

# How to use this deck

## **Name:**

Network Automation Workshop Deck

## **Purpose:**

This slide deck is part of a training course designed as an introduction to Ansible for network engineers and operators. The slides are meant to be taught in conjunction with hands-on exercises with a lab topology of Automation controller + 4 network devices.

## **Last updated:**

Sep 21, 2021 (check history for older versions)

## **What this deck is for?**

This deck corresponds to the prescriptive exercises available on [https://ansible.github.io/workshops/exercises/ansible\\_network/](https://ansible.github.io/workshops/exercises/ansible_network/)

The upstream source for exercises and provisioner are provided on <https://github.com/ansible/workshops>

## **What this deck is not for?**

This is not a replacement for Red Hat training. This is a small “taste” of Ansible Automation Platform and meant to help people understand what is possible for network engineers with automation. Please refer to <https://www.redhat.com/en/services/training-and-certification> for official training

## **Google Slides source link (Red Hat internal):**

<https://docs.google.com/presentation/d/1PIT-kGAGMVEEK8PsuZCoyzFC5ClzLBwdnftnUsdUNWQ/edit?usp=sharing>

# Network Automation Workshop

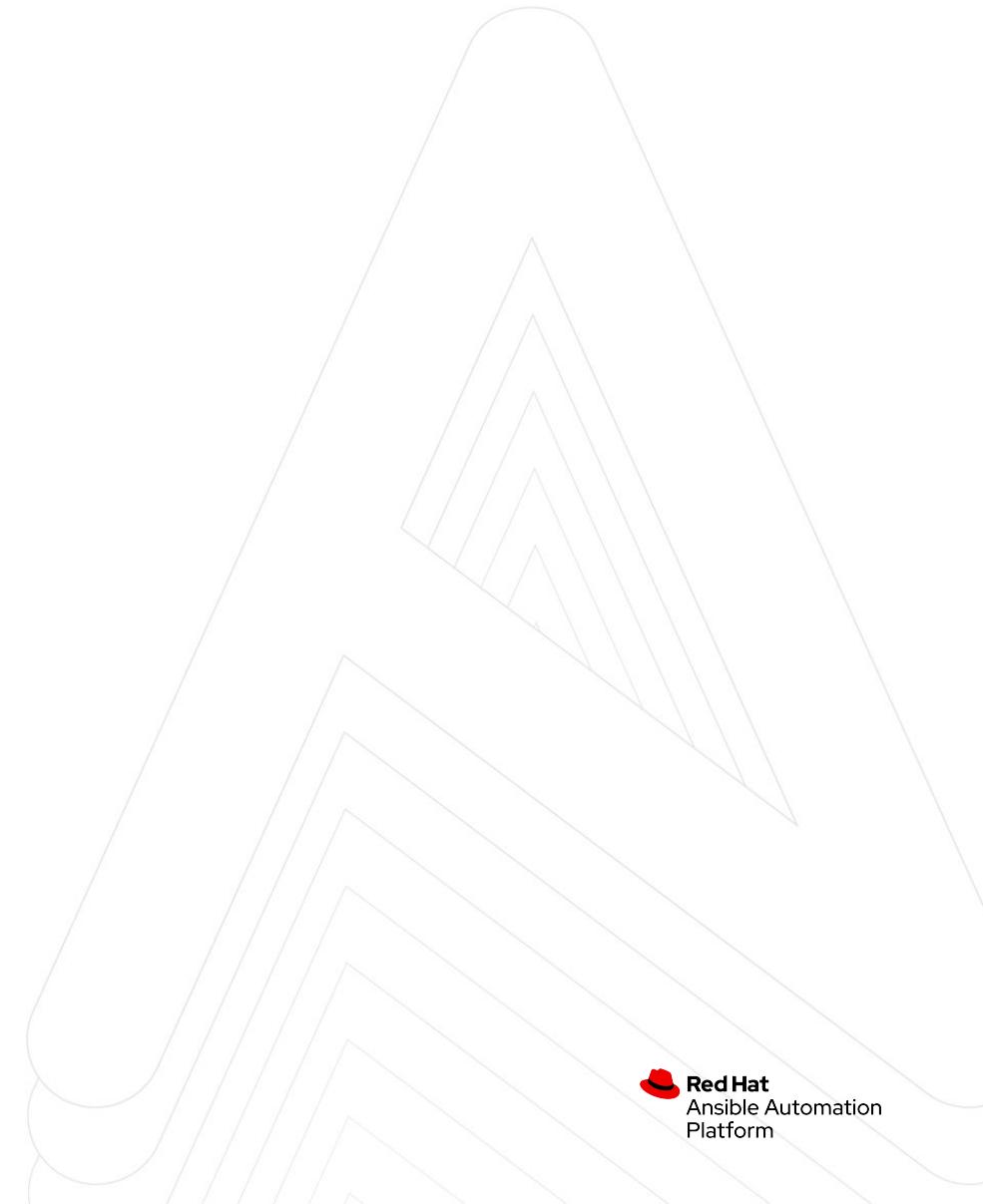
Introduction to Ansible for  
network engineers and operators

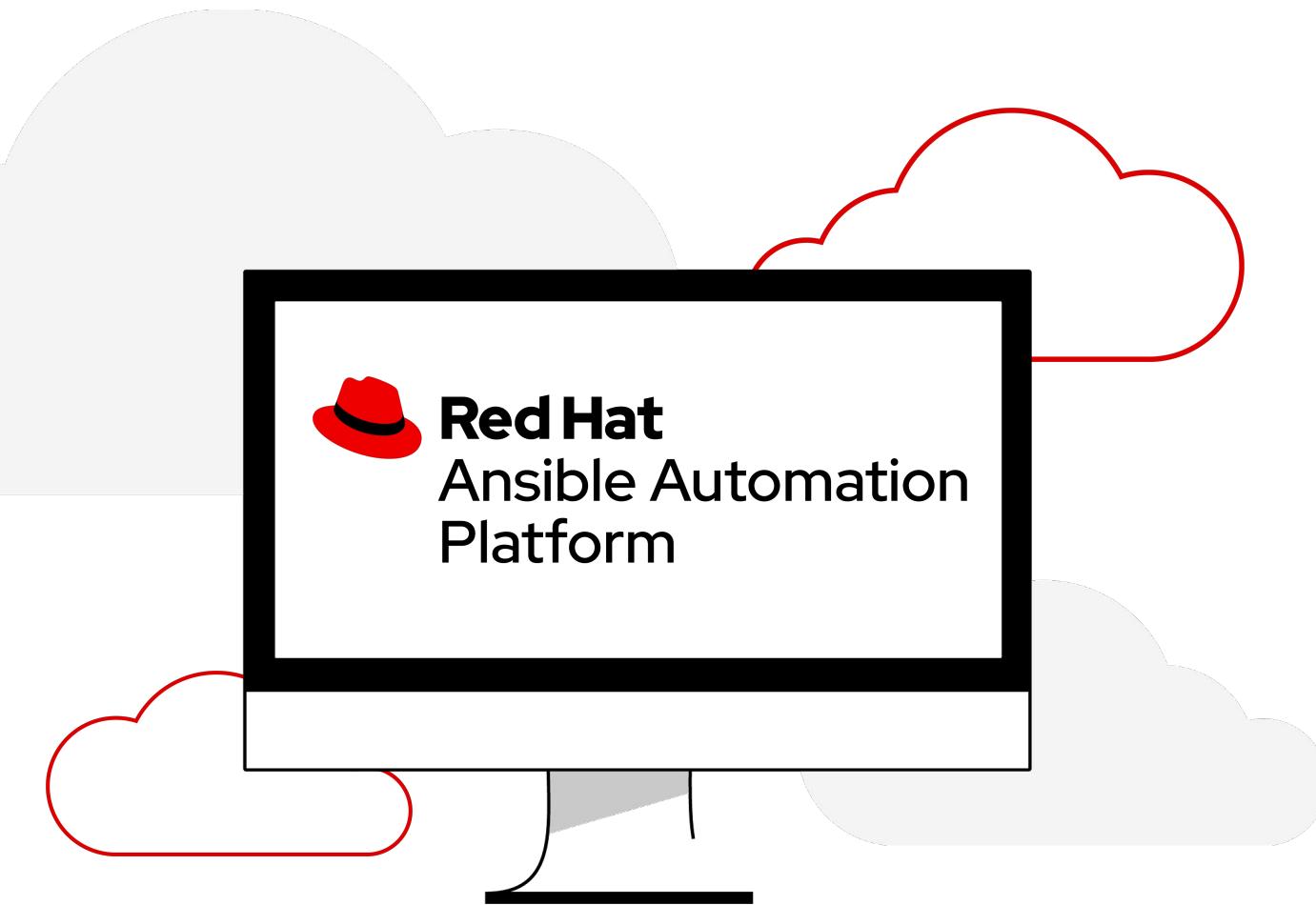


# Housekeeping

Understanding the format of this class

- Timing
- Breaks
- Takeaways





# What you will learn

- ▶ Introduction to Ansible automation
- ▶ How Ansible works for network automation
- ▶ Understanding Ansible modules and playbooks
- ▶ Executing Ansible playbooks to make configuration changes
- ▶ Gather information (Ansible facts)
- ▶ Network Resource Modules
- ▶ Using Automation controller to operationalize automation for your enterprise
- ▶ Major Automation controller features - RBAC, workflows

# Introduction

## Topics Covered:

- ▶ What is the Ansible Automation Platform?
- ▶ What can it do?
- ▶ Why Network Automation?
- ▶ How Ansible Network Automation works

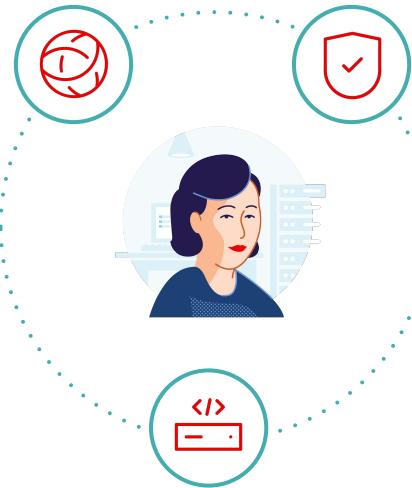




**Automation happens when  
one person meets a problem  
they never want to solve again**

# Many organizations share the same challenge

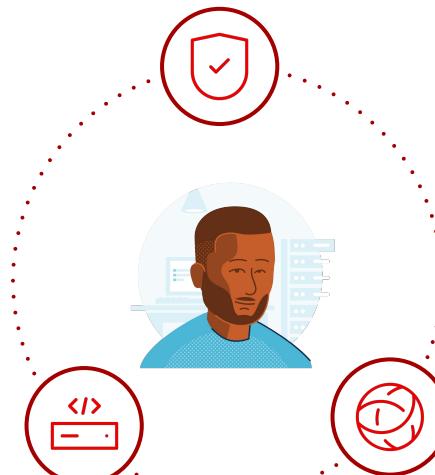
Too many unintegrated, domain-specific tools



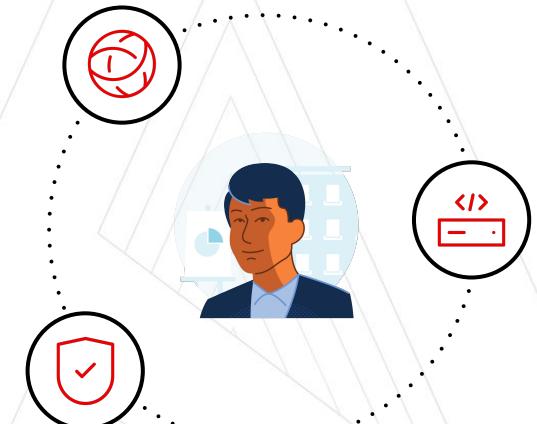
Network ops



SecOps

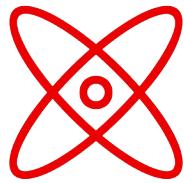


Devs/DevOps



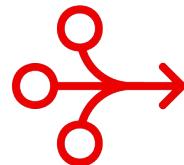
IT ops

# Why the Ansible Automation Platform?



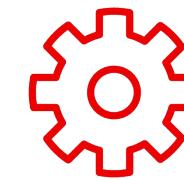
## Powerful

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



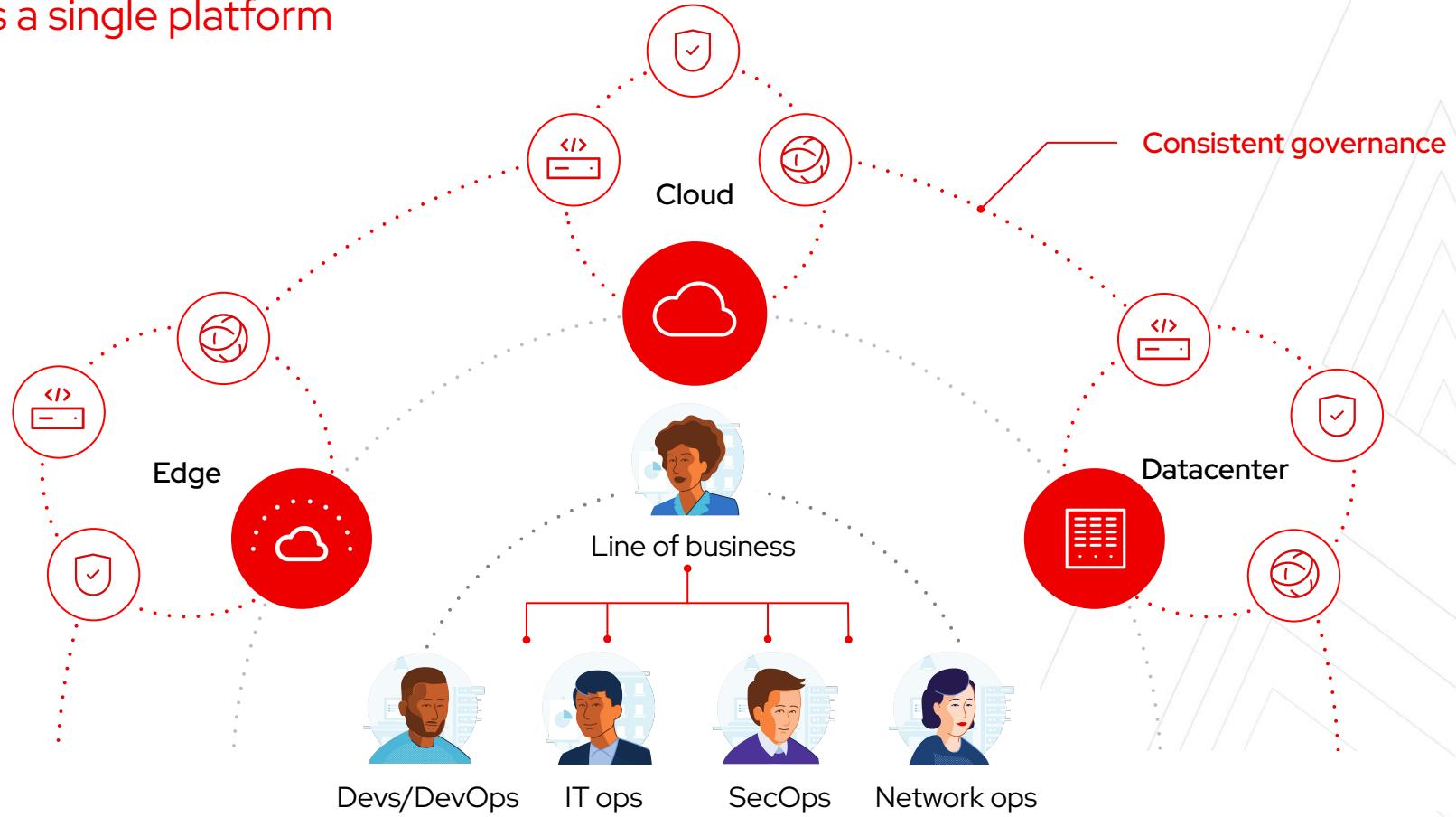
Network devices



And more ...

# Break down silos

## Different teams a single platform





## Red Hat Ansible Automation Platform



Content creators



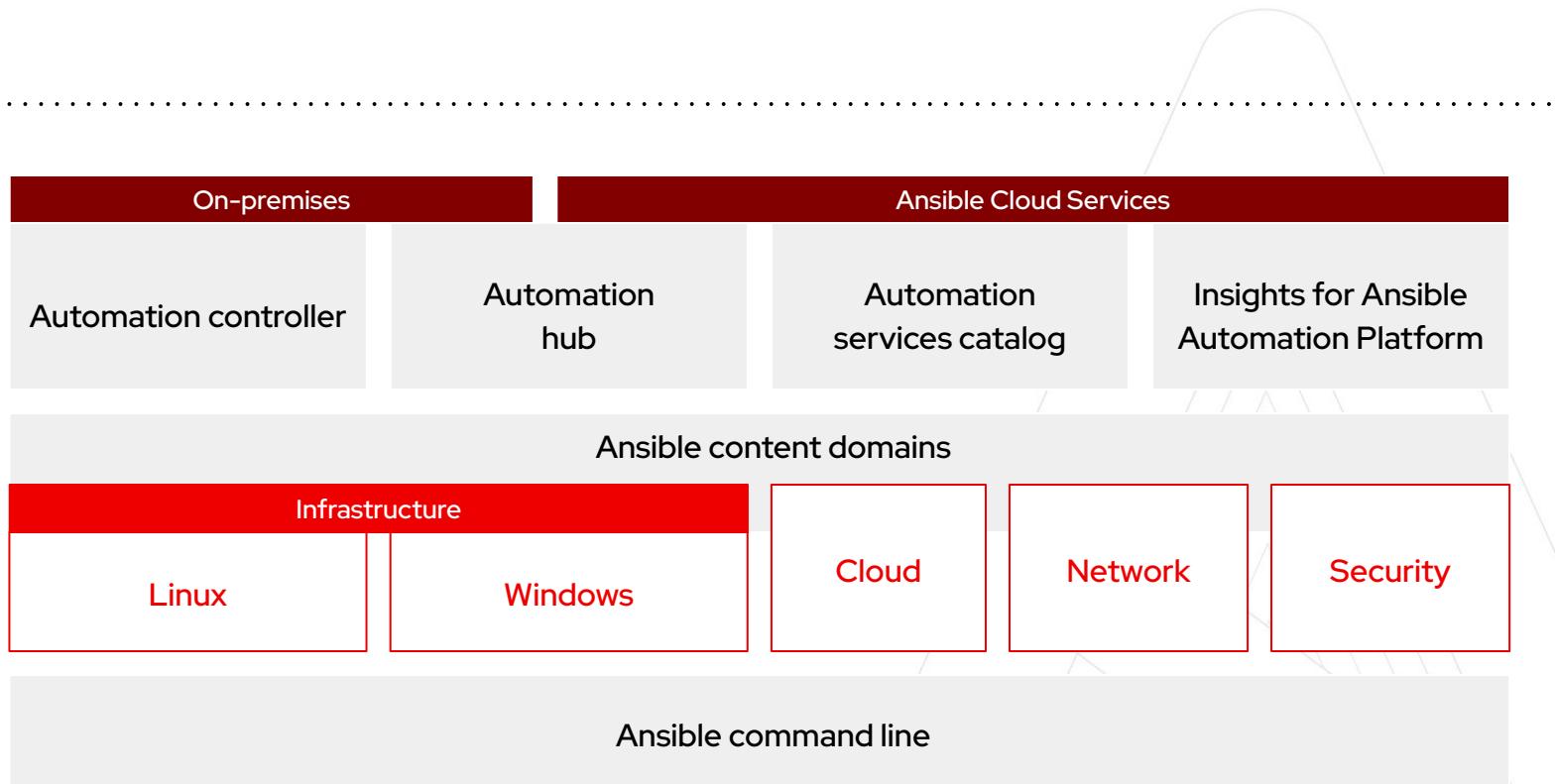
Operators



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
- Community support
- Partner Ecosystem
- 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:

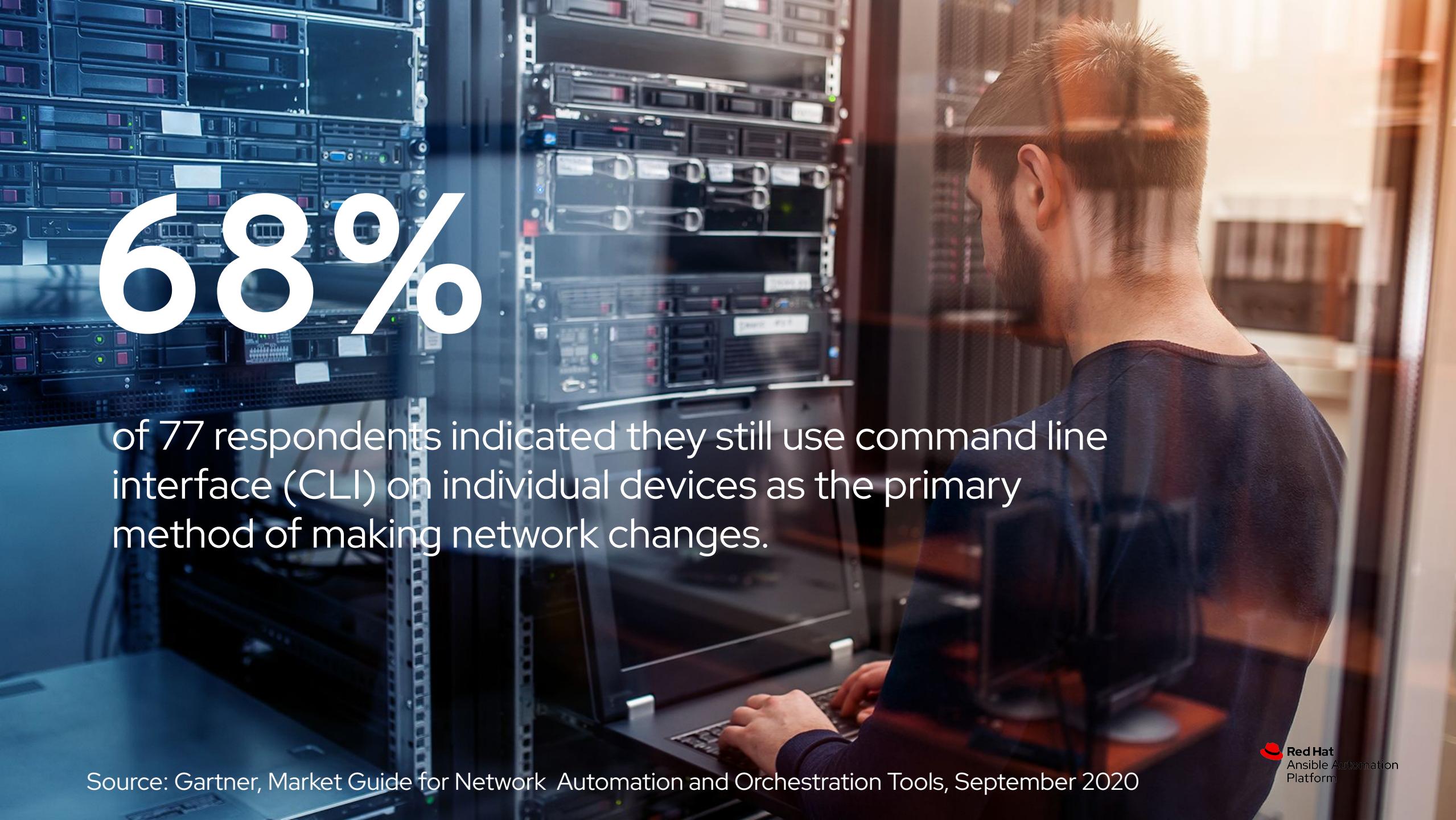
Gardner, Chris, Glenn O'Donnell, Robert Perdonii, and Diane Lynch. ["The Forrester Wave™: Infrastructure Automation Platforms, Q3 2020."](#) Forrester, 10 Aug. 2020.

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™. Information is based on best available resources. Opinions reflect judgment at the time and are subject to change.

## Use-Case

- ▶ Network Automation



A man with a beard, seen from the back, is working on a laptop in a server room. The room is filled with server racks, and the lighting is dim, with blue and red lights from the servers creating a moody atmosphere.

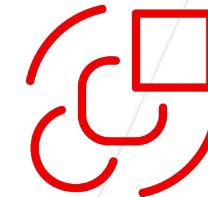
68%

of 77 respondents indicated they still use command line interface (CLI) on individual devices as the primary method of making network changes.

Source: Gartner, Market Guide for Network Automation and Orchestration Tools, September 2020

# Why hasn't networking changed?

Networking vendors are the trusted advisors



## PEOPLE

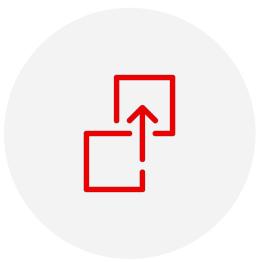
- Domain specific skill sets
- Vendor oriented experience
- Siloed organizations
- Legacy operational practices

## PRODUCTS

- Infrastructure-focused features
- CLI-based interfaces
- Siloed technologies
- Monolithic, proprietary platforms

# Next generation networking

Automation to effectively manage increasing diversity and scope



## Edge / IoT Devices

New device types entering networks at scale, with distributed computing.



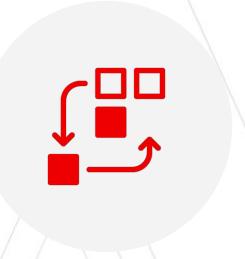
## Hybrid cloud

Numerous deployment forms across the globe



## Digital transformation

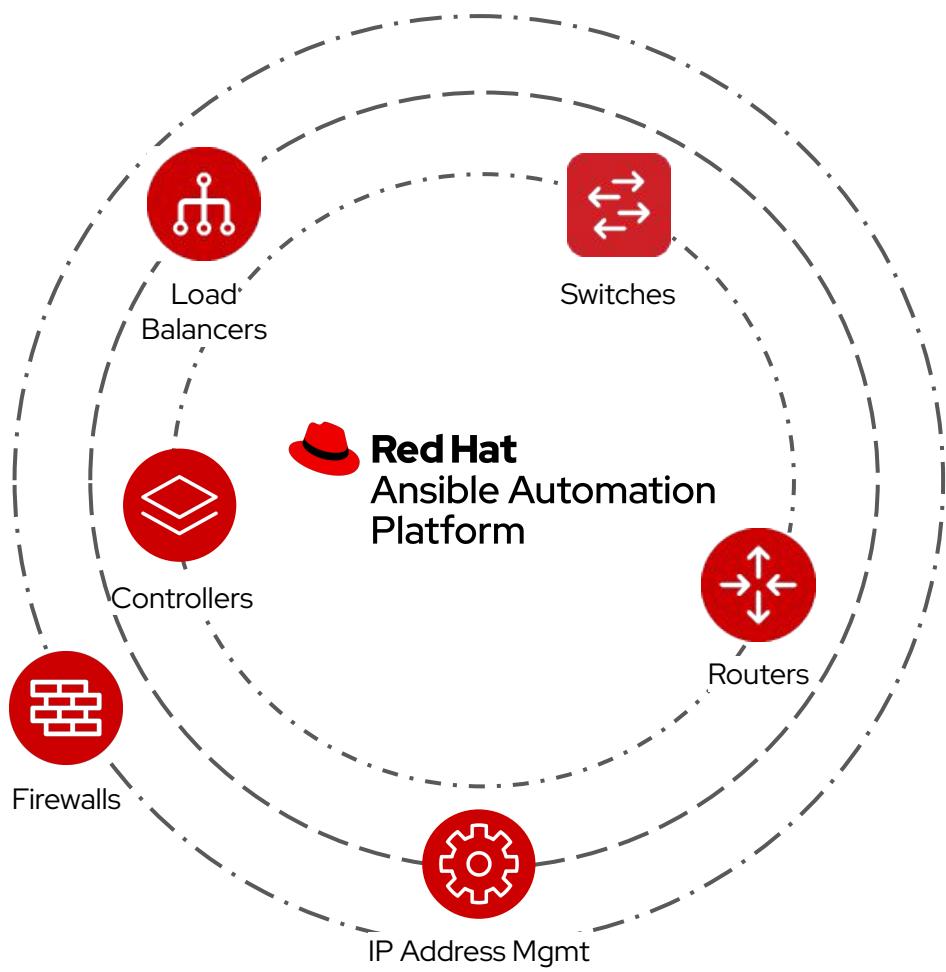
Responding with new applications is only as fast as the slowest process



## Data-intensive computing

Artificial intelligence, digital applications and growing data driving connectivity

# What is Ansible Network Automation?



**Ansible network automation** is our content domain focused on networking use cases. The goal is to provide network teams with the tools and an operational framework to implement next-generation network operations, manage network infrastructure-as-code, and better support digital transformation by connecting teams across the IT organization.

**Ansible network automation** is a set of Certified Content Collections designed to streamline and operationalize network operations across multiple platforms and vendors.

# Modernize and scale network operations

Choose what network tasks to automate at your own pace

## TRADITIONAL NETWORK OPERATIONS

- Traditional culture
- Risk averse
- Proprietary solutions
- Siloed from others
- “Paper” practices, MOPs
- “Artisanal” networks



## NEXT-GEN NETWORK OPERATIONS

- Community culture
- Risk-aware
- Open solutions
- Teams of heroes
- Infrastructure as code
- Virtual prototyping / DevOps

# What does it do?

Automate your network with a single tool



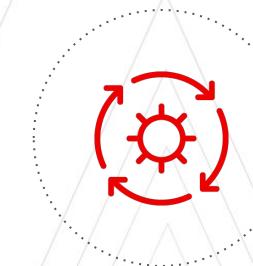
## Configuration Management

Platform agnostic configuration management to standardize and enforce best-practices.



## Infrastructure Awareness

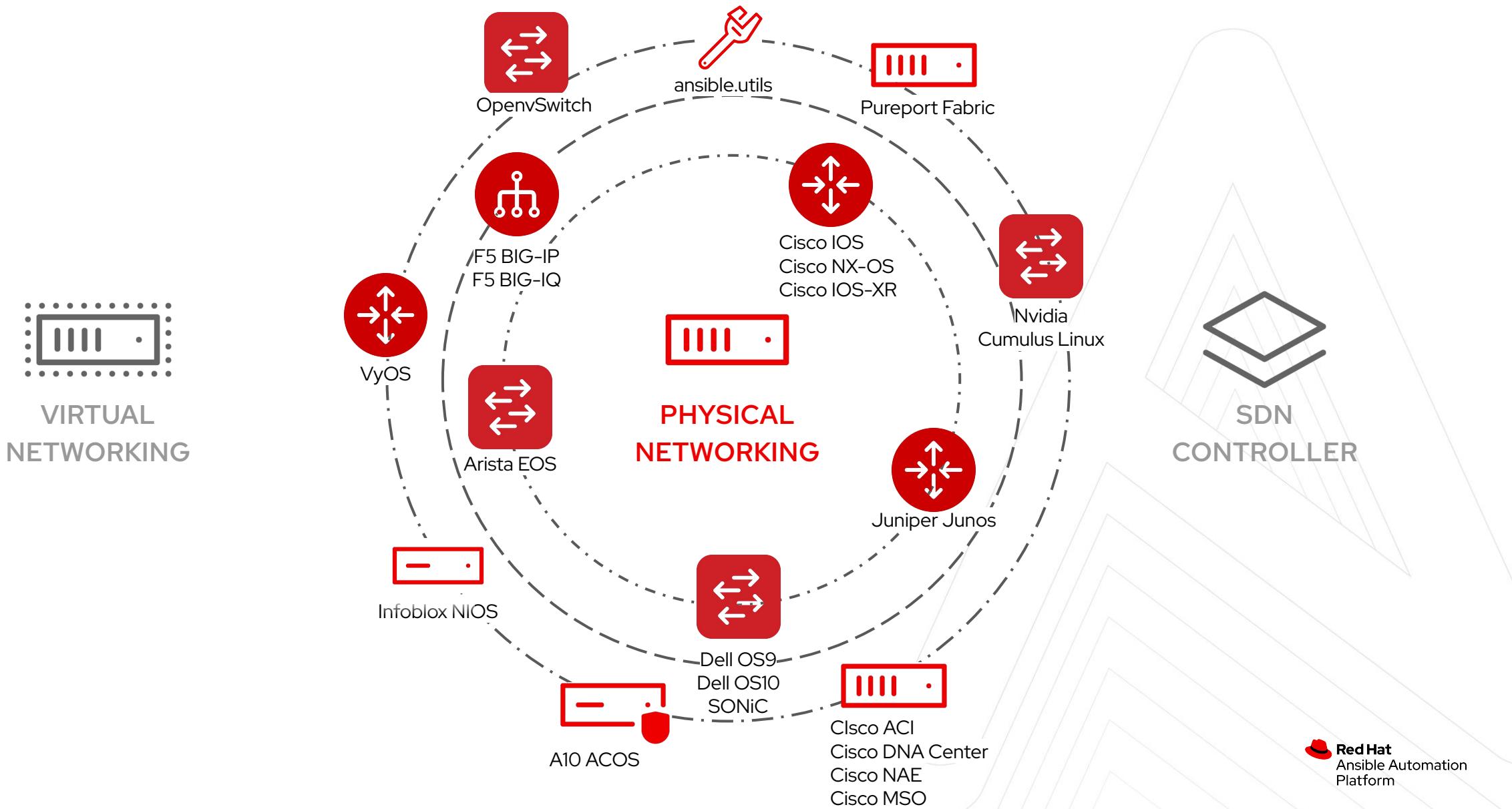
Track network resources through facts gathering, to perform preventive maintenance, reducing outage risks and costs of unnecessary hardware-refresh.



## Network Validation

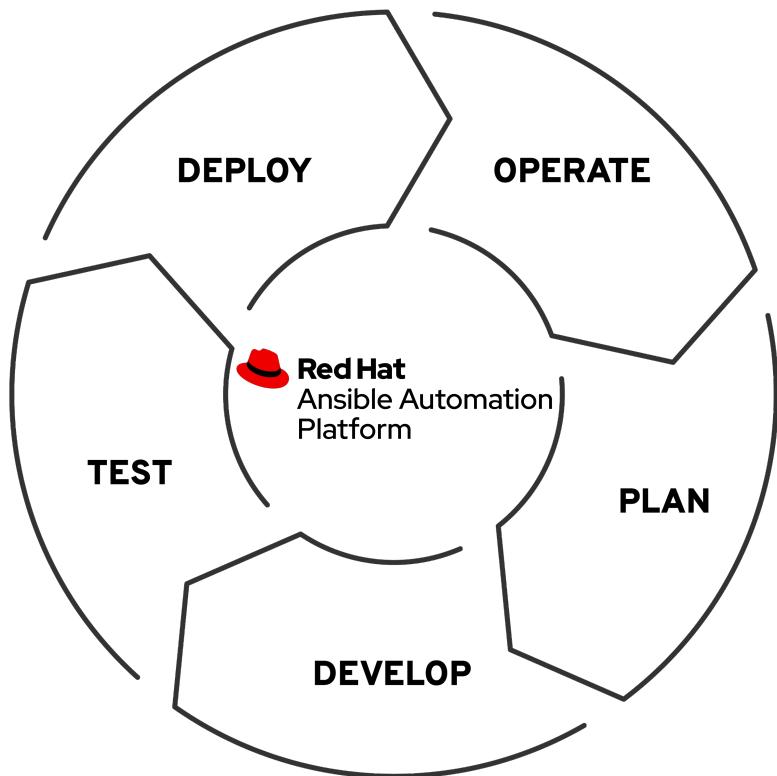
Examine operational state to check network connectivity and protocols and enhance operational workflows to help measure network intent.

# What is it for?



# Start Small, Think Big

Three high-level benefits for successful network operations



## Configuration Management

- Automate backup & restores
- Scoped Config Management

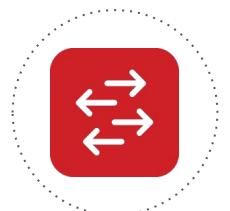
## Infrastructure Awareness

- Dynamic Documentation
- Compliance and traceability

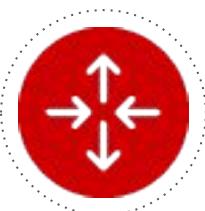
## Network Validation

- Validate operational steady-state
- Roll back if configuration changes don't meet goals

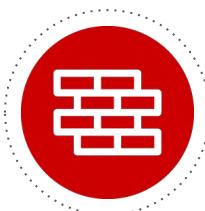
# Ansible Network Ecosystem



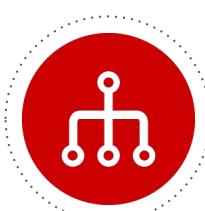
SWITCHES



ROUTERS



ENTERPRISE  
FIREWALLS



LOAD  
BALANCERS



CONTROLLERS



IP ADDRESS  
MGMT



ARISTA

aruba  
NETWORKS



Check Point  
SOFTWARE TECHNOLOGIES LTD

✓✓✓✓✓  
CISCO

DELL EMC

Infoblox  
NEXT LEVEL NETWORKING



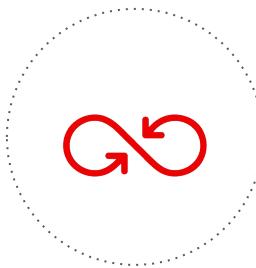
✓✓✓✓✓  
Open  
Switch

JUNIPER  
NETWORKS

VyOS

Red Hat  
Ansible Automation  
Platform

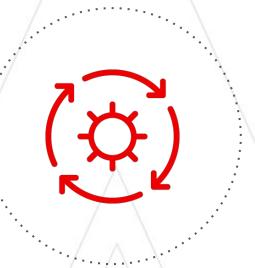
# Deep diving on use-cases



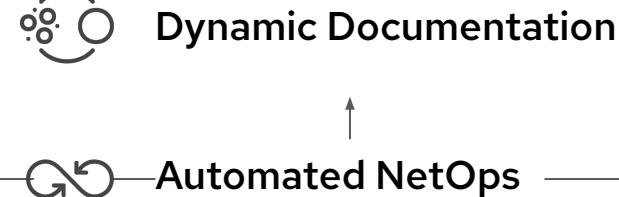
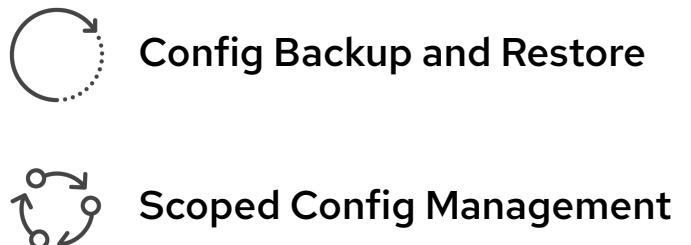
Configuration Management



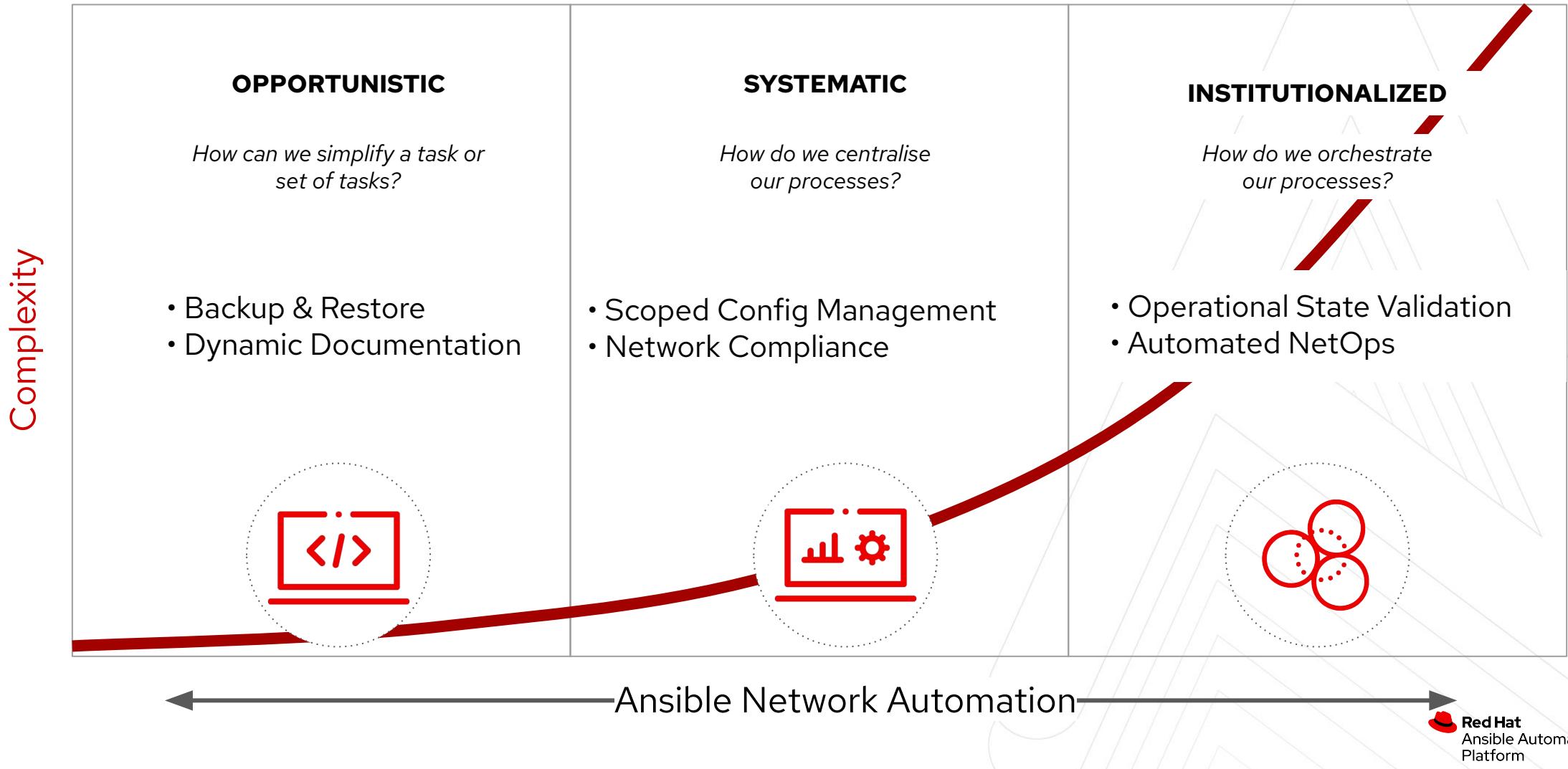
Infrastructure Awareness



Network Validation

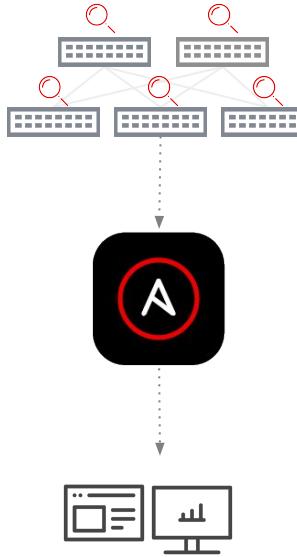
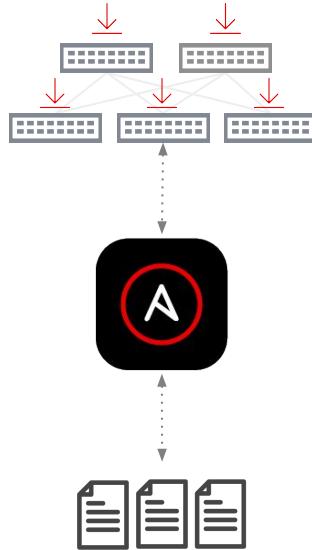


# Network Automation Journey



# Start Small

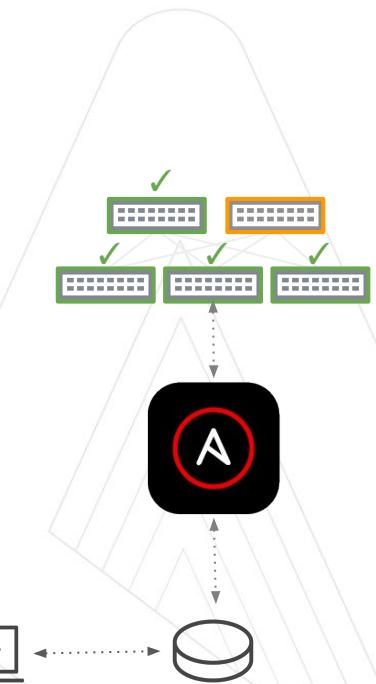
Quick automation victories for network engineers



## Config Backup and Restore

### Ubiquitous first touch use case

- Gain confidence in automation quickly
- First steps towards network as code
- Quickly recover network steady state



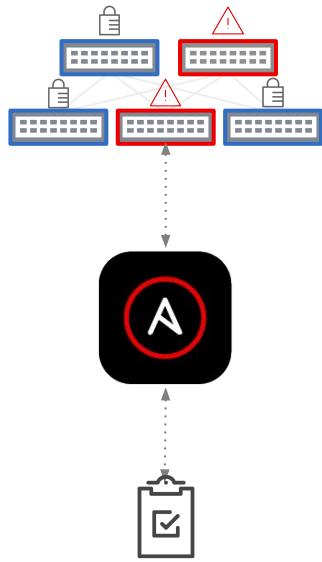
## Scoped Config Management

### Focus on high yield victories

- Automate VLANs, ACLs and SNMP config
- Introduce source of truth concepts
- Enforce Configuration policy

# Think Big

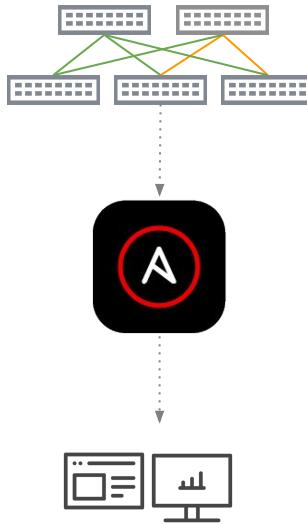
Institutionalizing automation into your organization



## Network Compliance

### Respond quickly and consistently

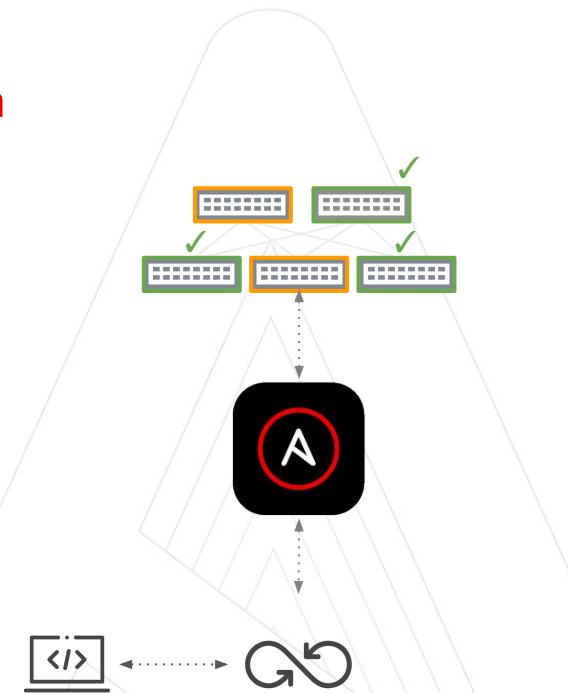
- Security and config compliance for network
- Remove human error from security responses
- Enforce Configuration policies and hardening



## Operational State Validation

### Going beyond config management

- Parsing operational state to structured values
- Schema validation and verification
- Enhance operational workflows



## Automated NetOps

### Infrastructure as code

- Data centric automation
- Deploy configuration pipelines
- GitOps for Network Automation

# Section 1

# Ansible Basics

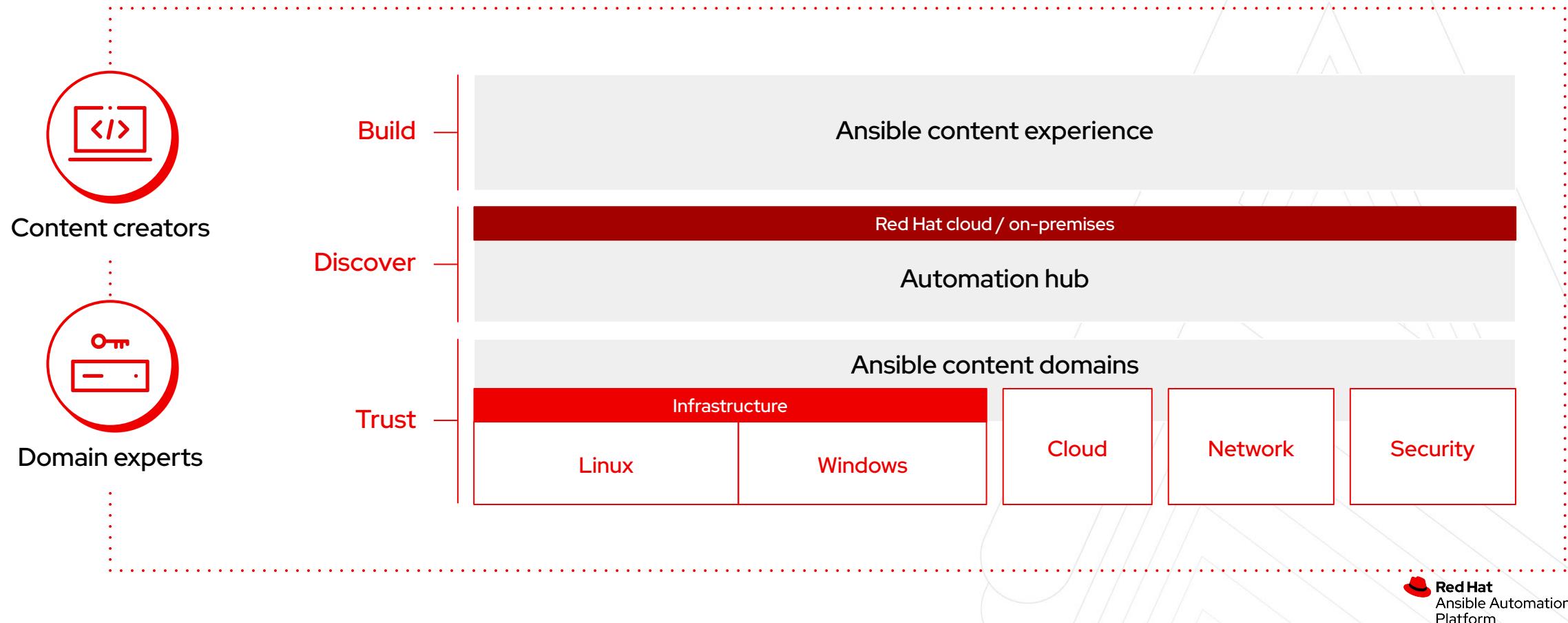
## Topics Covered:

- ▶ Understanding Inventory
- ▶ An example Ansible Playbook



# Create

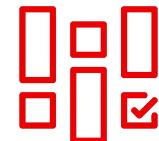
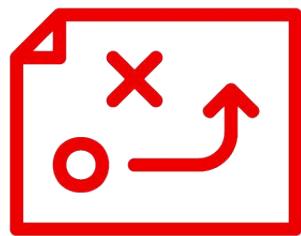
## The automation lifecycle





```
---  
- 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”



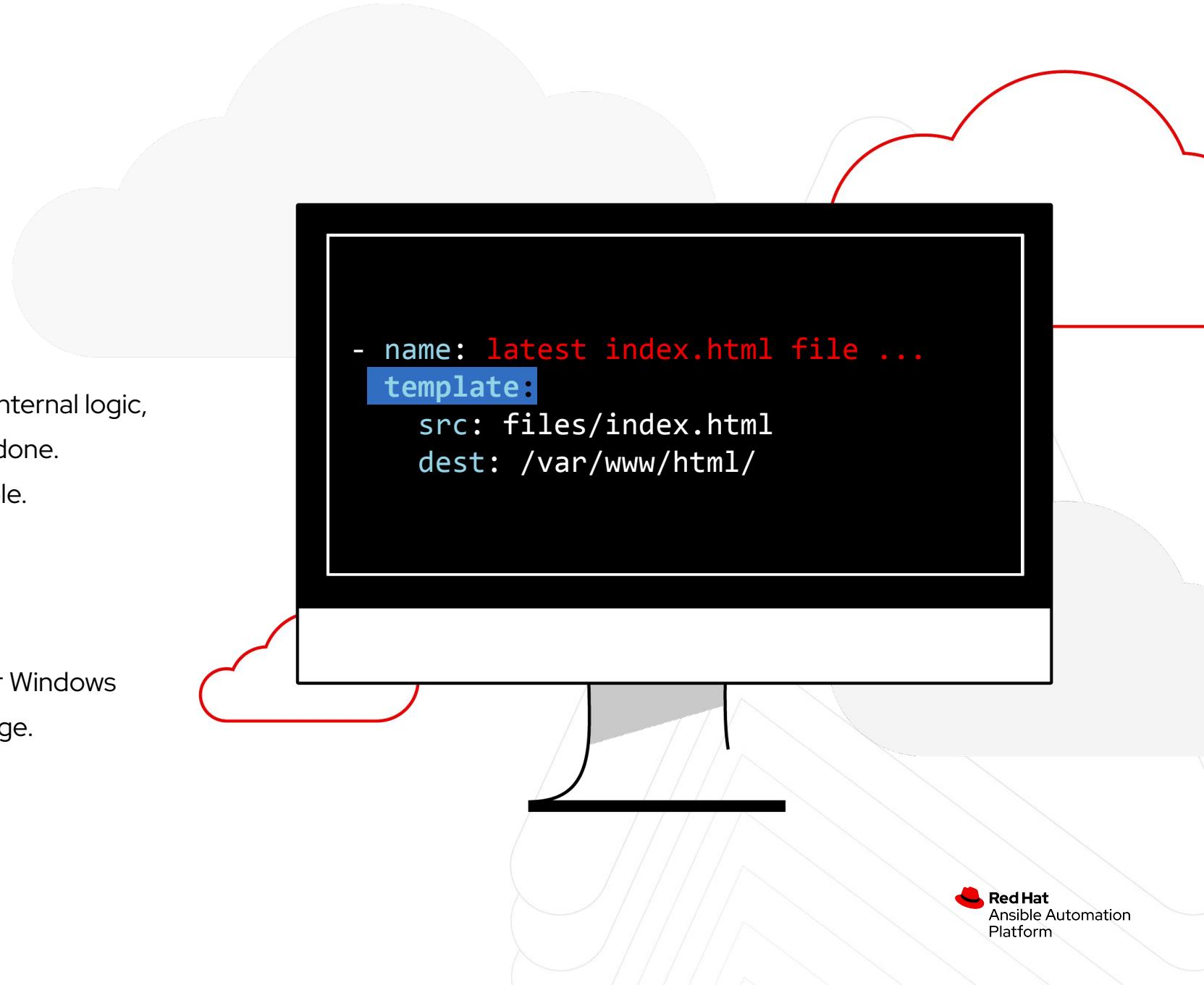
## What are they?

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.

#### 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

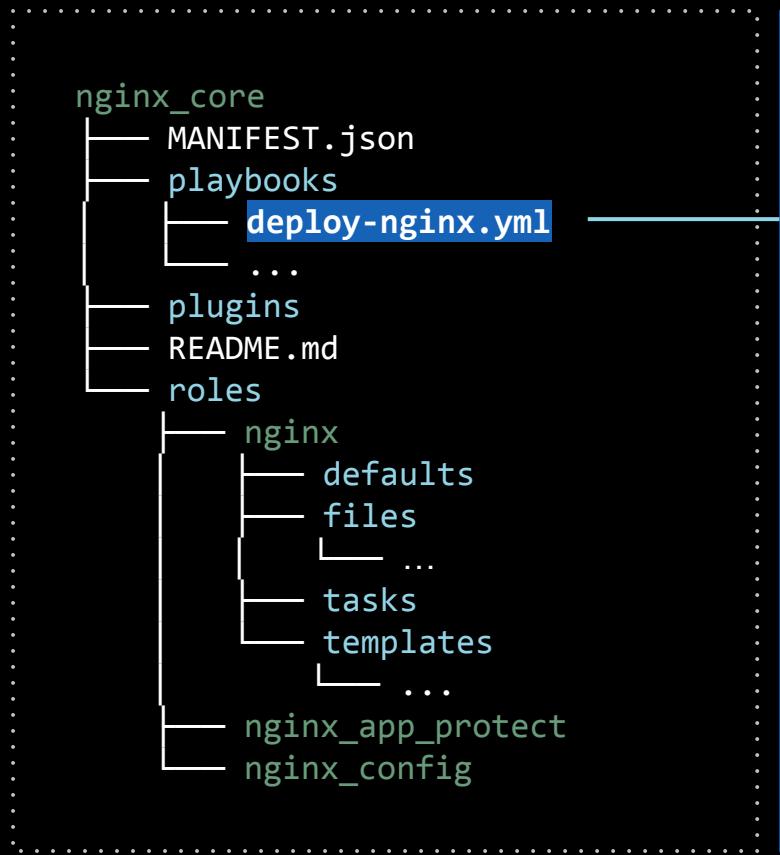


## What are they?

Collections are a data structure containing automation content:

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





```
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
```

90+  
certified platforms



Infrastructure



Cloud



Network



Security



Red Hat



ARISTA



NetApp™



CISCO™



FORTINET®

 Red Hat  
Ansible Automation  
Platform

# How is network automation different?



# Network Automation compared to servers

Module code is executed locally on the control node



Network Devices / API Endpoints

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



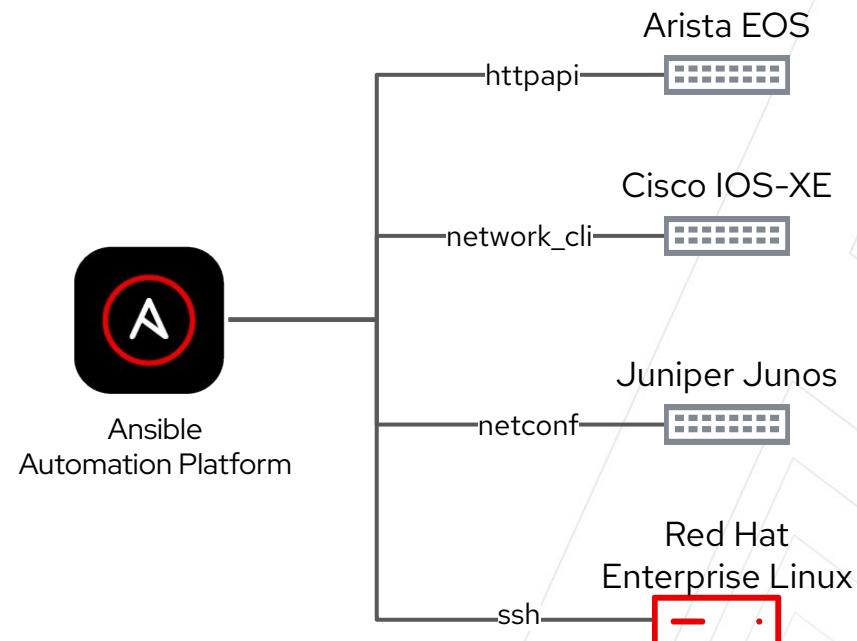
Linux / Windows Hosts

# Network Connection Plugins

Use your vendor connection of choice

## `ansible_connection`

- **netconf** - XML over netconf  
example: Juniper Junos
- **network\_cli** - command line over SSH  
example: Cisco IOS-XE, Arista EOS
- **httpapi** - vendor API  
example: Arista eAPI, Cisco NX-API



<https://docs.ansible.com/ansible/latest/plugins/connection.html>

# Understanding Inventory

```
● ● ●  
rtr1 ansible_host=18.220.156.59  
rtr2 ansible_host=18.221.53.11  
rtr3 ansible_host=13.59.242.237  
rtr4 ansible_host=3.16.82.231  
rtr5  
rtr6
```

# Understanding Inventory - Groups

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

```
[cisco]
rtr1 ansible_host=18.220.156.59 private_ip=172.16.184.164
[arista]
rtr2 ansible_host=18.221.53.11 private_ip=172.17.229.213
rtr4 ansible_host=3.16.82.231 private_ip=172.17.209.186
[juniper]
rtr3 ansible_host=13.59.242.237 private_ip=172.16.39.75
```

Groups can be nested

```
[routers:children]
cisco
juniper
arista
```

# Understanding Inventory - Variables

```
...  
[cisco]  
rtr1 ansible_host=18.220.156.59 private_ip=172.16.184.164  
[arista]  
rtr2 ansible_host=18.221.53.11 private_ip=172.17.229.213  
rtr4 ansible_host=3.16.82.231 private_ip=172.17.209.186  
[juniper]  
rtr3 ansible_host=13.59.242.237 private_ip=172.16.39.75  
  
[cisco:vars]  
ansible_user=ec2-user  
ansible_network_os=ios  
ansible_connection=network_cli
```

Host variables apply to the host and override group vars

Group variables apply for all devices in that group

# A Sample Ansible Playbook

```
---  
- name: configure VLANs  
  hosts: cisco  
  gather_facts: false  
  tasks:  
    - name: VLANs task  
      cisco.nxos.vlans:  
        config:  
        - vlan_id: 5  
          name: WEB  
        - vlan_id: 10
```

- A playbook is a list of plays.
- Each play is a list of tasks.
- Tasks invoke modules.
- A playbook can contain more than one play.

# Lab Time

## Exercise 1 - Exploring the lab environment

🔗 [red.ht/network-workshop-1](https://red.ht/network-workshop-1)

In this lab you will explore the lab environment and build familiarity with the lab inventory.

⌚ Approximate time: 10 mins

# Section 2

# Executing Ansible

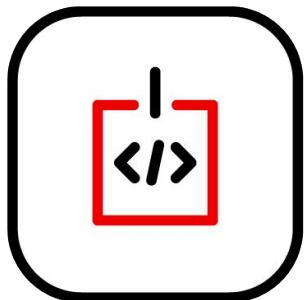
## Topics Covered:

- ▶ An Ansible Play
- ▶ Ansible Modules
- ▶ Execution Environments
- ▶ Running an Ansible Playbook

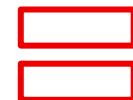


# Automation Execution Environments

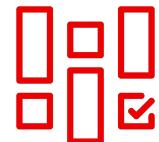
Components needed for automation, packaged in a cloud-native way



Execution  
Environments



Collections



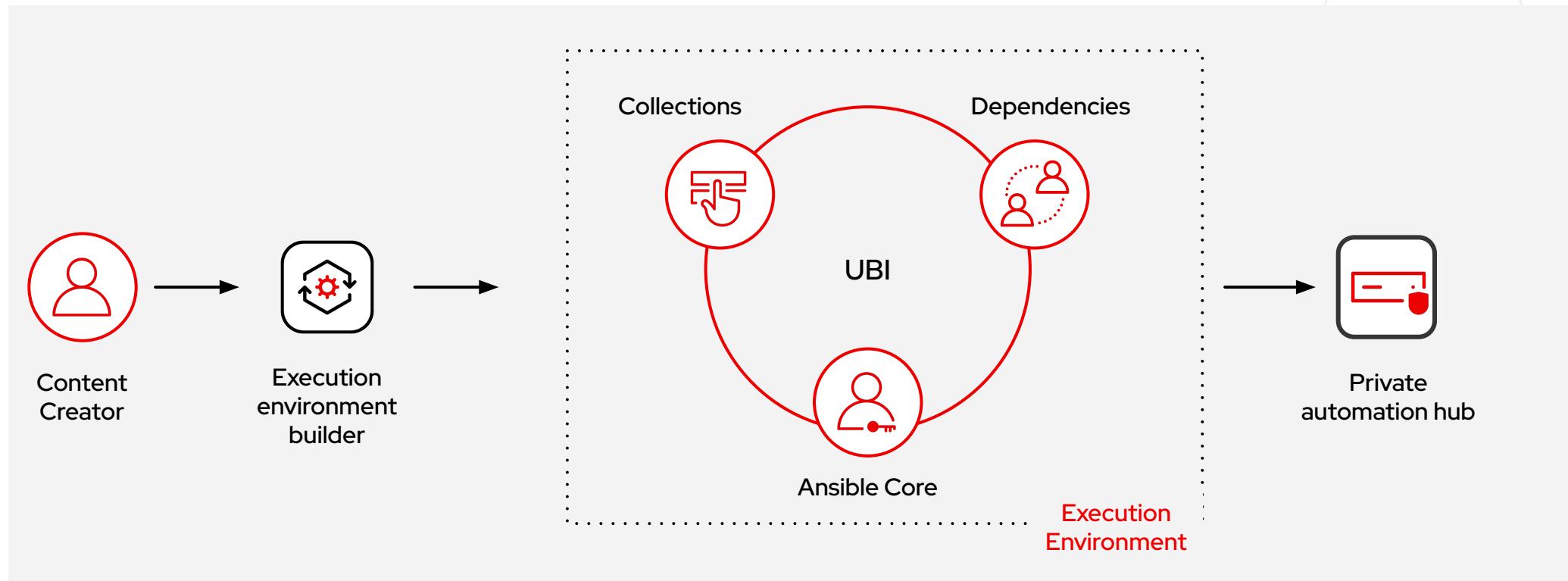
Libraries



Ansible Core

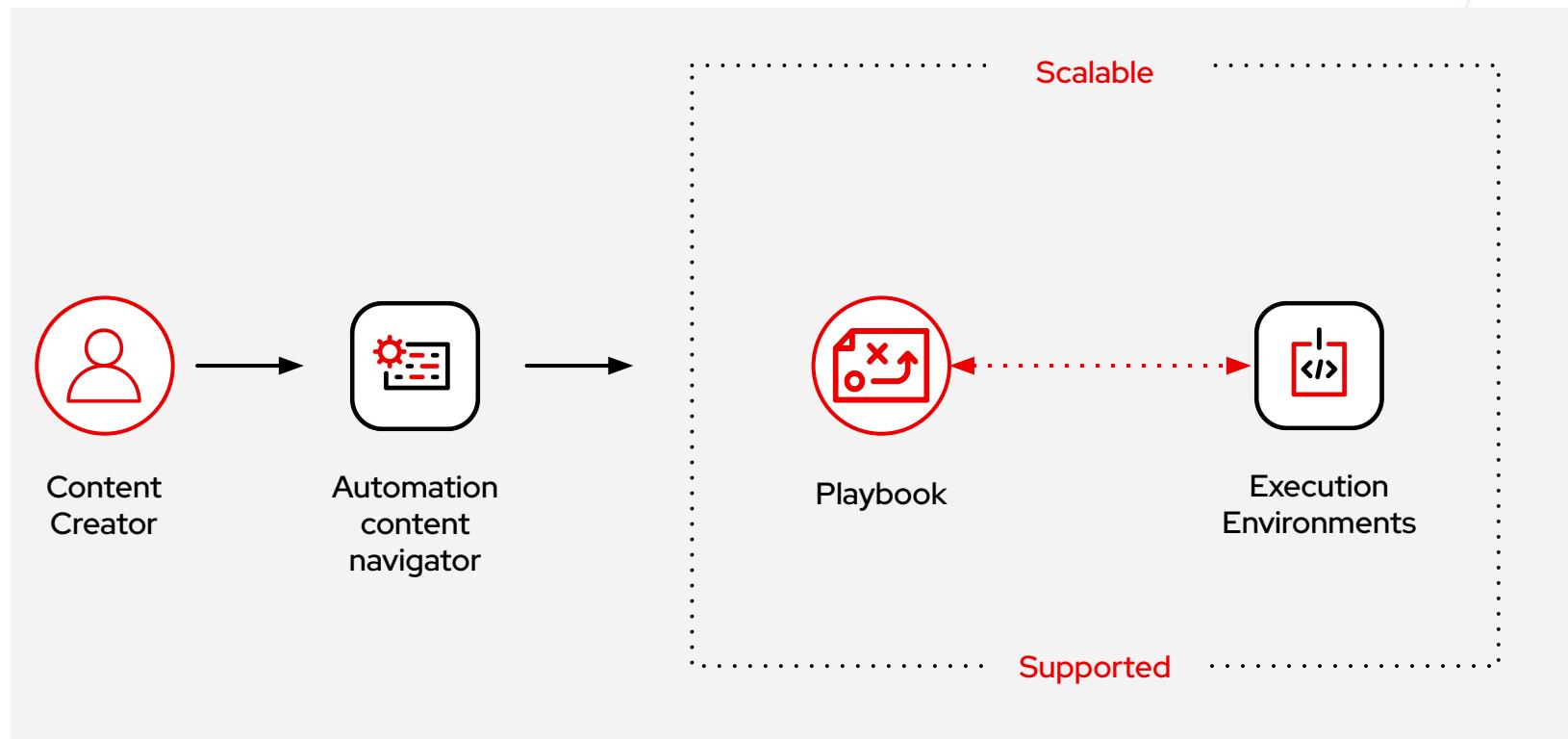
# Build, create, publish

Development cycle of an automation execution environment

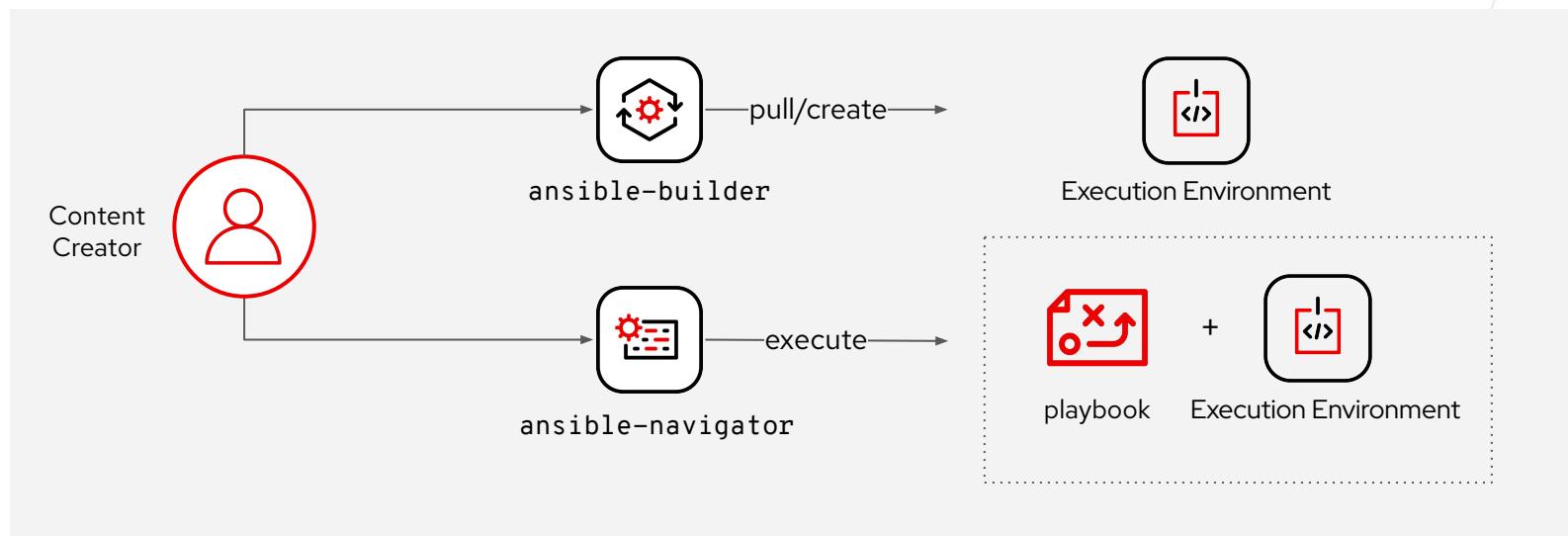


# Develop, test, run

How to develop, test and run containerized Ansible content



# Builder and Navigator



# Another Ansible Playbook Example

```
---  
- name: snmp ro/rw string configuration  
  hosts: cisco  
  gather_facts: false  
  
  tasks:  
    - name: ensure snmp strings are present  
      cisco.ios.config:  
        lines:  
          - snmp-server community ansible-public R0  
          - snmp-server community ansible-private RW
```

# Ansible Playbook - Play definition

- The **name** parameter describes the Ansible Play
- Target devices using the **hosts** parameter
- Optionally disable **gather\_facts**

```
---  
- name: snmp ro/rw string configuration  
  hosts: cisco  
  gather_facts: false
```

# Modules

Modules do the actual work in Ansible, they are what gets executed in each playbook task.

- Typically written in Python (but not limited to it)
- Modules can be idempotent
- Modules take user input in the form of parameters

```
tasks:
  - name: ensure snmp strings are present
    cisco.ios.config:
      lines:
        - snmp-server community ansible-public R0
        - snmp-server community ansible-private RW
```

# Network modules

Ansible modules for network automation typically references the vendor OS followed by the module name.

- namespace.collection.facts
- namespace.collection.command
- namespace.collection.config
- namespace.collection.resource

More modules depending on platform

Arista EOS = arista.eos.  
Cisco IOS/IOS-XE = cisco.ios  
Cisco NX-OS = cisco.nxos  
Cisco IOS-XR = cisco-iosxr  
F5 BIG-IP = f5networks.f5\_bigip\_bigip.  
Juniper Junos = junipernetworks.junos.  
VyOS = vyos.vyos.

# 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.



```
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 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

<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

# Running a playbook

```
● ● ●
---
- name: snmp ro/rw string configuration
  hosts: cisco
  gather_facts: false

  tasks:
    - name: ensure snmp strings are present
      cisco.ios.config:
        lines:
          - snmp-server community ansible-public RO
          - snmp-server community ansible-private RW
```

```
[student1@ansible networking-workshop]$ ansible-navigator playbook.yml --mode stdout

PLAY [snmp ro/rw string configuration] ****
TASK [ensure snmp strings are present] ****
changed: [rtr1]

PLAY RECAP ****
rtr1 : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

# Displaying output

```
[student1@ansible networking-workshop]$ ansible-navigator playbook.yml --mode stdout -v
Using /home/student1/.ansible.cfg as config file

PLAY [snmp ro/rw string configuration] ****
TASK [ensure that the desired snmp strings are present] ****
changed: [rtr1] => changed=true
  ansible_facts:
    discovered_interpreter_python: /usr/bin/python
  banners: {}
  commands:
    - snmp-server community ansible-public RO
    - snmp-server community ansible-private RW
  updates:
    - snmp-server community ansible-public RO
    - snmp-server community ansible-private RW

PLAY RECAP ****
rtr1      : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

**Increase the level of verbosity by adding more "v's" -vvvv**

# Lab Time

Exercise 2 - Execute your first network automation playbook

🔗 [red.ht/network-workshop-2](https://red.ht/network-workshop-2)

In this lab you will use Ansible to update the configuration of routers. This exercise will not have you create an Ansible Playbook; you will use an existing one.

⌚ Approximate time: 15 mins

# Section 3

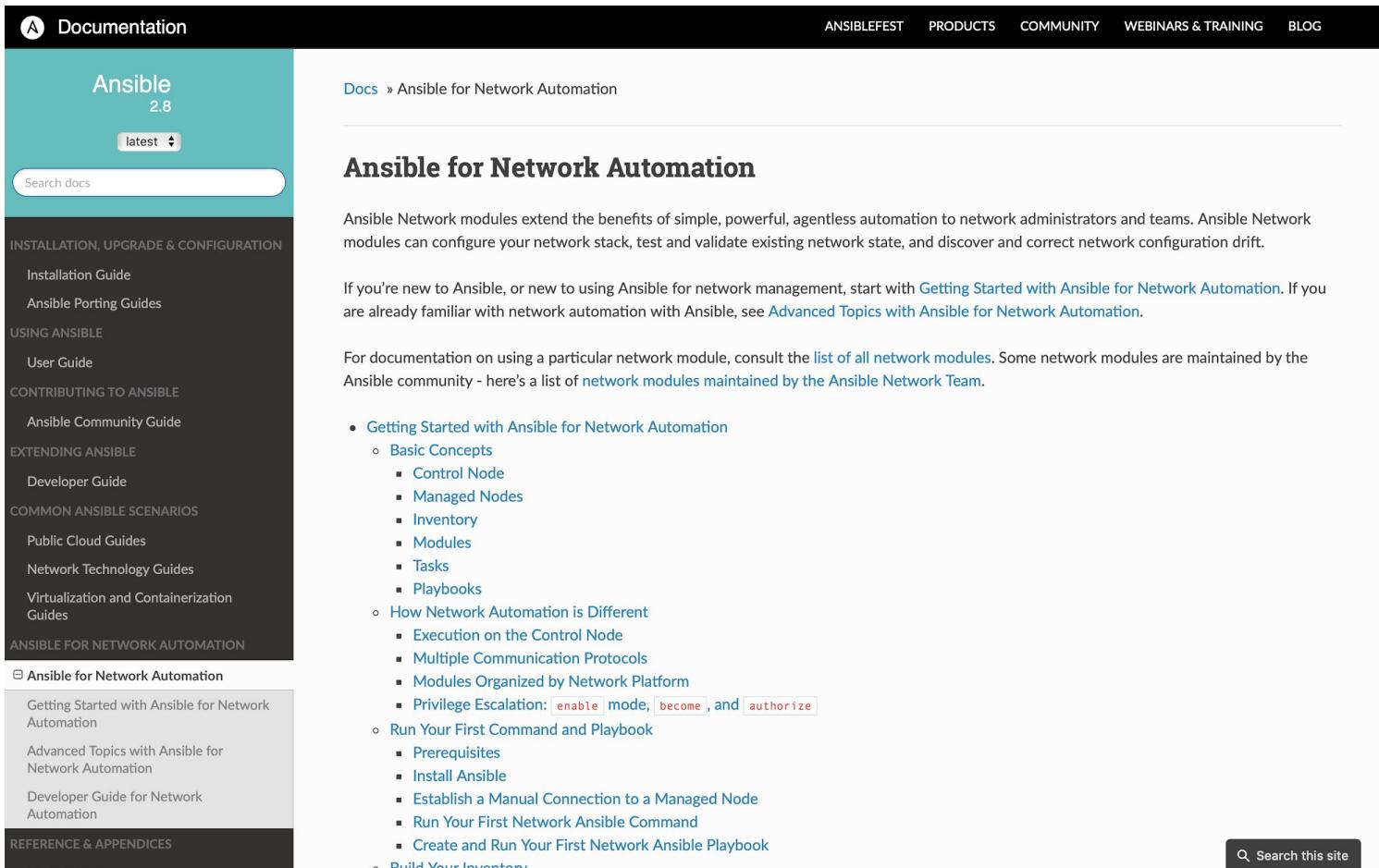
# Network Facts

## Topics Covered:

- ▶ Ansible Documentation
- ▶ Facts for Network Devices
- ▶ The debug module



# “Ansible for Network Automation” Documentation



The screenshot shows the Ansible Documentation website for Ansible 2.8. The main navigation bar includes links for ANSIBLEFEST, PRODUCTS, COMMUNITY, WEBINARS & TRAINING, and BLOG. The page title is "Ansible for Network Automation", which is also a link to the current page. The page content starts with a brief introduction: "Ansible Network modules extend the benefits of simple, powerful, agentless automation to network administrators and teams. Ansible Network modules can configure your network stack, test and validate existing network state, and discover and correct network configuration drift." It then provides links for "Getting Started with Ansible for Network Automation" and "Advanced Topics with Ansible for Network Automation". Below this, a list of network modules is provided, with a note that some are maintained by the Ansible community. The list includes:

- Getting Started with Ansible for Network Automation
  - Basic Concepts
    - Control Node
    - Managed Nodes
    - Inventory
    - Modules
    - Tasks
    - Playbooks
  - How Network Automation is Different
    - Execution on the Control Node
    - Multiple Communication Protocols
    - Modules Organized by Network Platform
    - Privilege Escalation: `enable` mode, `become`, and `authorize`
  - Run Your First Command and Playbook
    - Prerequisites
    - Install Ansible
    - Establish a Manual Connection to a Managed Node
    - Run Your First Network Ansible Command
    - Create and Run Your First Network Ansible Playbook
  - Build Your Inventory

At the bottom right of the page is a search bar with the placeholder "Search this site".

[red.ht/NetworkDocs](http://red.ht/NetworkDocs)

# Module Documentation

- Documentation is required as part of module submission
- Multiple Examples for every module
- Broken into relevant sections

Docs » Module Index

## Module Index

- [All Modules](#)
- [Cloud Modules](#)
- [Clustering Modules](#)
- [Commands Modules](#)
- [Crypto Modules](#)
- [Database Modules](#)
- [Files Modules](#)
- [Identity Modules](#)
- [Inventory Modules](#)
- [Messaging Modules](#)
- [Monitoring Modules](#)
- [Network Modules](#)
- [Notification Modules](#)
- [Packaging Modules](#)
- [Remote Management Modules](#)
- [Source Control Modules](#)
- [Storage Modules](#)
- [System Modules](#)
- [Utilities Modules](#)
- [Web Infrastructure Modules](#)
- [Windows Modules](#)

### service - Manage services.

- Synopsis
- Options
- Examples
  - Status
  - Support

#### Synopsis

● Controls services on remote hosts. Supported init systems include BSD init, OpenRC, SysV, Solaris SMF, systemd, upstart.

#### Options

parameter	required	default	choices	comments
arguments	no			Additional arguments provided on the command line aliases: args
enabled	no		• yes • no	Whether the service should start on boot. At least one of state and enabled are required.
name	yes			Name of the service.
pattern	no			If the service does not respond to the status command, name a substring to look for as would be found in the output of the <code>ps</code> command as a stand-in for a status result. If the string is found, the service will be assumed to be running.
runlevel	no	default		For OpenRC init scripts (ex: Gentoo) only. The runlevel that this service belongs to.
sleep (added in 1.3)	no			If the service is being <code>restarted</code> then sleep this many seconds between the stop and start command. This helps to workaround badly behaving init scripts that exit immediately after signaling a process to stop.
state	no		• started • stopped • restarted • reloaded	<code>started</code> / <code>restarted</code> are atomic actions that will not run commands unless necessary. <code>restarted</code> will always bounce the service. <code>reloaded</code> will always reload. At least one of state and enabled are required. Note that <code>reloaded</code> will start the service if it is not already started, even if your chosen init system wouldn't normally.
use (added in 2.2)	no	auto		The service module actually uses system specific modules, normally through auto detection, this setting can force a specific module. Normally it uses the value of the <code>'ansible_service_mgr'</code> fact and falls back to the old <code>'service'</code> module when none matching is found.

<https://docs.ansible.com/>

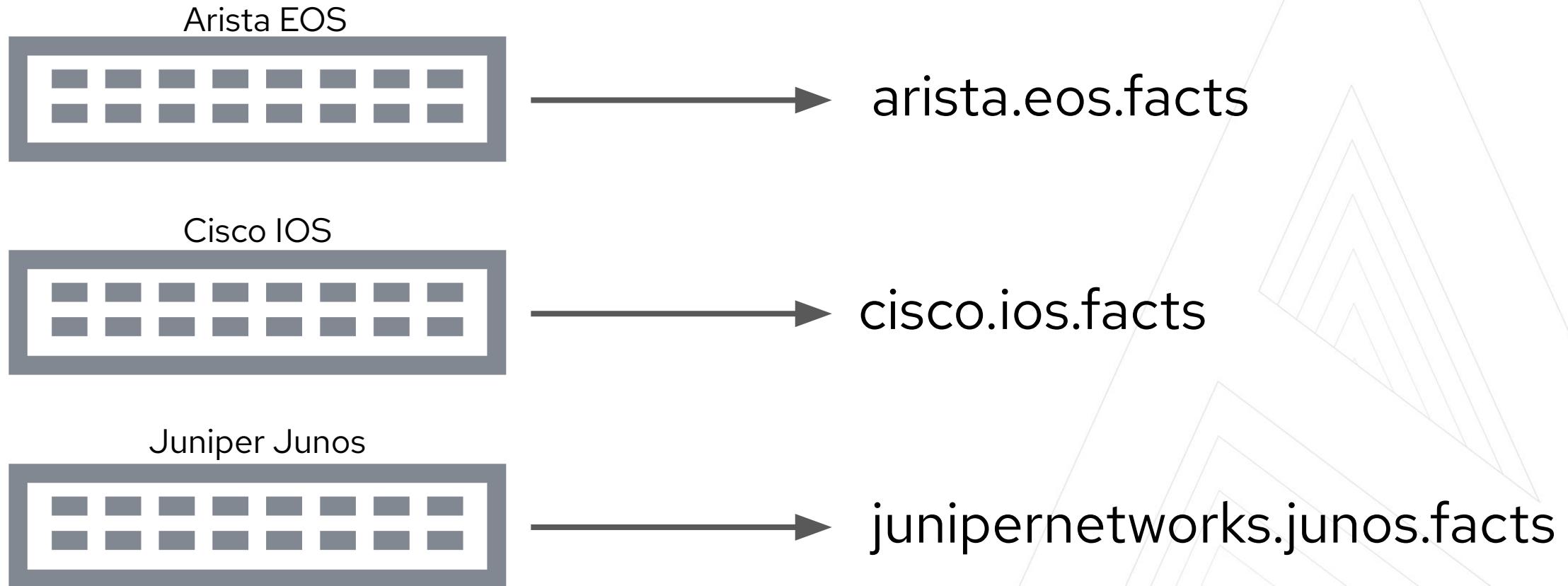
# Accessing the Ansible docs

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
.
.
.
.
.
```

# Fact modules



# What are facts?

Structured data, the Ansible way

```
cisco# show version
Cisco IOS XE Software, Version 16.09.02
Cisco IOS Software [Fuji], Virtual XE Software
(X86_64_LINUX_IOSD-UNIVERSALK9-M), Version 16.9.2,
RELEASE SOFTWARE (fc4)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
```

<<rest of output removed for slide brevity>>

Cisco IOS output

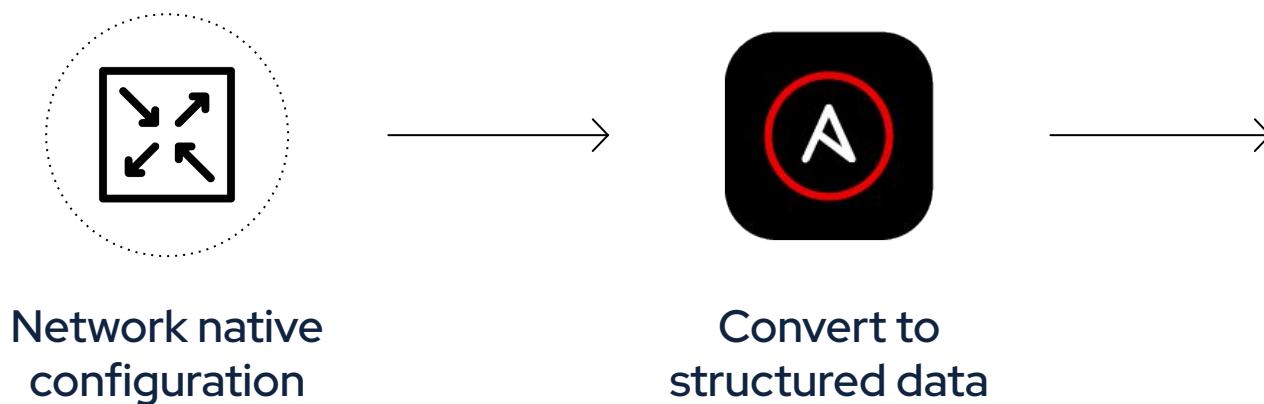
```
cisco# ansible -m ios_facts cisco
cisco | SUCCESS => {
    "ansible_facts": {
        "ansible_net_iostype": "IOS-XE",
        "ansible_net_version": "16.09.02",
        "ansible_net_serialnum": "9L8KQ482JFZ",
        "ansible_net_model": "CSR1000V",
```

<<rest of output removed for slide brevity>>

Ansible output

# Ansible Automation Platform facts

Network automation begins and ends with **facts**



```
ansible_facts": {  
    "ansible_net_iostype": "IOS-XE",  
    "ansible_net_version": "16.09.02",  
    "ansible_net_serialnum": "9L8KQ482JFZ",  
    "ansible_net_model": "CSR1000V",  
  
<<rest of output removed for brevity>>
```

# Displaying output - The “debug” module

The **debug** module is used like a "print" statement in most programming languages. Variables are accessed using "{{ }}"- quoted curly braces

```
...  
- name: display version  
  debug:  
    msg: "The IOS version is: {{ ansible_net_version }}"  
  
- name: display serial number  
  debug:  
    msg: "The serial number is: {{ ansible_net_serialnum }}"
```

# Working with Ansible facts

## 1. Gather facts

```
- name: gather eos facts
arista.eos.facts:
  gather_subset: config
  gather_network_resources: vlans
```



## 2. Use facts

```
- name: print out vlans
debug:
  var: ansible_network_resources.vlans
```

or

```
- name: gather eos facts
arista.eos.vlans:
  state: gathered
registered: vlanfacts
```



```
- name: print out vlans
debug:
  var: vlanfacts
```

# Simple and common approach

Arista EOS



Cisco IOS-XE



Juniper Junos



```
---  
- name: retrieve eos facts  
  arista.eos.facts:  
    gather_subset: config  
    gather_network_resources: all
```

```
---  
- name: retrieve ios facts  
  cisco.ios.facts:  
    gather_subset: config  
    gather_network_resources: all
```

```
---  
- name: retrieve junos facts  
  junipernetworks.junos.facts:  
    gather_subset: config  
    gather_network_resources: all
```

# Working with Ansible facts

## 2. Use facts

```
  - name: print out vlans
    debug:
      var: ansible_network_resources.vlans
```

..... or .....

```
  - name: print out vlans
    debug:
      var: vlanfacts
```

3 Displayed Results

```
  - name: dmz
    state: active
    vlan_id: 5
  - name: voip
    state: active
    vlan_id: 10
  - name: desktop
    state: active
    vlan_id: 30
```

playbook

terminal output window

# Running the Ansible Playbook with verbosity

```
$ ansible-navigator run facts.yml --mode stdout

PLAY [gather information from routers] *****

TASK [gather router facts] *****
ok: [rtr1]

TASK [display version] *****
ok: [rtr1] =>
  msg: 'The IOS version is: 16.09.02'

TASK [display serial number] *****
ok: [rtr1] =>
  msg: The serial number is: 964A1H0D1RM

PLAY RECAP *****
rtr1      : ok=3      changed=0      unreachable=0      failed=0      skipped=0      rescued=0      ignored=0
```

# Structured data is malleable

Create customized network reports

```
ansible_facts:
  ansible_net_api: cliconf
  ansible_net_fqdn: rtr2
  ansible_net_gather_network_resources:
    - interfaces
  ansible_net_gather_subset:
    - default
  ansible_net_hostname: rtr2
  ansible_net_image: flash:EOS.swi
  ansible_net_model: vEOS
  ansible_net_python_version: 2.7.5
  ansible_net_serialnum:
D00E130991A37B49F970714D8CCF7FCB
  ansible_net_system: eos
  ansible_net_version: 4.22.0F
  ansible_network_resources:
    interfaces:
      - enabled: true
        name: Ethernet1
      - enabled: true
        name: Loopback0
<<rest of output removed for slide brevity>>
```



Ansible Automation  
Platform



Customized  
Report

# Build reports with Ansible Facts

Hostname	Model Type	Mgmt0 IP Address	Code Version
n9k	Nexus9000 9000v Chassis	192.168.2.3	7.0(3)I7(1)
n9k2	Nexus9000 9000v Chassis	192.168.2.4	7.0(3)I7(1)
n9k3	Nexus9000 9000v Chassis	192.168.2.5	7.0(3)I7(1)
n9k4	Nexus9000 9000v Chassis	192.168.2.6	7.0(2)I7(1)
n9k5	Nexus9000 9000v Chassis	192.168.2.7	7.0(3)I7(1)
n9k6	Nexus9000 9000v Chassis	192.168.2.8	7.0(3)I7(1)

# Lab Time

## Exercise 3 - Ansible Facts

🔗 [red.ht/network-workshop-3](https://red.ht/network-workshop-3)

Demonstration use of Ansible facts on network infrastructure.

⌚ Approximate time: 15 mins

# Section 4

# Resource Modules



## Topics Covered:

- ▶ Resource modules
- ▶ state: merged
- ▶ state: gathered

# Network Automation Modules

How do we interact with network devices?

command



run arbitrary commands

facts



retrieve information

config



generic catch-all configuration  
and templating

resource



read and configure specific  
network resources

# Network Automation Modules

How do we interact with network devices?

command



namespace.collection.**command**  
Cisco IOS -> cisco.ios.command

facts



namespace.collection.**facts**  
Arista EOS -> arista.eos.facts

config



namespace.collection.**config**  
Juniper Junos-> junipernetworks.junos.config

resource

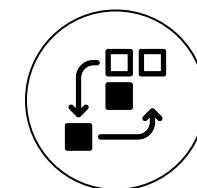


namespace.collection.**module**  
Cisco IOS-XR-> cisco.iosxr.acls

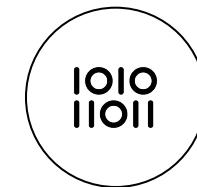
## Network resource modules

Managing device state across different devices and types

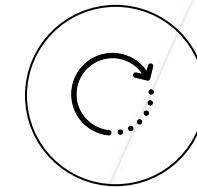
# Configuration to code



Built-in logic with commands  
and orchestration



Vendor-agnostic data model



Bidirectional with configuration to  
facts and facts to configuration

# Lab Time

## Exercise 4 - Ansible Network Resource Modules

[red.ht/network-workshop-4](https://red.ht/network-workshop-4)

This exercise will cover configuring VLANs on Arista EOS by building an Ansible Playbook using the `arista.eos.vlans` module.

Approximate time: 15 mins

# Section 5

# Automation controller

Topics Covered:

- ▶ What is Automation controller?
- ▶ Enterprise Features





## Red Hat Ansible Automation Platform



Content creators



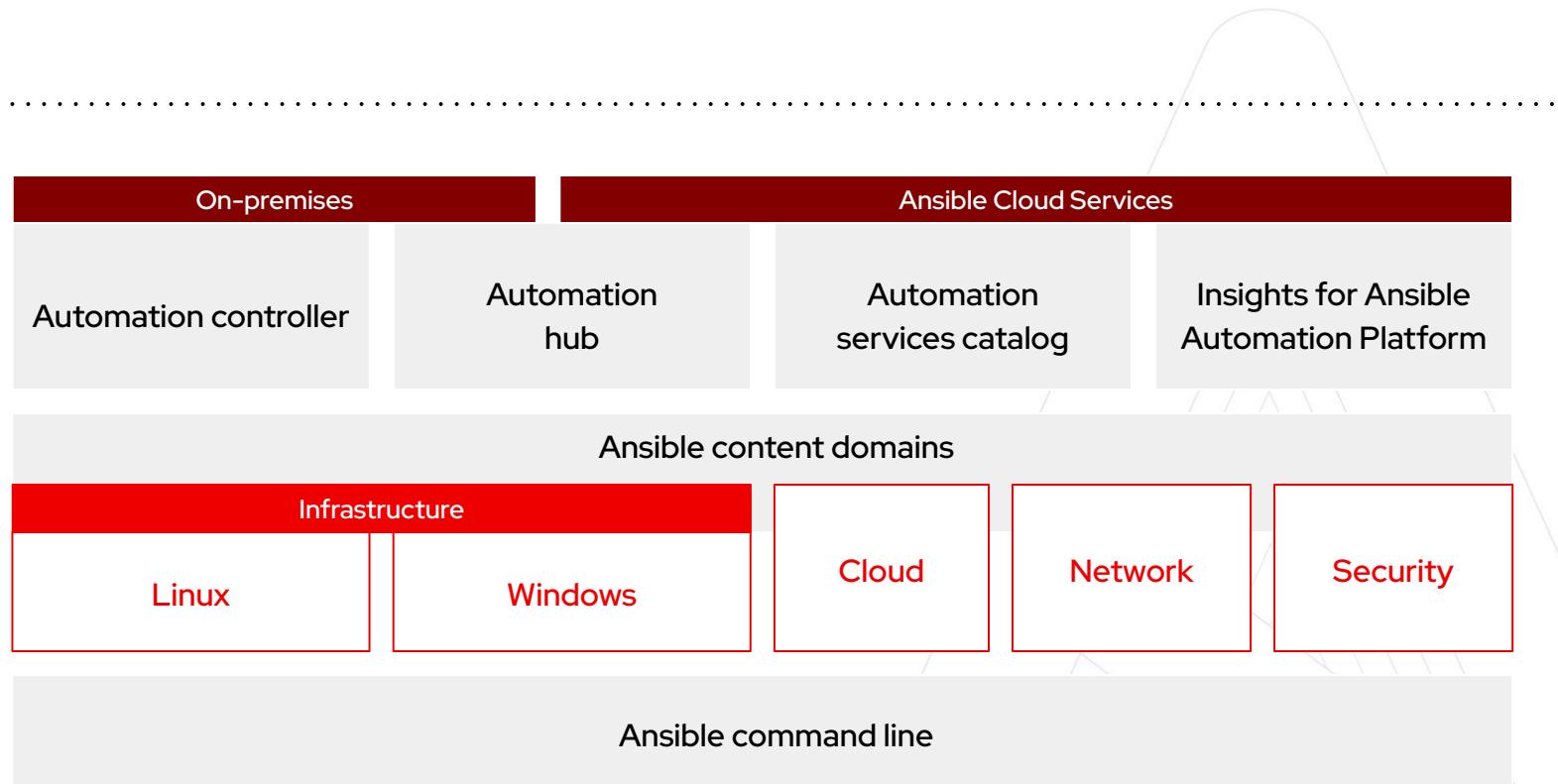
Operators



Domain experts



Users

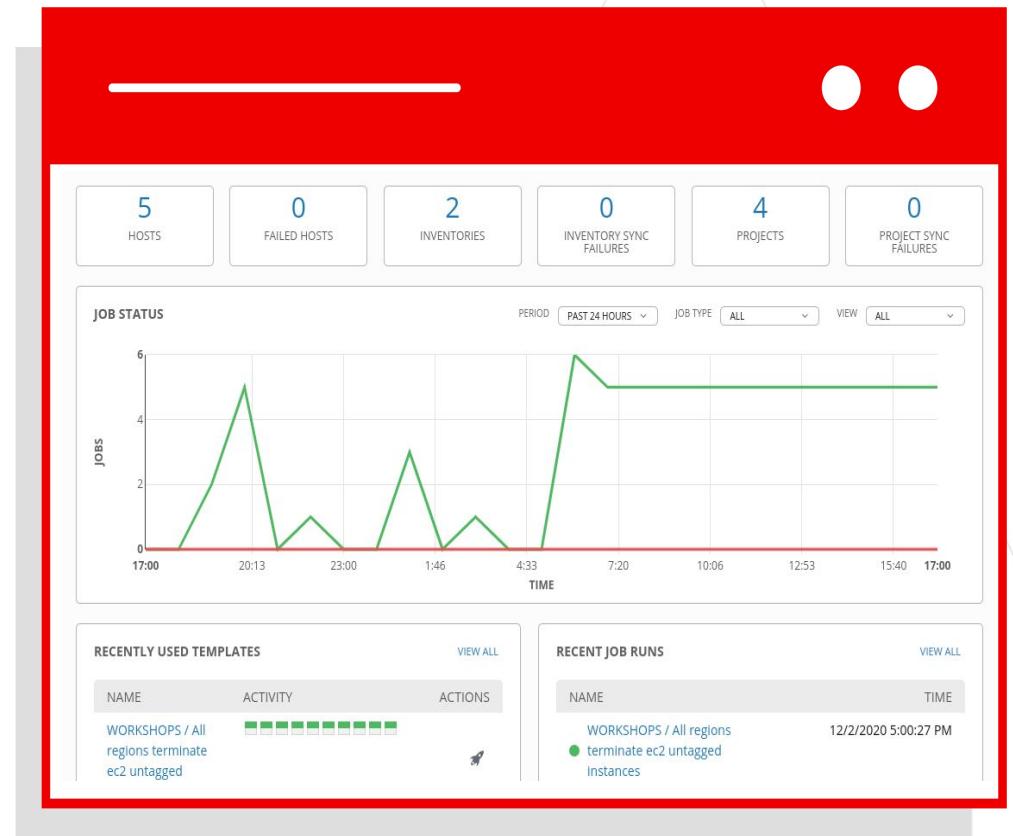


Fueled by an  
open source community

# 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



# 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.

# Lab Time

## Exercise 5: Explore Automation controller

🔗 [red.ht/network-workshop-5](https://red.ht/network-workshop-5)

Explore and understand the Automation controller lab environment.

⌚ Approximate time: 15 mins

# Section 6

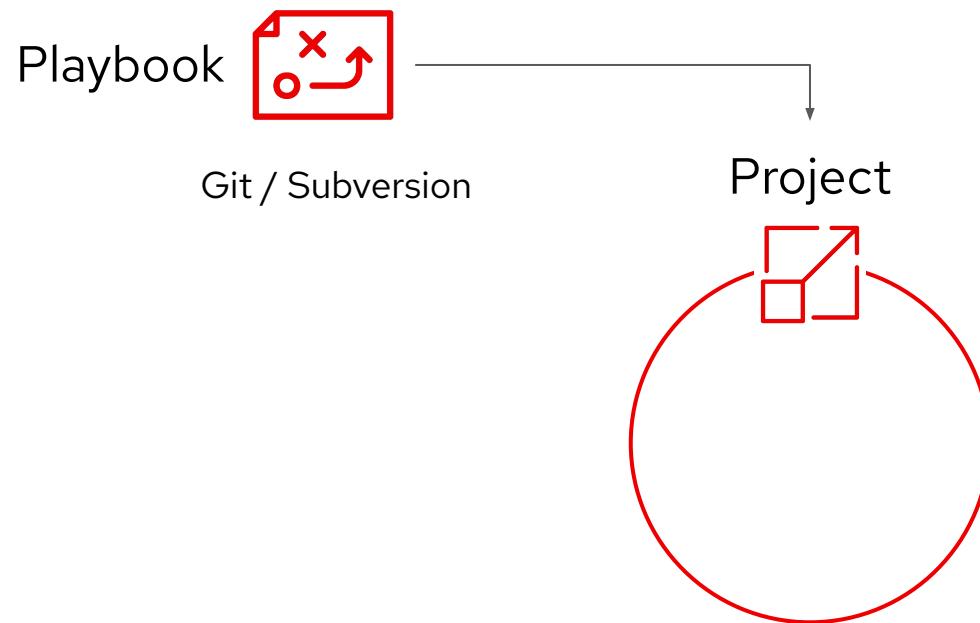
# Job Templates

Topics Covered:

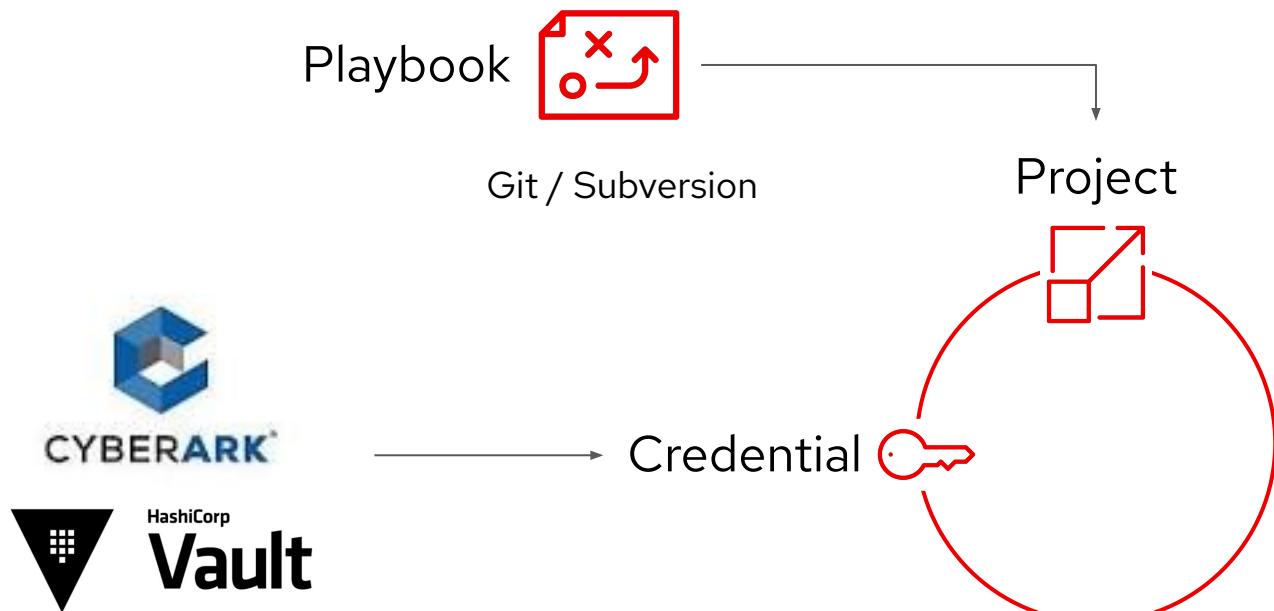
- ▶ Job Templates
  - Inventory
  - Credentials
  - Projects



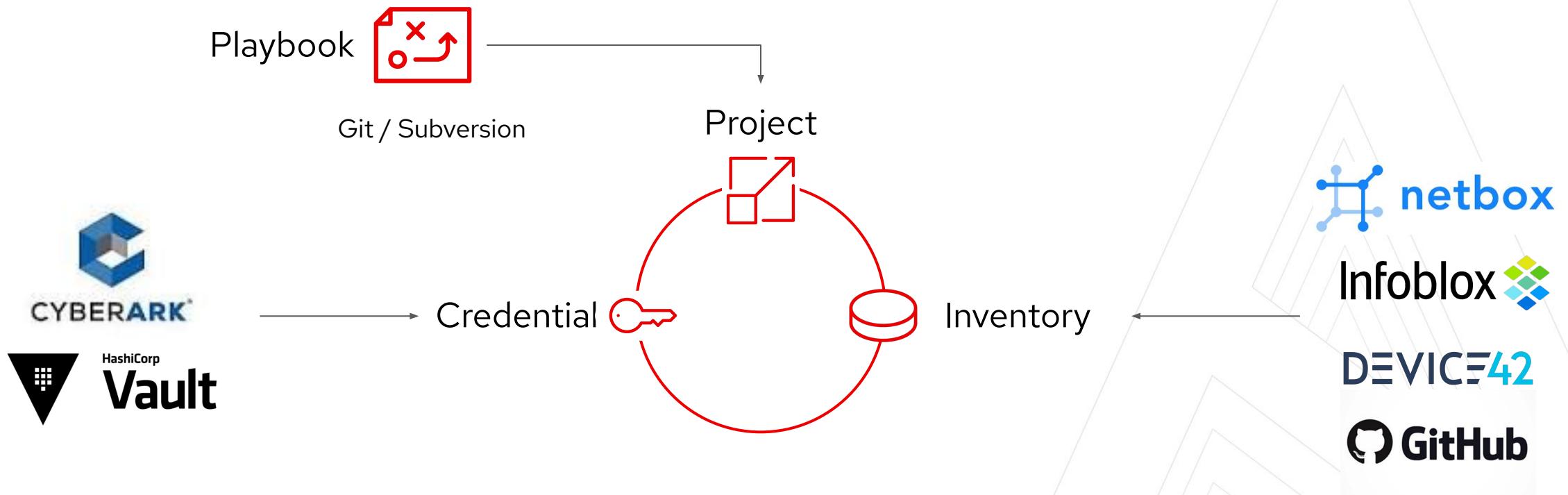
# Anatomy of an Automation Job



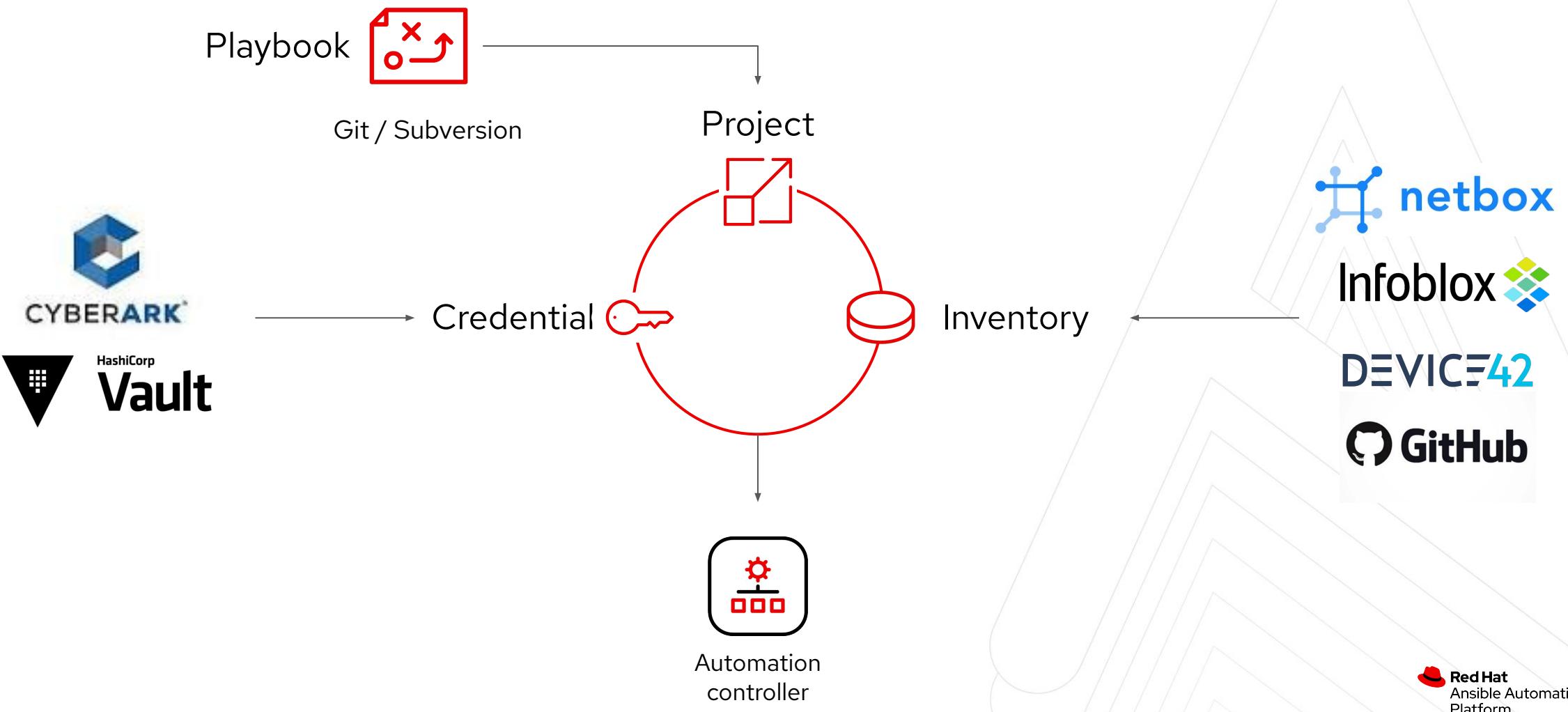
# Anatomy of an Automation Job



# Anatomy of an Automation Job



# Anatomy of an Automation Job



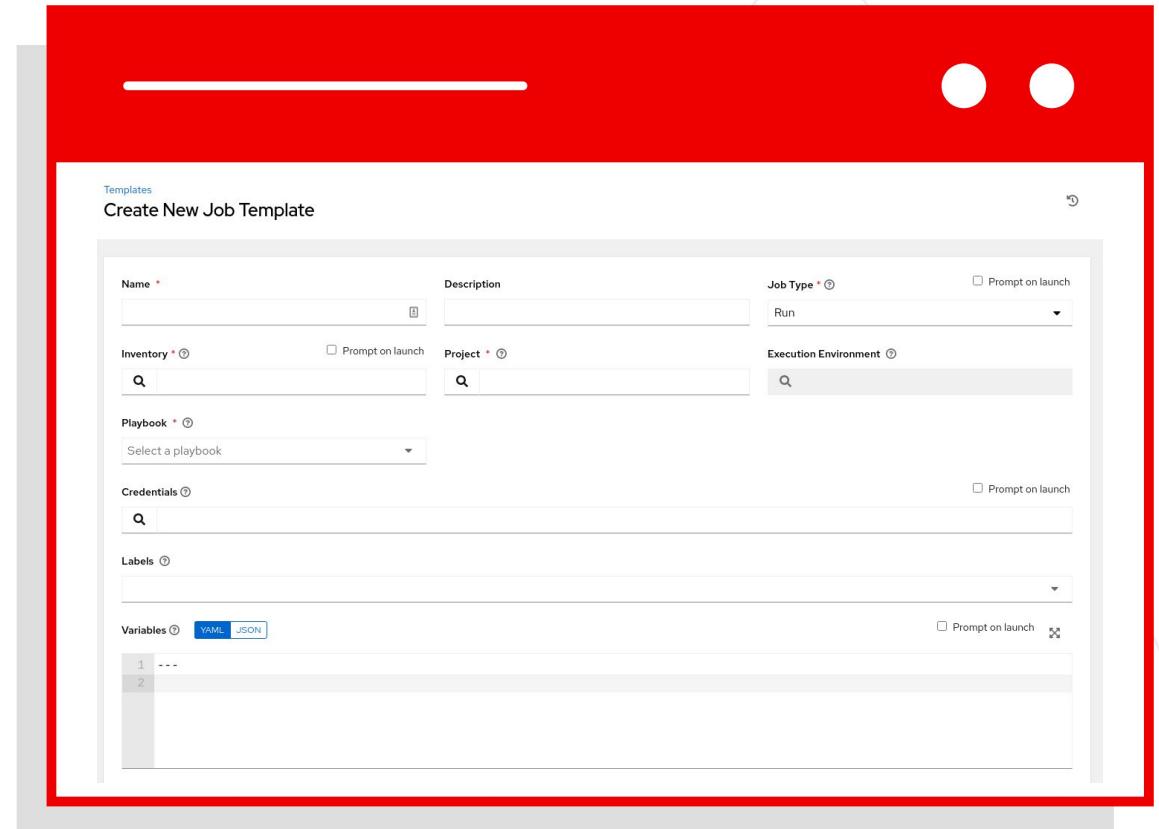
# Job Templates

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:

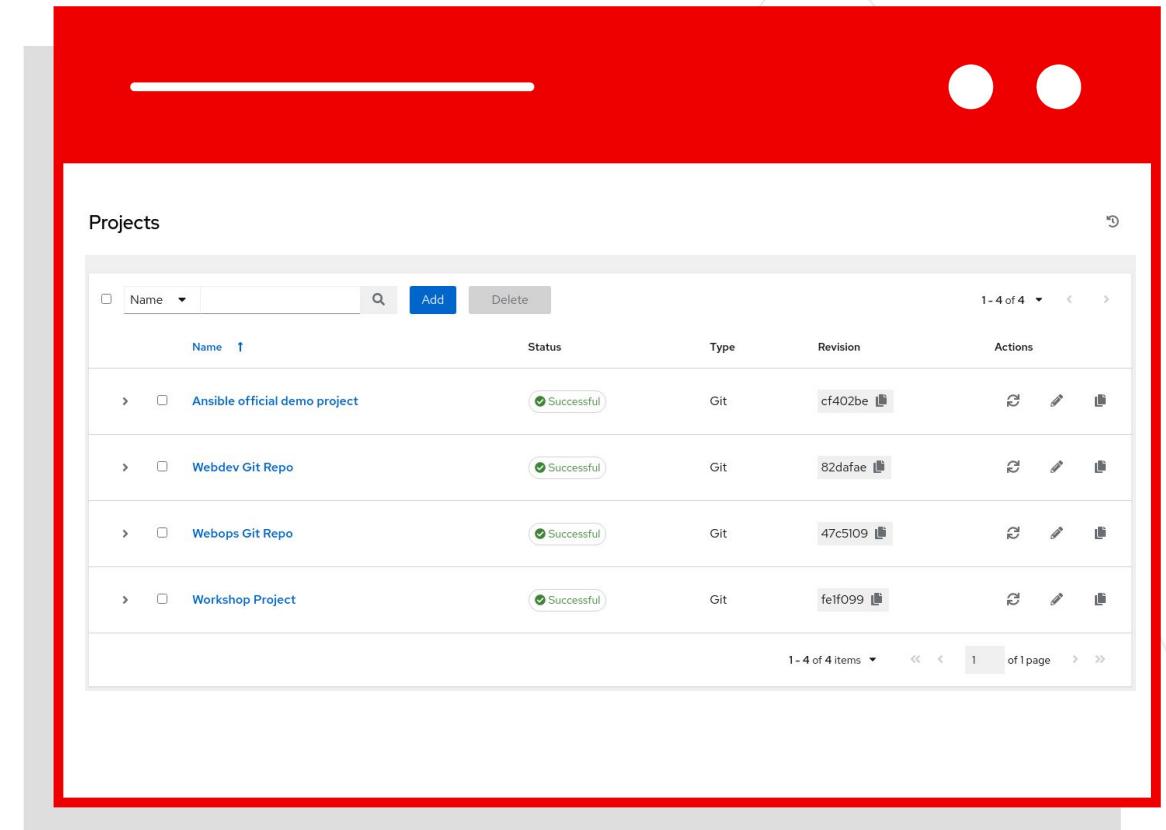
- ▶ A **Project** which contains Ansible Playbooks
- ▶ An **Inventory** to run the job against
- ▶ A **Credential** to login to devices.



# 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.

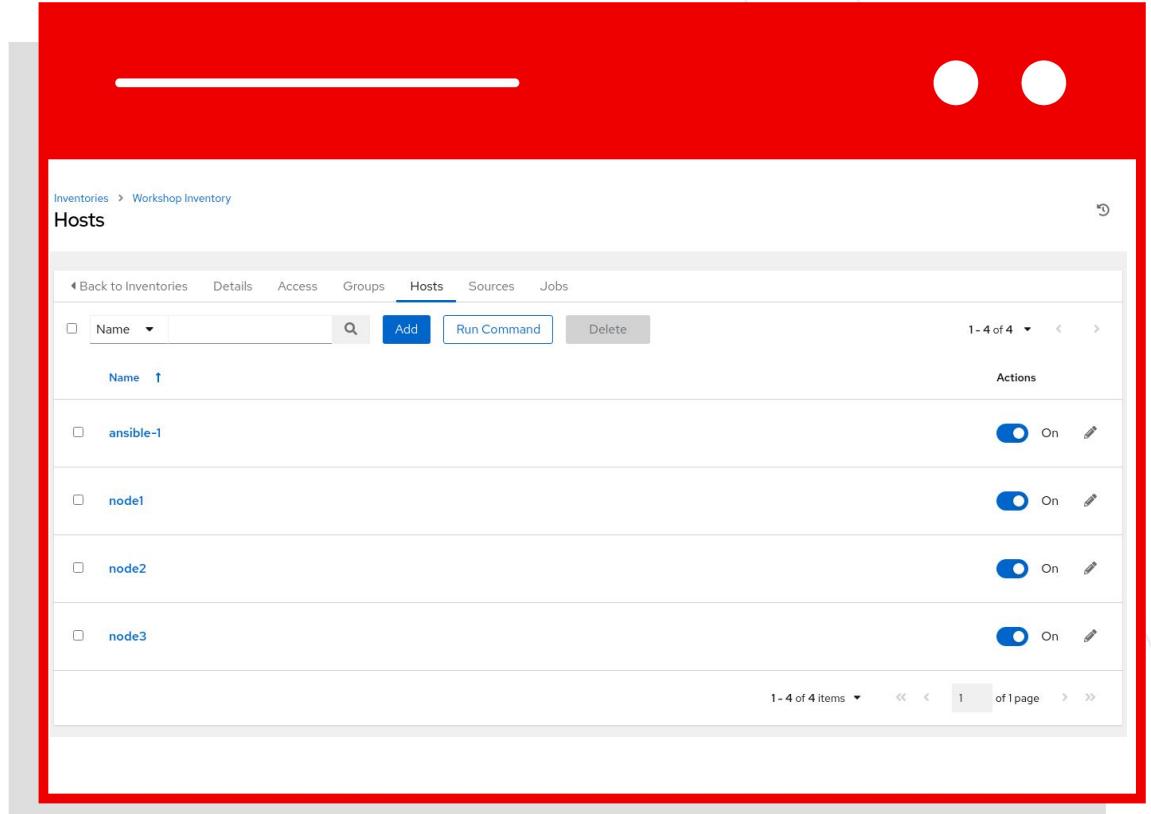


Name	Status	Type	Revision	Actions
Ansible official demo project	Successful	Git	cf402be	  
Webdev Git Repo	Successful	Git	82dafaef	  
Webops Git Repo	Successful	Git	47c5109	  
Workshop Project	Successful	Git	fe1f099	  

# 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



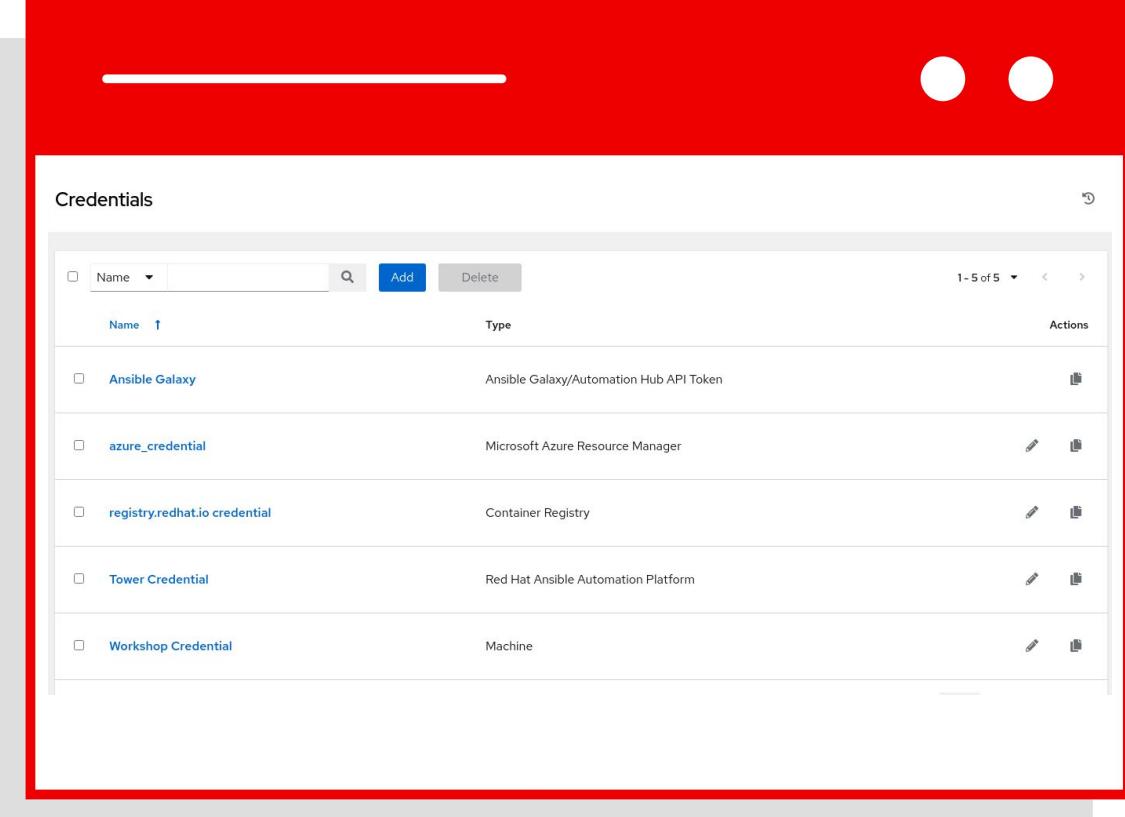
The screenshot shows a web-based interface for managing inventory. The top navigation bar includes 'Inventories', 'Workshop Inventory', and tabs for 'Hosts', 'Details', 'Access', 'Groups', 'Sources', and 'Jobs'. The 'Hosts' tab is selected. Below the tabs is a search bar with a dropdown for 'Name' and a 'Run Command' button. The main area displays a list of four hosts: 'ansible-1', 'node1', 'node2', and 'node3'. Each host entry includes a checkbox, a name, and an 'Actions' column with a switch (set to 'On') and an edit icon. At the bottom of the list, there is a pagination indicator showing '1-4 of 4 items' and a '1 of 1 page' button.

# 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

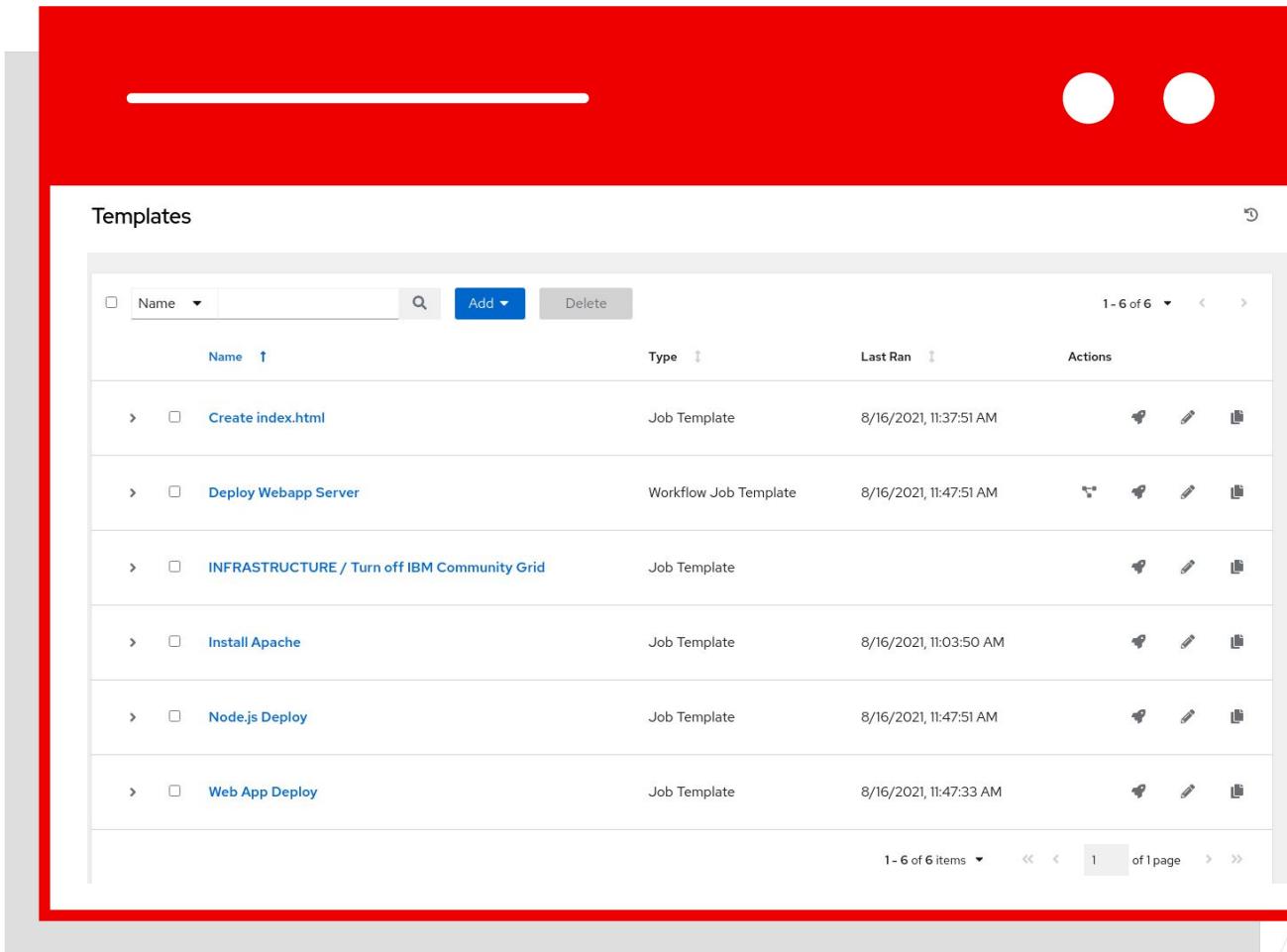
Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.



Name	Type	Actions
<a href="#">Ansible Galaxy</a>	Ansible Galaxy/Automation Hub API Token	 
<a href="#">azure_credential</a>	Microsoft Azure Resource Manager	 
<a href="#">registry.redhat.io credential</a>	Container Registry	 
<a href="#">Tower Credential</a>	Red Hat Ansible Automation Platform	 
<a href="#">Workshop Credential</a>	Machine	 

# Expanding on Job Templates

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

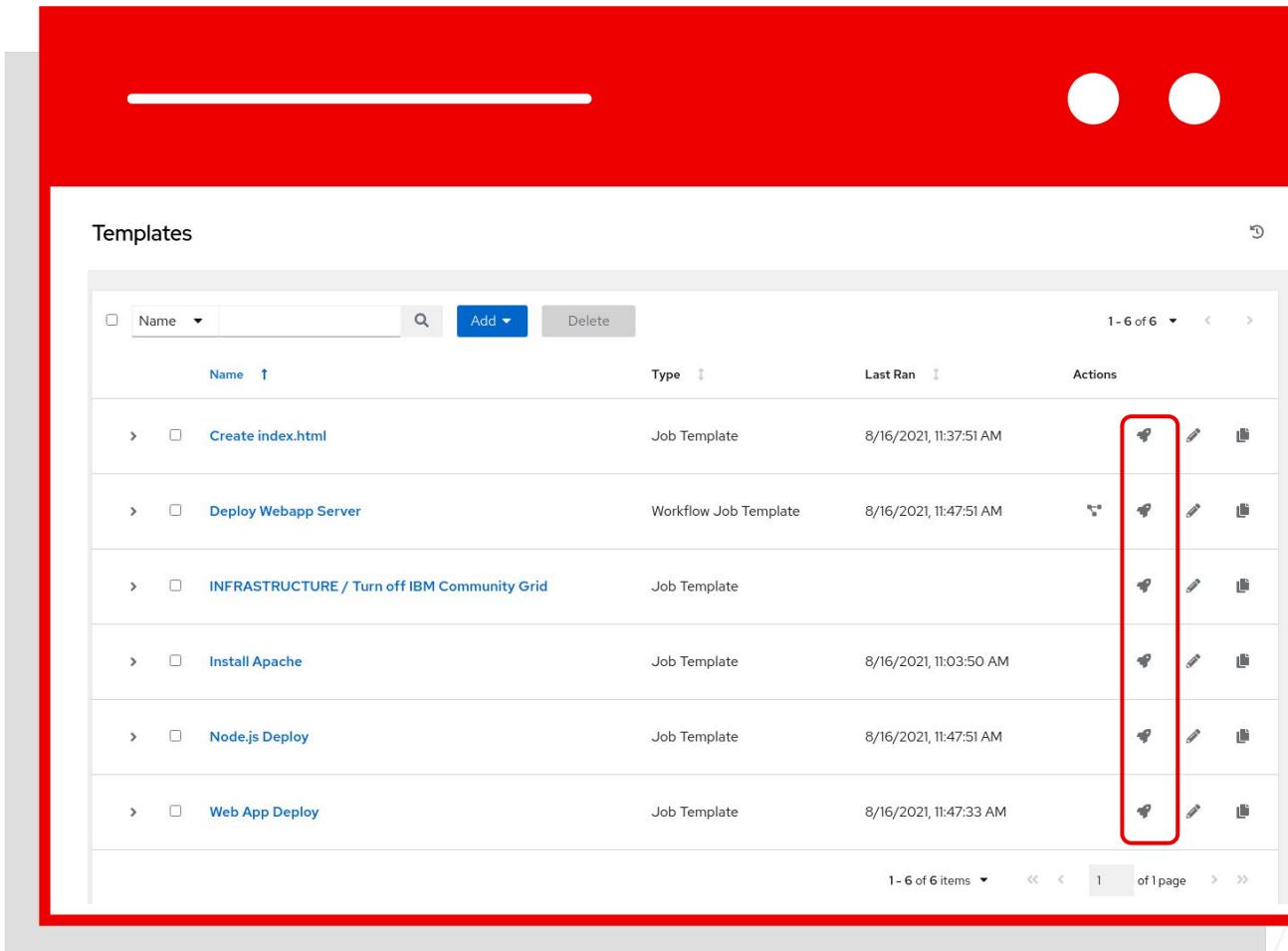


The screenshot shows a list of job templates in a table format. The columns are: Name, Type, Last Ran, and Actions. The table contains six items:

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	 

# Executing an existing Job Template

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

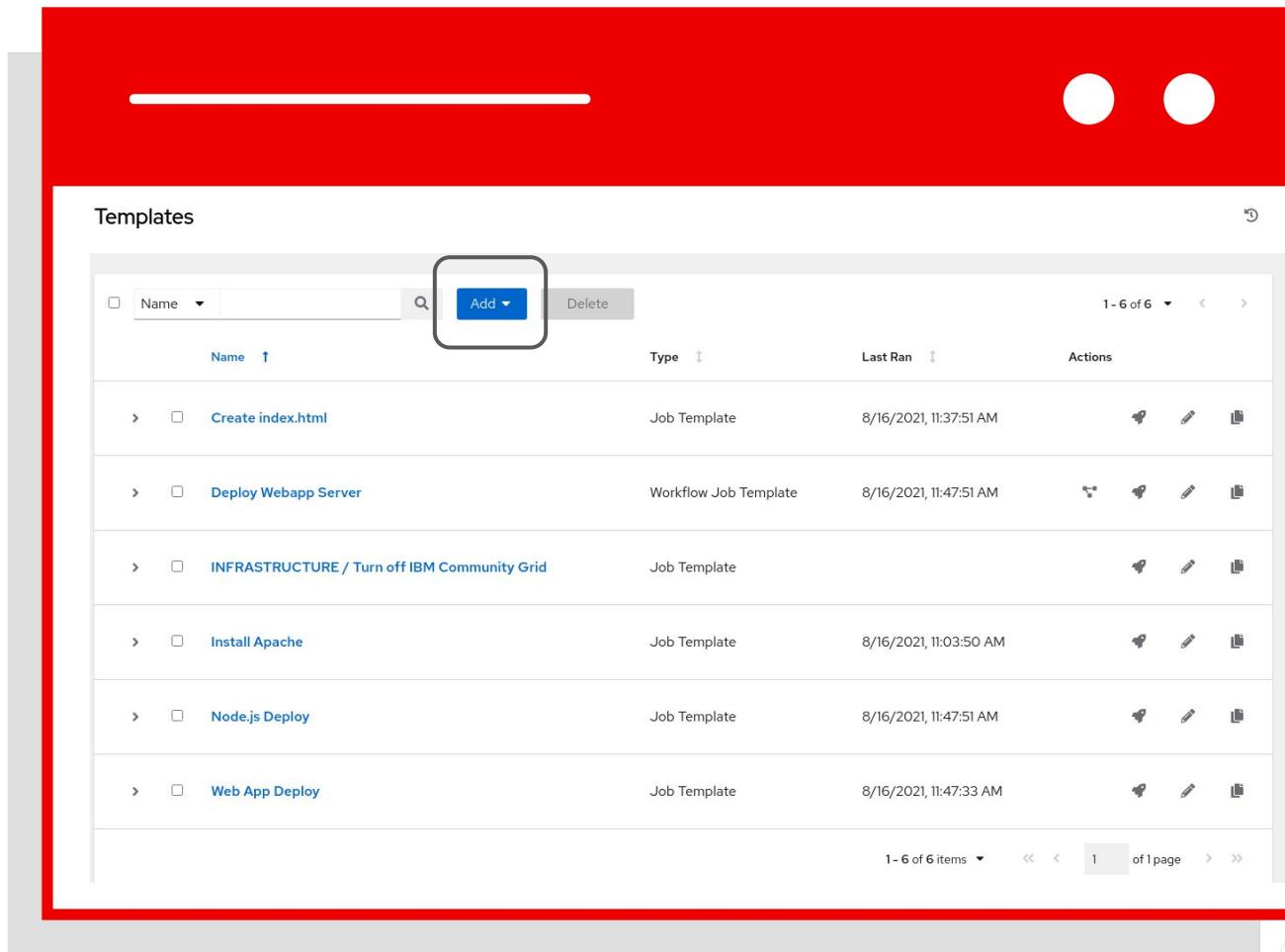


The screenshot shows a list of job templates in the Red Hat Ansible Automation Platform interface. The 'Actions' column for the first item, 'Create index.html', is highlighted with a red box, indicating where to click to execute the template. The interface includes a search bar, an 'Add' button, and a 'Delete' button at the top. The table columns are 'Name', 'Type', 'Last Ran', and 'Actions'.

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**

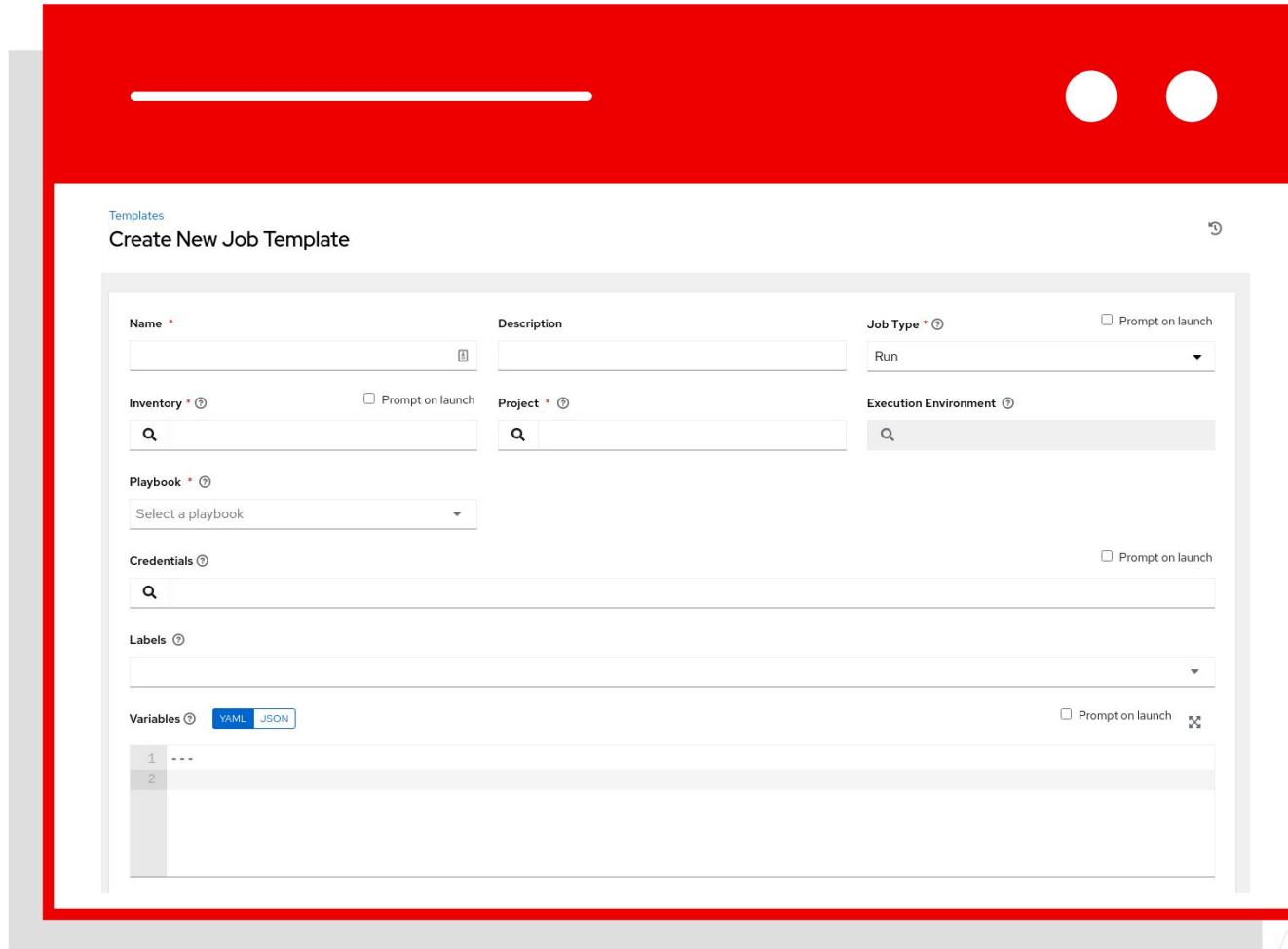


The screenshot shows a list of templates in a web-based interface. A red box highlights the 'Add' button in the top navigation bar. A blue box highlights the first template entry, 'Create index.html', which is a Job Template last run on 8/16/2021, 11:37:51 AM. The interface includes a search bar, a delete button, and a list of other templates: 'Deploy Webapp Server', 'INFRASTRUCTURE / Turn off IBM Community Grid', 'Install Apache', 'Node.js Deploy', and 'Web App Deploy'. The bottom of the screen shows a navigation bar with page numbers and arrows.

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' window. The window has a red border and a white background. At the top, there is a navigation bar with the text 'Templates' and 'Create New Job Template'. Below the navigation bar, there are several input fields and dropdown menus. The required fields are marked with a red asterisk (\*). The fields include:

- Name \***: A text input field.
- Description**: A text input field.
- Job Type \***: A dropdown menu set to 'Run'.
- Inventory \***: A dropdown menu with a search bar.
- Project \***: A dropdown menu with a search bar.
- Execution Environment**: A dropdown menu with a search bar.
- Playbook \***: A dropdown menu with a search bar and a placeholder 'Select a playbook'.
- Credentials**: A dropdown menu with a search bar.
- Labels**: A dropdown menu with a search bar.
- Variables**: A section with a 'YAML' tab selected and a 'JSON' tab. It contains a table with two rows, labeled 1 and 2.

# Lab Time

## Exercise 6: Creating an Automation controller Job Template

🔗 [red.ht/network-workshop-6](https://red.ht/network-workshop-6)

Demonstrate a network backup configuration job template with Automation controller.

⌚ Approximate time: 15 mins

# Section 7

# Survey

Topics Covered:

- ▶ Understanding Extra Vars
- ▶ Building a Survey
- ▶ Self-service IT with Surveys

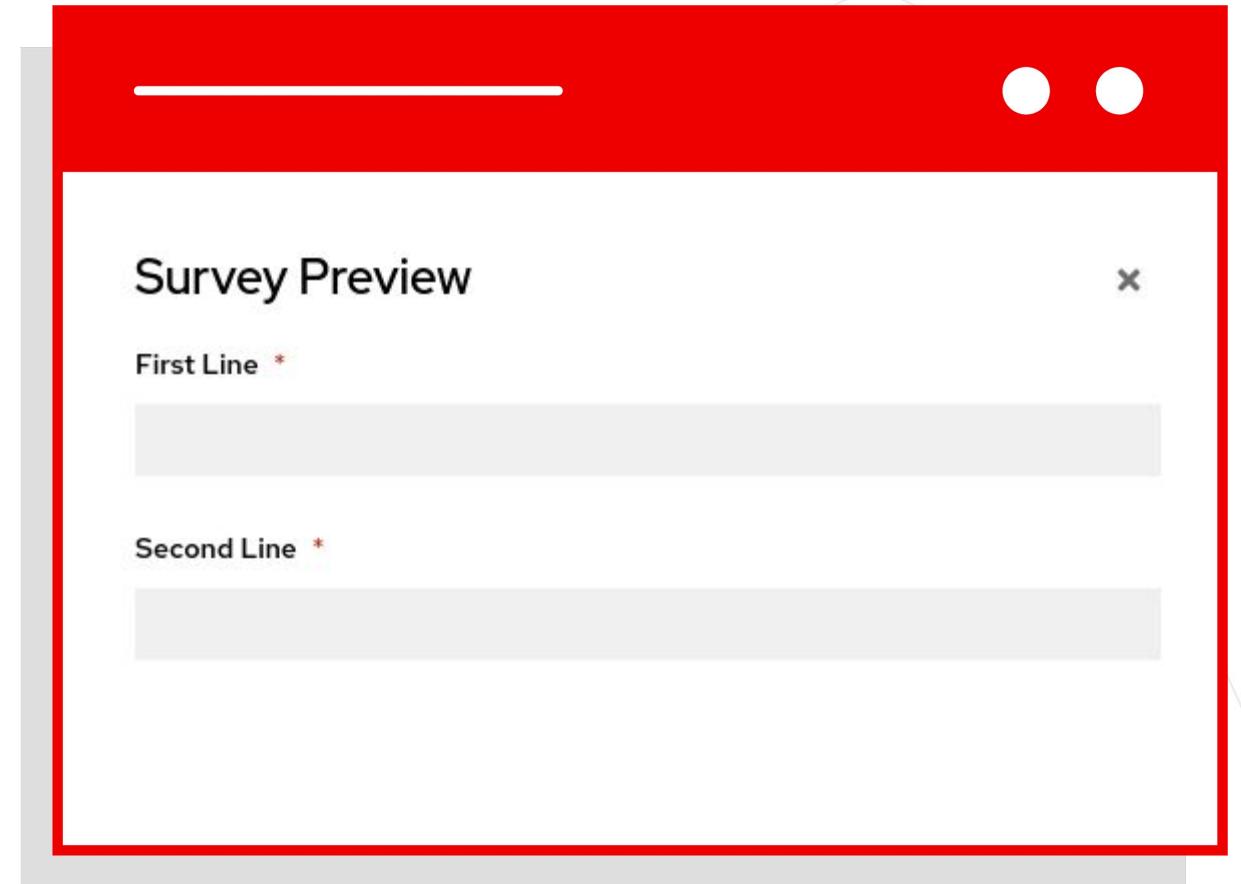


# Surveys

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.

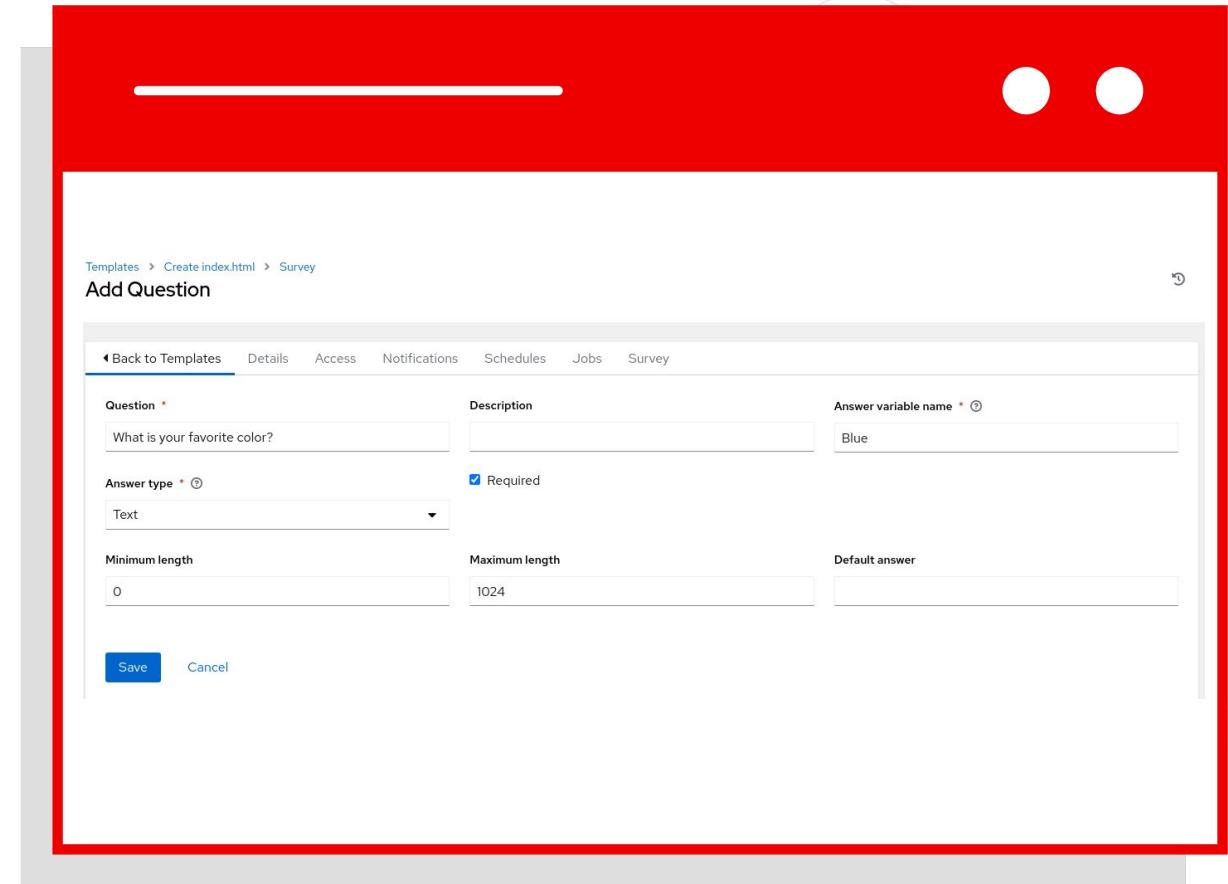
Combine that with Controller's role-based access control, and you can build simple, easy self-service for your users.



# 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.



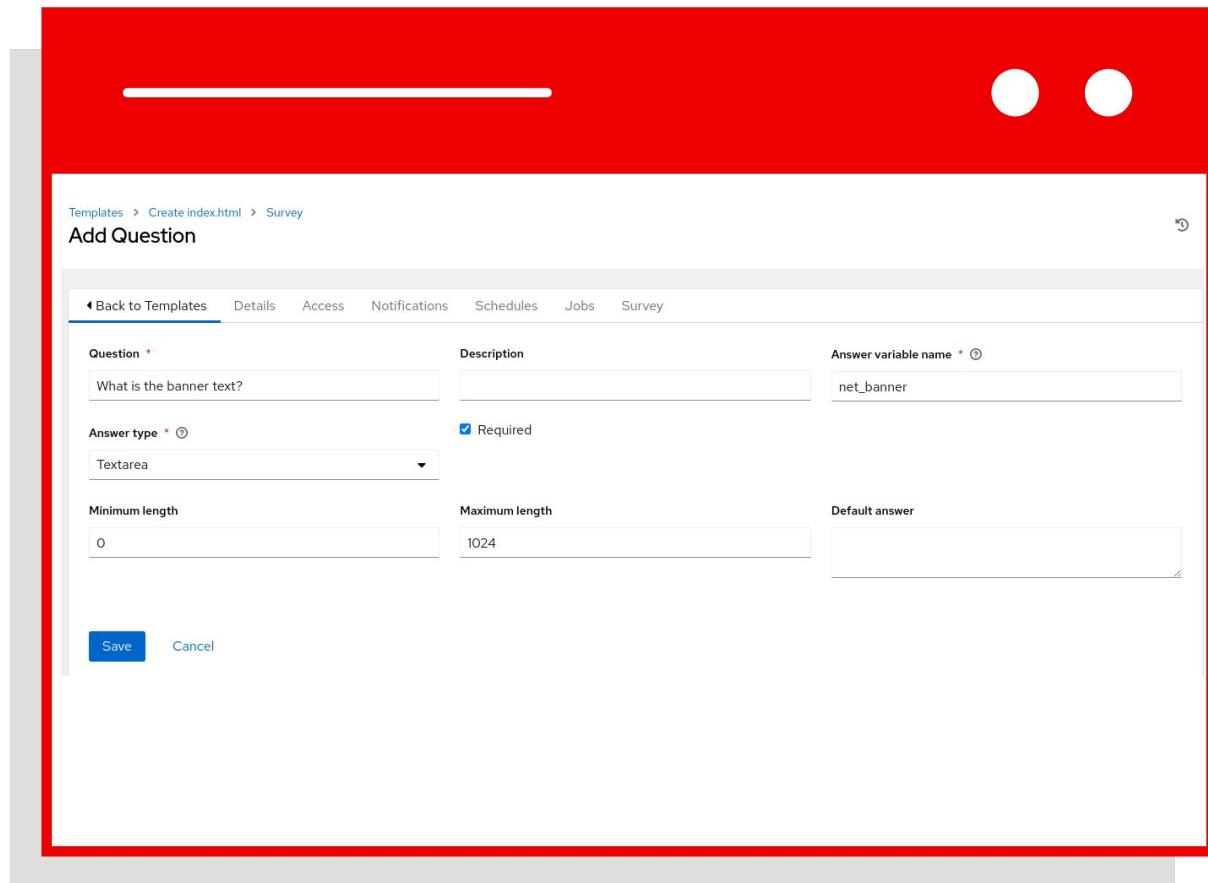
The screenshot shows a web-based configuration interface for a job template. The URL in the address bar is `Templates > Create index.html > Survey`. The main title is **Add Question**. The interface is divided into several sections:

- Question \***: A text input field containing the question "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". A checked checkbox labeled "Required" is next to it.
- 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 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.



Templates > Create index.html > Survey

### Add Question

Back to Templates Details Access Notifications Schedules Jobs Survey

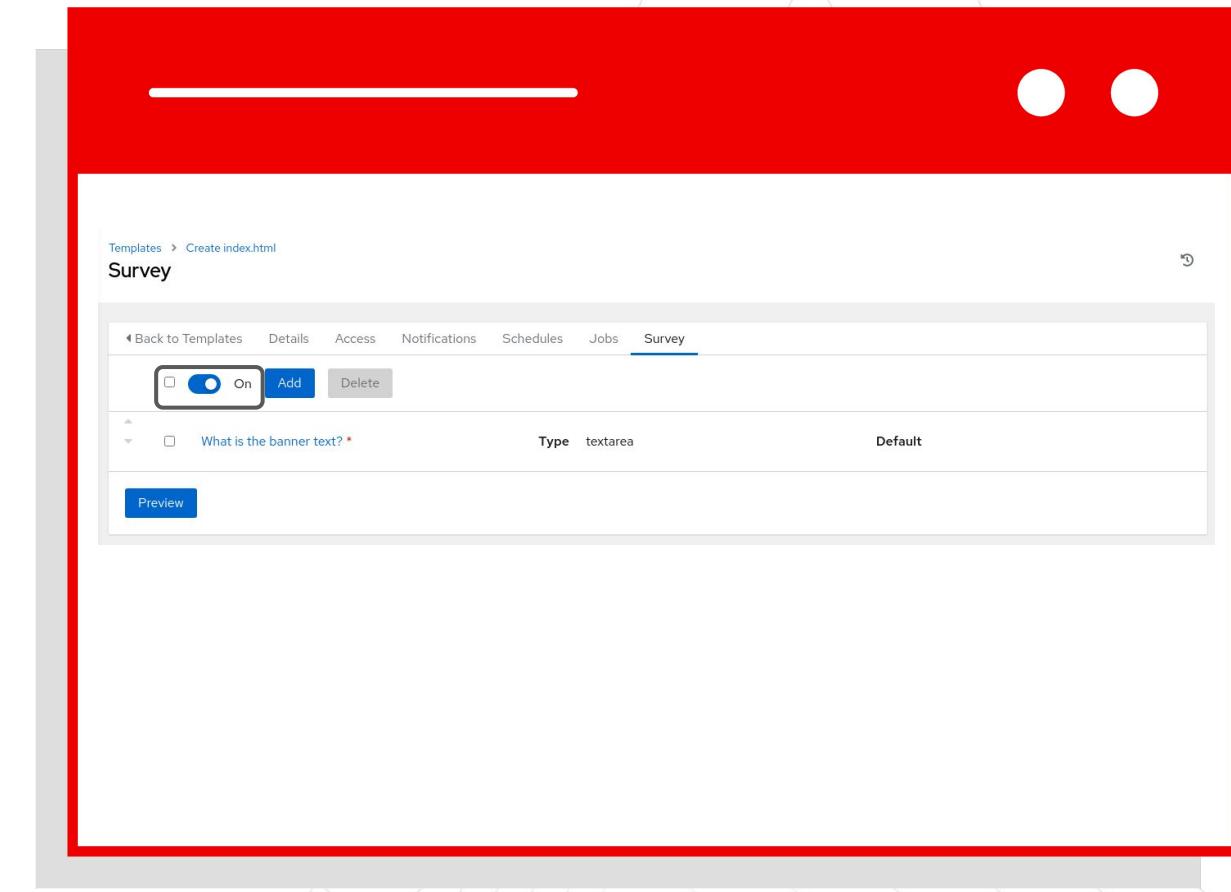
Question \* What is the banner text? Description net\_banner

Answer type \* Textarea  Required

Minimum length 0 Maximum length 1024 Default answer

Save Cancel

This screenshot shows the 'Add Question' window for a survey. It includes fields for the question text ('What is the banner text?'), a description ('net\_banner'), answer type ('Textarea'), and required status ('Required'). It also shows minimum and maximum length settings (0 and 1024 respectively) and a default answer field. The 'Survey' tab is selected in the navigation bar.



Templates > Create index.html

### Survey

Back to Templates Details Access Notifications Schedules Jobs Survey

On

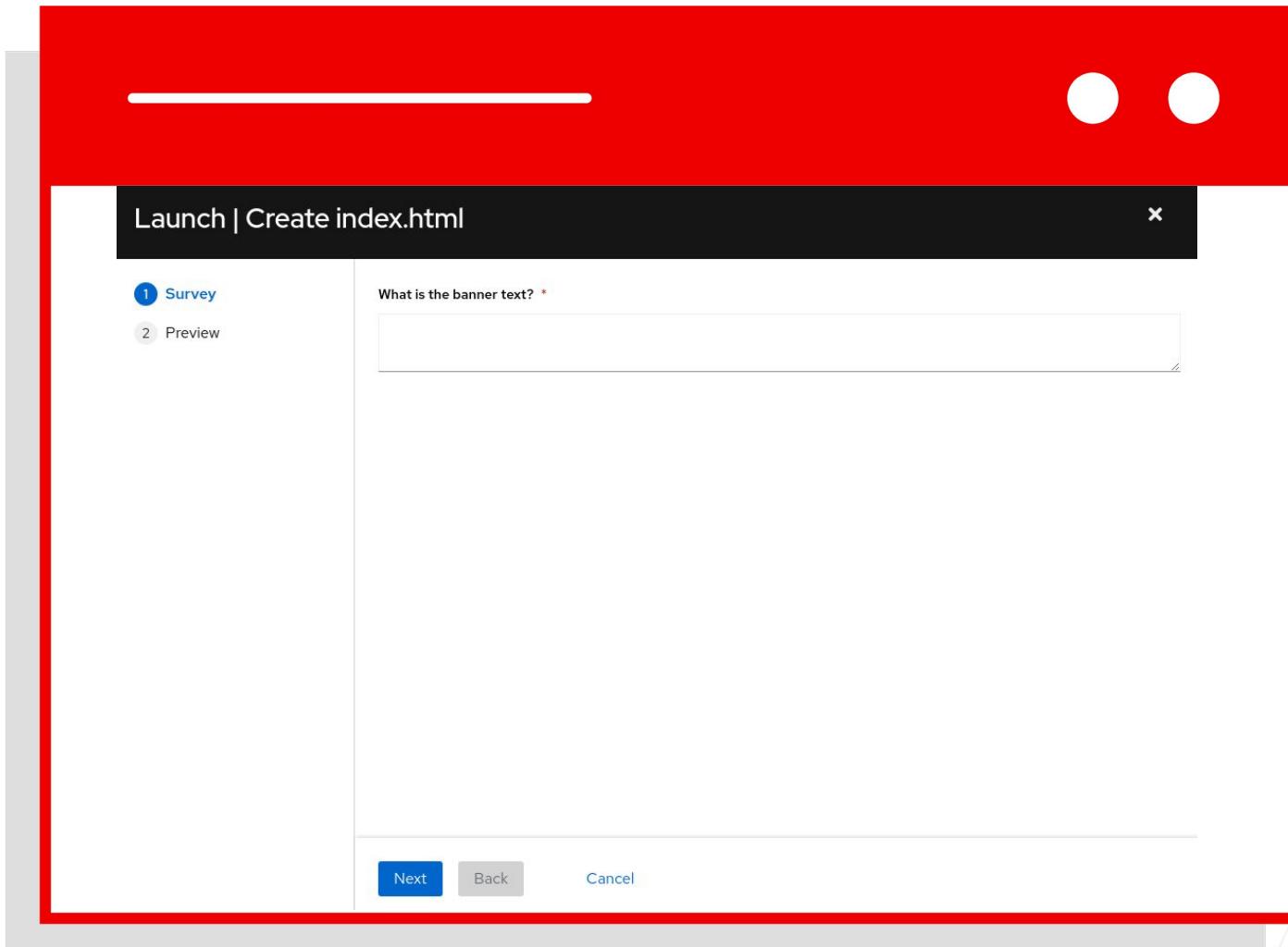
What is the banner text? \* Type textarea Default

Preview

This screenshot shows the 'Survey' window for the 'Create index.html' template. It displays the survey question ('What is the banner text?') with its type ('textarea') and default value ('Default'). The 'Survey' tab is selected in the navigation bar. A 'Preview' button is also visible.

# 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.



# Lab Time

## Exercise 7: Creating a Survey

🔗 [red.ht/network-workshop-7](https://red.ht/network-workshop-7)

Demonstrate the use of Automation controller survey feature.

⌚ Approximate time: 15 mins

# Section 8

# RBAC

## Topics Covered:

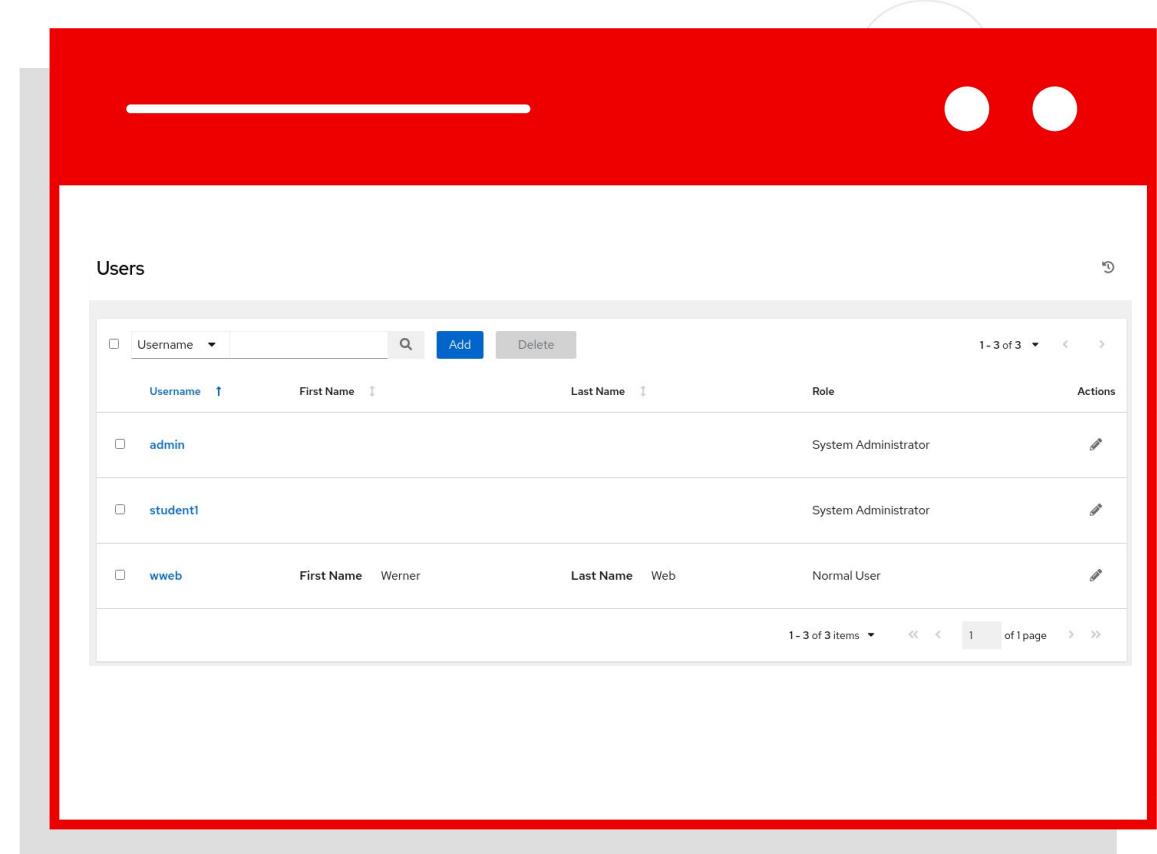
- ▶ Understanding Organizations
- ▶ Understanding Teams
- ▶ Understanding Users



# Role-based access control

## How to manage access

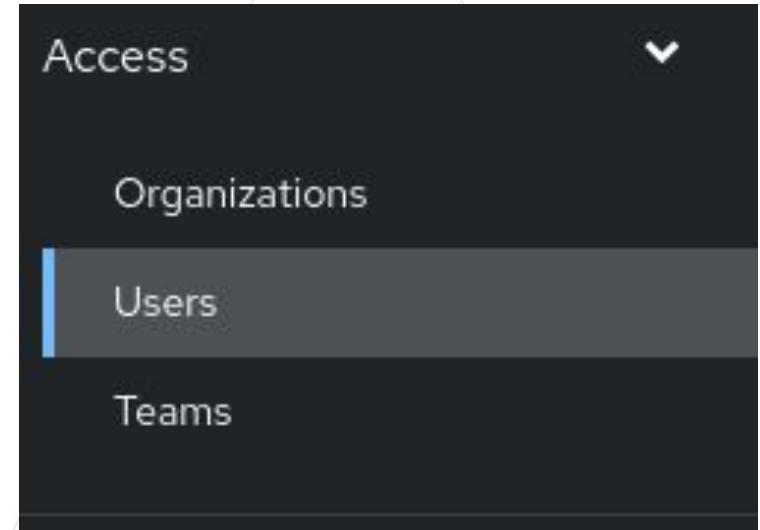
- ▶ Role-based access control system:  
Users can be grouped in teams, and roles can be assigned to the teams.
- ▶ Rights to edit or use can be assigned across all objects.
- ▶ All backed by enterprise authentication if needed.



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

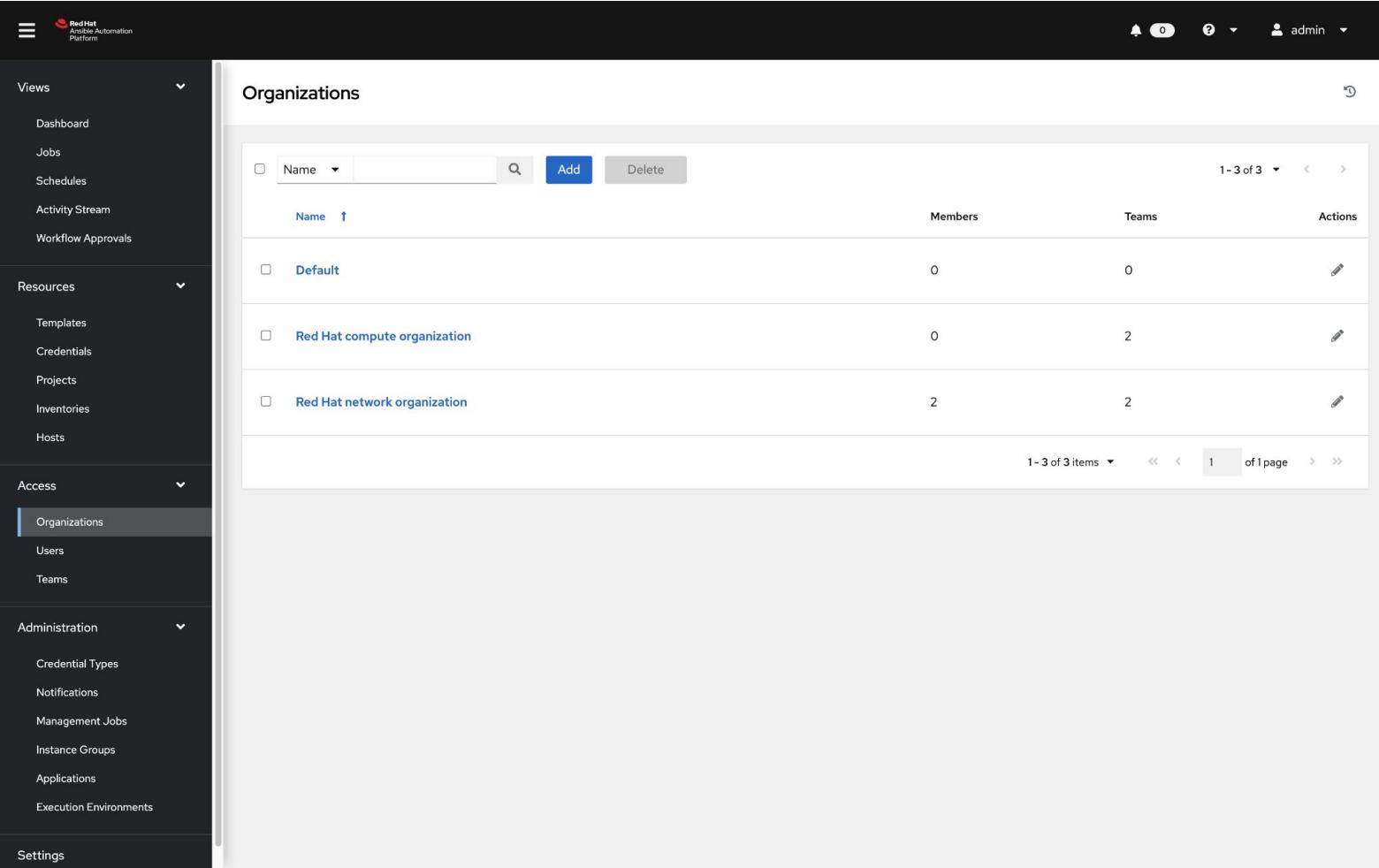
# User Management

- 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.



# Viewing Organizations

Clicking on the **Organizations** button in the left menu will open up the Organizations window



The screenshot shows the Red Hat Ansible Automation Platform interface. The left sidebar is a navigation menu with the following sections and items:

- Views**: Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals.
- Resources**: Templates, Credentials, Projects, Inventories, Hosts.
- Access**: **Organizations** (selected), Users, Teams.
- Administration**: Credential Types, Notifications, Management Jobs, Instance Groups, Applications, Execution Environments.
- Settings**.

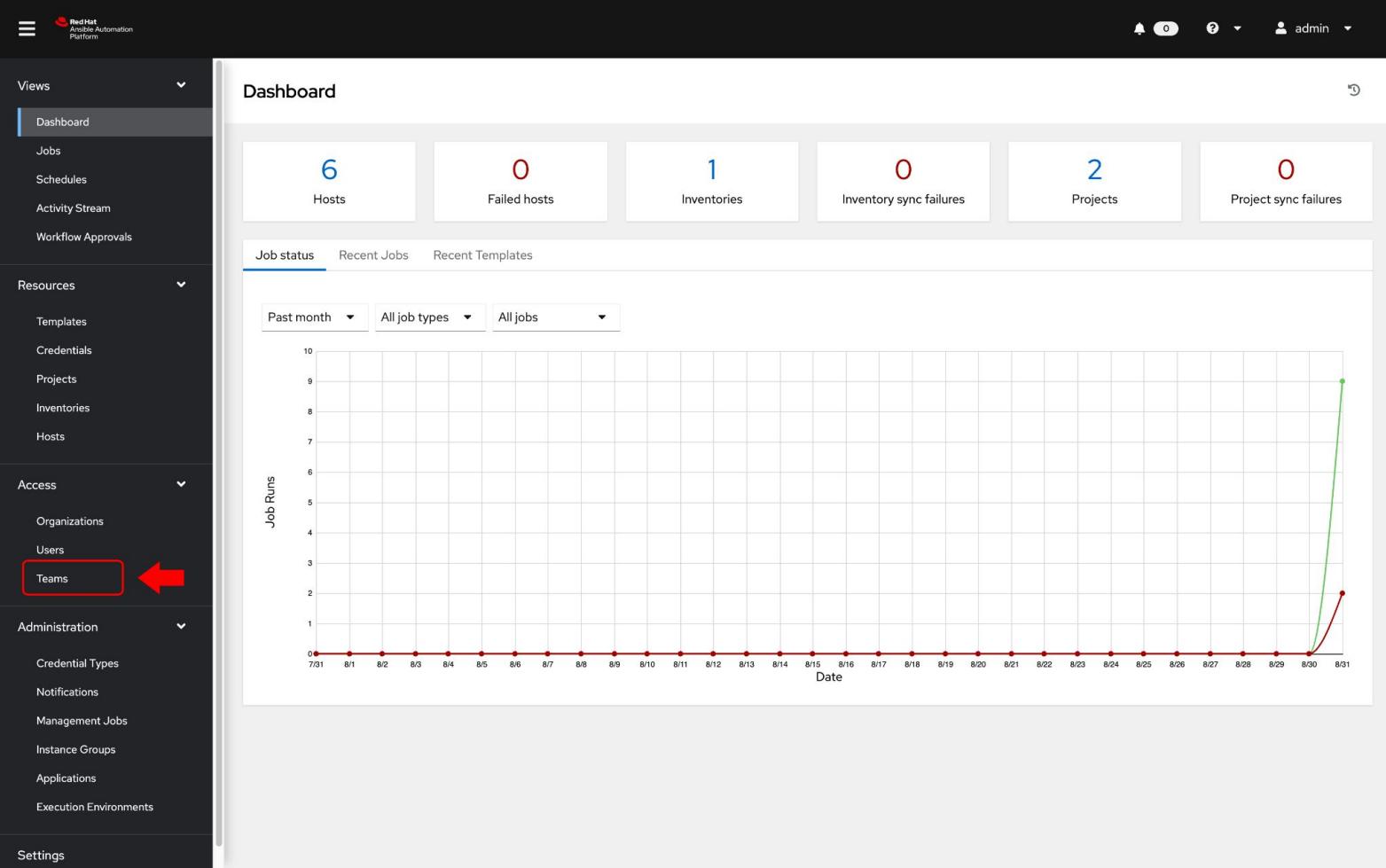
The main content area is titled "Organizations". It displays a table with three rows of organization data:

Name	Members	Teams	Actions
Default	0	0	
Red Hat compute organization	0	2	
Red Hat network organization	2	2	

At the bottom of the table, there is a page navigation bar showing "1-3 of 3 items" and "1 of 1 page".

# Viewing Teams

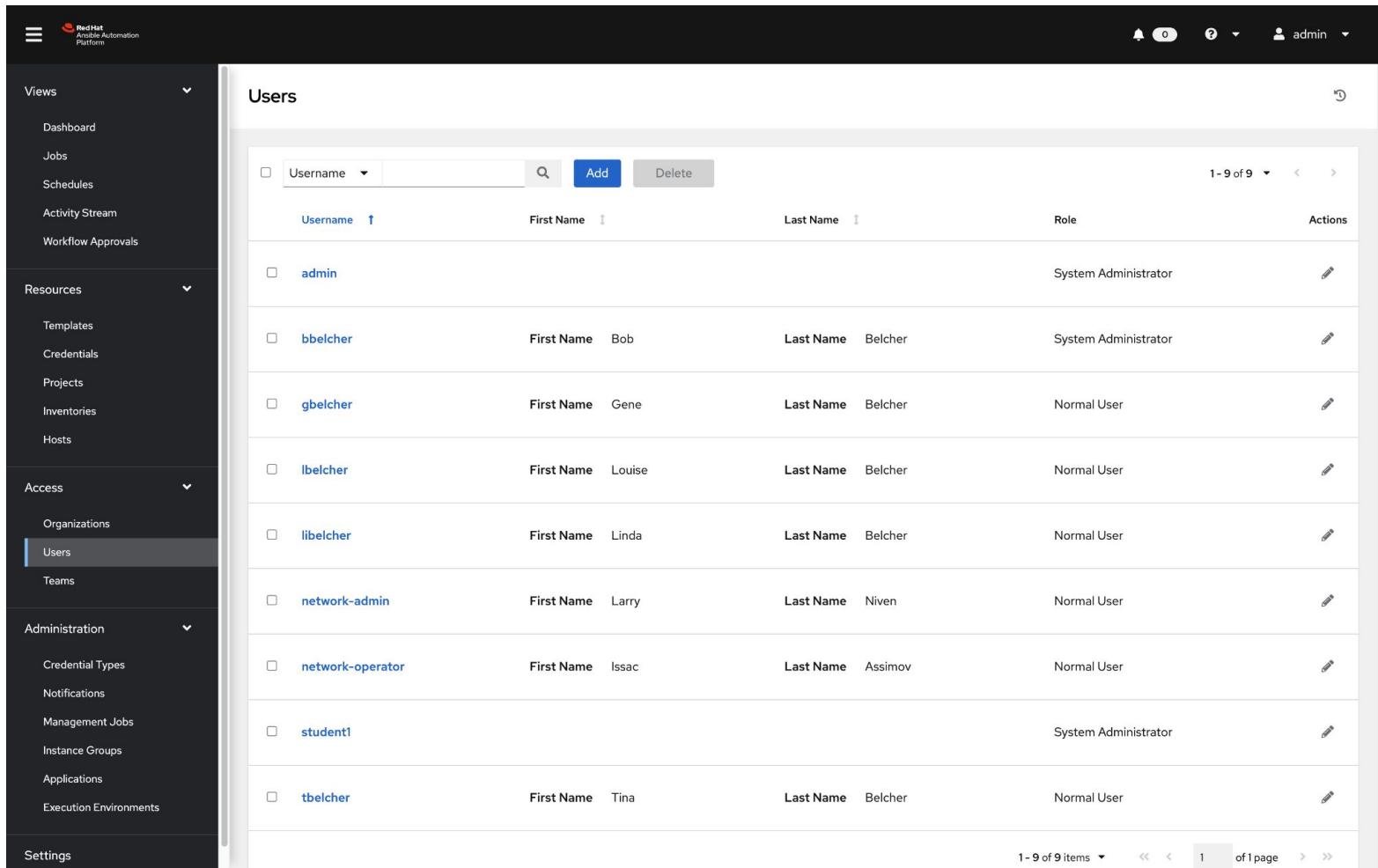
Clicking on the **Teams** buttons in the left menu will open up the Teams window



The screenshot shows the Red Hat Ansible Automation Platform dashboard. On the left, a dark sidebar contains a navigation menu with several categories: Views (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), Resources (Templates, Credentials, Projects, Inventories, Hosts), Access (Organizations, Users, Teams), Administration (Credential Types, Notifications, Management Jobs, Instance Groups, Applications, Execution Environments), and Settings. The 'Teams' item under the 'Access' section is highlighted with a red box and a red arrow pointing to it. The main dashboard area displays various metrics: 6 Hosts, 0 Failed hosts, 1 Inventories, 0 Inventory sync failures, 2 Projects, and 0 Project sync failures. Below these, a line chart titled 'Job status' shows 'Job Runs' over a 'Past month' (from 7/31 to 8/31). The chart shows a sharp increase in job runs starting around August 30th, with a single data point at 8/30 reaching a value of 9, and another point at 8/31 reaching a value of 2.

# Viewing Users

Clicking on the **Users** button in the left menu will open up the Users window



Username	First Name	Last Name	Role	Actions
admin			System Administrator	
bbelcher	Bob	Belcher	System Administrator	
gbelcher	Gene	Belcher	Normal User	
lbelcher	Louise	Belcher	Normal User	
libelcher	Linda	Belcher	Normal User	
network-admin	Larry	Niven	Normal User	
network-operator	Issac	Assimov	Normal User	
student1			System Administrator	
tbelcher	Tina	Belcher	Normal User	

# Lab Time

Exercise 8: Understanding RBAC in Automation controller

🔗 [red.ht/network-workshop-8](https://red.ht/network-workshop-8)

Demonstrate the use of role based access control on Automation controller.

⌚ Approximate time: 15 mins

# Section 9

# Workflows

Topics Covered:

- ▶ Understanding Workflows
- ▶ Branching
- ▶ Convergence / Joins
- ▶ Conditional Logic



# Lab Time

## Exercise 9: Creating a Workflow

🔗 [red.ht/network-workshop-9](https://red.ht/network-workshop-9)

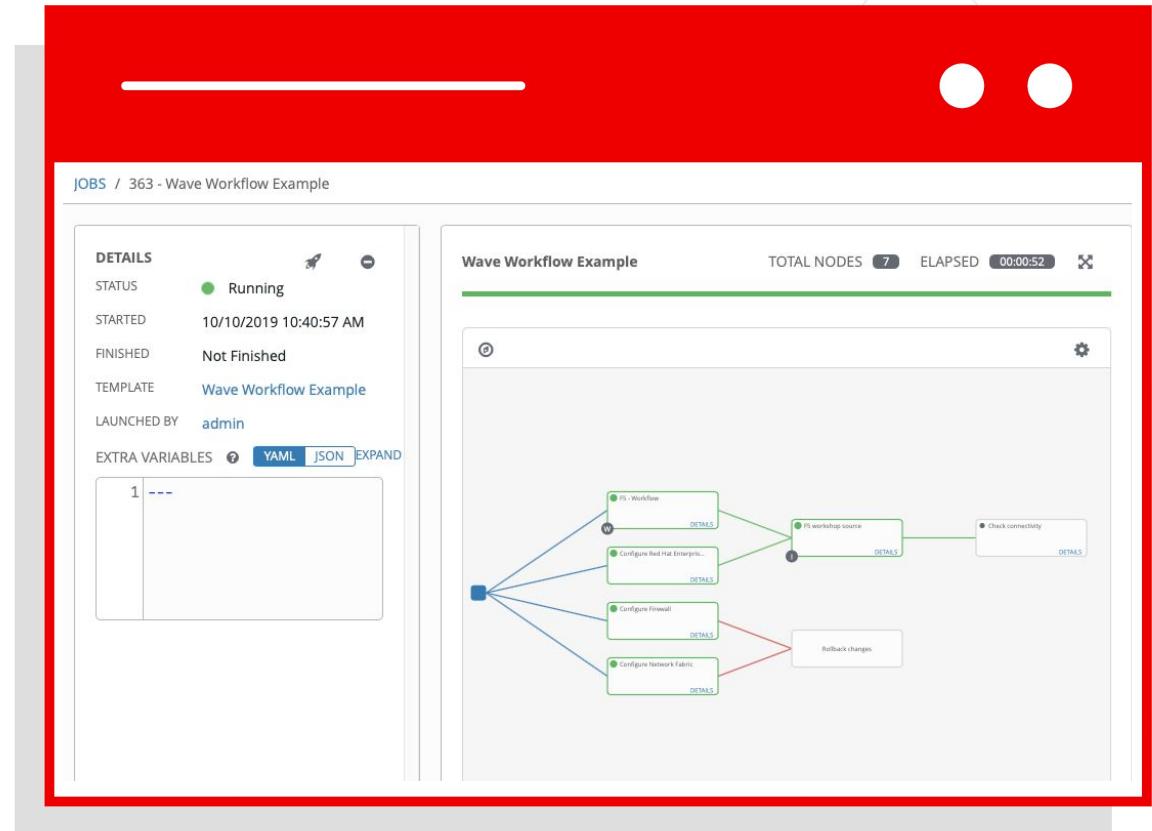
Demonstrate the use of Automation Controller workflow. Workflows allow you to configure a sequence of disparate job templates (or workflow templates) that may or may not share inventory, playbooks, or permissions.

⌚ Approximate time: 15 mins

# Workflows

Combine automation to create something bigger

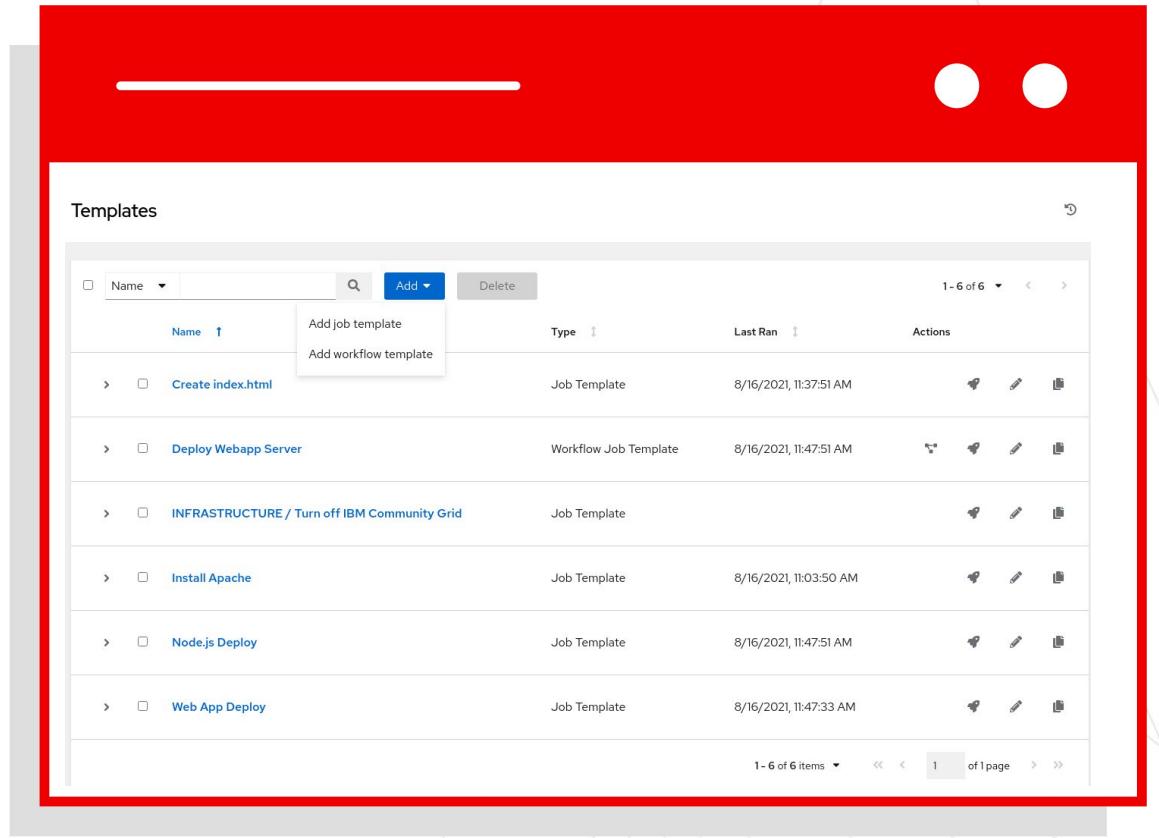
- ▶ Workflows enable the creation of powerful holistic automation, chaining together multiple pieces of automation and events.
- ▶ Simple logic inside these workflows can trigger automation depending on the success or failure of previous steps.



# Adding a New Template

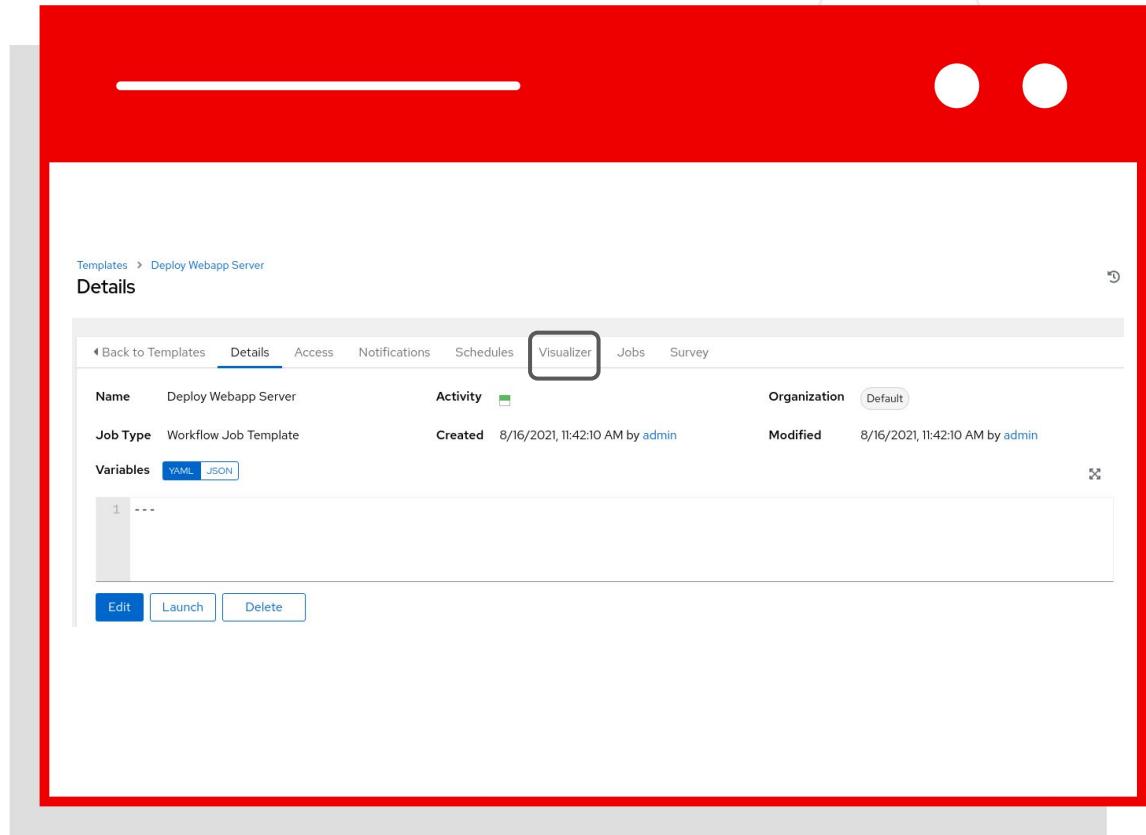
- ▶ To add a new **Workflow** click on the **Add** button.

This time select the **Add workflow template**



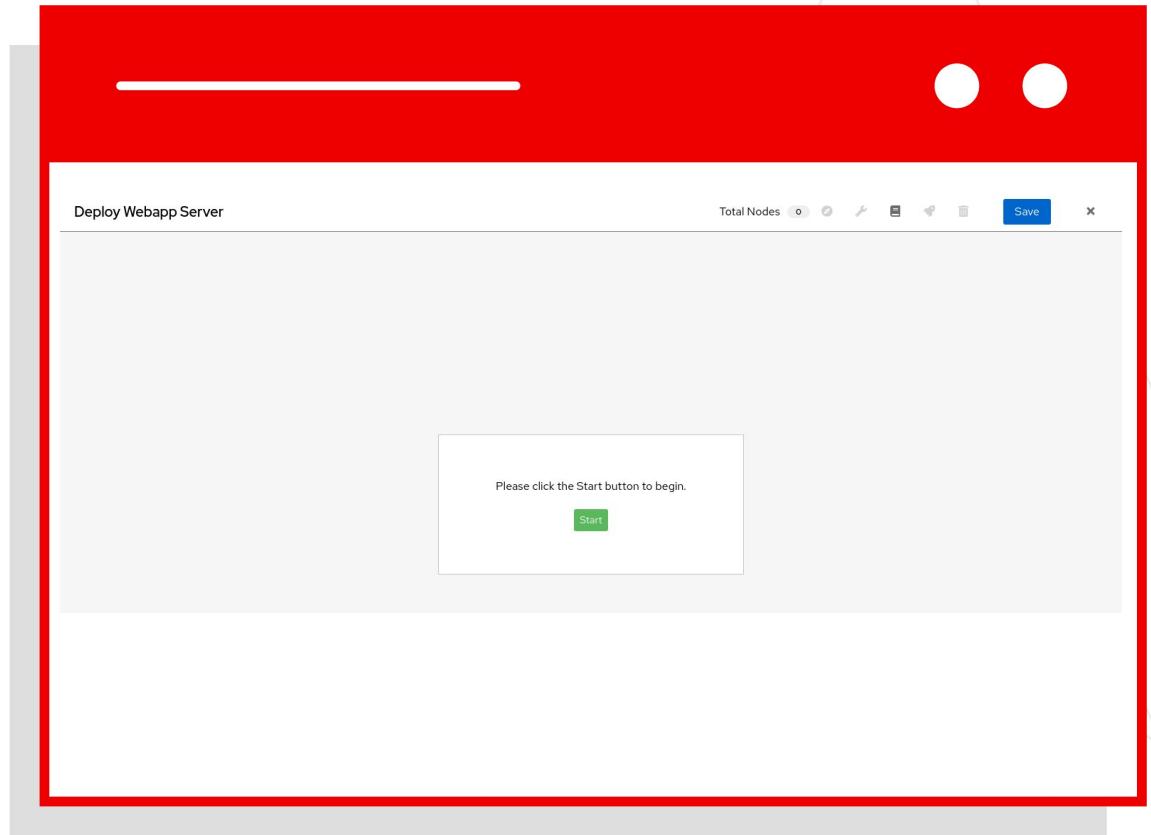
# Creating the Workflow

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



# Workflow Visualizer

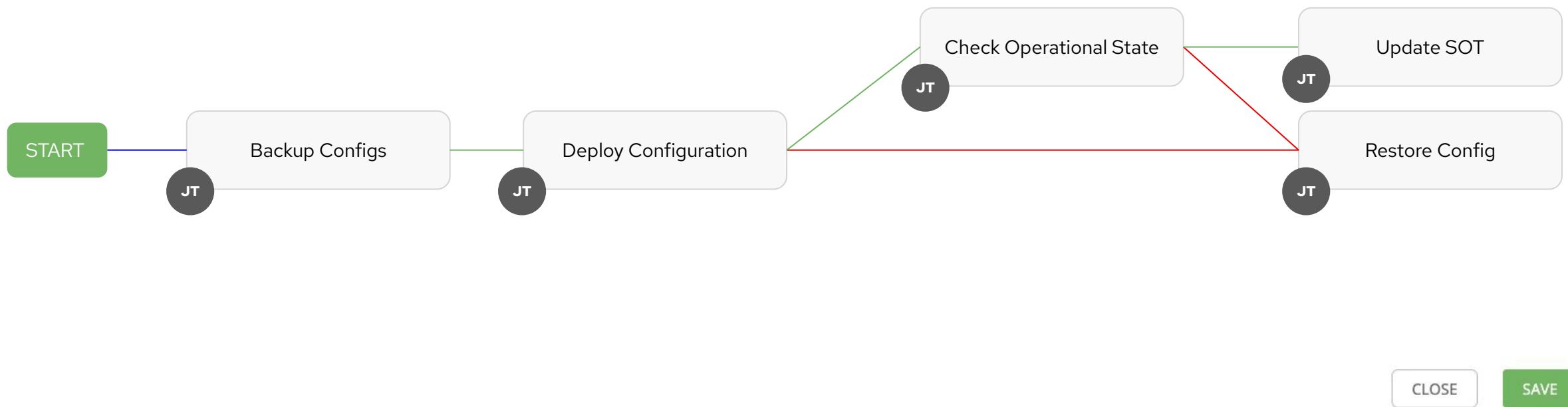
- ▶ The Workflow Visualizer will start as a blank canvas.
- ▶ Click the green Start button to start building the workflow.



# Ansible Automation Platform

Using workflows to enhance your automation

## WORKFLOW VISUALIZER | Operational State Workflow

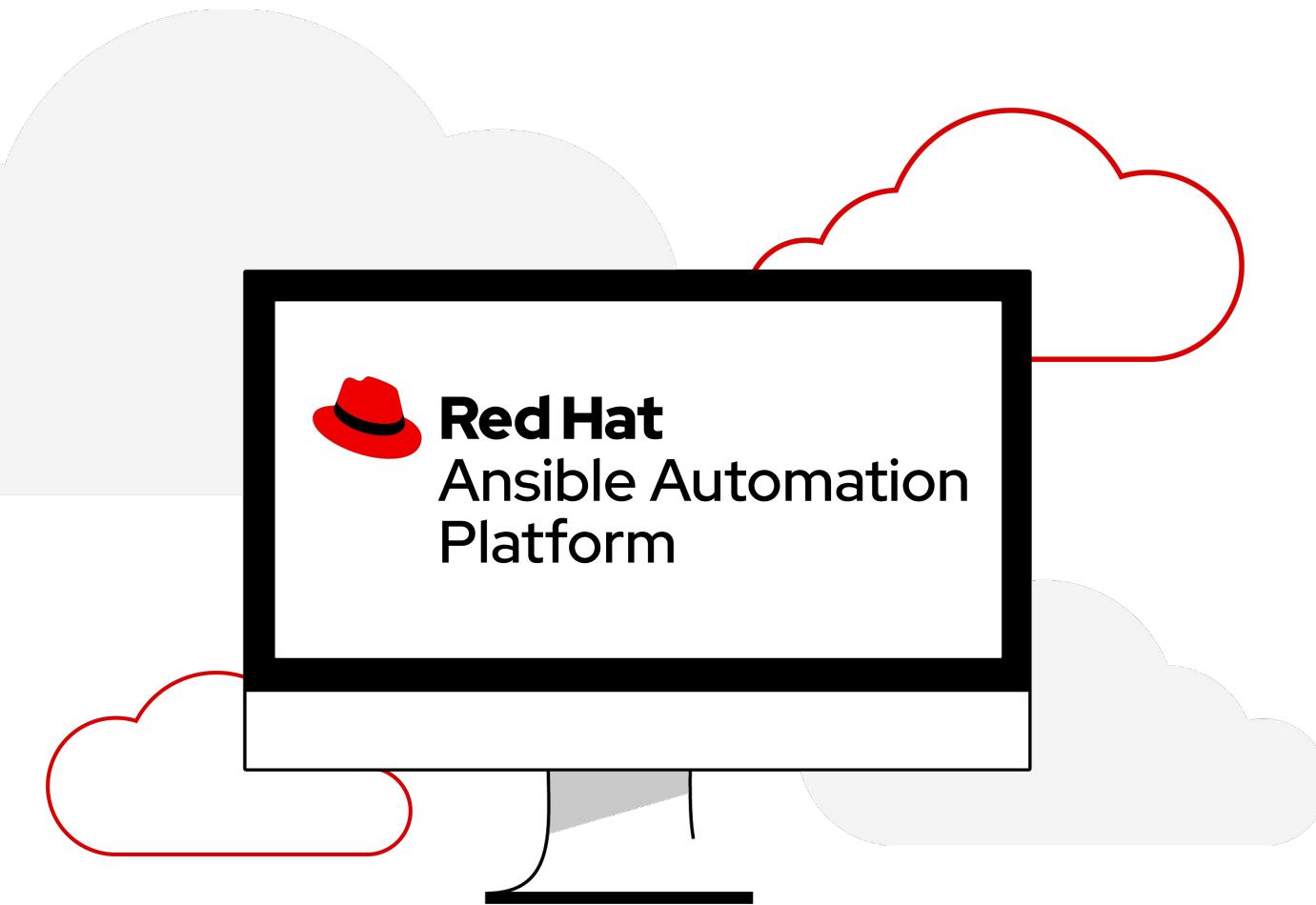


# Wrapping up

## Topics Covered:

- ▶ Next Steps
- ▶ Chat with us
- ▶ Consulting Services





# 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](#)

# Chat with us

- **Slack**

<https://ansiblenetwork.slack.com>

Join by clicking here <http://bit.ly/ansiblenetwork>

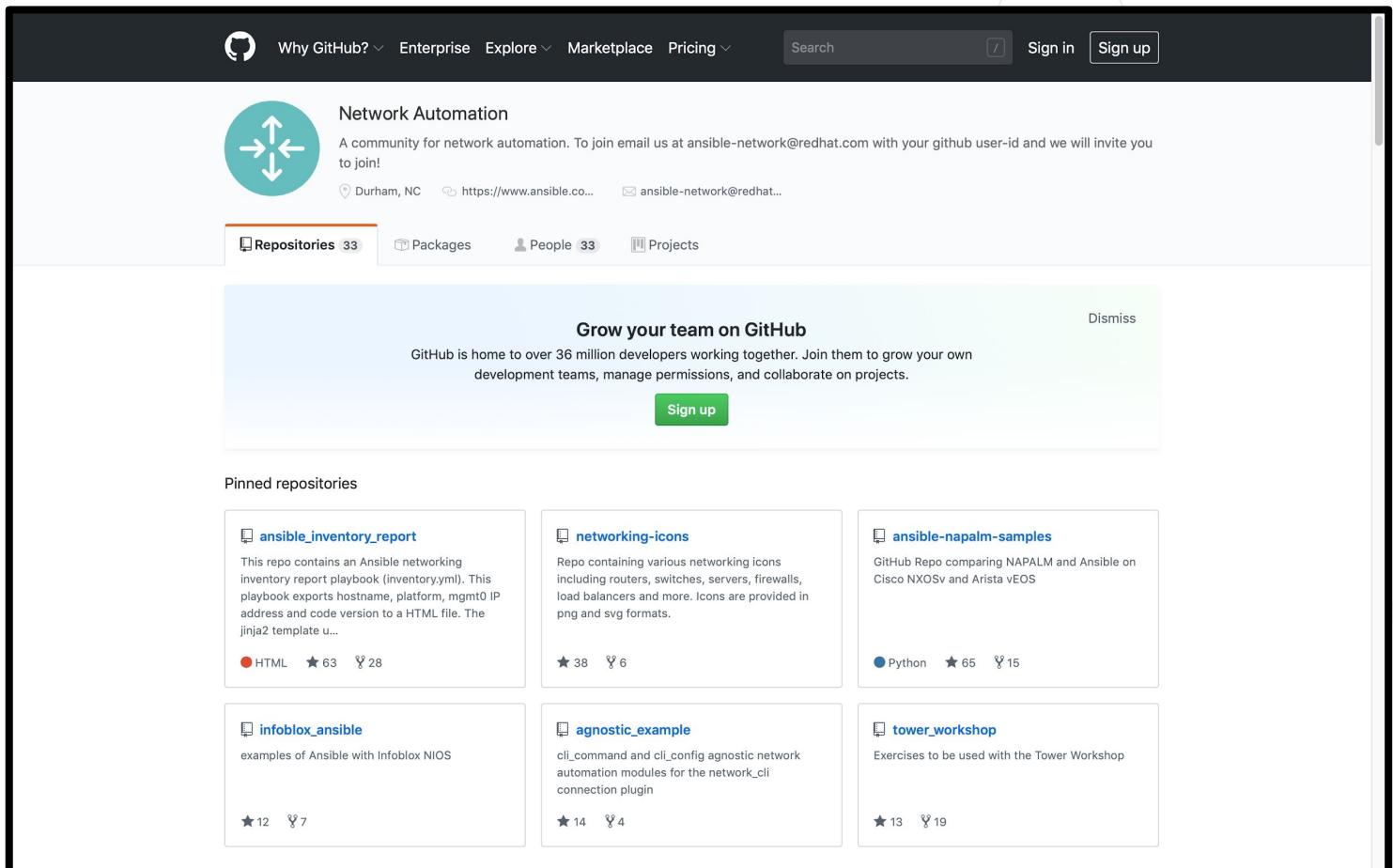
- **IRC**

#ansible-network on freenode

<http://webchat.freenode.net/?channels=ansible-network>

# Bookmark the Github organization

- Examples, samples and demos
- Run network topologies right on your laptop



# Red Hat Services

Accelerate standardization and automation of network configuration



## Challenge

### Slow

Time consuming, labor intensive procedures to propagate network changes

### Chaos

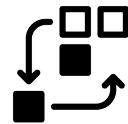
Rising number of devices, environments, and vendor-specific tooling create sprawl and skills gaps

### Errors

Over time, vulnerabilities, patches, and mistakes undermine known-good configurations.

### Mystery

No living source of truth for which patches, packages, or configurations are deployed where



## Approach

### Automate

Encode and execute procedures with human-readable Ansible playbooks

### Standardize

Automate common tasks using Ansible modules to abstract vendor-specific details

### Test

Iteratively refine and validate provisioning and configuration pre-PROD

### Catalog

Automate configuration reporting, inventory, and change tracking across all environments



## Benefits

### Speed

Reduce changes from days to hours and drive simultaneous config across 100s of endpoints

### Efficiency

Easily combine and execute complex configuration procedures across environments

### Reliability

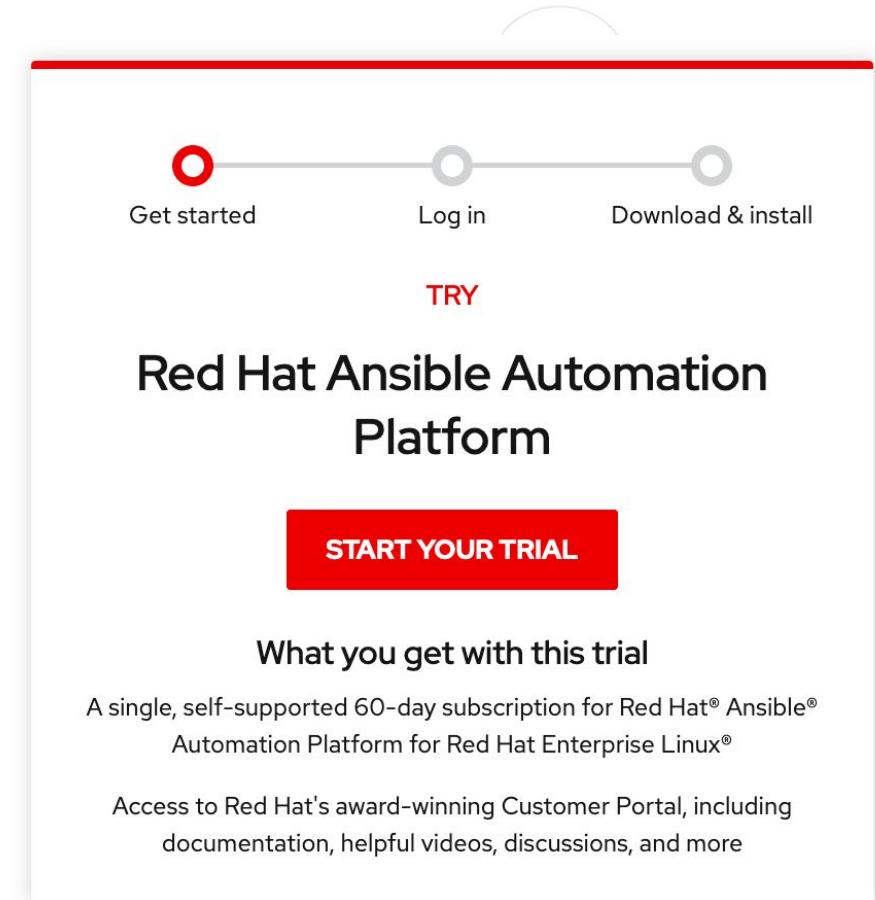
Eliminate human error in production changes

### Manageability

Centrally track and manage configuration rollout, drift, patching, and compliance

# Resources

- ▶ [Network automation for everyone](#) (Overview)
- ▶ [Automate your network with Red Hat](#) (Technical)
- ▶ [Online training: Red Hat Ansible for Network Automation](#)
- ▶ [Network Automation web page](#)
- ▶ [Red Hat Ansible Automation Platform blog](#)



The image shows a landing page for the Red Hat Ansible Automation Platform trial. At the top, there is a horizontal navigation bar with three items: 'Get started' (highlighted with a red dot), 'Log in', and 'Download & install'. Below this is a 'TRY' button. The main title is 'Red Hat Ansible Automation Platform'. A red button labeled 'START YOUR TRIAL' is prominently displayed. Below the trial button, the text 'What you get with this trial' is followed by two bullet points: 'A single, self-supported 60-day subscription for Red Hat® Ansible® Automation Platform for Red Hat Enterprise Linux®' and 'Access to Red Hat's award-winning Customer Portal, including documentation, helpful videos, discussions, and more'.

[red.ht/ansible\\_trial](http://red.ht/ansible_trial)

# 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)

# Supplemental

Topics Covered:

- ▶ Understand group variables
- ▶ Understand Jinja2
- ▶ cli\_config module



**Red Hat**  
Ansible Automation  
Platform

# Group variables

Group variables are variables that are common between two or more devices.

Group variables can be associated with an individual group (e.g. "cisco") or a nested group (e.g. routers).

Examples include

- NTP servers
- DNS servers
- SNMP information

Basically network information that is common for that group

# Inventory versus group\_vars directory

Group variables can be stored in a directory called **group\_vars** in YAML syntax. In exercise one we covered **host\_vars** and **group\_vars** with relationship to inventory. What is the difference?

## inventory

Can be used to set variables to connect and authenticate **to the device**.

Examples include:

- Connection plugins (e.g. network\_cli)
- Usernames
- Platform types  
**(ansible\_network\_os)**

## group\_vars

Can be used to set variables to configure **on the device**.

Examples include:

- VLANs
- Routing configuration
- System services (NTP, DNS, etc)

# Examining a group\_vars file

At the same directory level as the Ansible Playbook create a folder named **group\_vars**.  
Group variable files can simply be named the group name (in this case **all.yml**)

```
$ cat group_vars/all.yml

nodes:
  rtr1:
    Loopback100: "192.168.100.1"
  rtr2:
    Loopback100: "192.168.100.2"
  rtr3:
    Loopback100: "192.168.100.3"
  rtr4:
    Loopback100: "192.168.100.4"
```

# Jinja2

- Ansible has native integration with the Jinja2 templating engine
- Render data models into device configurations
- Render device output into dynamic documentation

Jinja2 enables the user to manipulate variables, apply conditional logic and extend programmability for network automation.



# Network Automation config modules

**cli\_config** (agnostic)

ios\_config:

nxos\_config:

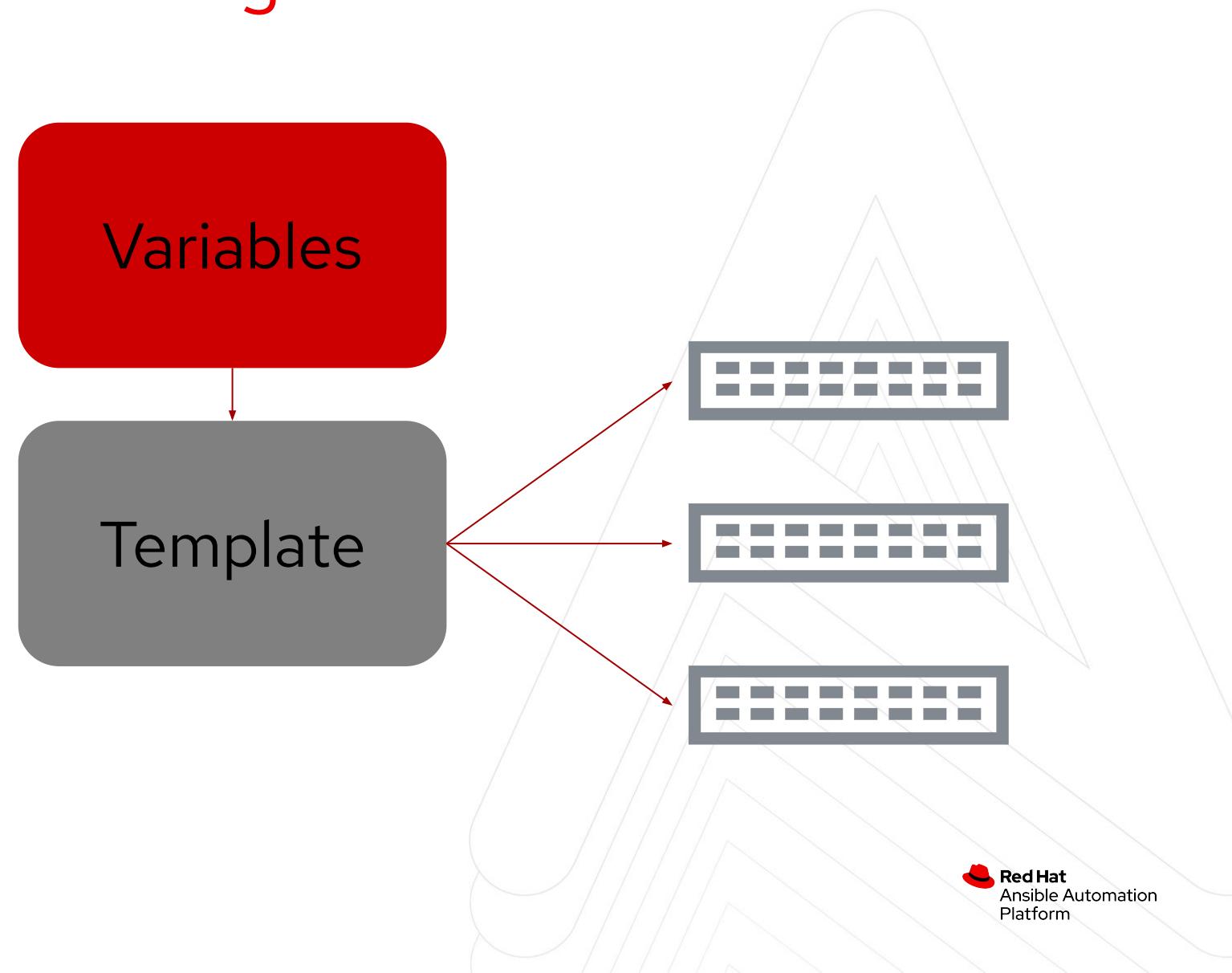
iosxr\_config:

eos\_config

.

.

\*os\_config:



# Jinja2 Templating Example (1/2)

## Variables

```
ntp_server: 192.168.0.250
name_server: 192.168.0.251
```

## Jinja2 Template

```
!
ntp server {{ntp_server}}
!
ip name-server {{name_server}}
!
```

## Generated Network Configuration

rtr1

```
!
ip name-server 192.168.0.251
!
ntp server 192.168.0.250
!
```

rtrX

```
!
ip name-server 192.168.0.251
!
ntp server 192.168.0.250
!
```

# Jinja2 Templating Example (2/2)

## Variables

```
nodes:  
  rtr1:  
    Loopback100: "192.168.100.1"  
  rtr2:  
    Loopback100: "192.168.100.2"  
  rtr3:  
    Loopback100: "192.168.100.3"  
  rtr4:  
    Loopback100: "192.168.100.4"
```

## Jinja2 Template

```
{% for interface,ip in nodes[inventory_hostname].items() %}  
  interface {{interface}}  
    ip address {{ip}} 255.255.255.255  
{% endfor %}
```

## Generated Network Configuration

rtr1

```
interface Loopback100  
  ip address 192.168.100.1  
!
```

rtr2

```
interface Loopback100  
  ip address 192.168.100.2  
!
```

rtrX

```
interface Loopback100  
  ip address X  
!
```

# The `cli_config` module

Agnostic module for network devices that uses the `network_cli` connection plugin.

```
---
```

```
- name: configure network devices
  hosts: rtr1,rtr2
  gather_facts: false
  tasks:
    - name: configure device with config
      cli_config:
        config: "{{ lookup('template', 'template.j2') }}"
```