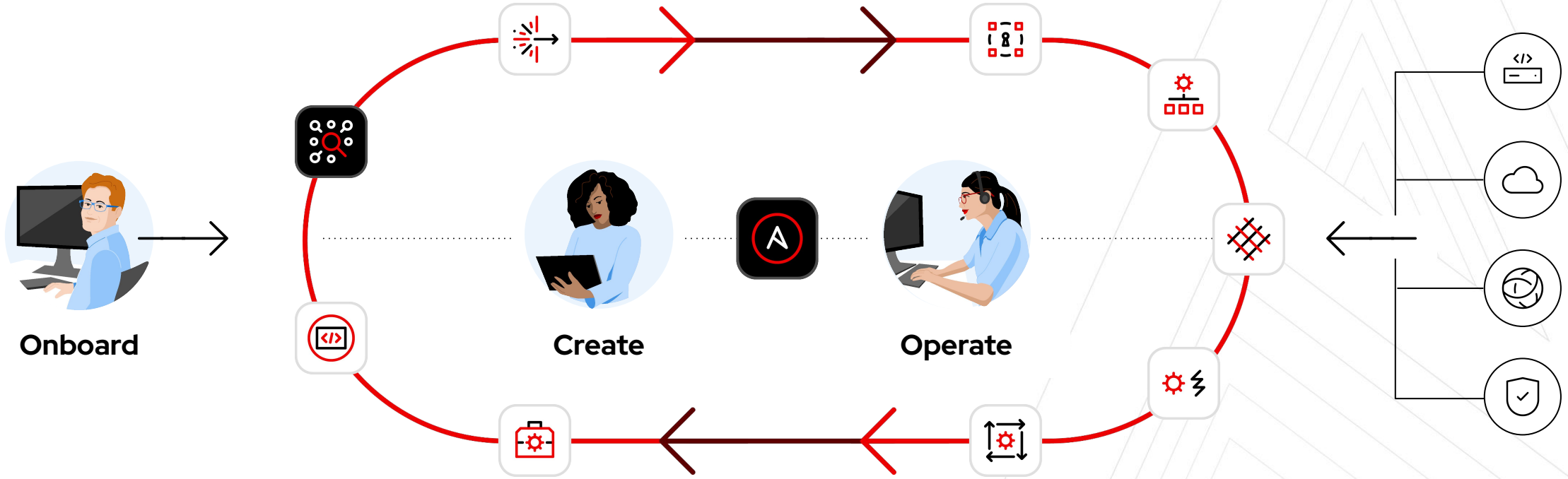
Topics Covered:

- Understanding the Ansible Content Lifecycle
- Ansible Development Tools
- What makes up an Ansible Playbook?

Red Hat

Automation Lifecycle: Development to Management.

Configuration as Code · Ansible plug-ins for Red Hat Developer Hub · Ansible Lightspeed · Automation hub · Automation Platform UI · Automation mesh



Ansible development workspaces · Ansible development tools · Event-Driven Ansible · Platform Operators

Red Hat

Ansible development tools. Streamlining creation

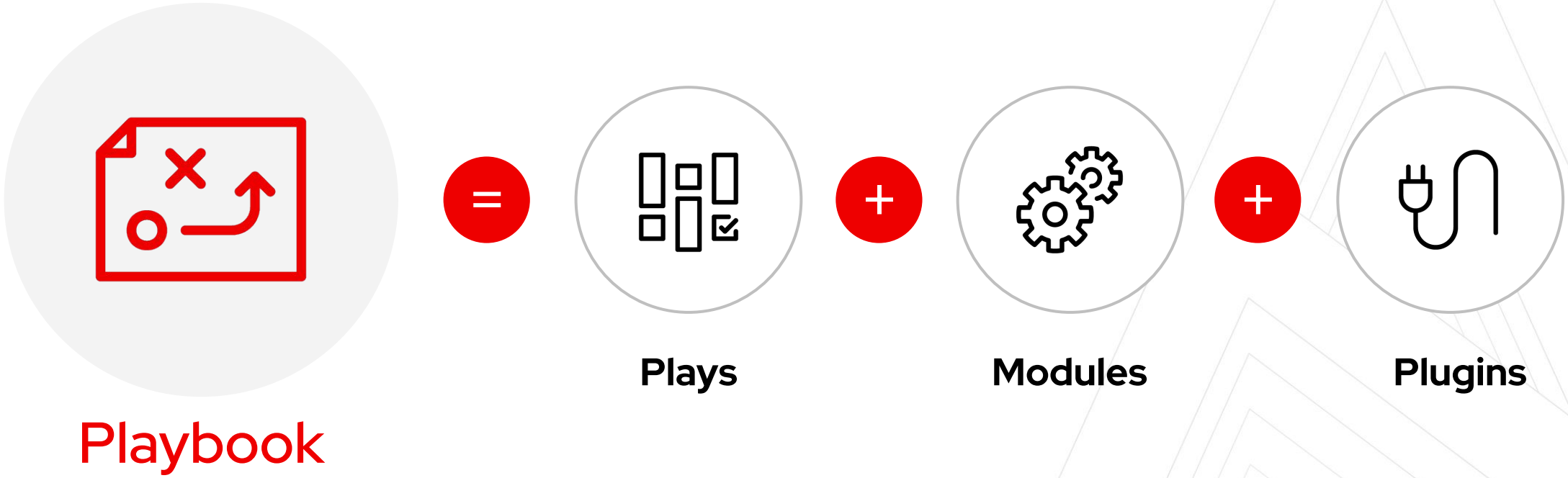
Supported, enterprise-grade components to enable creating, testing, and deployment of Ansible

Create	Test	Deploy
ansible-creator Scaffold projects and content	**Ansible Molecule** Functional test runner for collections, playbooks, and roles	**ansible-builder** Build Ansible execution environments
ansible-navigator Develop and troubleshoot with execution environments	**ansible-lint** Identify and correct stylistic errors and anti-patterns	**ansible-sign** Sign and verify Ansible content
Enhanced VS Code integration Native graphical user interface to Ansible development tools	**pytest-ansible** Added functions for module and plugin Python code unit testing	
Ansible development environment Build and manage virtual environments for content development	**tox-ansible** Test collections under different Python interpreters and Ansible core versions	

Ansible playbooks

```
---
- name: Install and start apache
  hosts: web
  become: true

  tasks:
    - name: Ensure the httpd package is installed
       ansible.builtin.package:
         name: httpd
         state: present

    - name: Create the index.html file
      ansible.builtin.template:
        src: files/index.html
        dest: /var/www/html/

    - name: Start the httpd service if needed
      ansible.builtin.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

```
---
- name: Ensure the httpd package is installed
  hosts: web
  become: true
```

Ansible modules. The “tools in the toolkit”.

What are they?

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

Language

- Usually created in Python, or Powershell for Windows setups, but can be developed in any language

```
- name: Create the index.html file
  ansible.builtin.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

```yaml
---
- name: Example Playbook Using json_query Filter
  hosts: localhost
  gather_facts: no

  vars:
    complex_data:
      users:
        - name: "Alice"
          age: 25
        - name: "Bob"
          age: 30
        - name: "Charlie"
          age: 35

  tasks:
    - name: Extract names of all users
      ansible.builtin.debug:
        msg: "{{ complex_data | community.general.json_query('users[*].name') }}"
```

Ansible Inventory. The systems that a playbook runs against.

What are they?

- List of systems in your infrastructure that automation is executed against

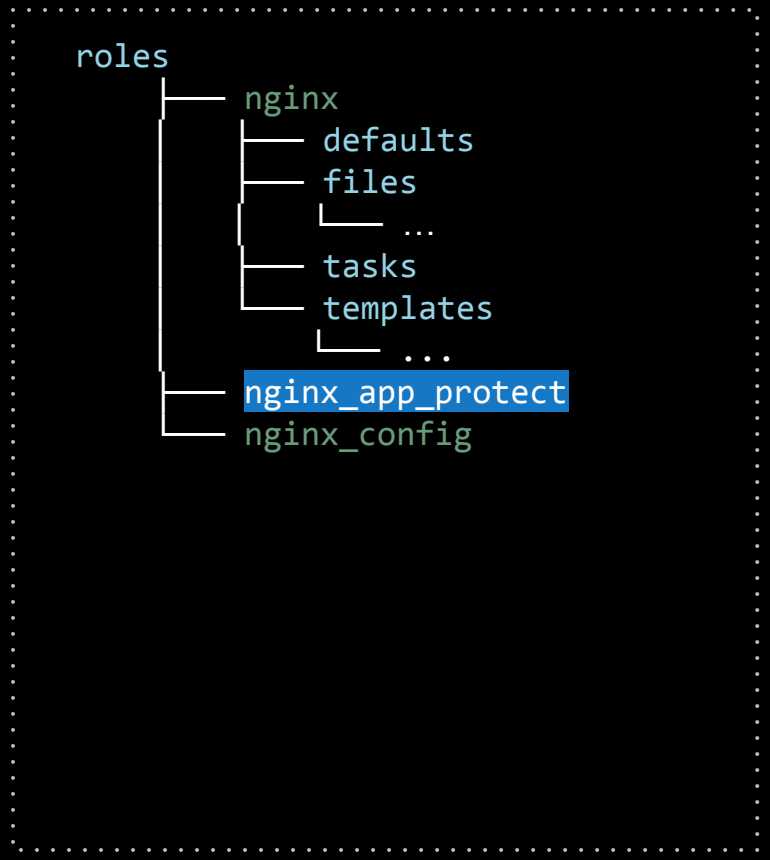
```
[web]
webserver1.example.com
webserver2.example.com

[db]
dbserver1.example.com

[switches]
leaf01.internal.com
leaf02.internal.com
```

Ansible Roles. Reusable automation actions.

What are they?

- Group 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
```



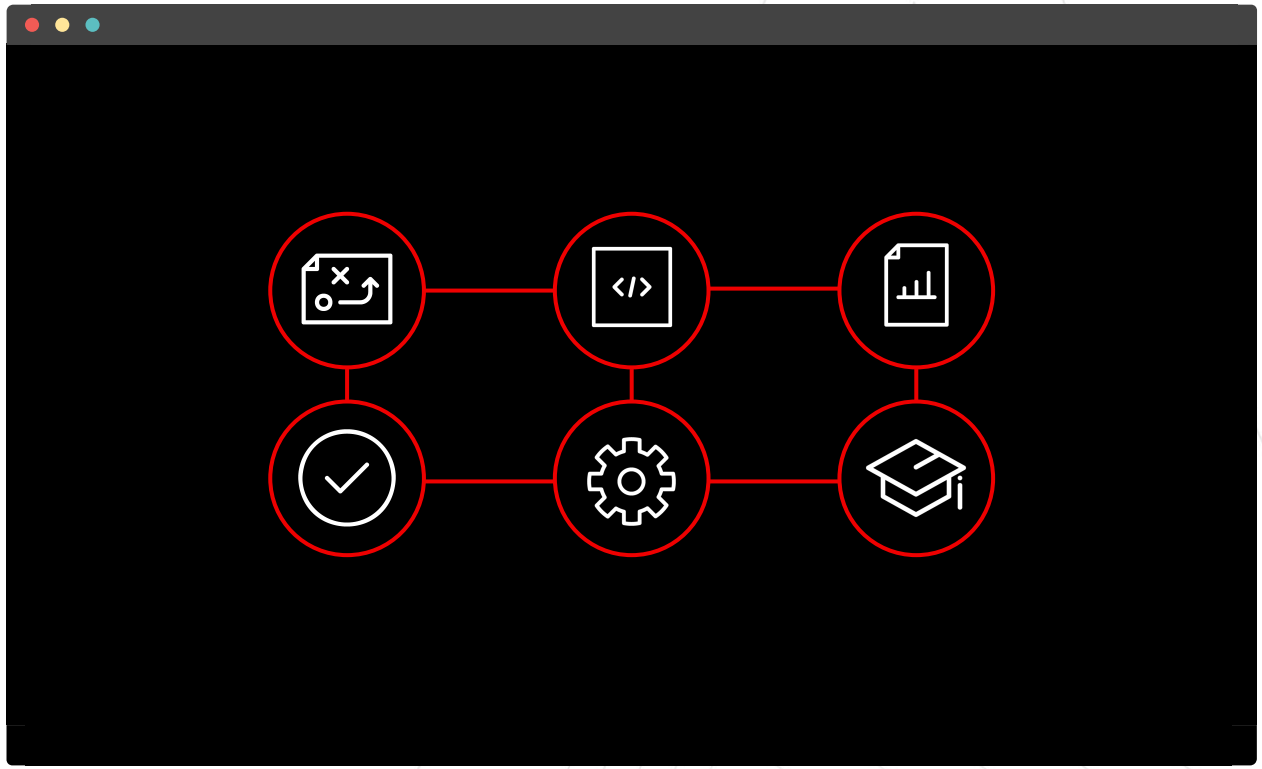
deploy-nginx.yml

```
---
- name: Install NGINX Plus
  hosts: all
  tasks:

    - name: Install NGINX App Protect
      ansible.builtin.include_role:
        name: nginx_app_protect
      vars:
        nginx_app_protect_setup_license: false
        nginx_app_protect_remove_license: false
        nginx_app_protect_install_signatures: false
```

Content Collections.
Simplified, consistent content delivery.

What are they?

- Contains automation content, including modules, multiple roles ,and playbooks
- Portable, reusable, and versioned enabling better collaboration



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

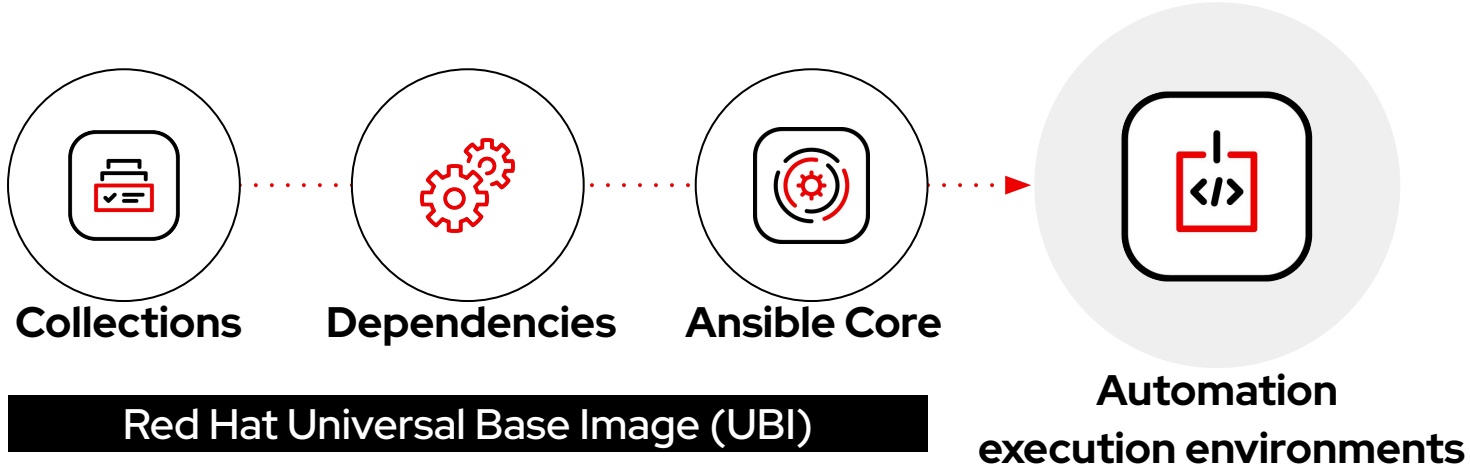
deploy-nginx.yml

```yaml
---
- name: Install NGINX Plus
  hosts: all
  tasks:
    - name: Install NGINX
      ansible.builtin.include_role:
        name: nginxinc.nginx
      vars:
        nginx_type: plus

    - name: Install NGINX App Protect
      ansible.builtin.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
```

Automation execution environments.
Reuse and scale automation content.

What are they?

- Containerized environments built using an Red Hat Universal Base Image that bundles essential collections, dependencies and Ansible core to ensure consistent and portable automation.
- Provide a reliable and repeatable way to run your automation consistently throughout your automation lifecycle.



Exercise 1.1

- Follow the steps to access your environment.
- Use the assigned IP address (the script contains only a placeholder IP).
- Choose your preferred command-line editor.
- New to editors? Don’t worry—check out our quick introduction!

Lab Time

Complete Exercise 1.1

Exercise 1.2

Topics Covered:

- Ansible Inventories
- Accessing Ansible docs
- Ansible Modules
- Getting help

Red Hat

Ansible Inventory. The systems that a playbook runs against.

What are they?

- List of systems in your infrastructure that automation is executed against

How do they work?

- Defines the systems that Ansible manages and targets for automation.
- Organizes hosts into groups (e.g., web servers, databases) for better management.
- Group variables apply settings across multiple systems efficiently.
- Host-specific variables allow detailed customization for individual systems.

```
[web]
webserver1.example.com
webserver2.example.com

[db]
dbserver1.example.com

[switches]
leaf01.internal.com
leaf02.internal.com
```

Ansible Inventory. The systems that a playbook runs against.

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
```

Ansible Inventory

```
[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
```

Ansible Inventory

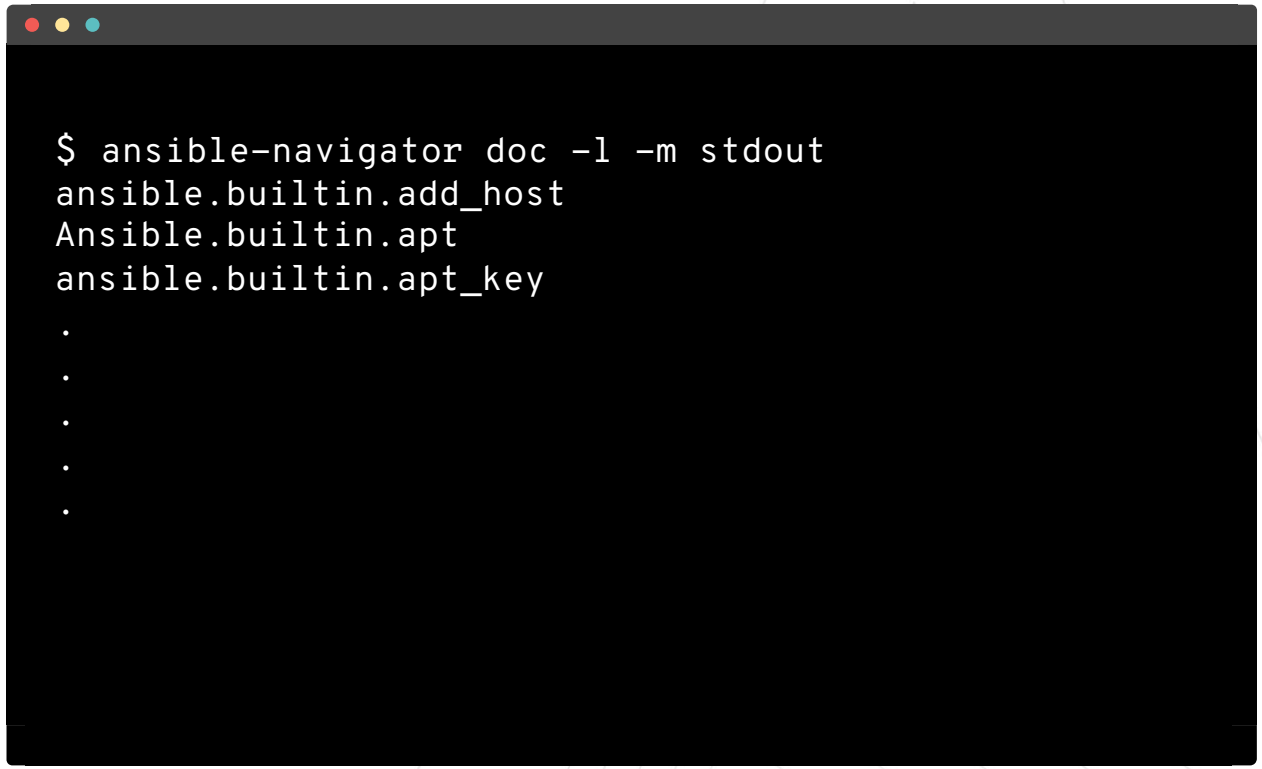
```
[app1srv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[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
```

Ansible Docs. Knowledge at your fingertips.

Documentation

- 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 Docs. Knowledge at your fingertips.

Documentation

- 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

> MODULE ansible.builtin.user
(/usr/lib/python3.12/site-packages/ansible/modules
/user.py)

Manage user accounts and user attributes.
For Windows targets, use the
ansible.windows.win_user module instead.
```

Lab Time

Complete Exercise 1.2

Exercise 1.3

Topics Covered:

- Ansible Playbooks Basics
- Running an Ansible Playbook

Red Hat

Ansible playbook

A play

```
---
- name: Install and start apache
  hosts: web
  become: true

  tasks:
    - name: Ensure the httpd package is installed
      ansible.builtin.package:
        name: httpd
        state: present

    - name: Create the index.html file
      ansible.builtin.template:
        src: files/index.html
        dest: /var/www/html/

    - name: Start the httpd service if needed
      ansible.builtin.service:
        name: httpd
        state: started
```

Ansible playbook

A task

```
---
- name: Install and start apache
  hosts: web
  become: true

  tasks:
    - name: Ensure the httpd package is installed
      ansible.builtin.package:
        name: httpd
        state: present

    - name: Create the index.html file
      ansible.builtin.template:
        src: files/index.html
        dest: /var/www/html/

    - name: Start the httpd service if needed
      ansible.builtin.service:
        name: httpd
        state: started
```

Ansible playbook

A module 

```
---
- name: Install and start apache
  hosts: web
  become: true

  tasks:
    - name: Ensure the httpd package is installed
      ansible.builtin.package:
        name: httpd
        state: present

    - name: Create the index.html file
      ansible.builtin.template:
        src: files/index.html
        dest: /var/www/html/

    - name: Start the httpd service if needed
      ansible.builtin.service:
        name: httpd
        state: started
```

Running an Ansible playbook. The most important colors of Ansible.

A task executed as expected, no change was made.

A task executed as expected, making a change.

A task failed to execute successfully.

Running an Ansible Playbook with `ansible-navigator`

What is it?

It is a command line utility and text-based user interface (TUI) for running, testing and developing Ansible automation content

- Review EEs
- Develop collections
- Develop playbooks
- Troubleshoot problems

```
# Direct command-line interface method
$ ansible-navigator run playbook.yml \
-i inventory.ini \
-m stdout


# Text-based User Interface method
$ ansible-navigator run playbook.yml -i inventory.ini
```

Ansible content navigator (ansible-navigator)

Mapping to previous Ansible commands

ansible command	ansible-navigator command
ansible-config	ansible-navigator config
ansible-doc	ansible-navigator doc
ansible-inventory	ansible-navigator inventory
ansible-playbook	ansible-navigator run

Ansible content navigator (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

Lab Time

Complete Exercise 1.3

Exercise 1.4

Topics Covered:

- Working with Ansible Variables
- Working with Ansible Facts

Red Hat

Ansible Variable Playbook

```
---
- 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
      ansible.builtin.debug:
        msg: "{{ var_three }}"
```

Ansible Variable Playbook

```
---
- 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
      ansible.builtin.debug:
        msg: "{{ var_three }}"
```

ansible is awesome

Ansible Facts

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
    ansible.builtin.setup:
      gather_subset:
        - 'all'
```

Ansible Facts Playbook

```
---
- name: Ansible Facts playbook
  hosts: localhost
  gather_facts: true

  tasks:
    - name: Define custom fact
      ansible.builtin.set_fact:
        server_role: “frontend”

    - name: Use custom and built-in fact
      ansible.builtin.debug:
        msg: ‘The {{ ansible_hostname }} is assigned the role: {{ server_role }}’
```

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

```
Play name                 Ok   Changed   Unreachable    Failed     Skipped      Ignored      In progress        Task count   Progress
0|Ansible Facts playbook   3         0             0         0           0            0                0                 3   Complete
```

```
Result      Host             Number      Changed     Task                          Task action                    Duration
0|Ok        localhost             0      False       Gathering Facts               gather_facts                         4s
1|Ok        localhost             1      False       Define custom fact            ansible.builtin.set_fact             0s
2|Ok        localhost             2      False       Use custom and built-in fact  ansible.builtin.debug                0s
```

```
Play name: Ansible Facts playbook:2
Task name: Use custom and built-in fact
Ok: localhost The web.example.com is assigned the role: frontend

.
.
 4|host: localhost
 5|play: Ansible Facts playbook
 6|play_pattern: localhost
 7|playbook: /Users/facts_playbook.yml
 8|remote_addr: 127.0.0.1
 9|res:
10|  _ansible_no_log: false
11|  _ansible_verbose_always: true
12|  changed: false
13|  msg: 'The web.example.com is assigned the role: frontend'
```

Lab Time

Complete Exercise 1.4

Exercise 1.5

Topics Covered:

- Conditionals
- Handlers
- Loops

Conditionals

What are they?

- Conditionals allow tasks to run **only if certain conditions are met.**
- Enable **dynamic automation** by checking the values of variables & making decisions at runtime.

```
vars:
   my_mood: happy

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

Ansible Conditionals

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

  vars:
    my_mood: happy

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

Alternatively

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

Ansible Conditionals with Facts

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

  tasks:
  - name: Install httpd
    ansible.builtin.package:
      name: httpd
      state: latest
    when: ansible_distribution == 'RedHat'

  - name: Install apache
    ansible.builtin.package:
      name: apache2
      state: latest
    when: ansible_distribution == 'Debian' or
          ansible_distribution == 'Ubuntu'
```

Ansible Conditionals using Previous Task State

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

  tasks:
  - name: Ensure httpd package is present
    ansible.builtin.package:
      name: httpd
      state: latest
    register: httpd_results

  - name: Restart httpd
    ansible.builtin.service:
      name: httpd
      state: restarted
    when: httpd_results.changed
```

Ansible Handler Tasks

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

  tasks:
  - name: Ensure httpd package is present
    ansible.builtin.package:
      name: httpd
      state: latest
  notify: restart_httpd

  handlers:
  - name: restart_httpd
    ansible.builtin.service:
      name: httpd
      state: restarted
```

Ansible Handler Tasks

```yaml
tasks:
- name: Ensure httpd package is present
  ansible.builtin.package:
    name: httpd
    state: latest
  notify: restart_httpd

- name: Standardized index.html file
  ansible.builtin.copy:
    content: "This is my index.html file for {{ ansible_host }}"
    dest: /var/www/html/index.html
  notify: restart_httpd
```

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

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

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

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

unchanged

changed

handler runs once

Ansible Handler Tasks

```
tasks:
- name: Ensure httpd package is present
  ansible.builtin.package:
    name: httpd
    state: latest
  notify: restart_httpd

- name: Standardized index.html file
  ansible.builtin.copy:
    content: "This is my index.html file for {{ ansible_host }}"
    dest: /var/www/html/index.html
  notify: restart_httpd
```

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

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

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

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

changed

changed

handler runs once

Ansible Handler Tasks

```
tasks:
- name: Ensure httpd package is present
  ansible.builtin.package:
    name: httpd
    state: latest
  notify: restart_httpd

- name: Standardized index.html file
  ansible.builtin.copy:
    content: "This is my index.html file for {{ ansible_host }}"
    dest: /var/www/html/index.html
  notify: restart_httpd
```

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

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

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

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

unchanged

unchanged

Ansible Variables and Loops

```
---
- name: Ensure users
  hosts: node1
  become: true

  tasks:
    - name: Ensure user is present
      ansible.builtin.user:
        name: dev_user
        state: present

    - name: Ensure user is present
      ansible.builtin.user:
        name: qa_user
        state: present

    - name: Ensure user is present
      ansible.builtin.user:
        name: prod_user
        state: present
```

Ansible Variables and Loops

```
---
- name: Ensure users
  hosts: node1
  become: true

  tasks:
    - name: Ensure user is present
      ansible.builtin.user:
        name: "{{ item }}"
        state: present
      loop:
        - dev_user
        - qa_user
        - prod_user
```

Lab Time

Complete Exercise 1.5

Exercise 1.6

Topics Covered:

- Templates

Red Hat

Ansible Templates

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

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

Ansible Templates

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

  tasks:
    - name: Verify correct config file is present
      ansible.builtin.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 }}
```

Lab Time

Complete Exercise 1.6

Exercise 1.7

Topics Covered:

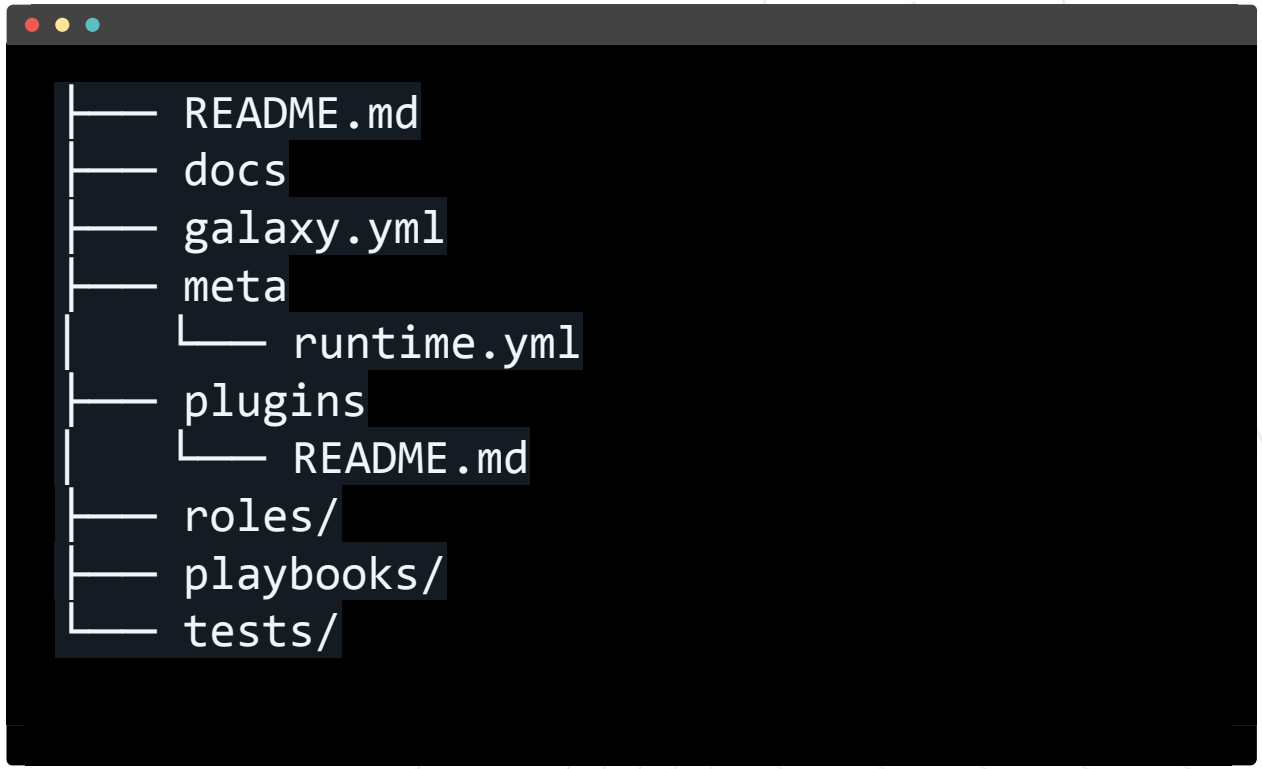
- What are Ansible Collections?
- How do you create an Ansible Collection?
- What and how do I use ansible-galaxy?

Red Hat

Ansible Collections

What are they?

- A way to **organize**, **distribute**, and **reuse** automation content.
- Group components like **roles**, **modules**, **plugins** and **playbooks**.
- Distributed via:
  - Ansible Galaxy
  - Automation Hub
- Improves **content management** and **collaboration** within teams

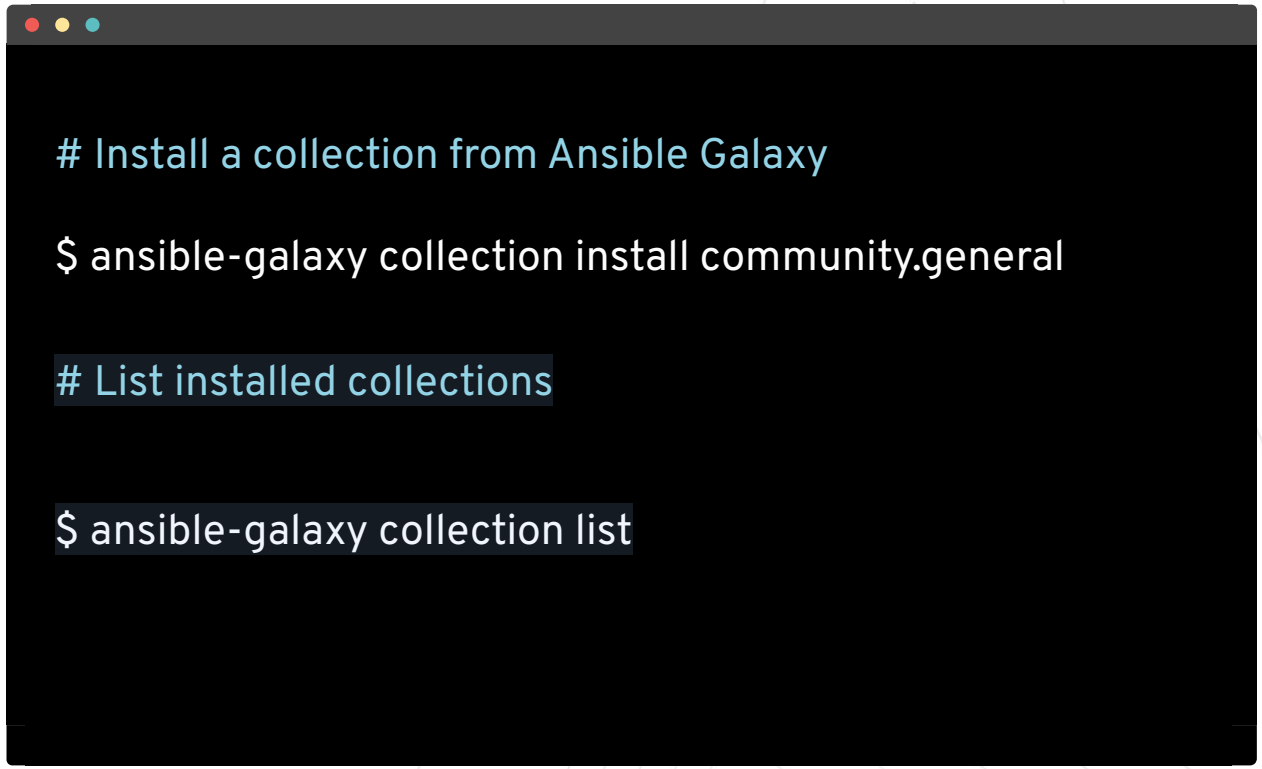


Ansible Galaxy

What is Ansible Galaxy?

- **Ansible Galaxy** is a **community platform** to **discover, share, and download** automation content like **roles** and **collections**.

- It enables users to **reuse existing content** instead of building everything from scratch, fostering collaboration and efficiency.



Lab Time

Complete Exercise 1.7

Exercise 1.8

Topics Covered:

- Debugging in Ansible

Red Hat

Debugging in Ansible. Identify and Resolve Issues.

How does it work?

- Debugging helps identify and resolve issues in playbooks.

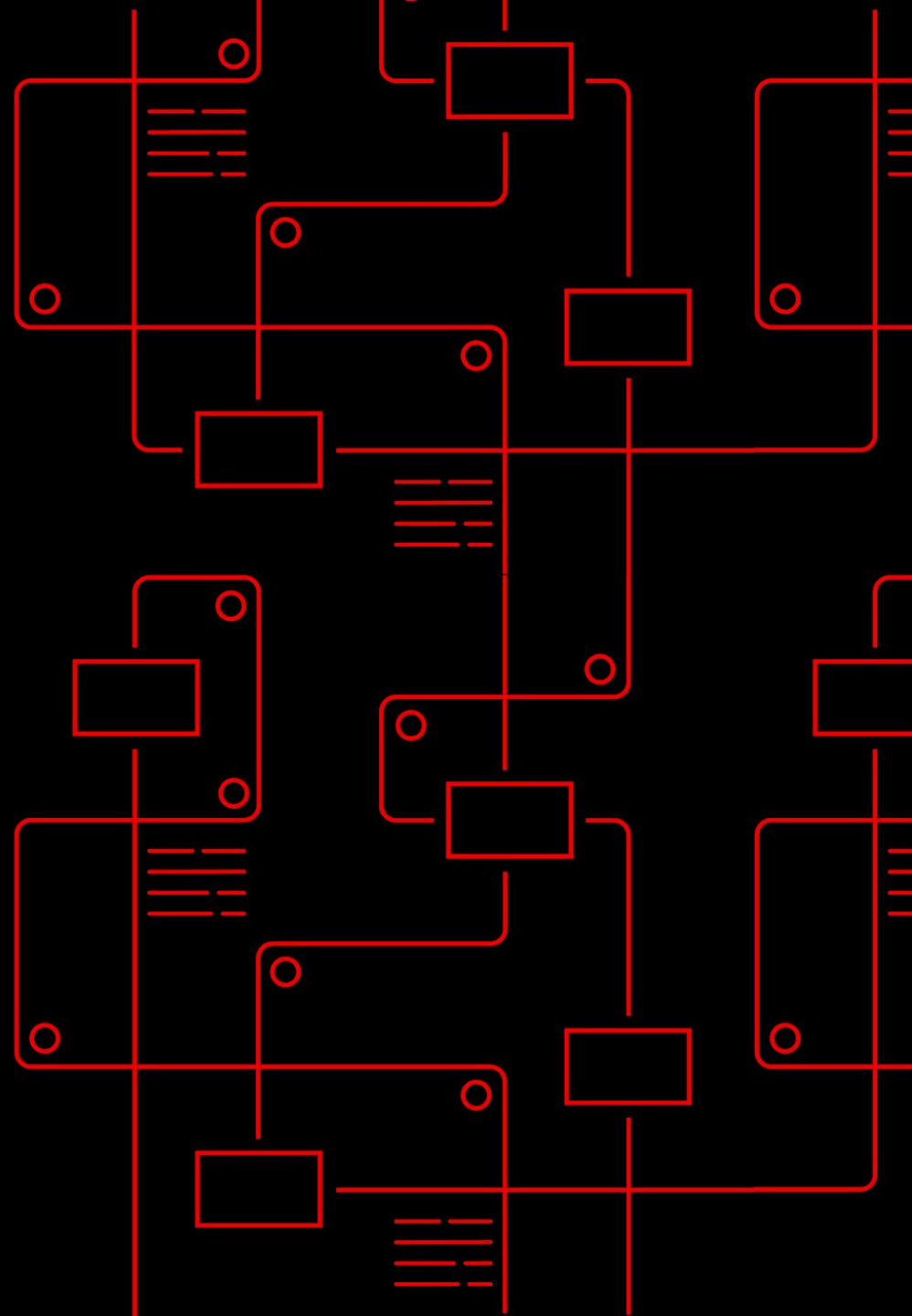
- Ansible offers several methods for debugging, including:
  - Debug module
  - Increased verbosity levels

```
- name: Display Variable Value
  ansible.builtin.debug:
    var: apache_service_name

- name: Display Custom Message
  ansible.builtin.debug:
    msg: "Apache service name is {{ apache_service_name }}"
```

Lab Time

Complete Exercise 1.8

Section 2
Automation Controller



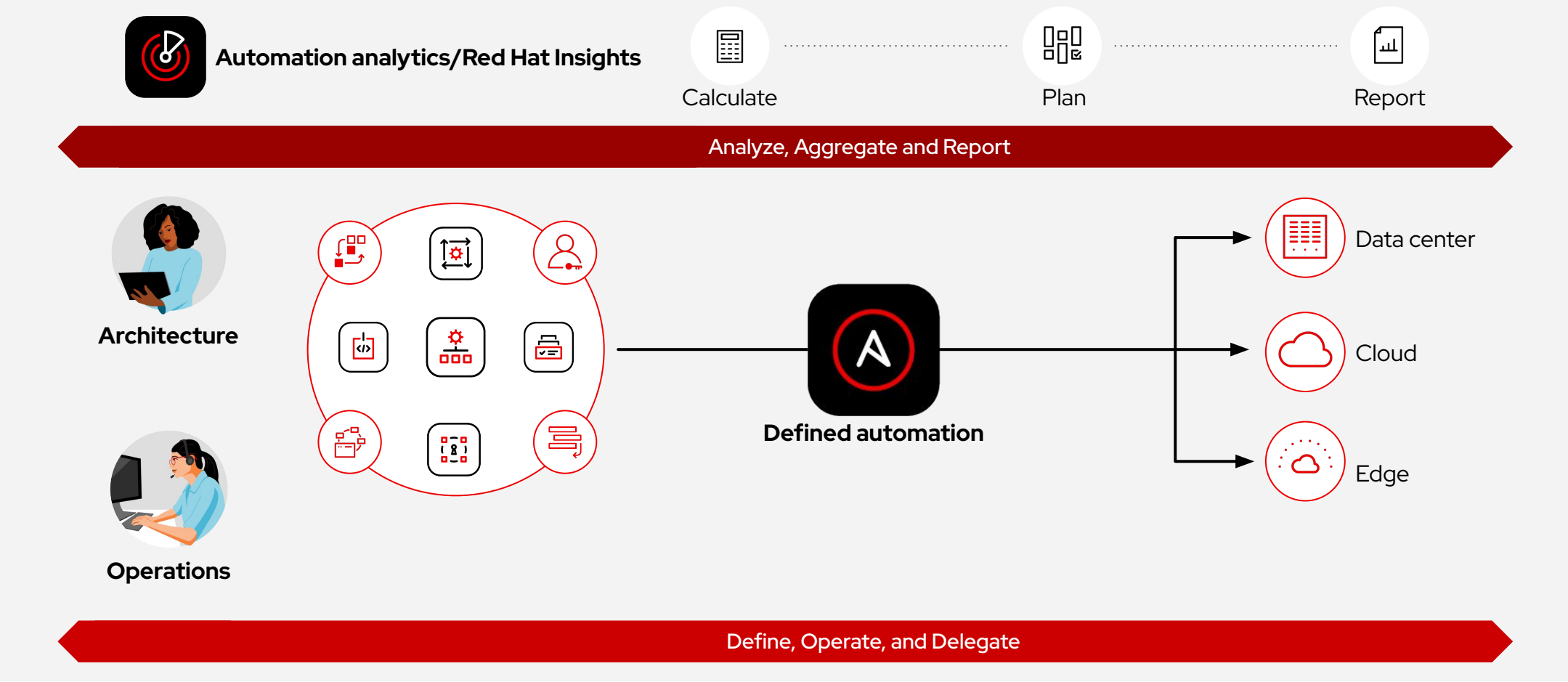
Exercise 2.1

Topics Covered:

- Intro to Automation Controller

Red Hat

The automation life cycle



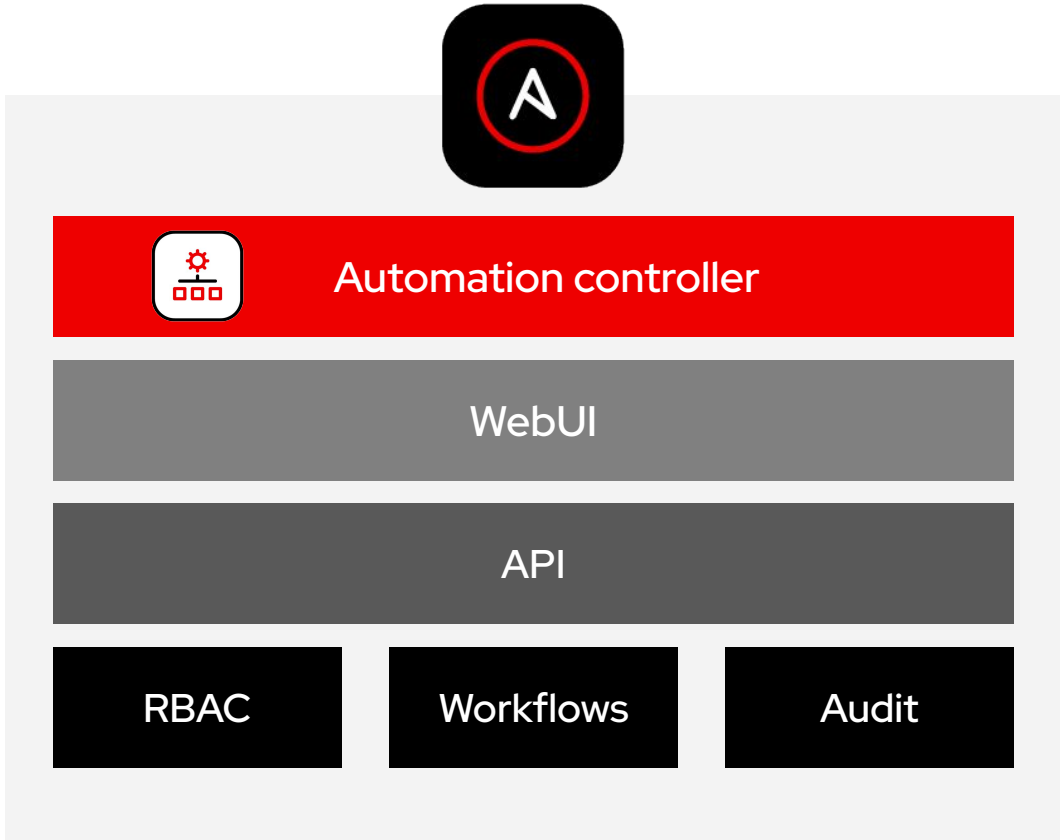
Automation controller. Define, operate, and delegate.

What is it?

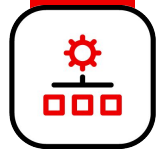
Automation controller is the Ansible Automation Platform control plane which enables users to define, operate, and delegate automation across their enterprise

Automation controller provides:

- WebUI and API
- Role-based access control
- Powerful workflows
- Centralized logging
- Credential management
- Push-button automation



Automation controller. Define, operate, and delegate.



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.

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.

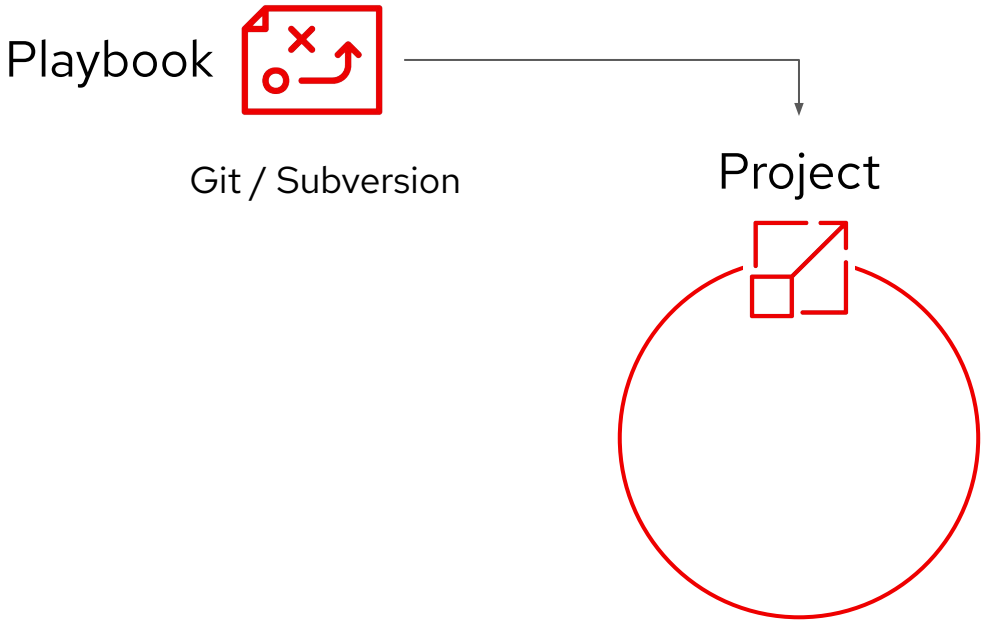
Workflows

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.

Credential Management

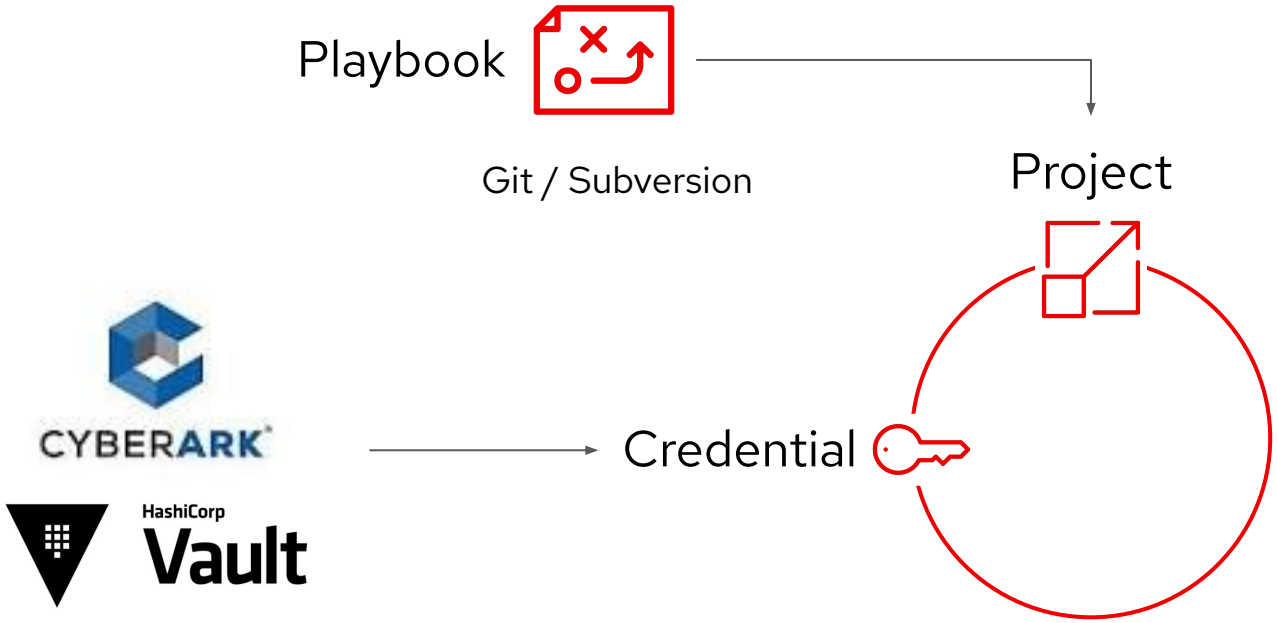
Ability to access and authenticate with external resources, repositories, or target endpoints.

Anatomy of an Automation Job

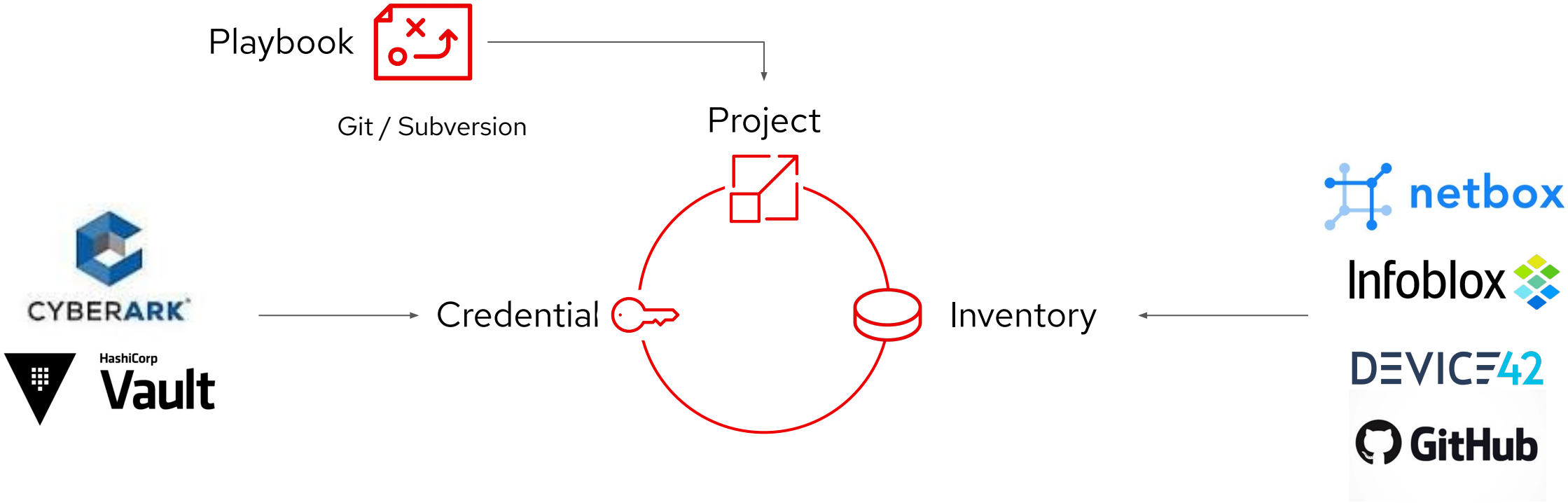


Red Hat

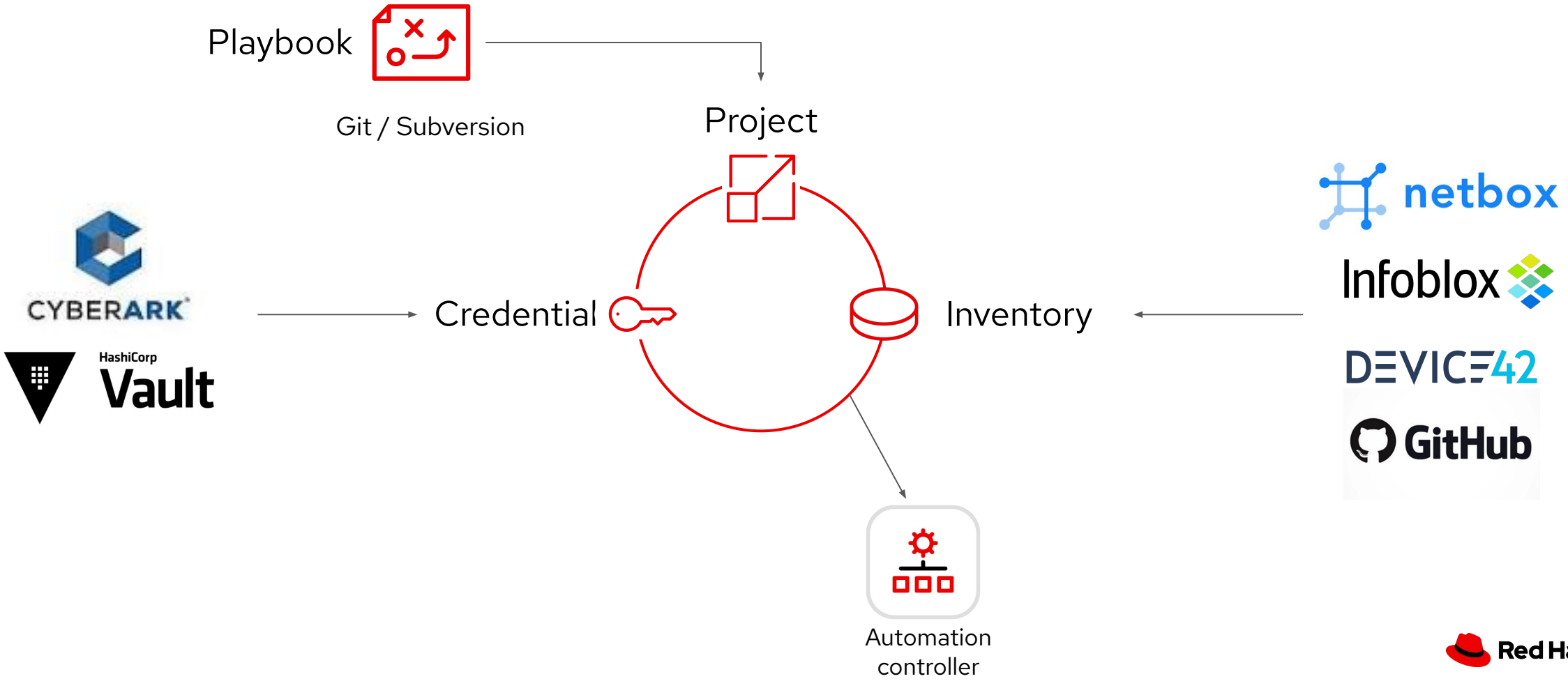
Anatomy of an Automation Job



Anatomy of an Automation Job

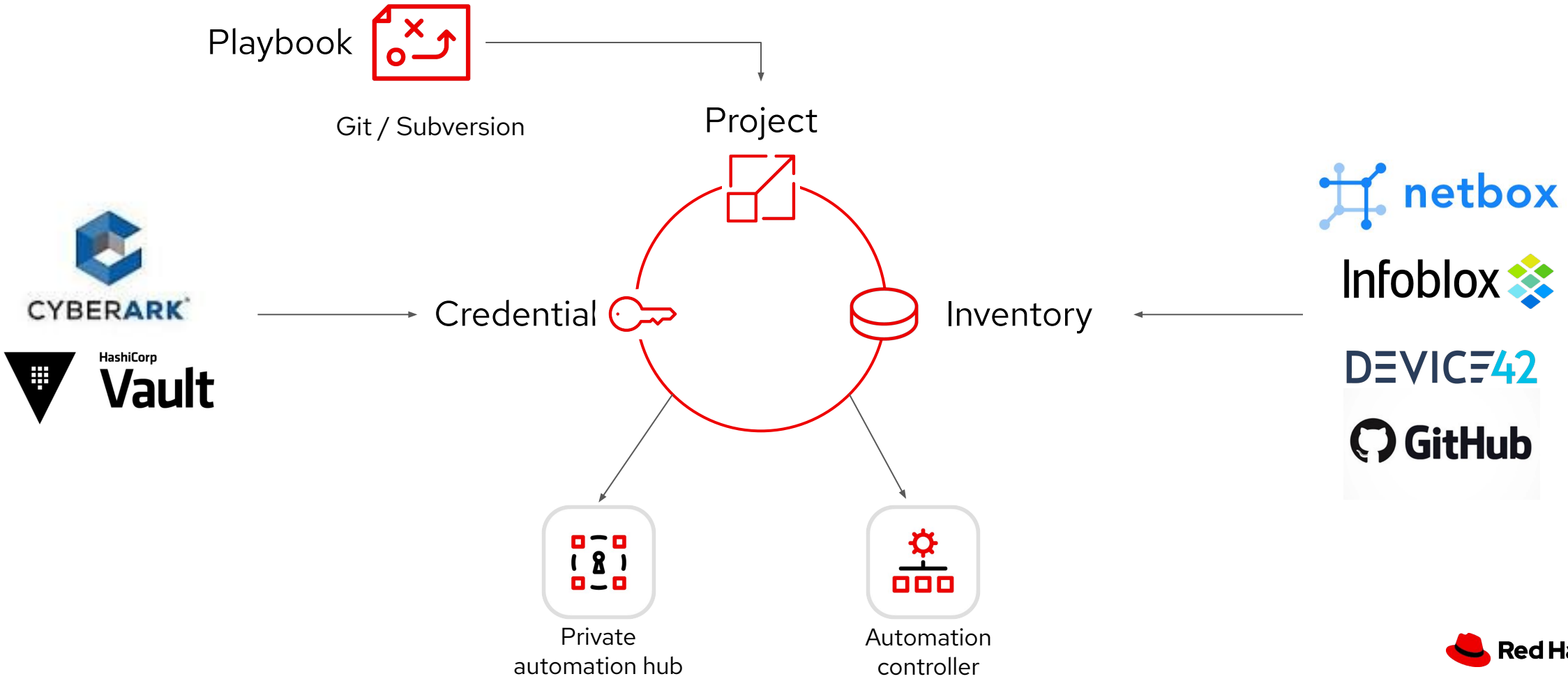


Anatomy of an Automation Job



Red Hat

Anatomy of an Automation Job



Red Hat

Lab Time

Complete Exercise 2.1

Exercise 2.2

Topics Covered:

- Inventories
- Credentials

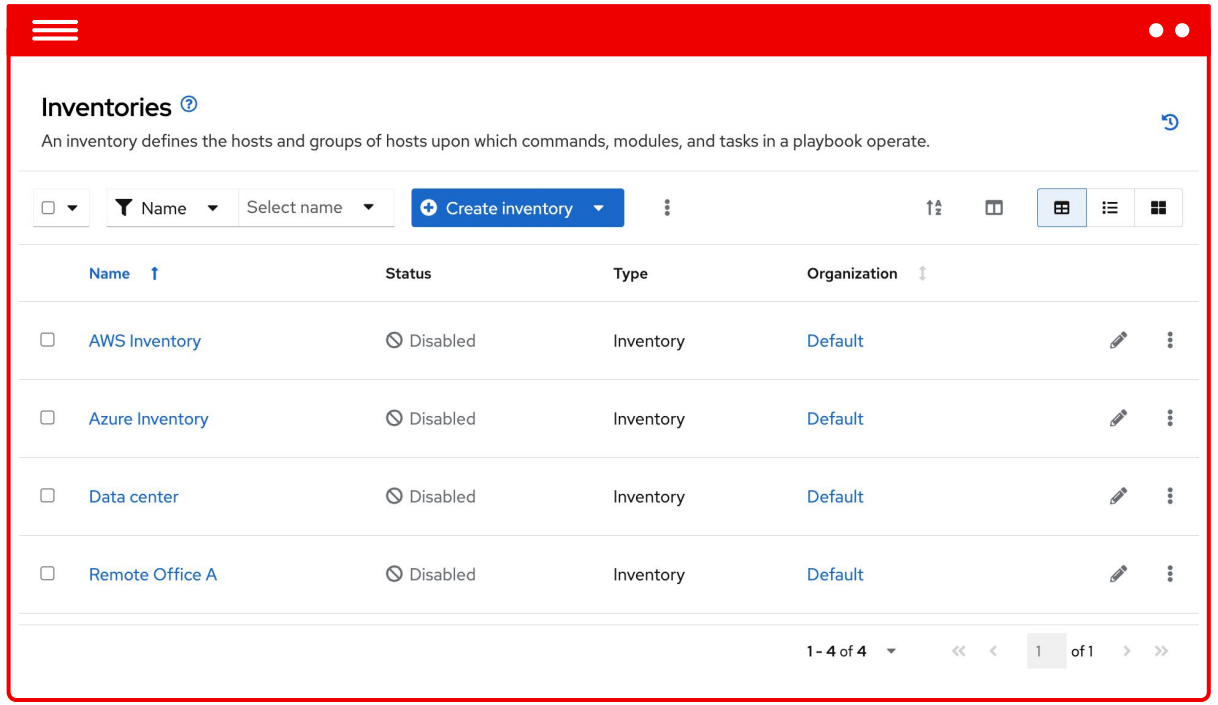
Red Hat

Inventories. What do I want to run my automation on?

What is it?

Collection of endpoints against which jobs may be launched

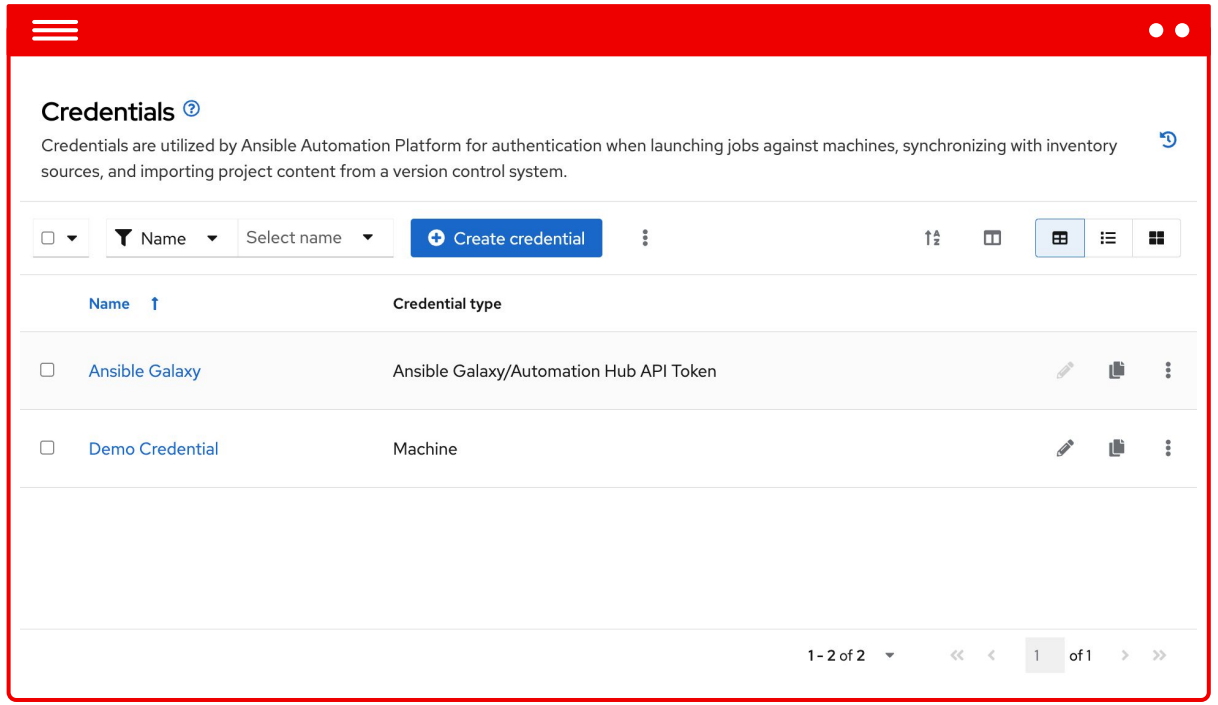
- Multiple inventory sources supported
- Dynamic endpoint discovery
- Logically group endpoints by metadata or user-defined filters
- Granular RBAC permissions



Credentials. Securing resource and endpoint access.

What is it?

- Securely manage credentials needed for automation resources
- Multiple credential types supported
- Integrate external secret management systems
- Create custom credential types and plugins
- Use RBAC to govern access
- Actual credential never exposed



Red Hat Ansible Automation Platform

Lab Time

Complete Exercise 2.2

Exercise 2.3

Topics Covered:

- Projects
- Templates

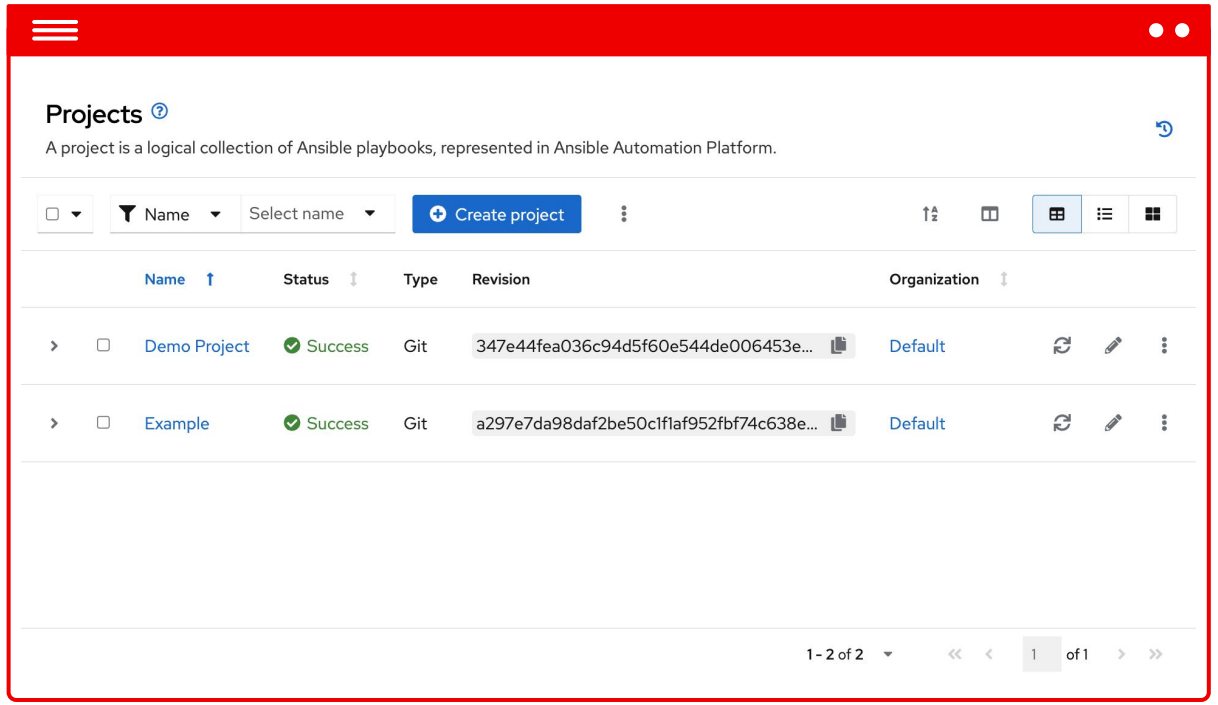
Red Hat

Projects. Adding your automation content to controller.

What is it?

Logical collection of your playbooks:

- Multiple source types supported
- Source Control Management (SCM) integration and update strategies
- Red Hat Insights integration
- Role-based access control (RBAC) and schedules

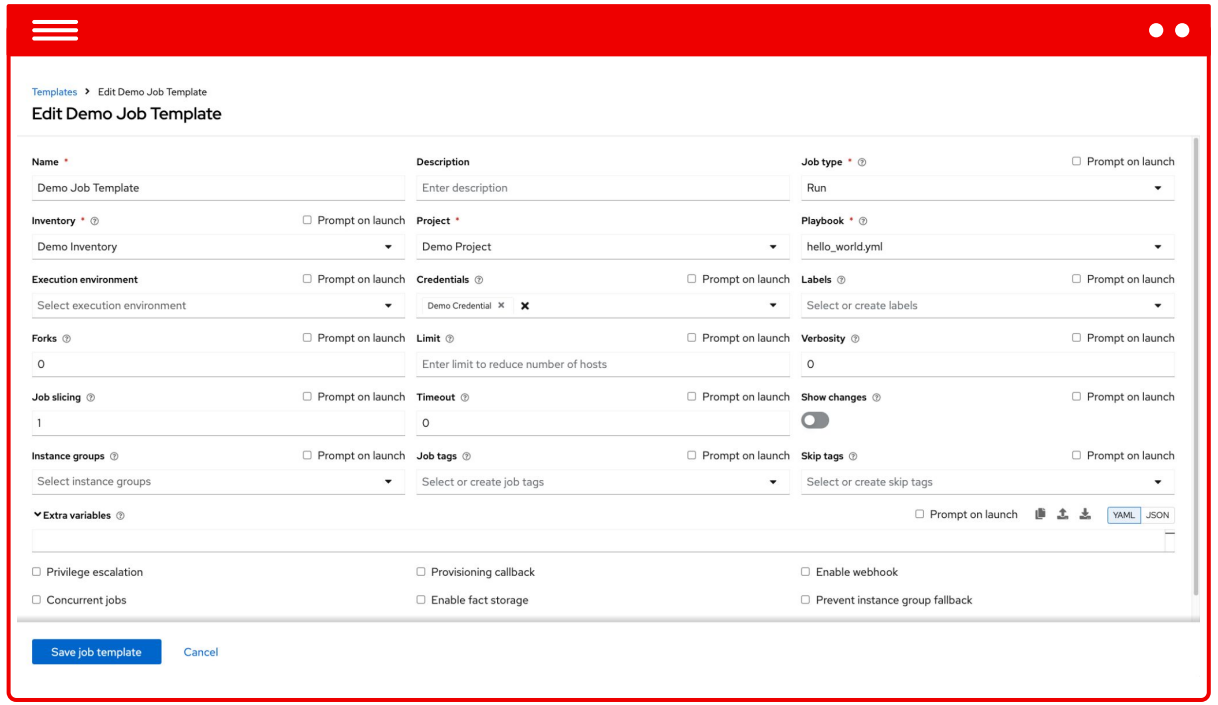


Red Hat
Ansible Automation
Platform

Job Templates. Bringing it all together.

What is it?

- Define and standardize running automation
- Reusable and shareable
- Leverage agile practices, such as GitOps and event-driven automation

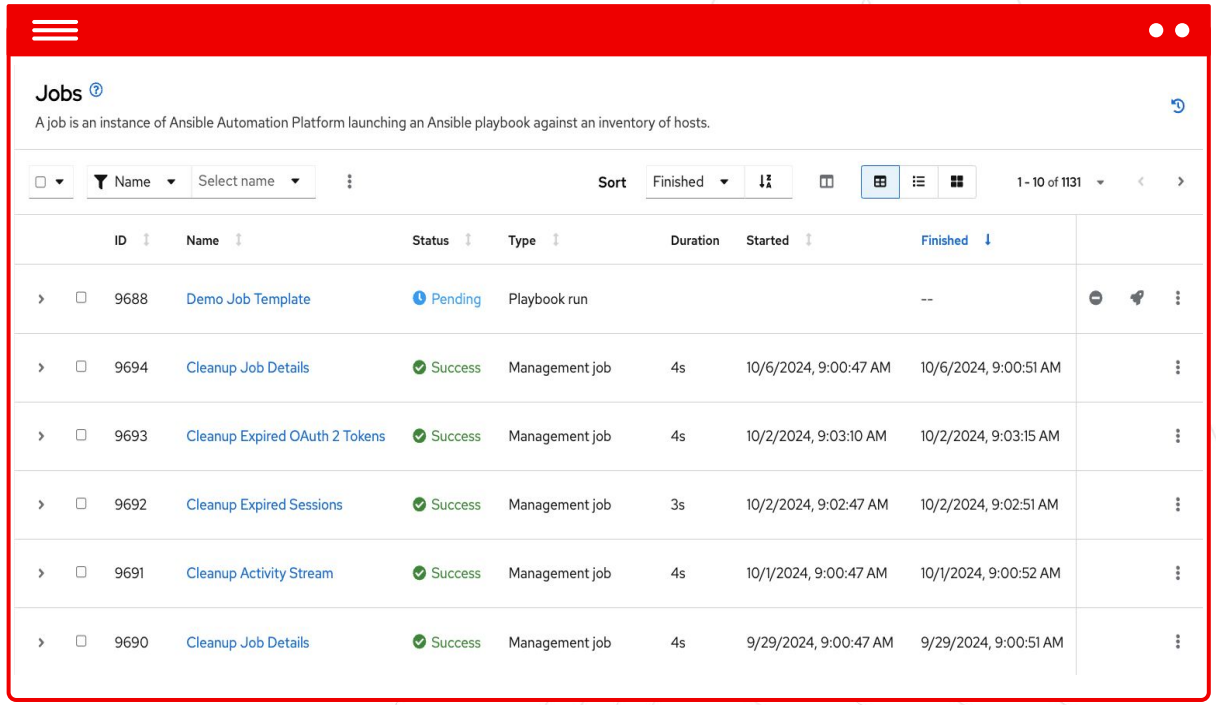


Red Hat
Ansible Automation
Platform

Automation jobs. Executing your defined automation.

What is it?

- Controller launching an instance of defined automation
- Relaunch automation jobs
- Use Job Details to view job outputs
- Troubleshoot automation execution using filtered views



Red Hat
Ansible Automation
Platform

Lab Time

Complete Exercise 2.3

Exercise 2.4

Topics Covered:

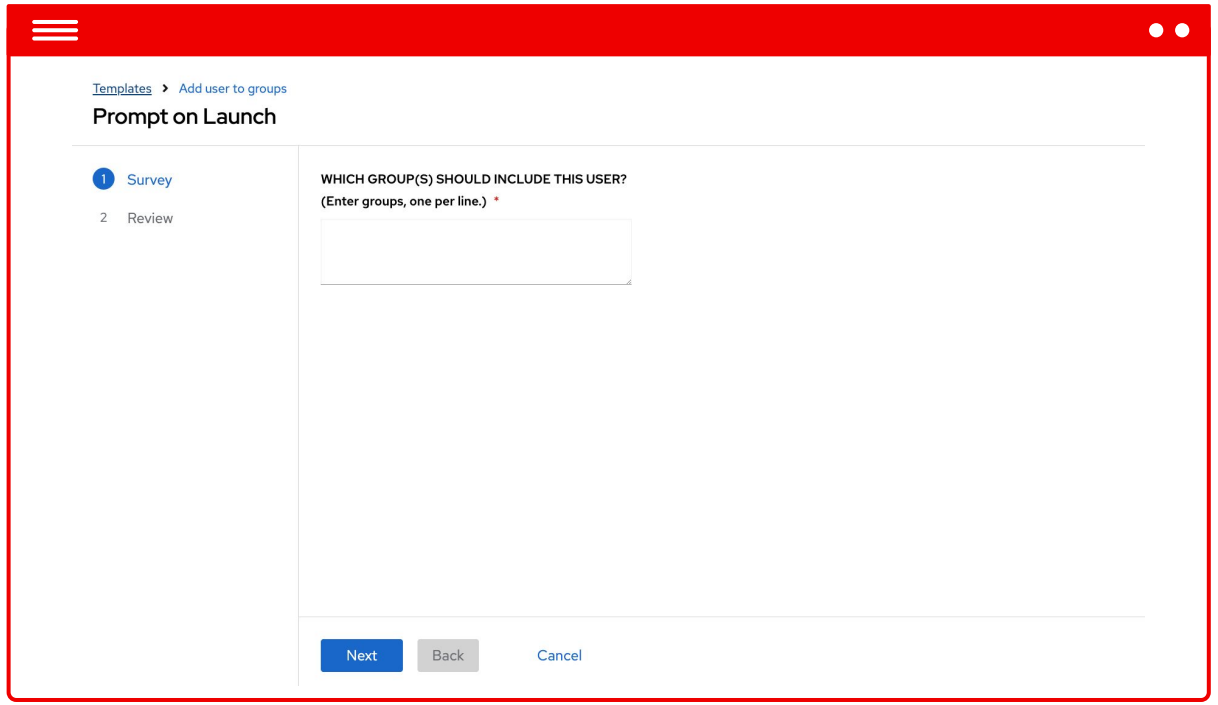
- Ansible Surveys

Red Hat

Automation controller surveys. Adopt and grow.

What is it?

- User-friendly, self-service interface in automation controller
- Abstracts complexity using question and answer format
- Best suited for teams directly accessing automation and close to the automation practice
- Access and execution governed using controller features



Lab Time

Complete Exercise 2.4

Exercise 2.5

Topics Covered:

- Role Based Access Control (RBAC)

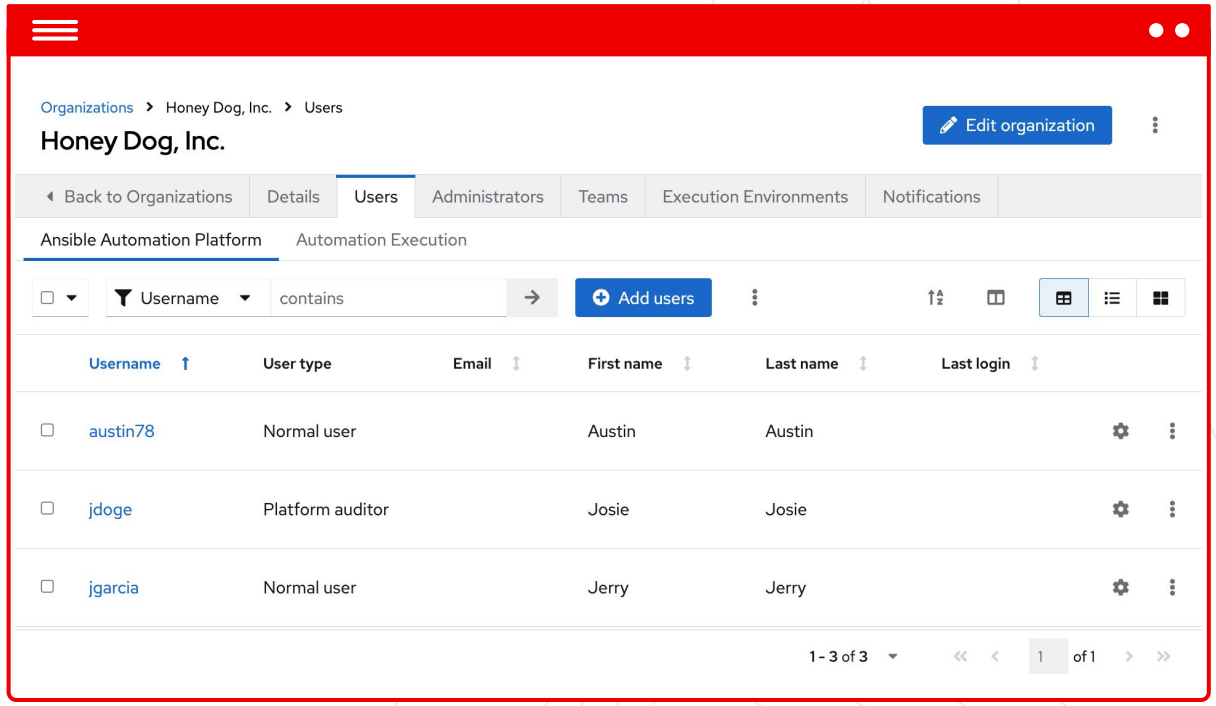
Red Hat

Role-Based Access Control. Who can use my automation?

What is it?

Securely govern access to your automation

- Logically group controller objects and grant users and teams read, execute, edit permissions
- Use predefined roles to grant access
- Integrates with your existing enterprise authentication systems



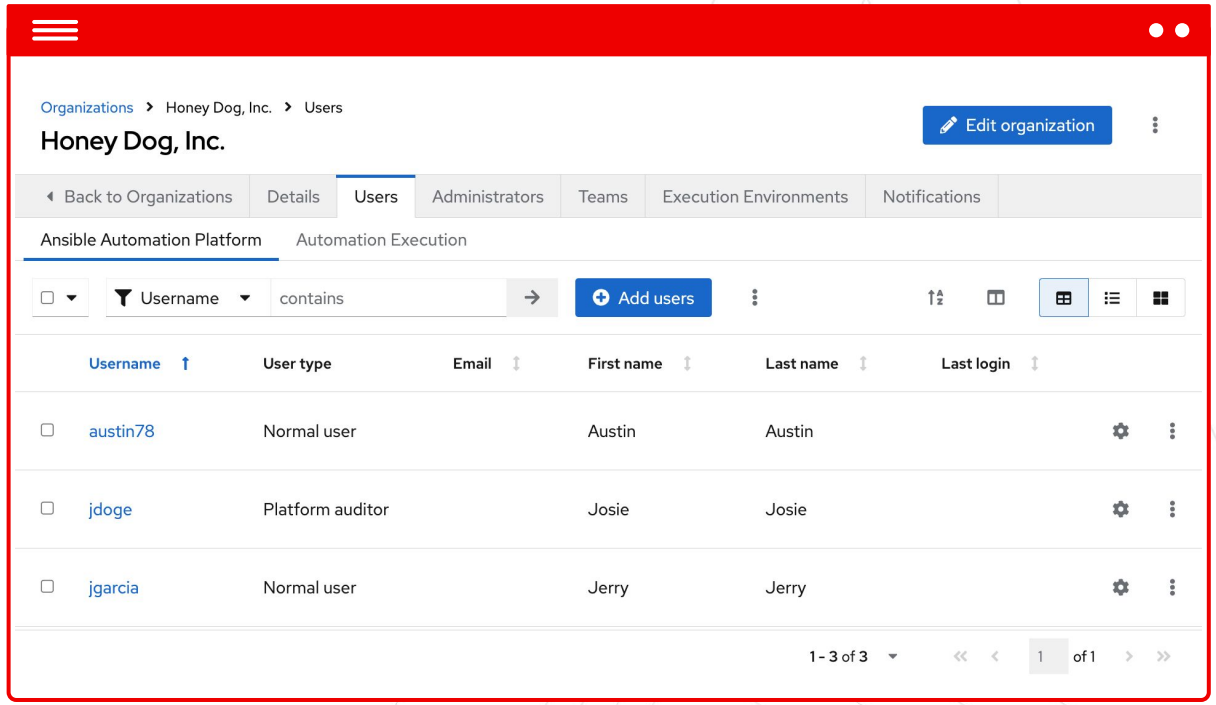
Red Hat
Ansible Automation
Platform

User Management

What is it?

Securely govern access to your automation

- Logically group controller objects and grant users and teams read, execute, edit permissions
- Use predefined roles to grant access
- Integrates with your existing enterprise authentication systems

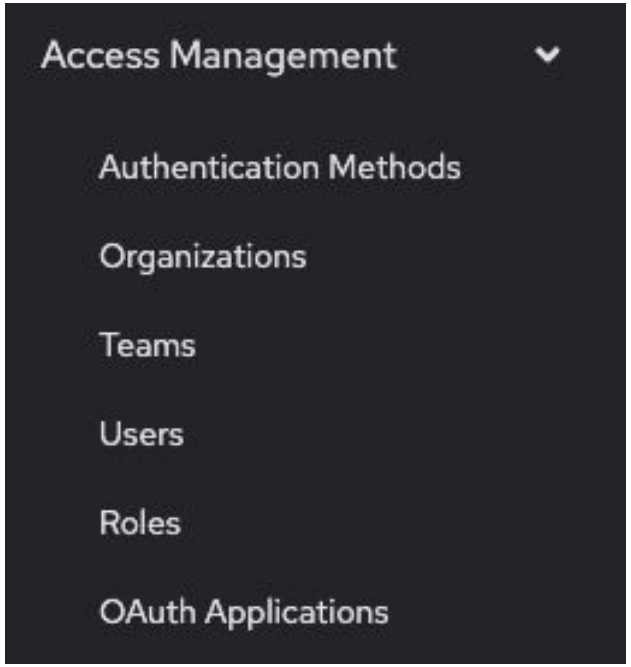


Red Hat
Ansible Automation
Platform

User Management

Govern access to your automation

- 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

Lab Time

Complete Exercise 2.5

Exercise 2.6

Topics Covered:

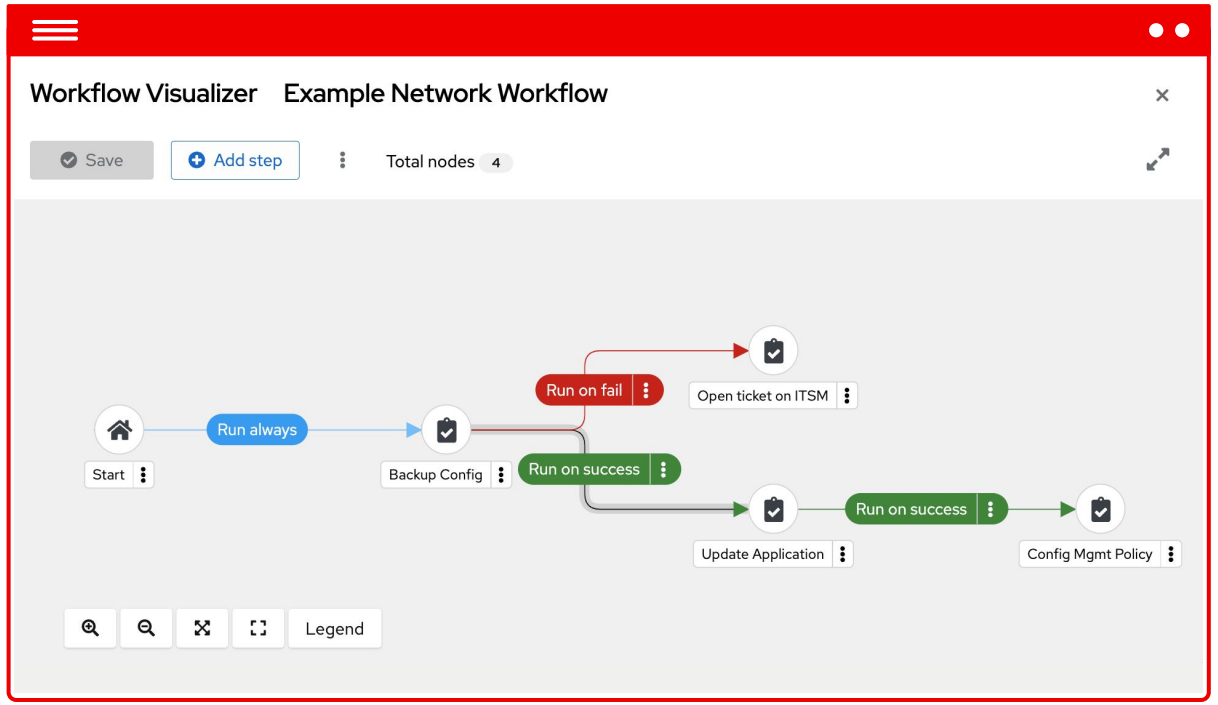
- Workflows

Red Hat

Workflows. Solve complex problems.

What is it?

- Overhauled in AAP 2.5
- 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
- Add approvals to your workflows to enhance governance
- Integrate other systems, such as ITSM to fit with your existing controls and processes



Lab Time

Complete Exercise 2.6

Exercise 2.7

Topics Covered:

- Wrap-Up

Red Hat

Lab Time

Complete Exercise 2.7

Where to go next

Learn more

- Workshops
- Documents
- Youtube
- Twitter

Get started

- Self-paced labs
- Ansible Automation Platform trial
- console.redhat.com
- Ansible Lightspeed trial

Get serious

- Red Hat Automation Adoption Journey
- Red Hat Training
- Red Hat Consulting

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/c/AnsibleAutomation


facebook.com/redhatinc


twitter.com/ansible