Terraform Developer

TEKsystems Global Services

WORKFORCE DEVELOPMENT

PARTICIPANT GUIDE

Content Usage Parameters

Content refers to material including instructor guides, student guides, lab guides, lab or hands-on activities, computer programs, etc. designed for use in a training program

1

Content is subject to copyright protection

2

Content may only be leveraged by students enrolled in the training program

3

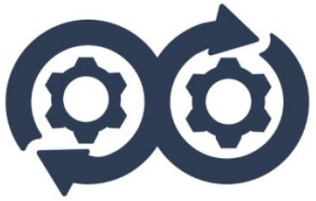
Students agree not to reproduce, make derivative works of, distribute, publicly perform and publicly display content in any form or medium outside of the training program

4

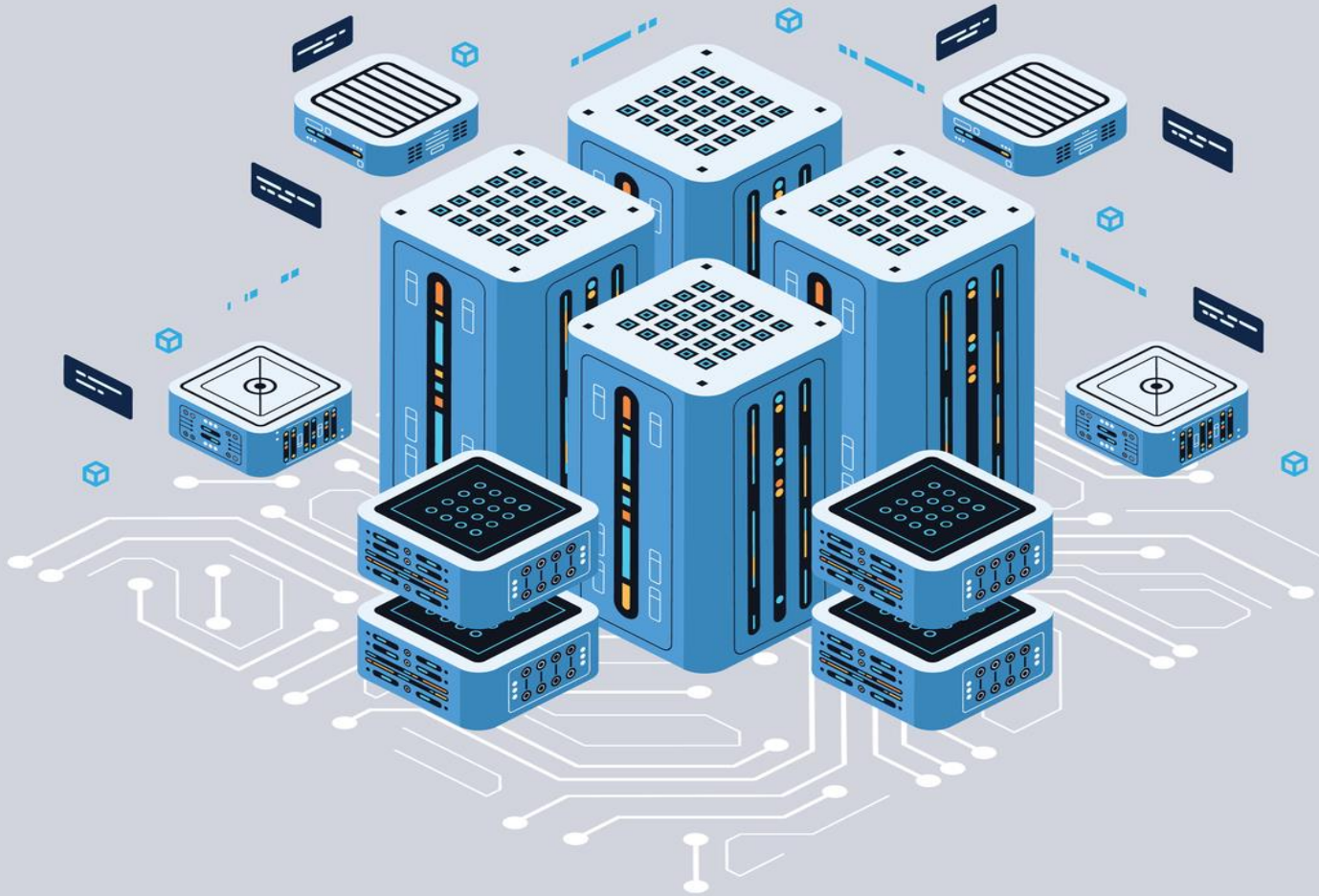
Content is intended as reference material only to supplement the instructor-led training

Lab page

https://jruels.github.io/tf-dev



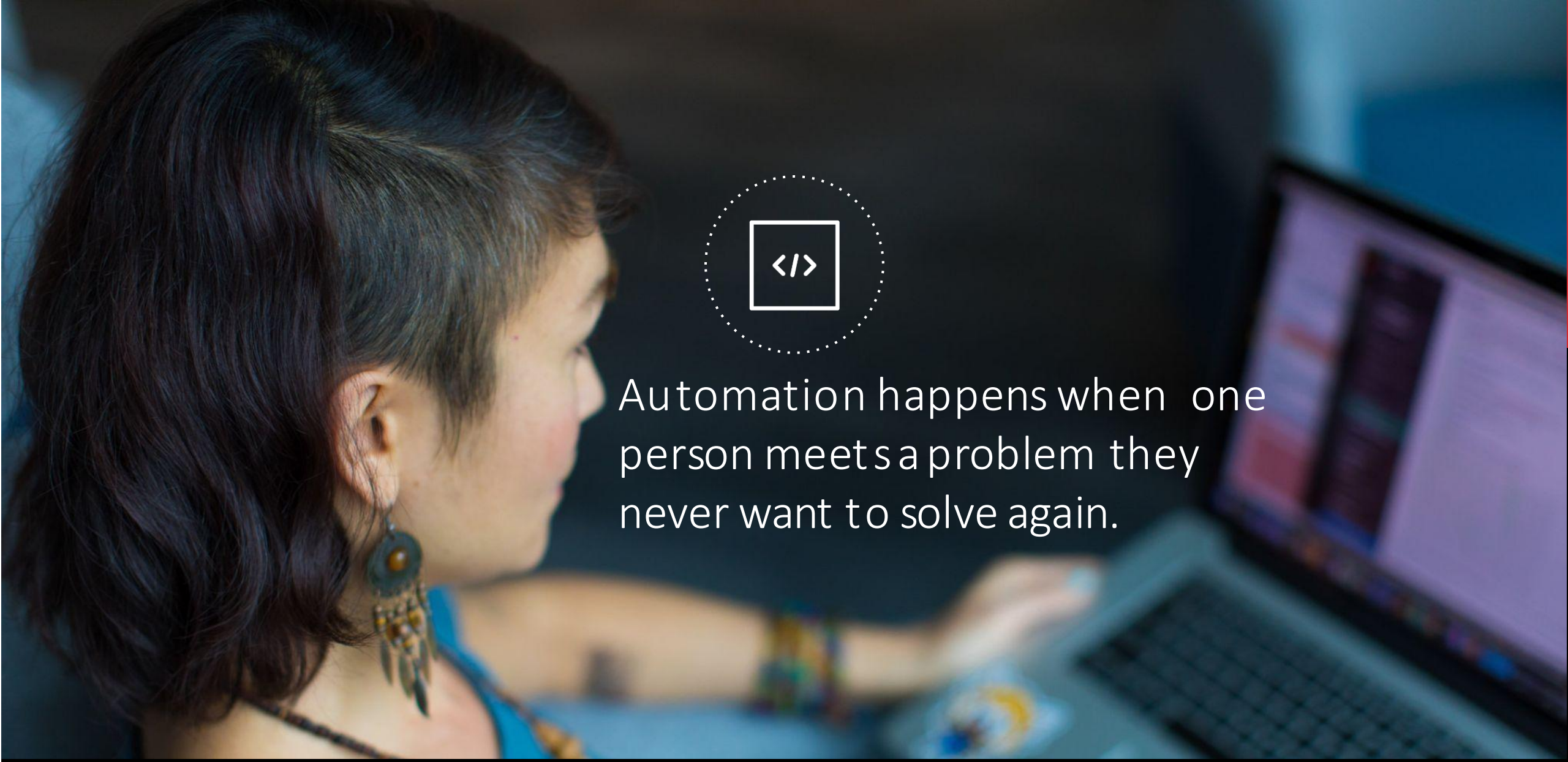
Ansible



POP QUIZ: Automation

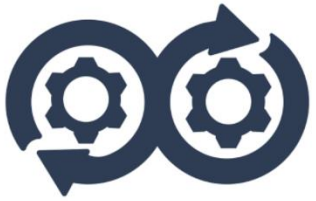
Why do we automate tasks?

5 MINUTES



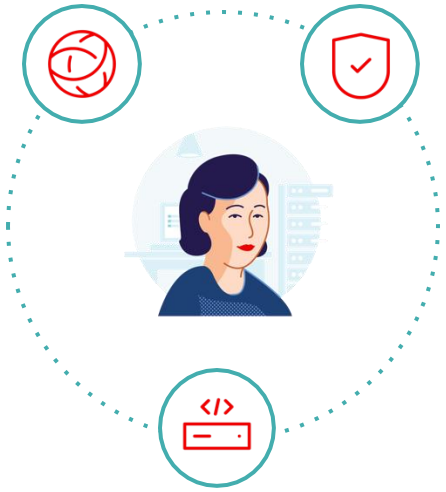
Automation

Repeatability by automating activities.

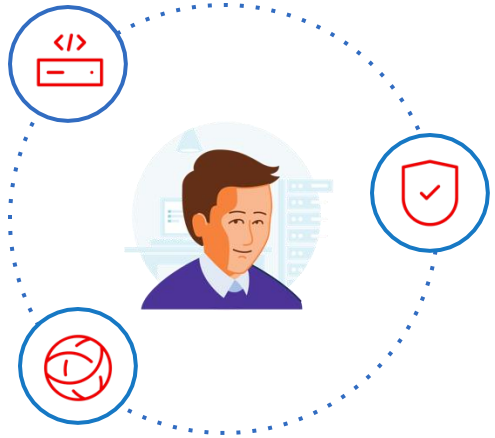


Many organizations share the same challenge

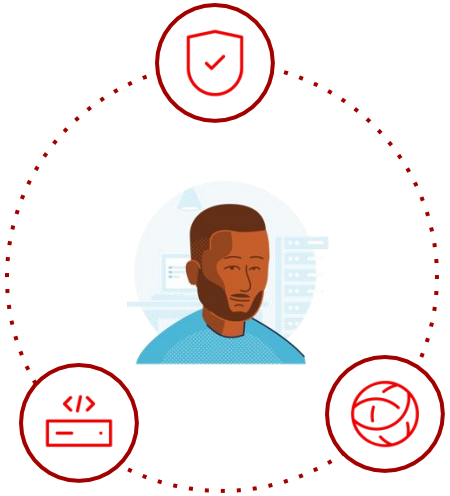
Too many unintegrated, domain-specific tools



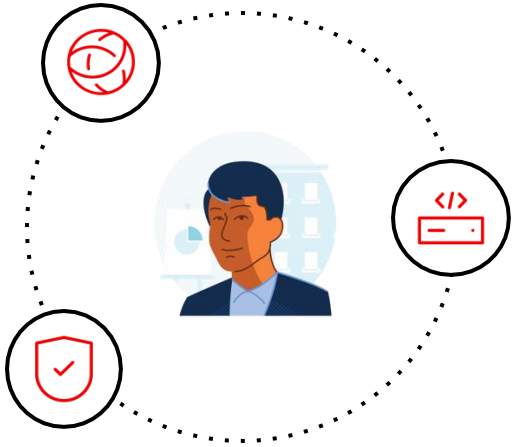
Network ops



SecOps

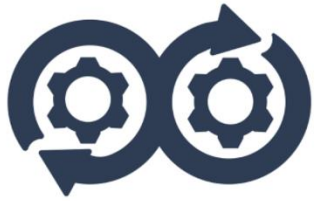


Devs/DevOps



IT ops

Ansible Architecture



Ansible Architecture

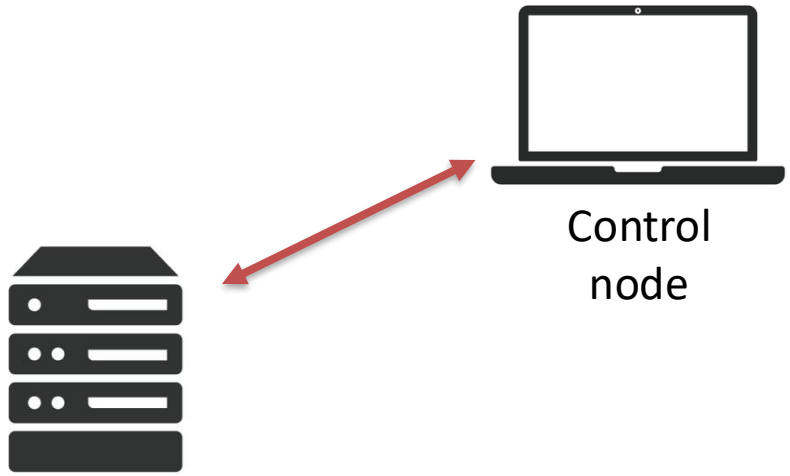
Control
node

Ansible Control Node

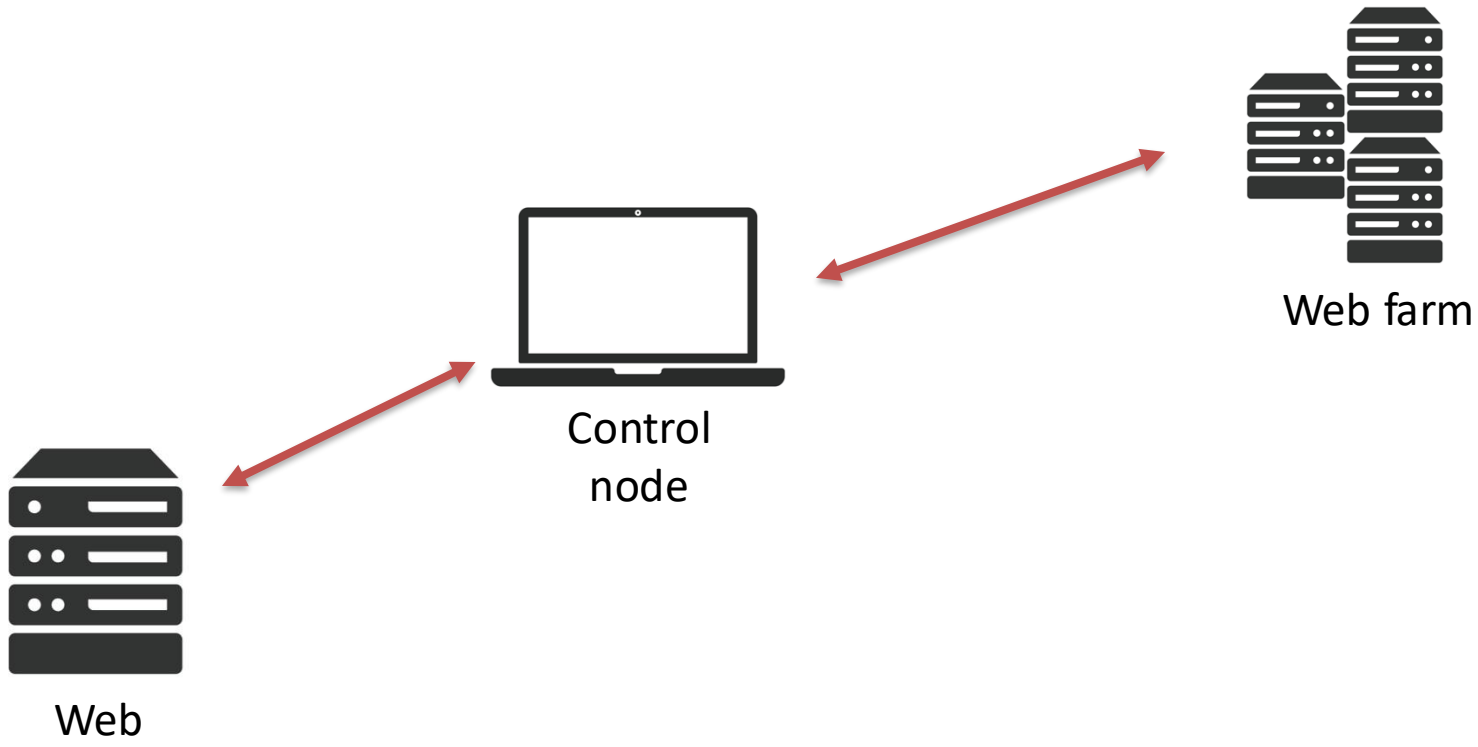
The machine from which you run the Ansible CLI tools (`ansible-playbook` , `ansible`, `ansible-vault` and others).

You can use any computer that meets the software requirements as a control node - laptops, shared desktops, and servers can all run Ansible. Multiple control nodes are possible, but Ansible itself does not coordinate across them, see AAP for such features.

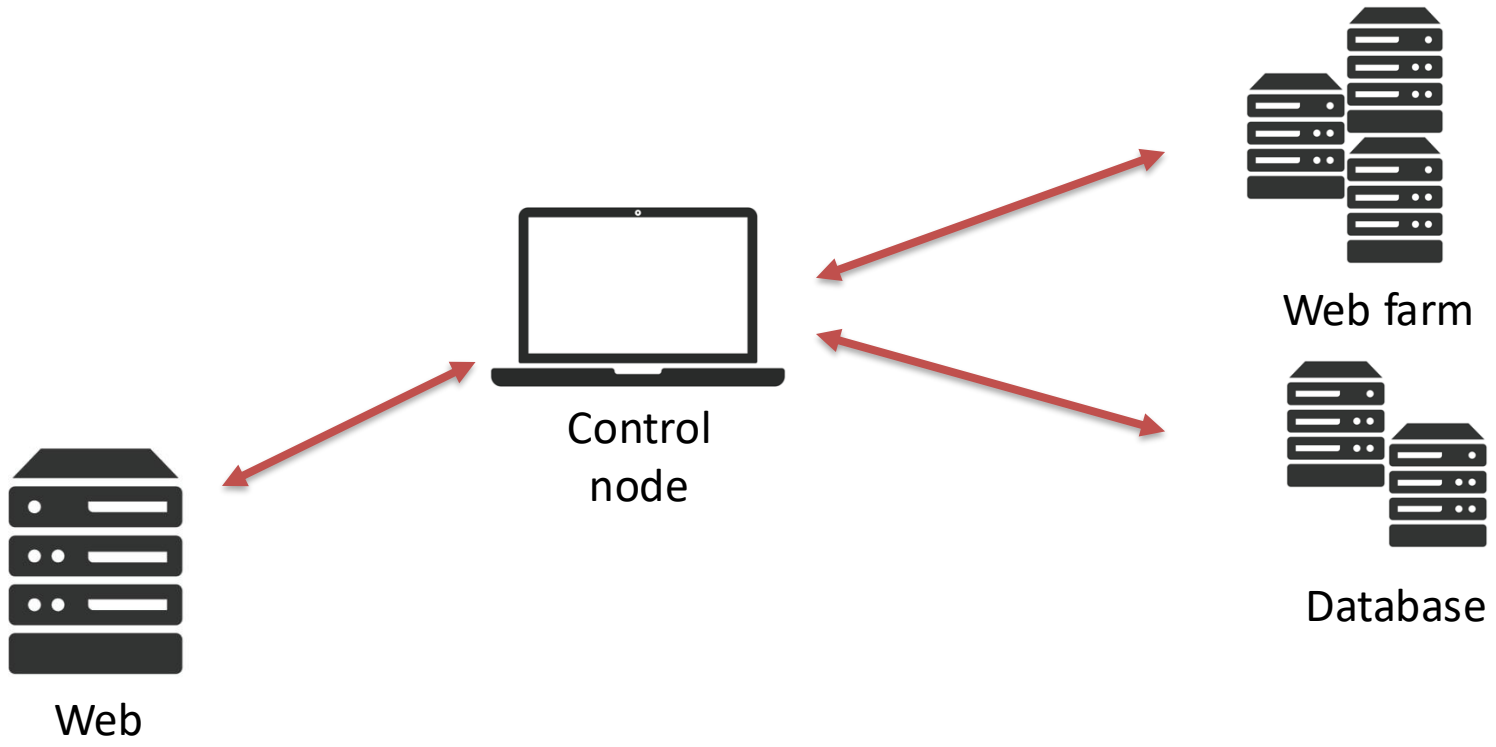
Ansible Architecture



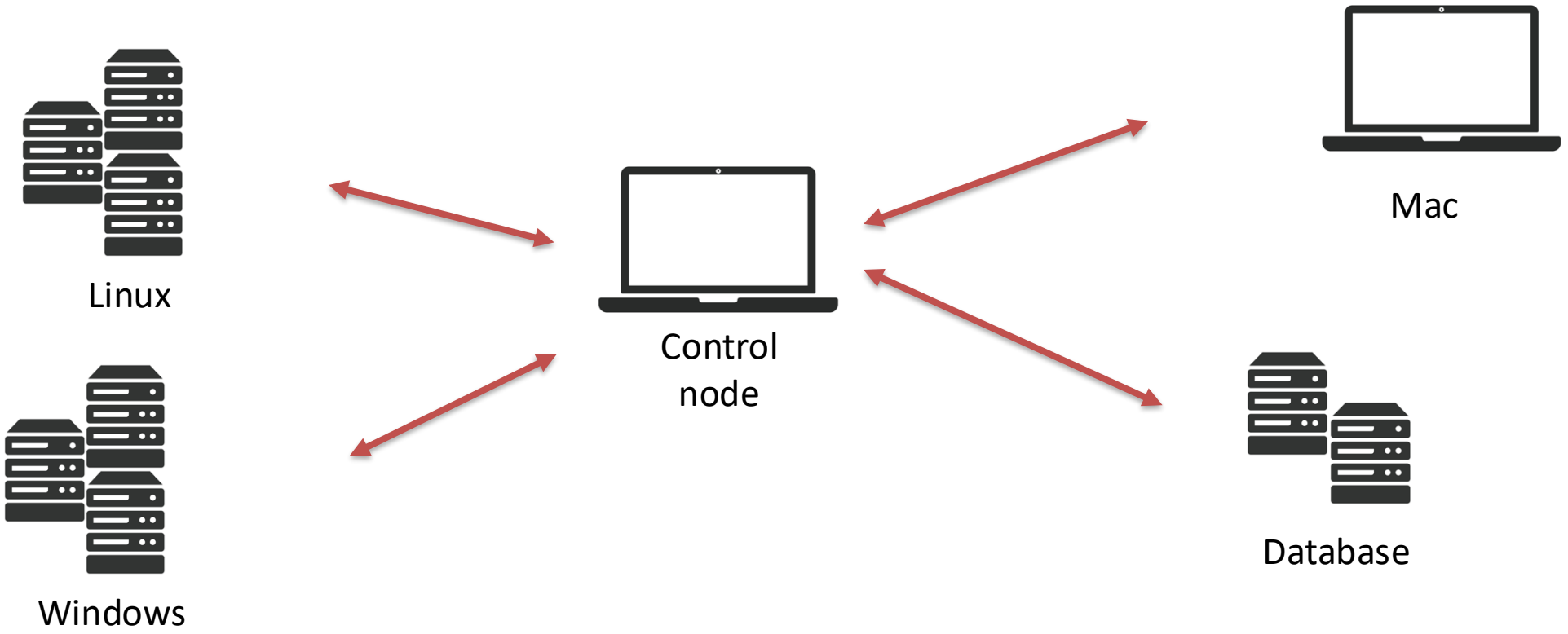
Ansible Architecture



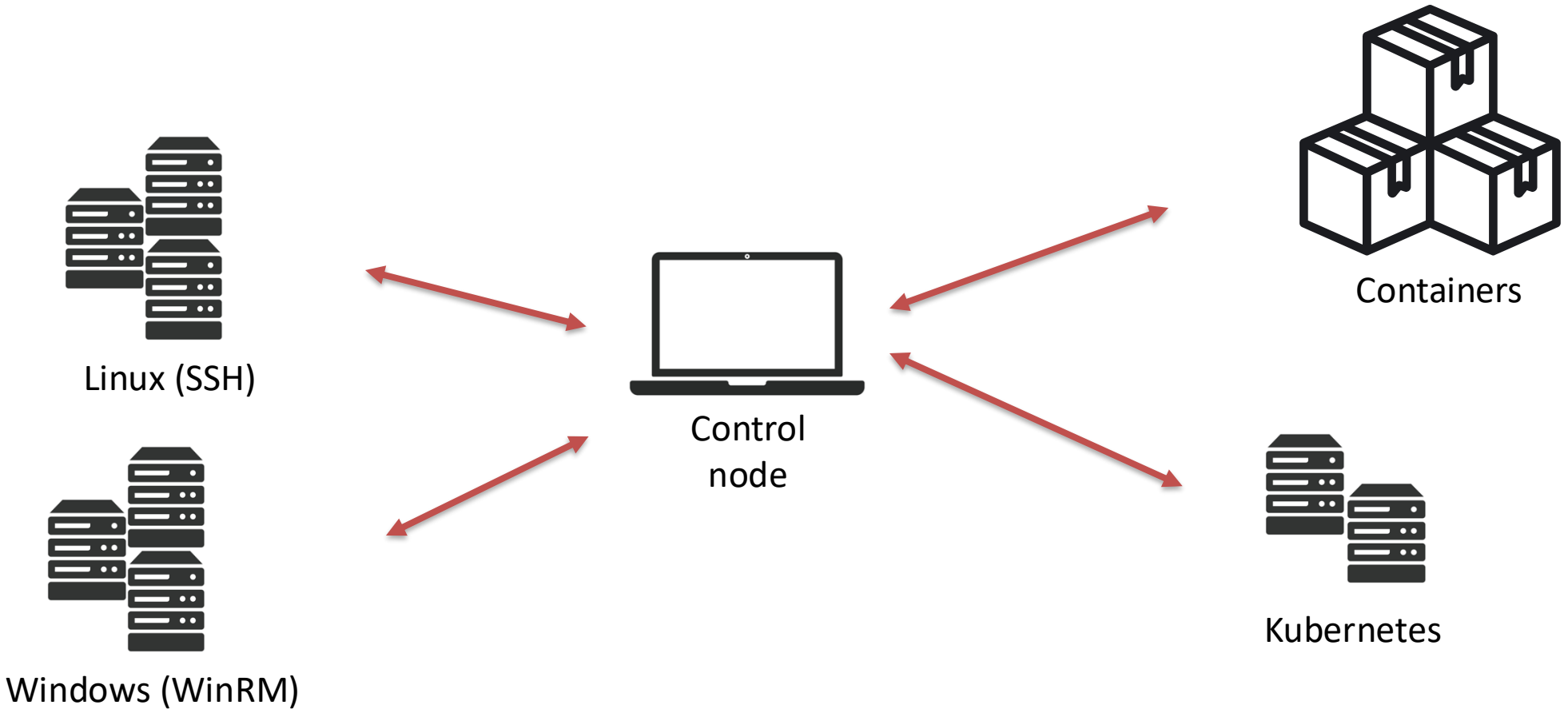
Ansible Architecture



Connecting Anywhere



Connecting Anywhere



Managed Node

Also referred to as ‘hosts’, these are the target devices (servers, network appliances or any computer) you aim to manage with Ansible.

Ansible is not normally installed on managed nodes, unless you are using `ansible-pull`, but this is rare and not the recommended setup.

Inventory

Ansible works against multiple managed nodes or “hosts” in your infrastructure at the same time, using a list or group of lists known as inventory.

Once your inventory is defined, you use patterns to select the hosts or groups you want Ansible to run against.

Inventory

The default location for inventory is a file called

- `/etc/ansible/hosts`

You can specify another inventory file/directory at the command-line using the `-i <path>` option.

You can also use multiple inventory files at the same time aand/or pull inventory from dynamic or cloud sources.

Inventory

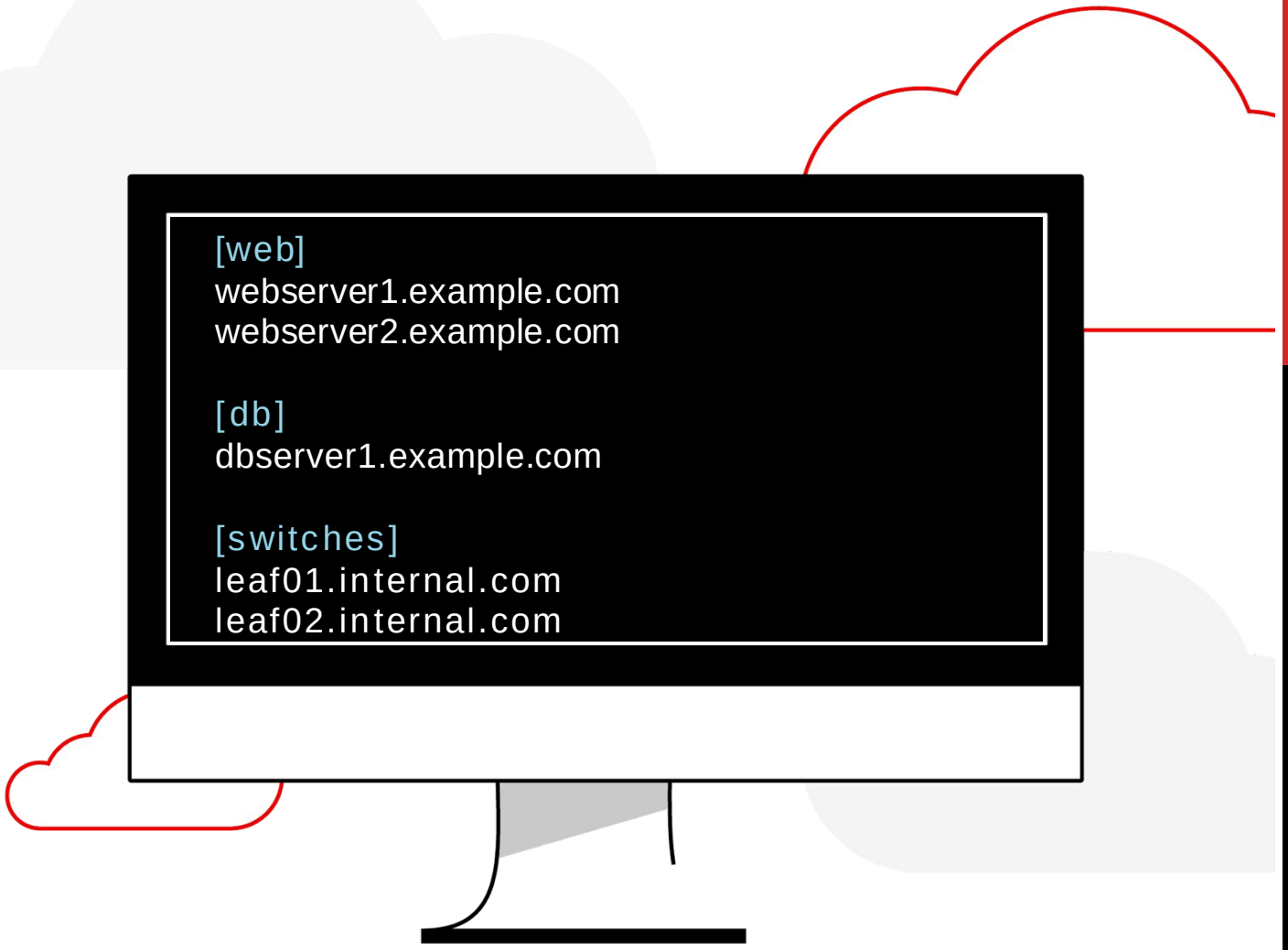
- Ansible works against multiple systems in an inventory
- Inventory is usually file based
- Can have multiple groups
- Can have variables for each group or even host

Ansible Inventory

The systems that a playbook runs against

What are they?

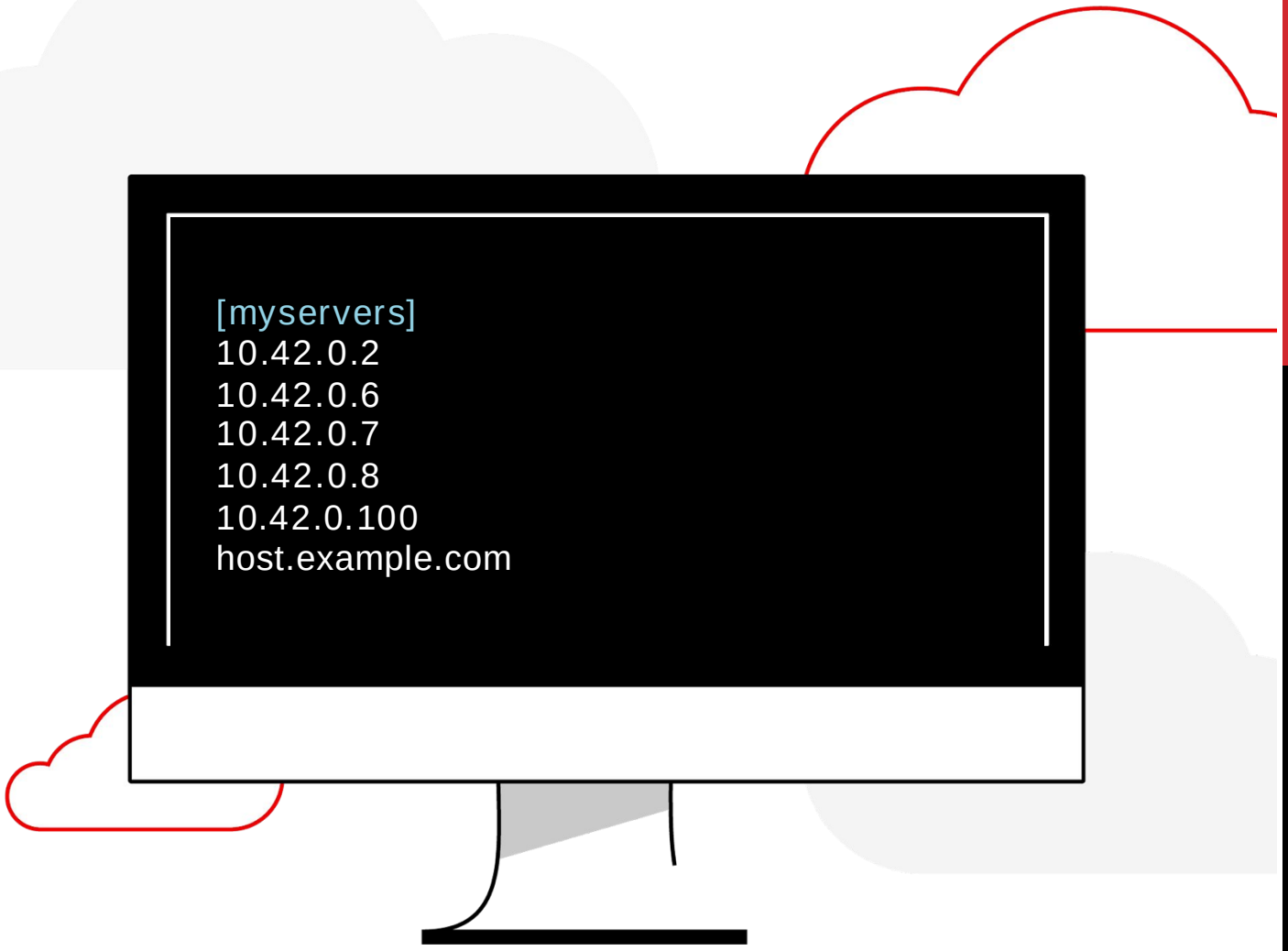
List of systems in your infrastructure that automation is executed against



Ansible Inventory

The Basics

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



Inventory Variables


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

[web]
node-[1:30]  ansible_host=10.42.0.[31:60]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=kev
ansible_ssh_private_key_file=/home/kev/.ssh/id_rsa
```

Ansible Inventory Formats

Ansible inventory can be expressed in `'INI'` or `'YAML'` format.

Example (INI):

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

Ansible Inventory Formats

Here's the same basic inventory file in YAML format:

Example (YAML):

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
```

Inventory Groups


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

[atlanta]
node-[1:30]  ansible_host=10.42.0.[31:60]

[south:children]
Atlanta
Nashville
hsvapp05
```

Ansible Inventory Groups

There are two default groups:

- all: Contains every host
- ungrouped: Contains all hosts that don't have another group aside from all.

Every host will always belong to at least 2 groups (all and ungrouped or all and some other group).

Though all and ungrouped are always present, they can be implicit and not appear in group listings.

Ansible Inventory Groups

You can (and probably will) put each host in more than one group. For example, a production webserver in a datacenter in Atlanta might look like this:

```
all:
  children:
    prod:
      hosts:
        web1.example.com:
    atlanta:
      hosts:
        web1.example.com
    webservers:
      hosts:
        web1.example.com
```

Ansible Inventory Ranges

If you have a lot of hosts with a similar pattern, you can add them as a range rather than listing each hostname separately:
Example (INI):

```
[webservers]
www[01:50].example.com
```

```
...
  webservers:
    hosts:
      www[01:50].example.com:
```

Ansible Dynamic Inventory

If your Ansible inventory fluctuates over time, with hosts spinning up and shutting down in response to business demands you may need to track hosts from multiple sources: cloud providers, LDAP, Cobbler, and/or enterprise CMDB systems.

Ansible integrates all of these options through a dynamic inventory system. Ansible uses inventory plugins to keep track of managed hosts.

Ansible Dynamic Inventory

Dynamic inventory supports many solutions including Cloud Providers.
Example (AWS):

```
plugin: aws_ec2
regions:
  - us-west-1
filters:
  instance-state-name: running
keyed_groups:
  - key: tags['role']
    prefix: tag_role
hostnames:
  - ip-address
```

POP QUIZ: DISCUSSION

Where is the default inventory location?

POP QUIZ: DISCUSSION

Where is the default inventory location?

- `/etc/ansible/hosts`

POP QUIZ: DISCUSSION

Which formats are valid for an Ansible inventory?

POP QUIZ: DISCUSSION

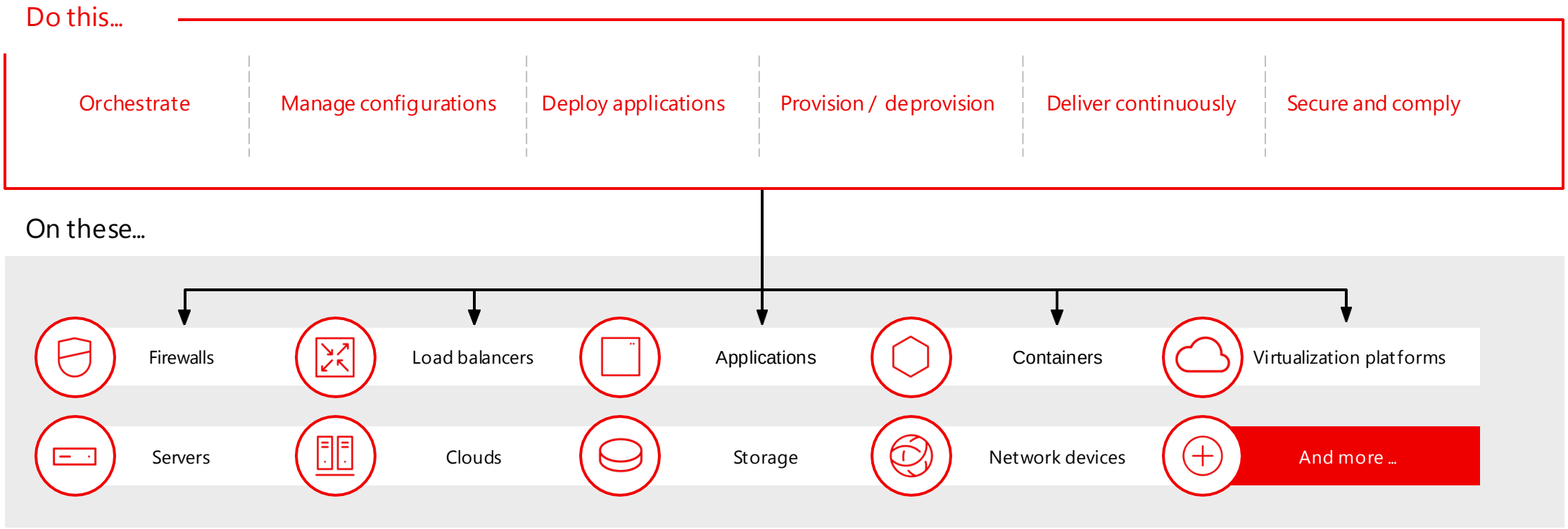
Which formats are valid for an Ansible inventory?

- INI
- YAML

Why The Ansible Automation Platform?

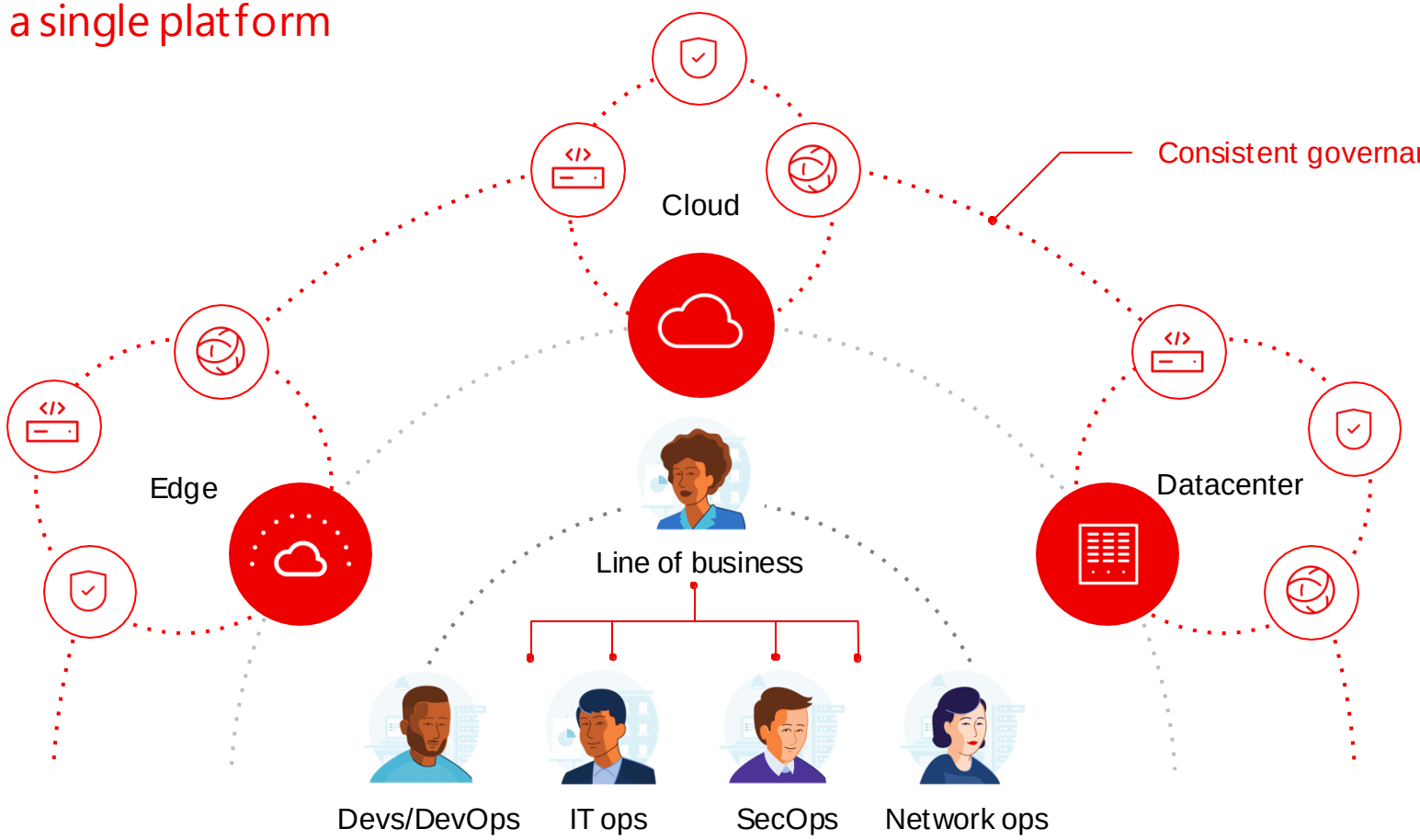
Automate the deployment and management of automation

Your entire IT footprint



Break Down Silos

Different teams a single platform



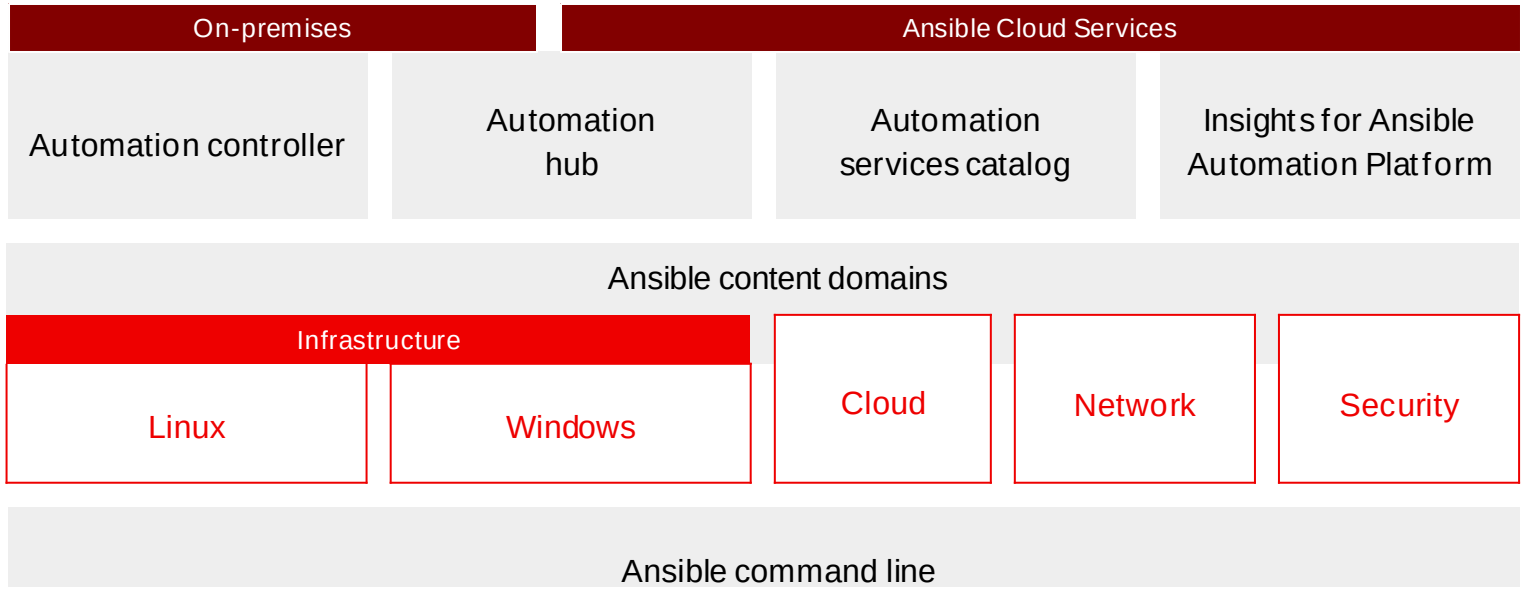
Red Hat
Ansible Automation
Platform

Content creators

Operators

Domain experts

Users



Fueled by an
open source community

Automation Platform Concepts

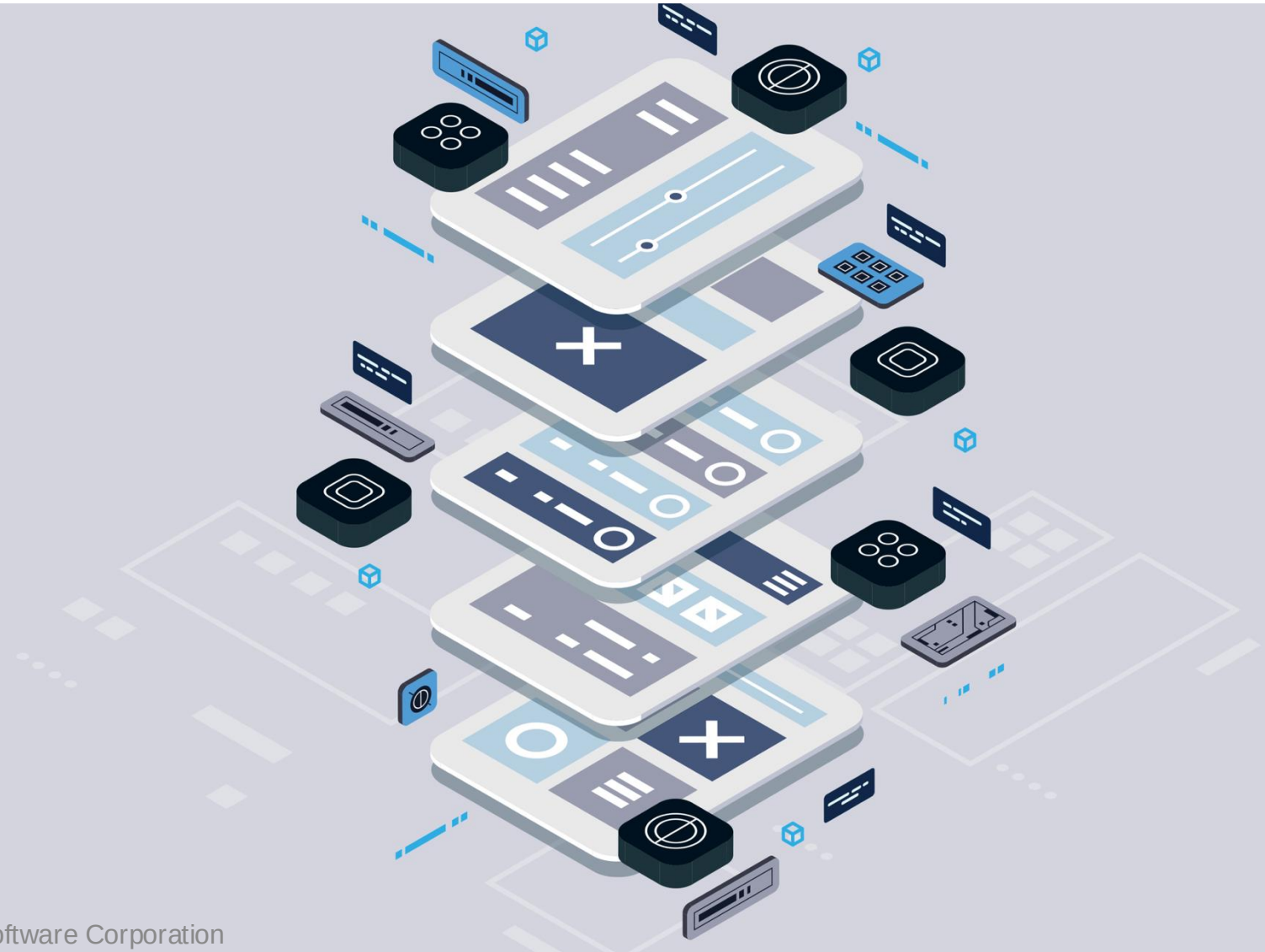
A control node is installed with Ansible and is used to run playbooks.

- Contains the Ansible Engine software, the playbook, and it's supporting files.
- Red Hat Automation Platform is a control node that also provides a central web interface, authentication, and API for Ansible.

A managed host is a machine that is managed by Ansible automation.

- Does not have Ansible installed
- Does need to be configured to allow Ansible to connect to the host
- Must be listed in the inventory (or generated by a dynamic inventory script or plugin)

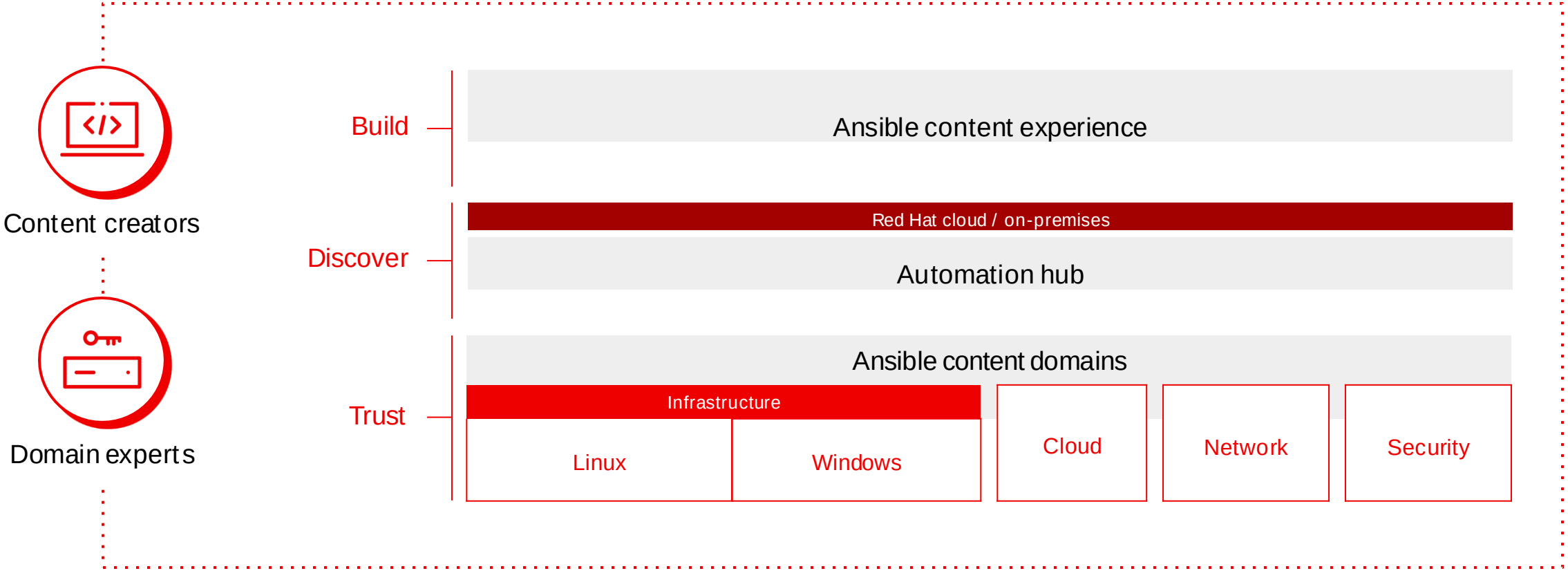
Ansible Automation Platform Infrastructure



Create Code

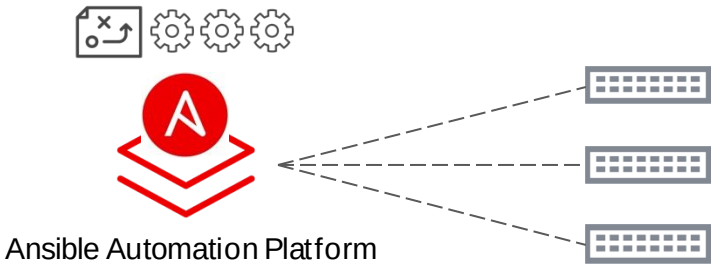
Create

The automation lifecycle



How Ansible Automation Works

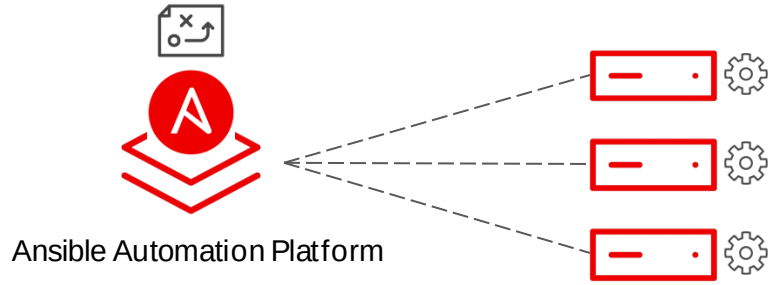
Module code is executed locally on the control node



Local Execution

Network Devices / API Endpoints

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



Remote Execution

Linux / Windows Hosts

Installation Architecture Options

Red Hat Ansible Automation Platform can be implemented using one of the following architectures:

- Single Machine with Integrated Database
- Single Machine with Remote Database
- Multi Machine Cluster with Remote Database
- OpenShift Pod with Remote Database

Lab Environment

This class uses a single-node installation with an integrated database:

- Need a system installed with Red Hat Enterprise Linux (bare metal, virtual machine, or cloud instance)
  - Cloud VM
  - 2 vCPU / 4 GB RAM / at least 40GB of storage

Sign up for an evaluation of Automation Platform from Red Hat:

- https://www.redhat.com/en/technologies/management/ansible/try-it

Installer

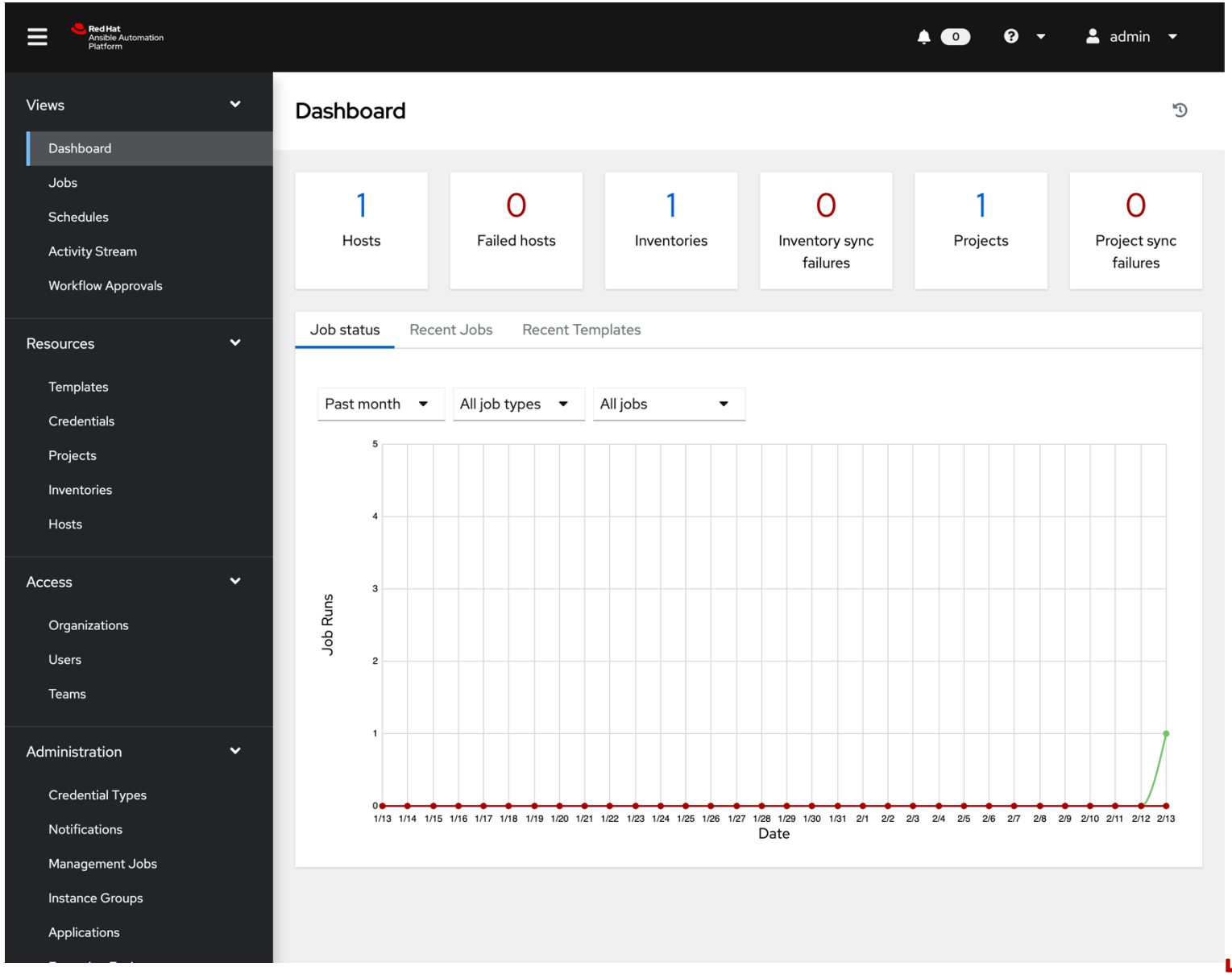
Two different installation packages are available for Automation Platform:

- Standard setup
  - https://access.redhat.com/downloads/content/480
  - Requires internet connectivity to download Automation Platform packages from repositories.

- Bundled installer
  - Download from same link as “Standard setup”, but choose the packages with “Bundle” in the name.
  - Includes initial RPM packages for Automation Platform
  - May be installed on systems without internet access.

Automation Platform Dashboard

- The main control center for Red Hat Ansible Tower.
- Displayed when you log in.
- Composed of four reporting sections:
  - Summary
  - Job Status
  - Recently Used
  - TemplatesRecent Job Runs

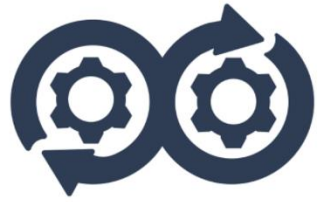


Dashboard Navigation

The dashboard contains links to common resources.

Organizations	Control what resources are visible to which users.
Teams	Groups of users that need to access the same resources
Users	User accounts
Jobs	A history of previous Ansible runs.
Templates	Prepared playbooks settings that can be “launched” to run a job.
Credentials	Authentication secrets for managed hosts and Git repositories
Projects	Sources of Ansible Playbooks (usually Git repo)
Inventories	Inventories of managed hosts
Inventory Scripts	Dynamic inventory
Notifications	Configurable for job completion or failure
Management Jobs	Special jobs used to maintain Ansible Platform.

Lab: Setup Ansible Tower



Ansible Ad-hoc

An Ansible ad hoc command uses the `/usr/bin/ansible` command-line tool to automate a single task on one or more managed nodes. ad hoc commands are quick and easy, but they are not reusable.

Why learn about ad hoc commands first? ad hoc commands demonstrate the simplicity and power of Ansible. The concepts you learn will port over directly to the playbook language.

POP QUIZ: DISCUSSION

What are some use-cases for ad-hoc mode?

POP QUIZ: DISCUSSION

What are some use-cases for ad-hoc mode?

- Copy files

POP QUIZ: DISCUSSION

What are some use-cases for ad-hoc mode?

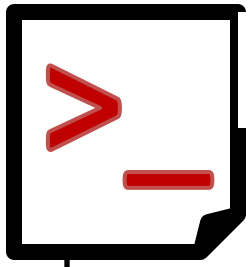
- Copy files
- Manage packages, users, groups

POP QUIZ: DISCUSSION

What are some use-cases for ad-hoc mode?

- Copy files
- Manage packages, users, groups
- Reboot servers

Ansible ad-hoc



Scripted Ad-hoc

```
ansible -m copy -a "src=master.gitconfig dest=~/.gitconfig" localhost

ansible -m homebrew -a "name=bat state=latest" localhost

ansible -i hosts all -m ping -u ubuntu
```

Ansible ad-hoc

ad hoc commands are great for tasks you repeat rarely. For example, if you want to power off all the machines in your lab for Christmas vacation, you could execute a quick one-liner in Ansible without writing a playbook. An ad hoc command looks like this:

```
$ ansible [pattern] -m [module] -a "[module options]"
```

Example of installing packages on nodes in webservers group.

```
$ ansible webservers -m yum -a "name=apache state=present"
```

YAML

Playbooks are written in YAML.
YAML is used because it is easier for humans to read and write than other data formats like XML and JSON.
It is a format widely used by many tools (Kubernetes, Docker compose, Machine Learning, etc.)

YAML Basics

For Ansible, nearly every YAML file starts with a list. Each item in the list is a list of key/value pairs, commonly called a “hash” or a “dictionary”. So, we need to know how to write lists and dictionaries in YAML.

There’s another small quirk to YAML. All YAML can optionally begin with --- and end with …. This is part of the YAML format and indicates the start and end of a document.

YAML Basics

All members of a list are lines beginning at the same indentation level starting with a "- " (a dash and a space):

Example:

```
---
# A list of tasty fruits
- Apple
- Orange
- Strawberry
- Mango
...
```

YAML Basics

A dictionary is represented in a simple key: value form (the colon must be followed by a space):

Example:

```
---
# An employee record
martin:
  name: Martin D'vloper
  job: Developer
  skill: Elite
```

YAML Basics

More complicated data structures are possible, such as lists of dictionaries, dictionaries whose values are lists or a mix of both:

Example:

```
---
# Employee records
- martin:
    name: Martin D'vloper
    job: Developer
    skills:
      - python
      - perl
      - pascal
- tabitha:
    name: Tabitha Bitumen
    job: Developer
    skills:
      - lisp
      - fortran
      - erlang
```

YAML Basics

Values can span multiple lines using | or >.

Spanning multiple lines using a “Literal Block Scalar” | will include the newlines and any trailing spaces.

Using a “Folded Block Scalar” > will fold newlines to spaces; it’s used to make what would otherwise be a very long line easier to read and edit.

In either case the indentation will be ignored.

YAML Basics

Example:

```
include_newlines: |
        exactly as you see
        will appear these three
        lines of poetry

fold_newlines: >
        this is really a
        single line of text
        despite appearances
```

Ansible Playbook

Ansible Playbooks offer a repeatable, reusable, simple configuration management and multi-machine deployment system, one that is well suited to deploying complex applications.

If you need to execute a task with Ansible more than once

- Write a playbook
- Put it under source control.

Then you can use the playbook to push out new configurations or confirm the configuration of remote systems.

Ansible Playbook

Playbooks can:

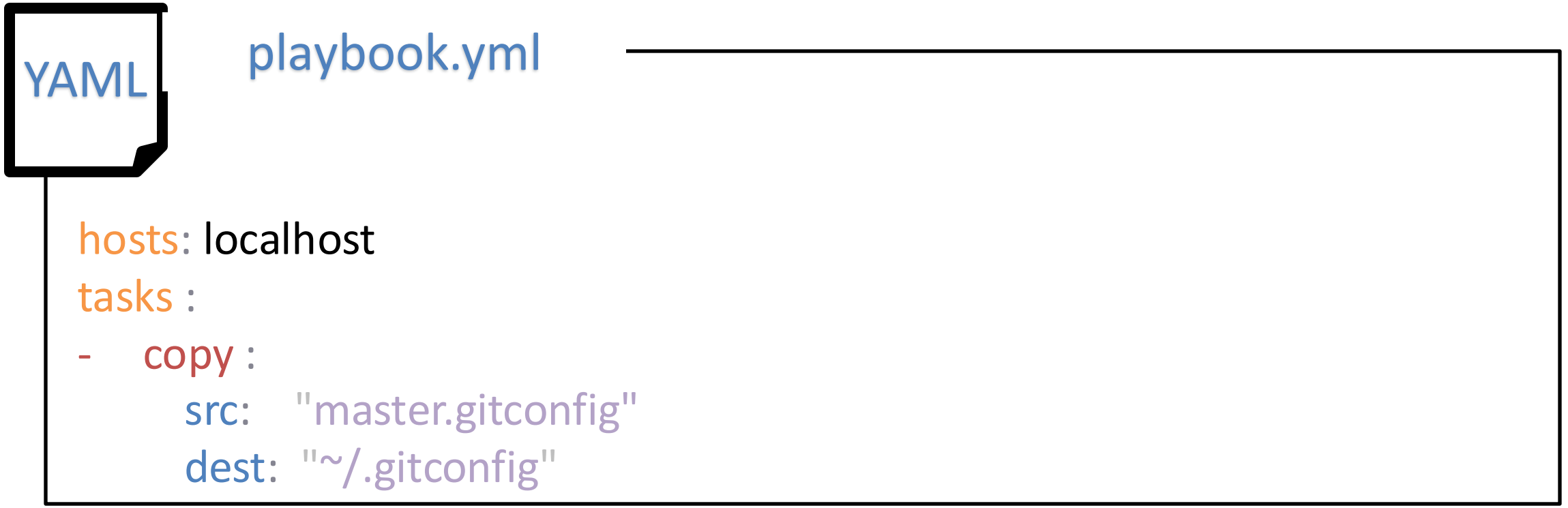
- Declare configurations
- Orchestrate steps of any manual ordered process, on multiple sets of machines, in a defined order
- Launch tasks synchronously or asynchronously

Ansible Playbook

A playbook is composed of one or more ‘plays’ in an ordered list.

The terms ‘playbook’ and ‘play’ are sports analogies. Each play executes part of the overall goal of the playbook, running one or more tasks. Each task calls an Ansible module.

Playbook



Playbook

playbook.yml

```
hosts: localhost
tasks :
-    copy :
        src:    "master.gitconfig"
        dest:  "~/.gitconfig"
```

>_

```
ansible-playbook playbook.yml
```

Ansible Playbook Execution

A playbook runs in order from top to bottom. Within each play, tasks also run in order from top to bottom.

Playbooks with multiple ‘plays’ can orchestrate multi-machine deployments, running one play on your webservers, then another play on your database servers, then the third play on your network infrastructure, and so on.

Ansible Playbook Tips

At a minimum, each play defines two things:

- The managed nodes to target, using a pattern
- At least one task to execute

- Ansible creates a <playbook>.retry playbook for hosts where it failed. You can execute the <playbook>.retry playbook and it will try to run it ONLY on the hosts that failed.
- Limit: Used to run Ansible playbook only on hosts you specify. Great for testing on one host.
- Whitespace: Ansible is like Python, it requires correct indentation. (ansible lint, syntax-check, etc.)

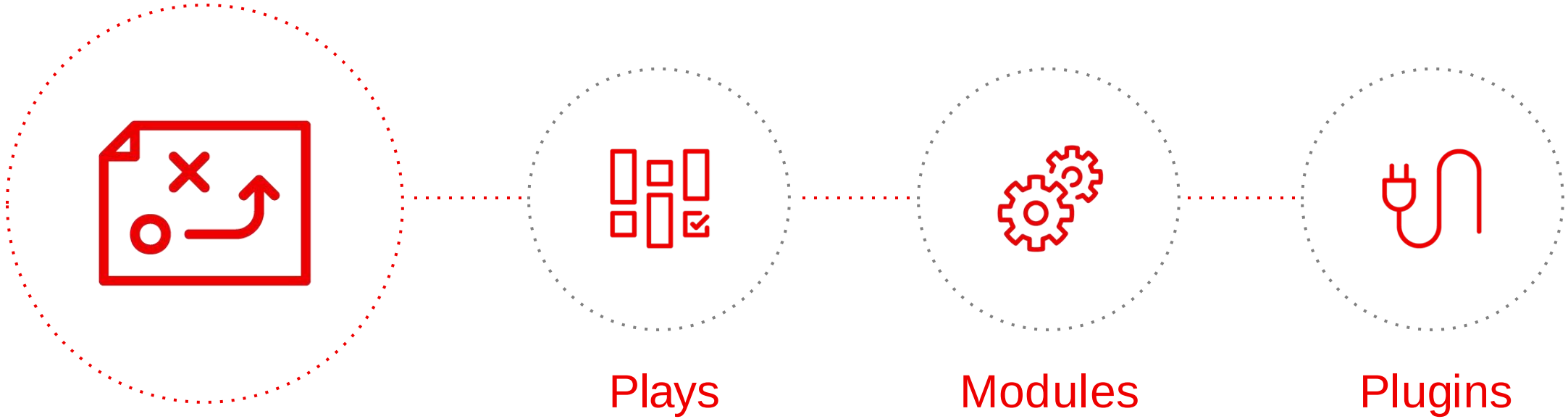
Playbooks

Simple playbook to install and configure Apache web server

```
---
- 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
```

Playbooks

What makes up an Ansible playbook?



Ansible Plays

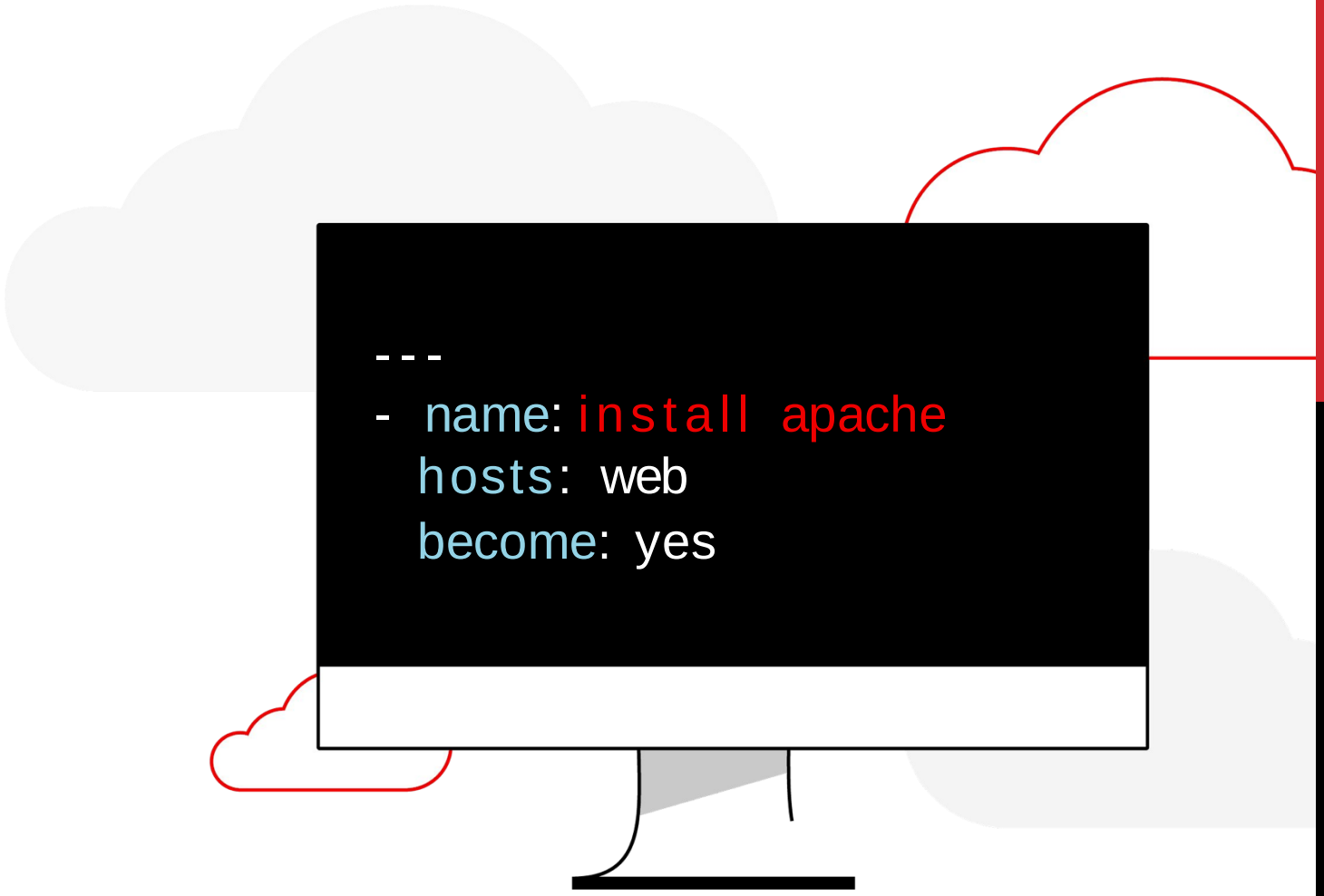
What am I automating?

What are they?

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

Building blocks for playbooks

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



Ansible Modules

The “tools in the toolkit”

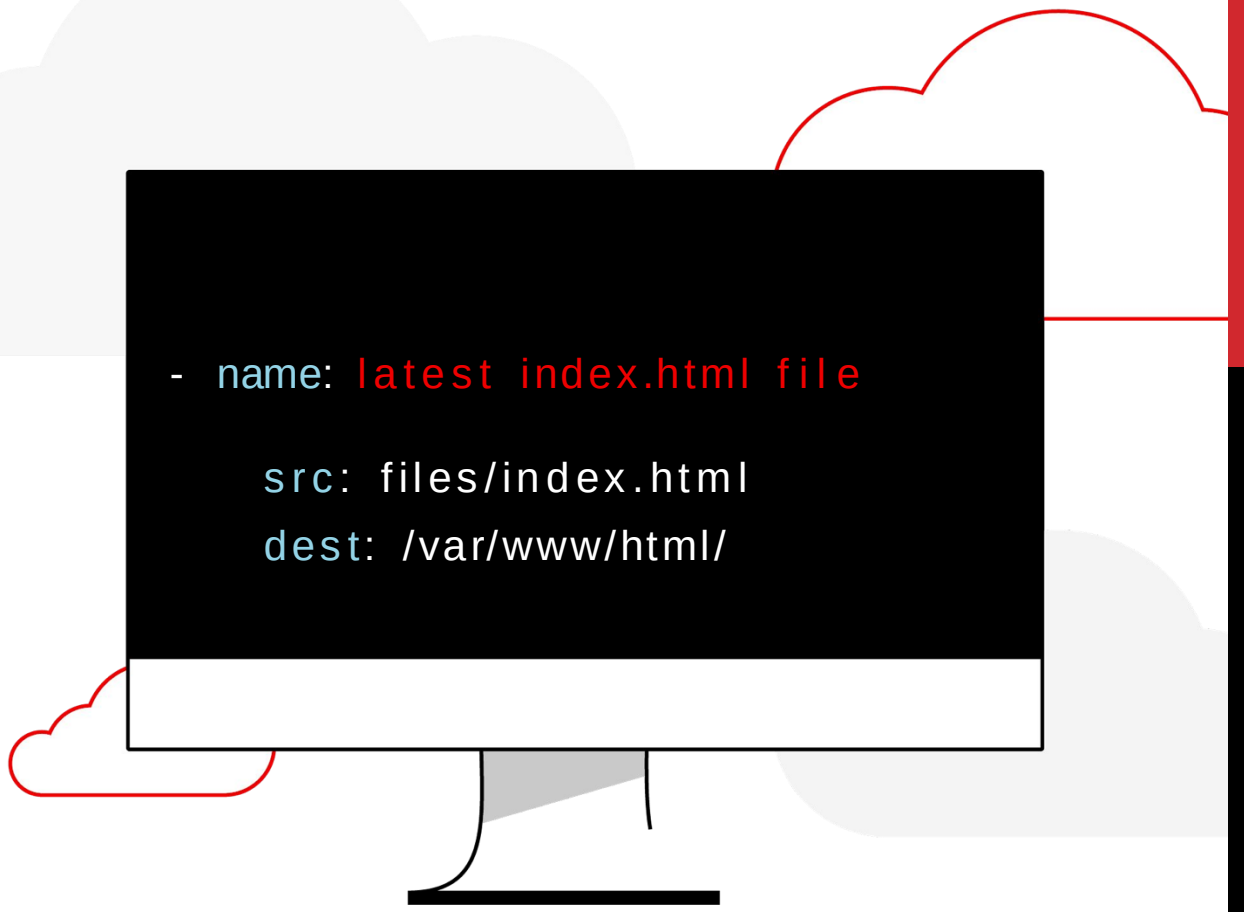
What are they?

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.



More Complete Playbook

```
- name: Install Docker
  hosts: tag_role_k8s_master:tag_role_k8s_member
  remote_user: ubuntu
  gather_facts: no
  handlers:
    - include: roles/handlers/main.yml
  tasks:
    - package:
        name: "docker-ce"
        state: latest
    - user:
        name: "{{ ansible_ssh_user }}"
        groups: docker
        append: yes
      notify: restart docker
```

Name of Play

More Complete Playbook

```yaml
- name: Install Docker
  hosts: tag_role_k8s_master:tag_role_k8s_member
  remote_user: ubuntu
  gather_facts: no
  handlers:
    - include: roles/handlers/main.yml
  tasks:
    - package:
        name: "docker-ce"
        state: latest
    - user:
        name: "{{ ansible_ssh_user }}"
        groups: docker
        append: yes
      notify: restart docker
```

Hosts to run Play on

More Complete Playbook

```
- name: Install Docker
  hosts: tag_role_k8s_master:tag_role_k8s_member
  remote_user: ubuntu
  gather_facts: no
  handlers:
    - include: roles/handlers/main.yml
  tasks:
    - package:
        name: "docker-ce"
        state: latest
    - user:
        name: "{{ ansible_ssh_user }}"
        groups: docker
        append: yes
      notify: restart docker
```

User to run Play as

More Complete Playbook

```
- name: Install Docker
  hosts: tag_role_k8s_master:tag_role_k8s_member
  remote_user: ubuntu
  gather_facts: no
  handlers:
    - include: roles/handlers/main.yml
  tasks:
    - package:
        name: "docker-ce"
        state: latest
    - user:
        name: "{{ ansible_ssh_user }}"
        groups: docker
        append: yes
      notify: restart docker
```

Define what to do after package is installed.

More Complete Playbook

```
- name: Install Docker
  hosts: tag_role_k8s_master:tag_role_k8s_member
  remote_user: ubuntu
  gather_facts: no
  handlers:
    - include: roles/handlers/main.yml
  tasks:
    - package:
        name: "docker-ce"
        state: latest
    - user:
        name: "{{ ansible_ssh_user }}"
        groups: docker
        append: yes
      notify: restart docker
```

Call the defined handler

More Complete Playbook

```yaml
- name: Install Docker
  hosts: tag_role_k8s_master:tag_role_k8s_member
  remote_user: ubuntu
  gather_facts: no
  handlers:
    - include: roles/handlers/main.yml
  tasks:
    - package:
        name: "docker-ce"
        state: latest
    - user:
        name: "{{ ansible_ssh_user }}"
        groups: docker
        append: yes
      notify: restart docker
```

Tasks to run
- install Docker

More Complete Playbook

```yaml
- name: Install Docker
  hosts: tag_role_k8s_master:tag_role_k8s_member
  remote_user: ubuntu
  gather_facts: no
  handlers:
    - include: roles/handlers/main.yml
  tasks:
    - package:
        name: "docker-ce"
        state: latest
    - user:
        name: "{{ ansible_ssh_user }}"
        groups: docker
        append: yes
      notify: restart docker
```

Create user account

Ansible Variables

Ansible uses variables to manage differences between systems. With Ansible, you can execute tasks and playbooks on multiple different systems with a single command.

- Create variables with YAML syntax, including lists and dictionaries
- Define these variables

Ansible Variables

Variables can be defined in many locations:

- Playbooks
- Inventory
- re-usable files or roles
- command line.

You can also create variables during a playbook run by registering the return value or values of a task as a new variable.

Ansible Variable Valid Names

A variable name can only contain:

- Letters
- Numbers
- Underscores

A variable name cannot begin with a number, but can begin with an underscore.

POP QUIZ: DISCUSSION

Are these valid variable names?

- foo

POP QUIZ: DISCUSSION

Are these valid variable names?

- foo - valid

POP QUIZ: DISCUSSION

Are these valid variable names?

- app.port

POP QUIZ: DISCUSSION

Are these valid variable names?

- app.port - invalid

POP QUIZ: DISCUSSION

Are these valid variable names?

- *ssl_key

POP QUIZ: DISCUSSION

Are these valid variable names?

- *ssl_key - invalid

POP QUIZ: DISCUSSION

Are these valid variable names?

- _web_root

POP QUIZ: DISCUSSION

Are these valid variable names?

- _web_root - valid

Simple Variable

Defining simple variables

```
remote_install_path: /opt/my_app_config
```

After you define a variable, use Jinja2 syntax to reference it. Jinja2 variables use double curly braces.

```
ansible.builtin.template:
  src: foo.cfg.j2
  dest: '{{ remote_install_path }}/foo.cfg'
```

Variable Quotation

If you start a value with {{ foo }}, you must quote the whole expression to create valid YAML syntax. If you do not quote the whole expression, the YAML parser cannot interpret the syntax - it might be a variable or it might be the start of a YAML dictionary.

This example will error.

```
- hosts: app_servers
  vars:
    app_path: {{ base_path }}/22
```

ERROR! Syntax Error while loading YAML

Variable Quotation

If you start a value with {{ foo }}, you must quote the whole expression to create valid YAML syntax. If you do not quote the whole expression, the YAML parser cannot interpret the syntax - it might be a variable or it might be the start of a YAML dictionary.

Fix the issue by quoting the entire expression.

```
- hosts: app_servers
  vars:
    app_path: "{{ base_path }}/22"
```

Ansible Modules

Modules are the main building blocks of Ansible playbooks. Although we do not generally speak of "module plugins", a module is a type of plugin.

Common modules:

- Working with files: copy, archive, unarchive, get_url
- user, group
- ping
- service
- yum, apt, package
- template

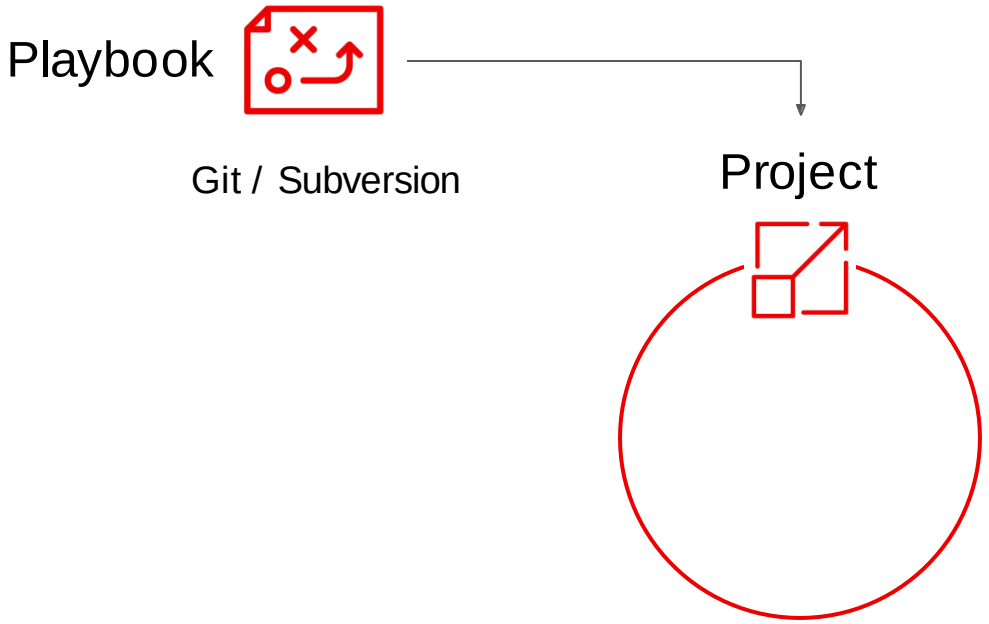
Ansible Modules

Common modules:

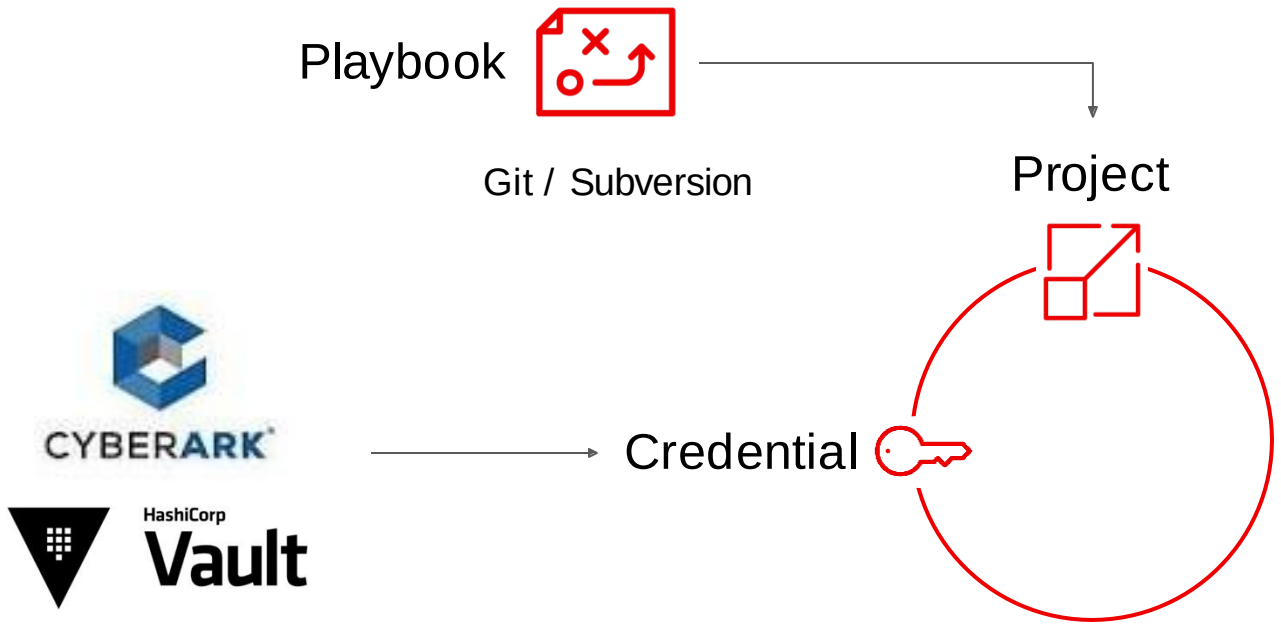
- Lineinfile
  - Manipulate text in files
    - Add alias for hosts
    - Supports regex
    - Idempotent
- Shell/command
- Script module
- Debug module

Automation Controller

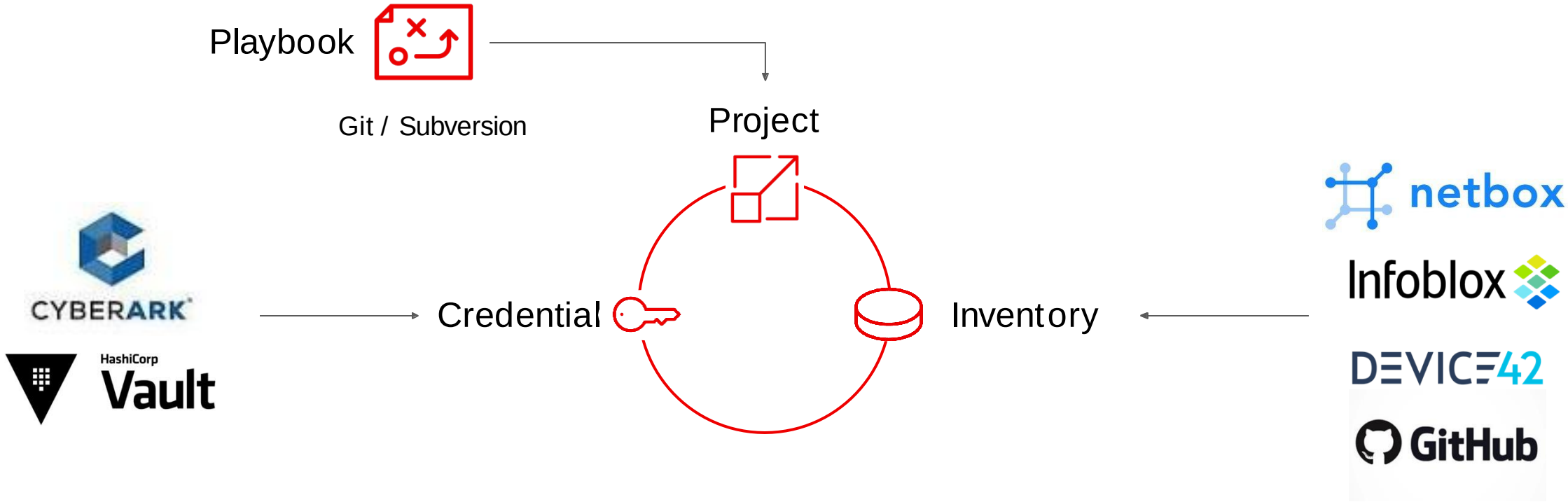
Anatomy of An Automation Job



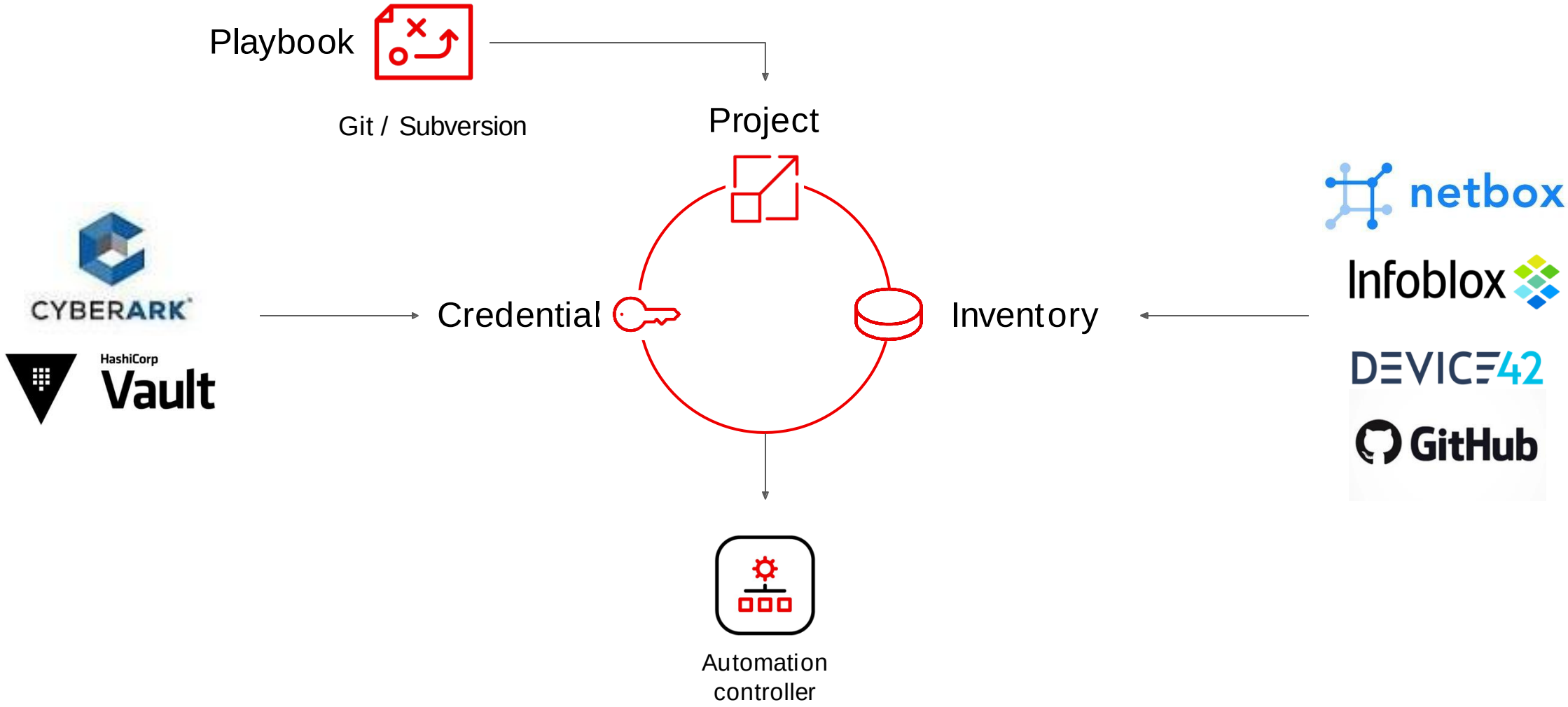
Anatomy of An Automation Job



Anatomy of An Automation Job

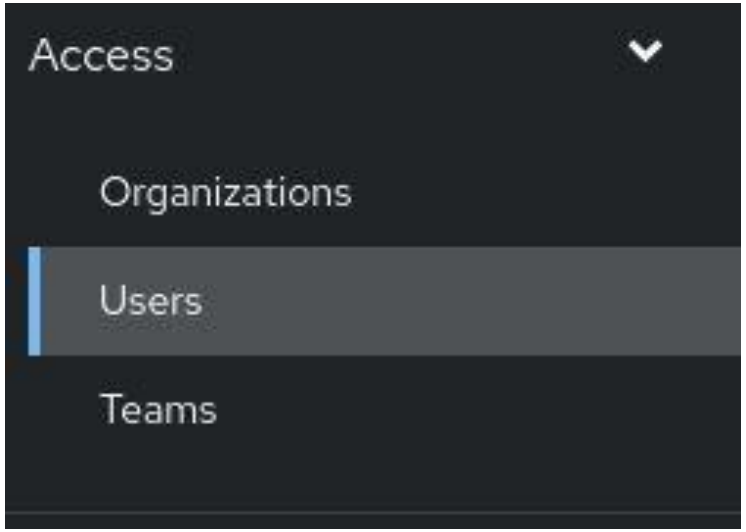


Anatomy of An Automation Job



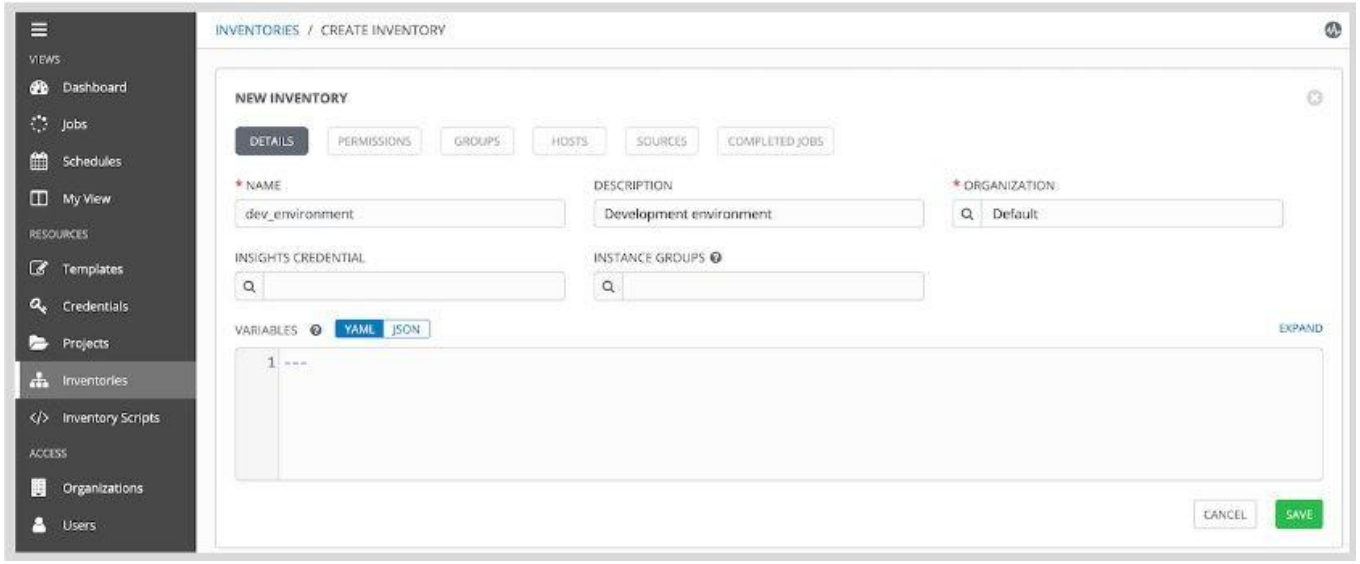
User Management

- 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.



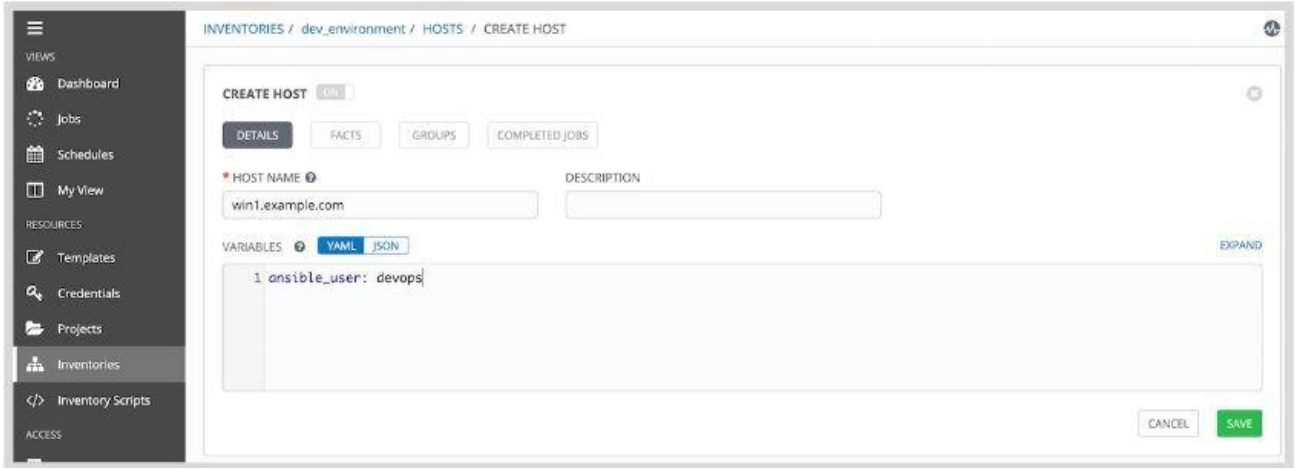
Inventory

- Log into Ansible Platform (the admin user will work for this example).
- Click on **Inventories**.
- In the INVENTORIES window, click the **+** button.
- Enter a NAME for the inventory and its ORGANIZATION (often “Default”)



Inventory

- In the Ansible Platform GUI, click the **Inventories** menu, then click on the name of the inventory.
  - Click the **HOSTS** button, then click on **+**. This displays the “Create a new host” tooltip.
  - In the HOST NAME field enter the hostname or IP address of the managed host.
  - In the VARIABLES text box, you can set values for variables that apply only to this host.
  - Click **SAVE**.

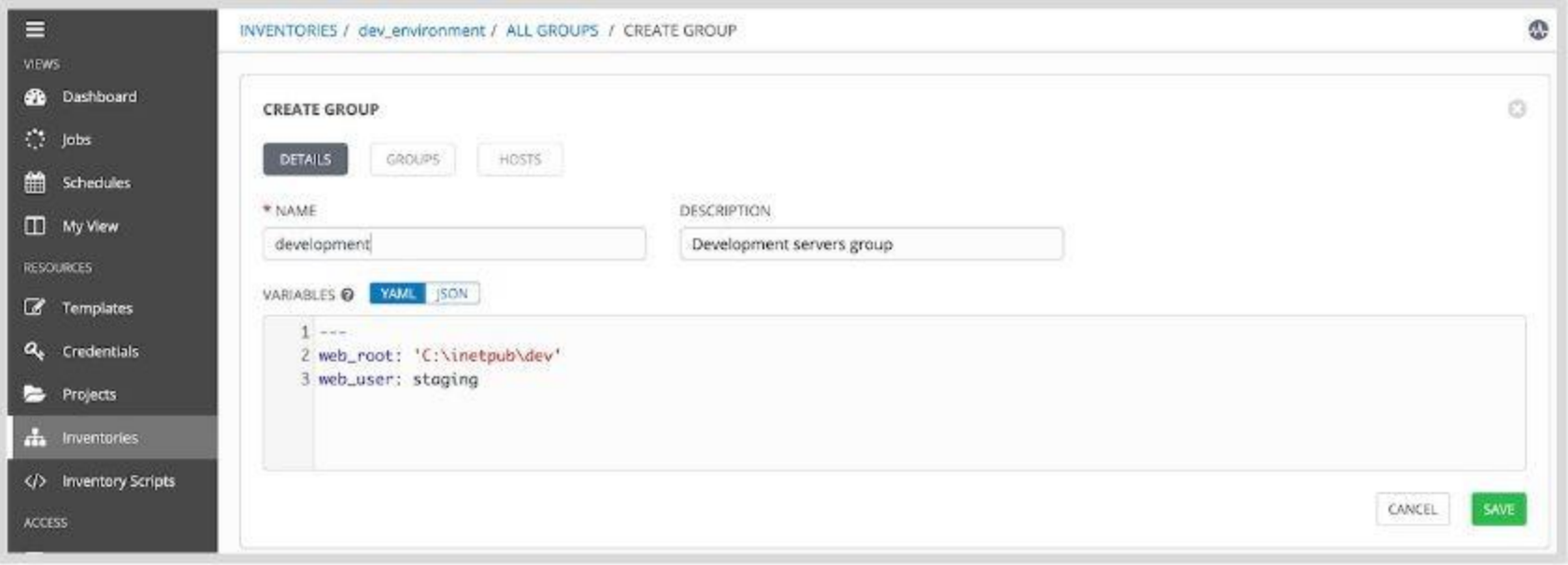


Inventory Groups

- Groups allow you to organize hosts into a set that can be managed together
- Hosts may be in multiple groups at the same time
  - All hosts that are in a particular data center
  - All hosts that have a particular purpose
  - Dev / Test / Prod hosts can be grouped
- Groups can be nested
  - The *europe* group might include a *paris_dc* group and a *london_dc* group
- This allows you to run playbooks on particular groups
- This allows you to set a variable to a specific value for all hosts in a group

Inventory Groups

- In the Ansible Platform GUI, click the **Inventories** menu, and click on the inventory to edit.
- Click the **GROUPS** button, then click on **+**. This will open the “Create a new group” tooltip.
- In the NAME field, enter the name of the group.
- Define any values for variables
- Click **SAVE**.

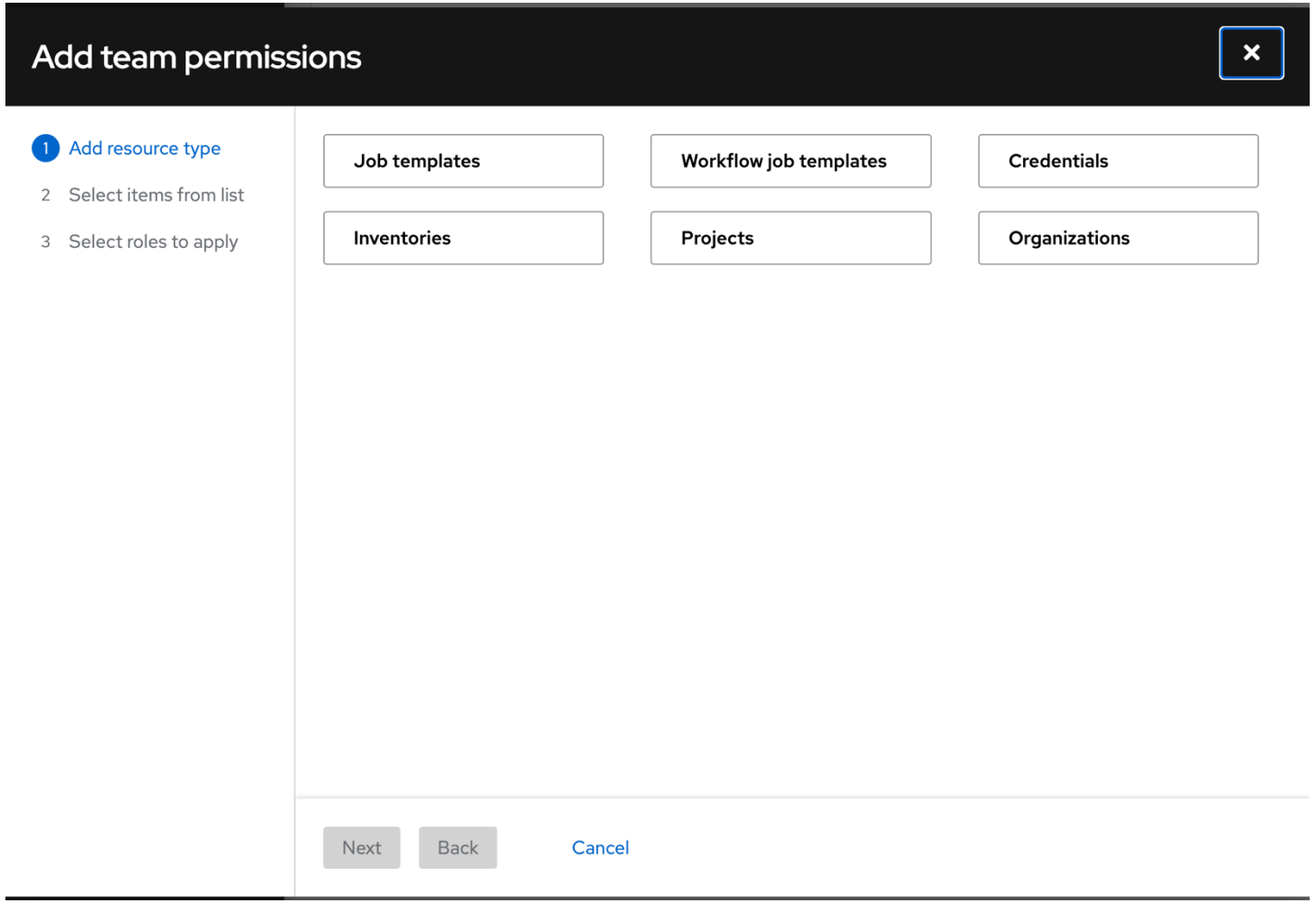


Inventory Groups

- In the Ansible Platform GUI, click the **Inventories** menu, and click on the inventory to edit.
- Click the **GROUPS** button, then click on the group to edit.
- Click the **HOSTS** button, then click on **+**. This will open the “Add a host” tooltip. Select “New Host”.
- In the HOST NAME field, enter the hostname or IP address of the managed host to add.
- Define any values for variables that affect only that host (overriding any group variables).
- Click **SAVE**.

Roles

- Create custom roles with specific permissions.



Inventory Roles

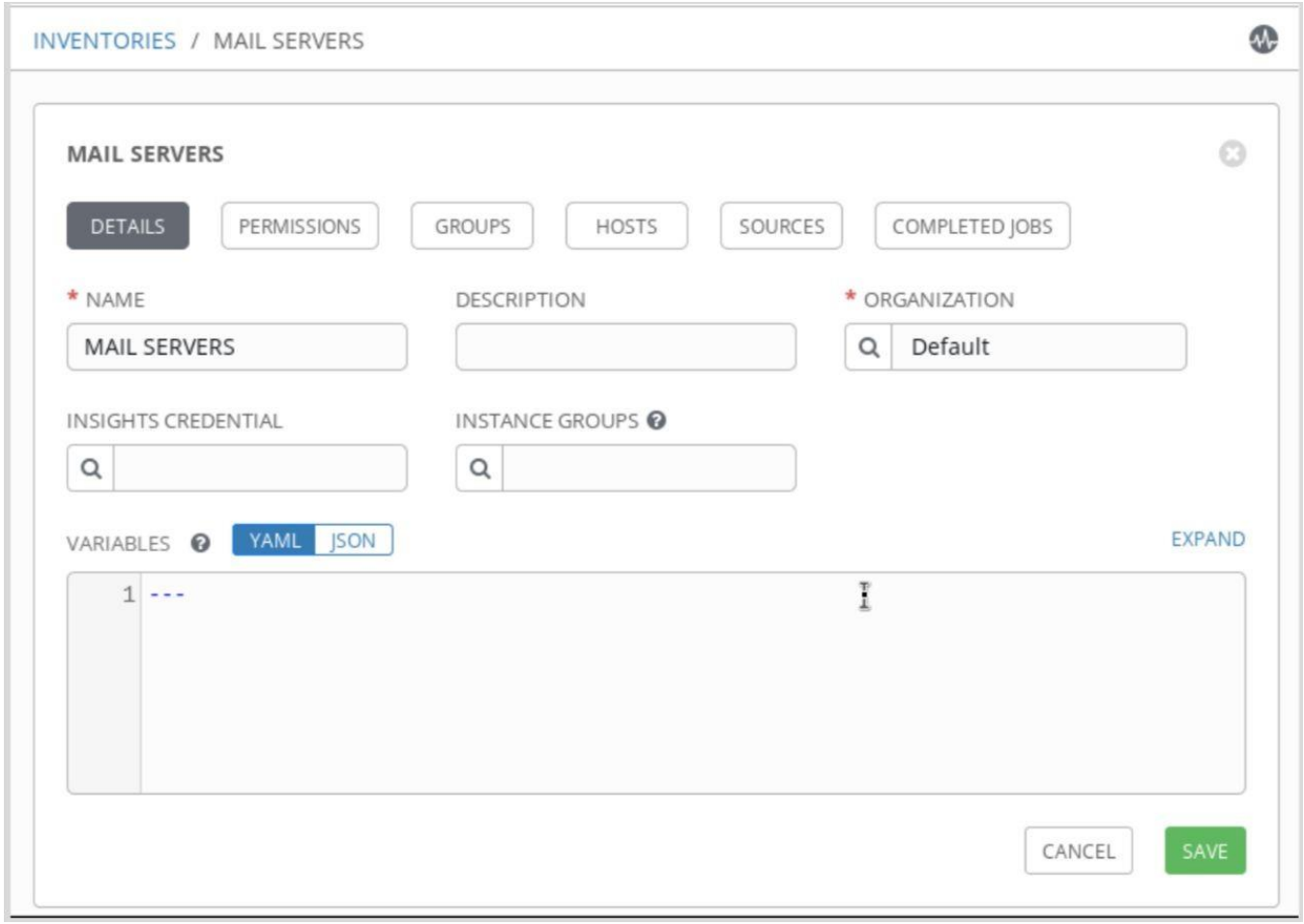
Role	Description
Admin	The inventory **Admin** role grants users full permissions over an inventory. These permissions include deletion and modification of the inventory. In addition, this role also grants permissions associated with the inventory roles **Use**, **Ad Hoc**, and **Update**.
Use	The inventory **Use** role grants users the ability to use an inventory in a job template resource. This controls which inventory is used to launch jobs using the job template's playbook.
Ad Hoc	The inventory **Ad Hoc** role grants users the ability to use the inventory to execute ad hoc commands.
Update	The inventory **Update** role grants users the ability to update a dynamic inventory from its external data source.
Read	The inventory **Read** role grants users the ability to view the contents of an inventory.

Inventory Variables

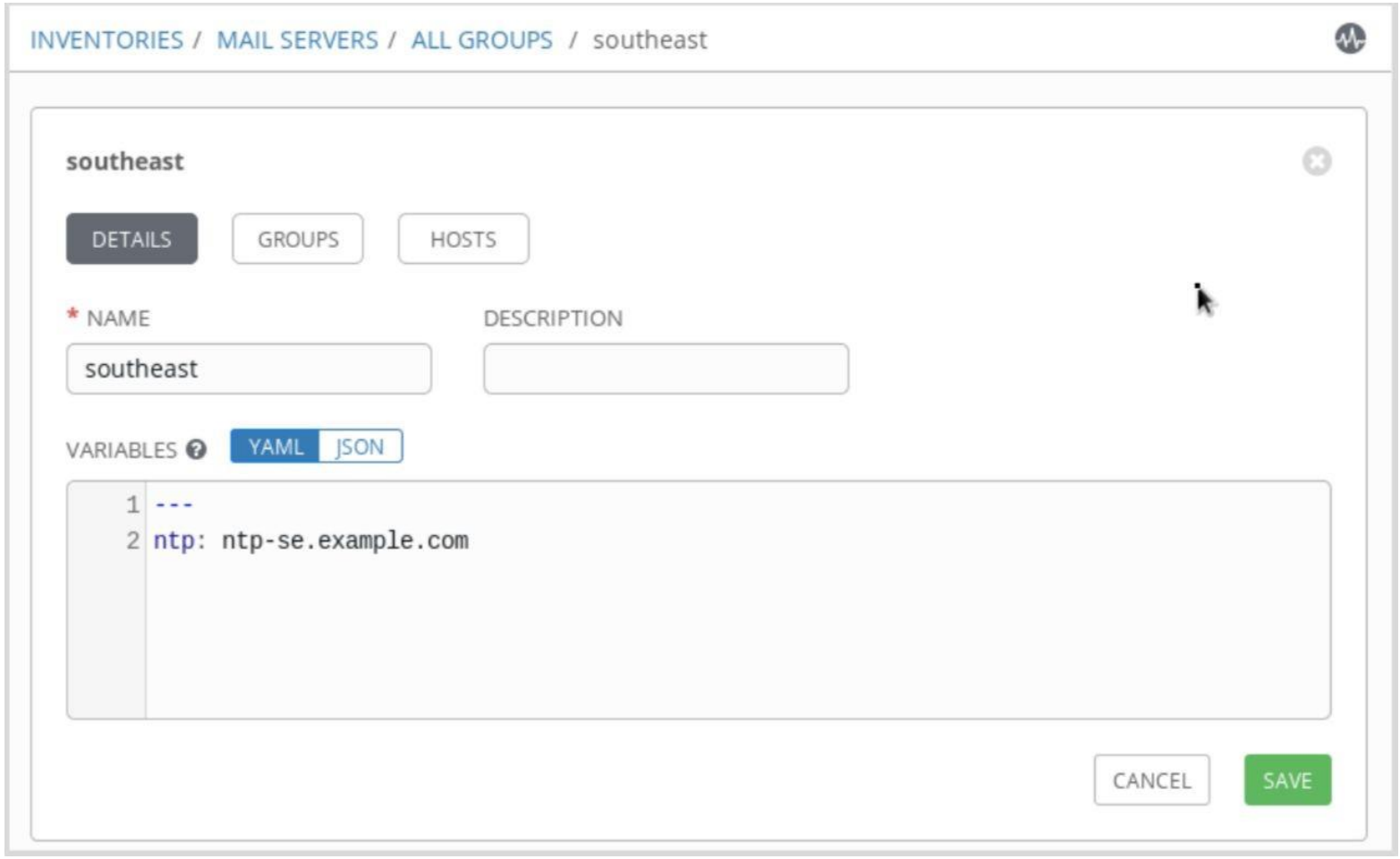
When you manage a static inventory in the Ansible Platform web UI, you may define inventory variables directly in the inventory objects.

- Variables set in the inventory details affect all hosts in the inventory.
- Variables set in a group's details are the equivalent of **`group_vars`**.
- Variables set in a host's details are the equivalent of **`host_vars`**.

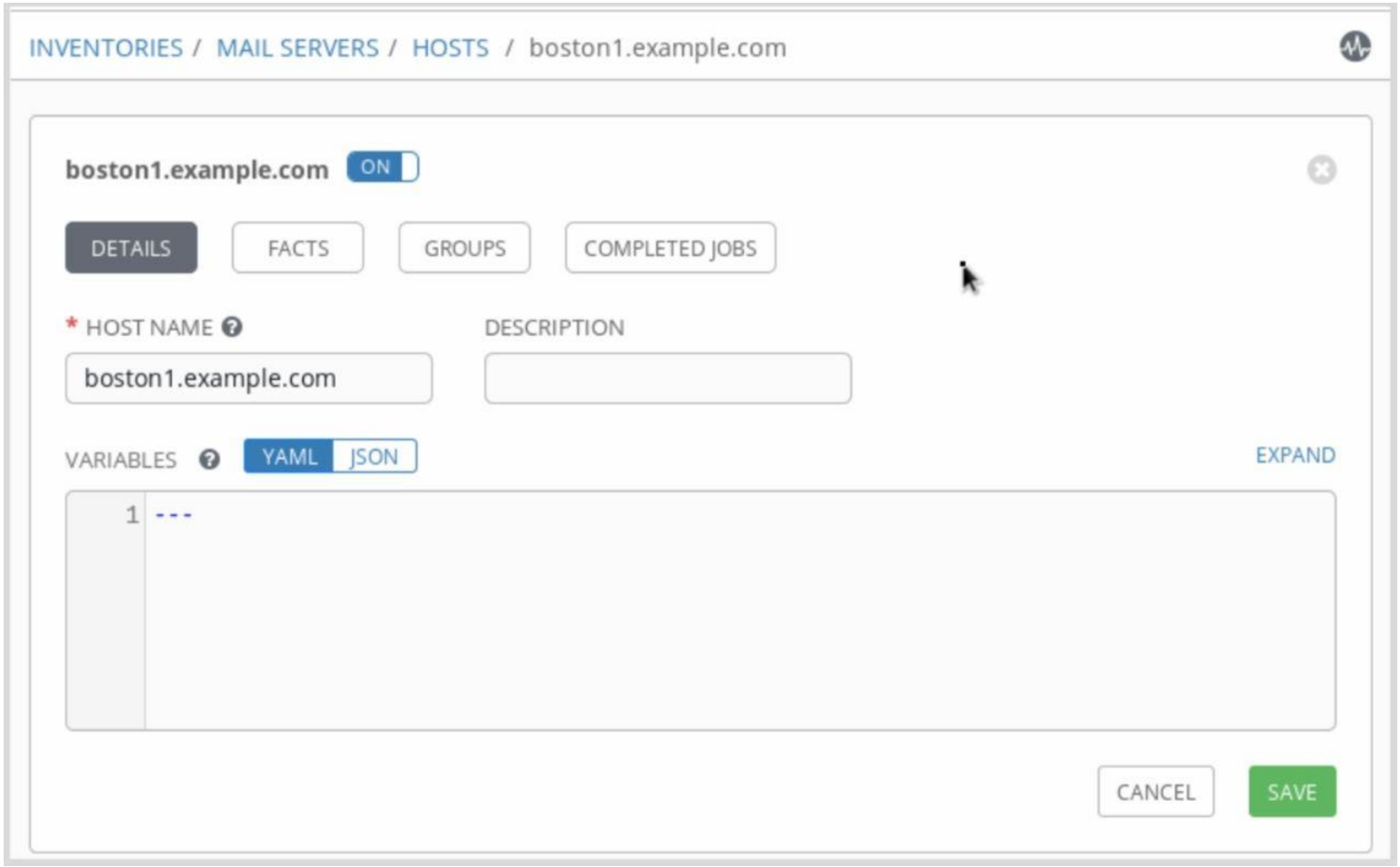
Inventory Variables



Inventory Group Variables



Inventory Host Variables

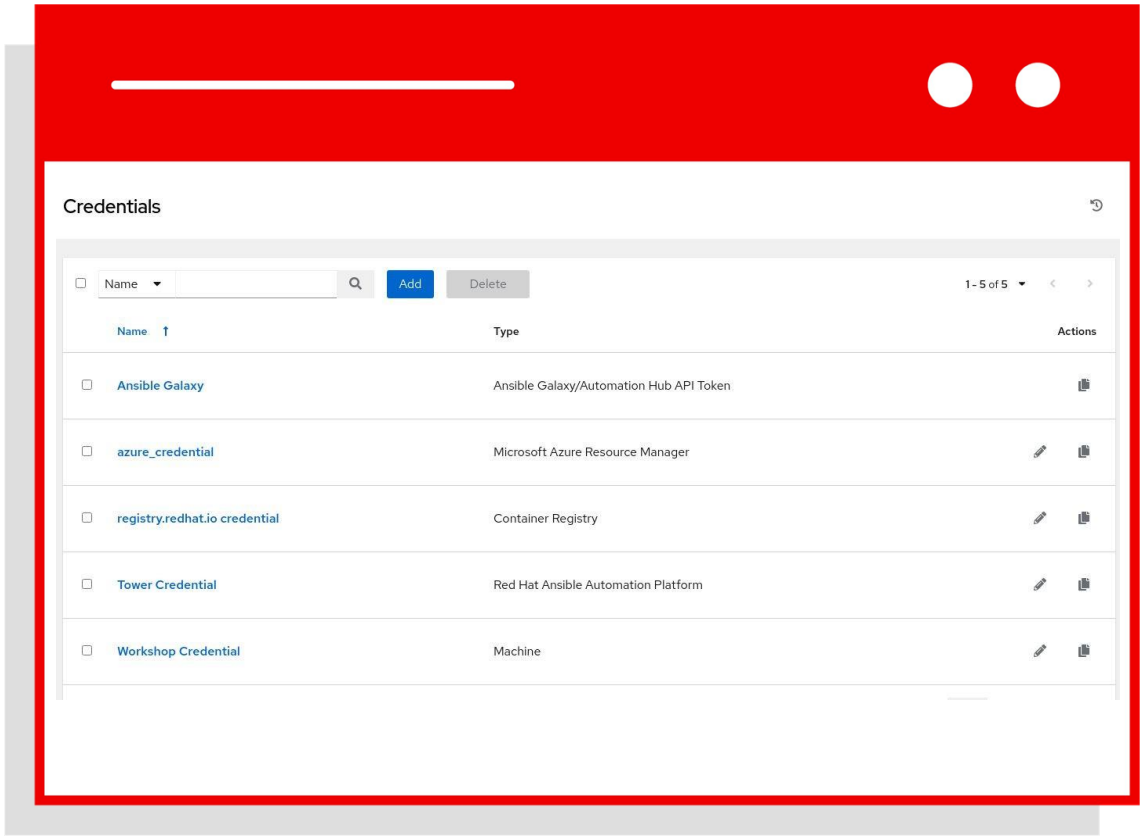


Credentials

Credentials are utilized by Automation Controller for authentication with various external resources:

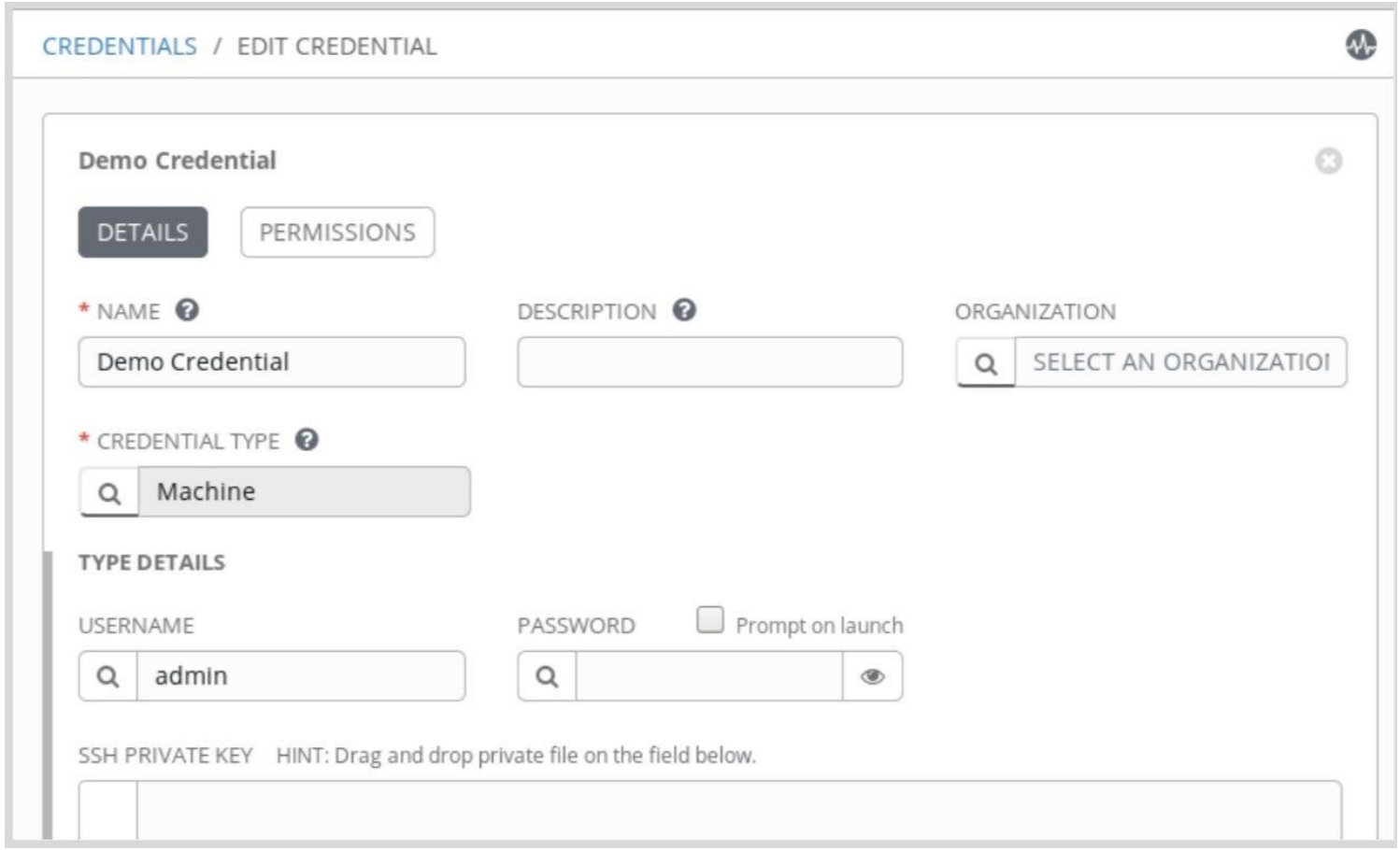
- Connecting to remote machines to run jobs
- Syncing with inventory sources
- Importing project content from version control systems
- Connecting to and managing network devices

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



Credentials

- Create credentials in the UI
- This credential contains information that is used to access managed hosts in the **Inventory**.

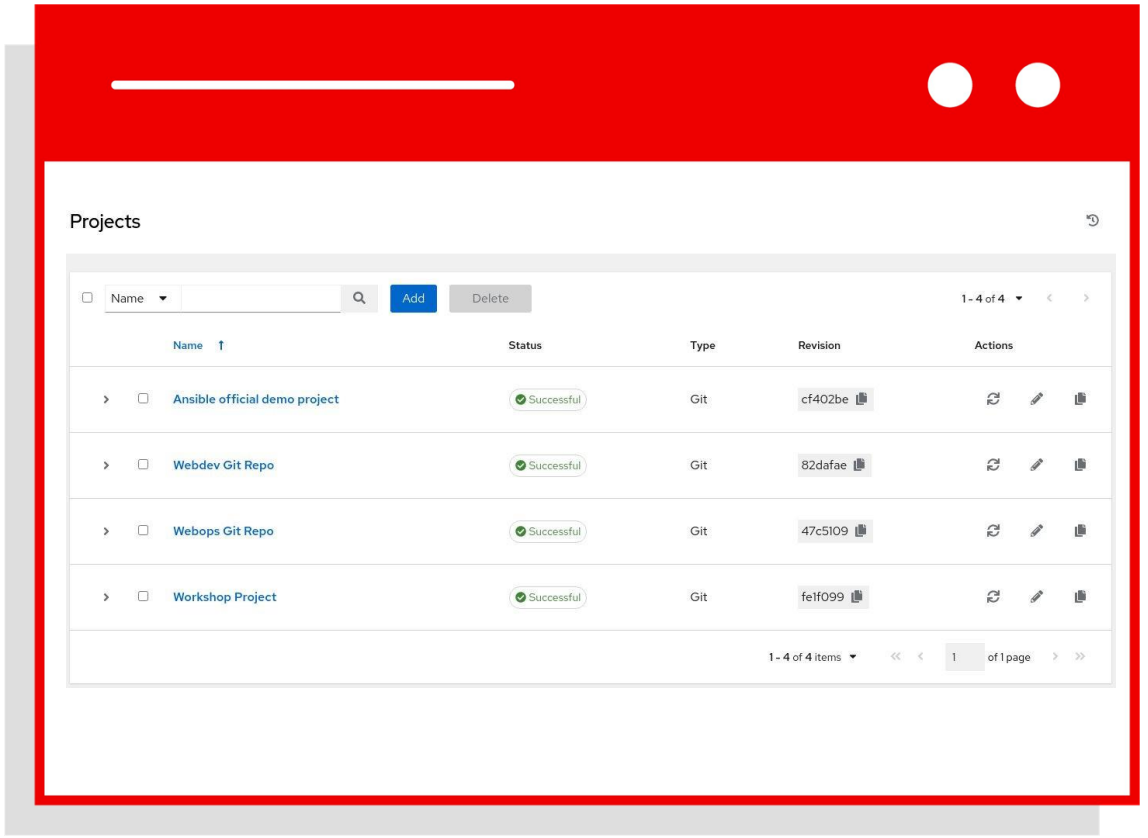


Lab: AAP Inventories, credentials, and ad-hoc commands

PROJECT

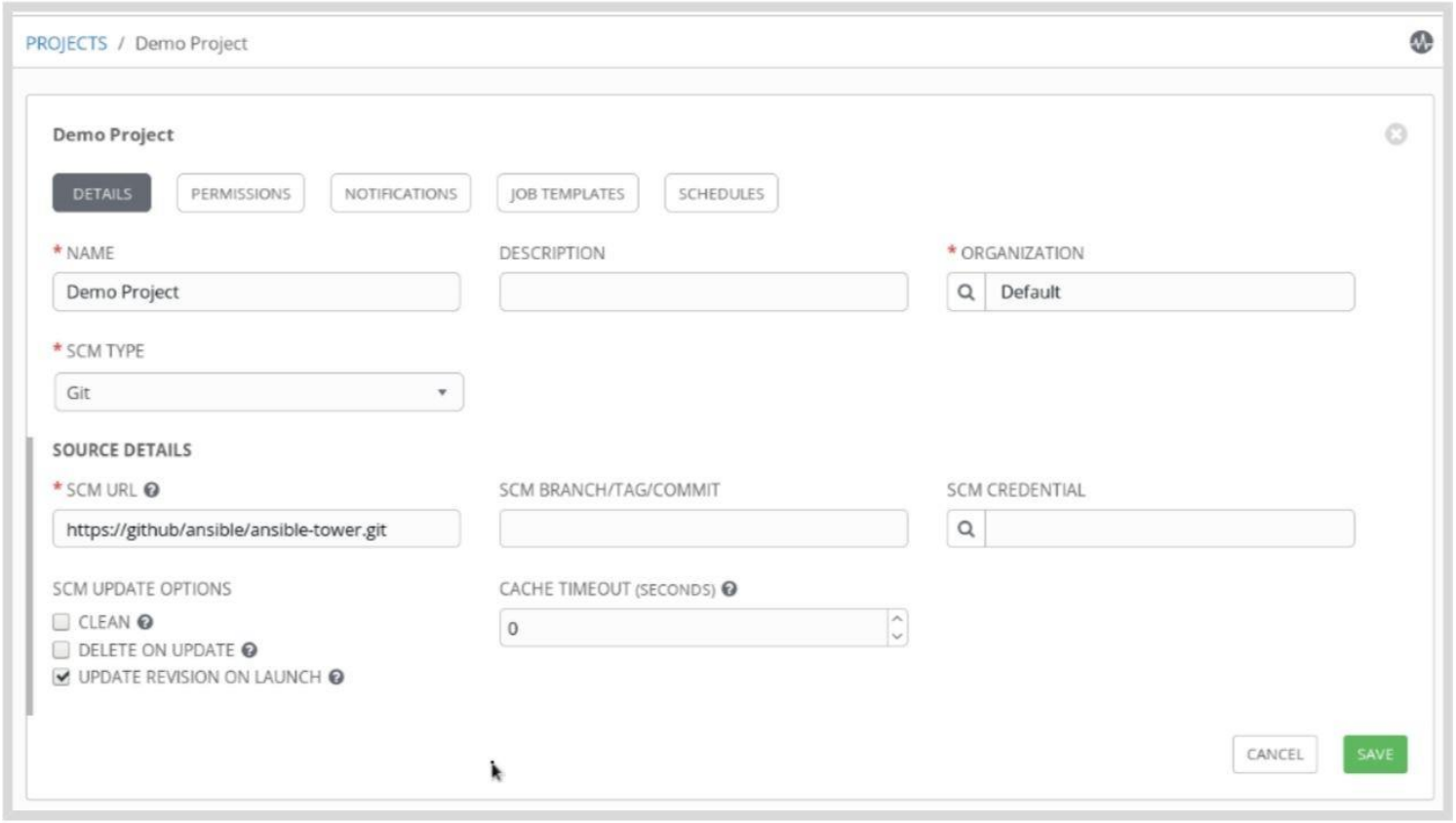
A project is a logical collection of Ansible Playbooks, represented in Ansible Automation Controller.

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



PROJECT

- Under **Projects** in the left navigation bar, an example project named **Demo Project** is displayed.
- This project is configured to get Ansible project materials, including a playbook, from a public Git repository.
- You can also prepare a credential so you can access a private Git repository that needs authentication.



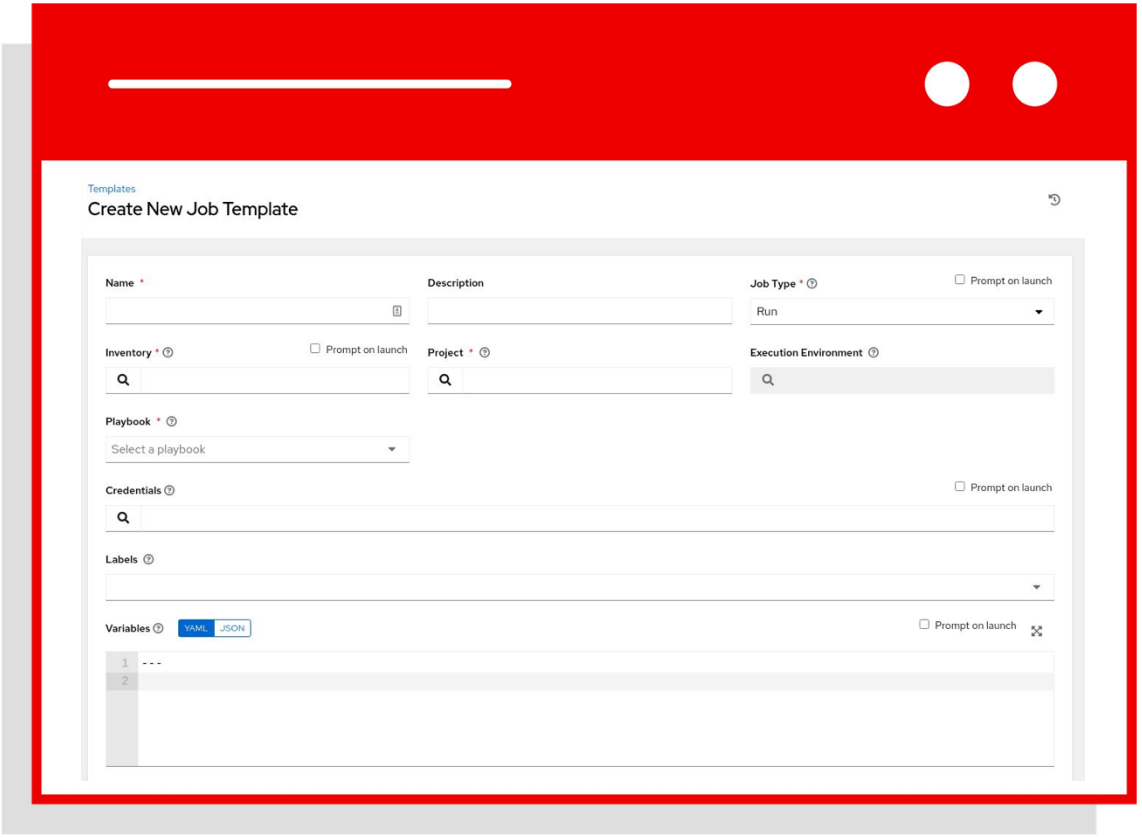
JOB TEMPLATES

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

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

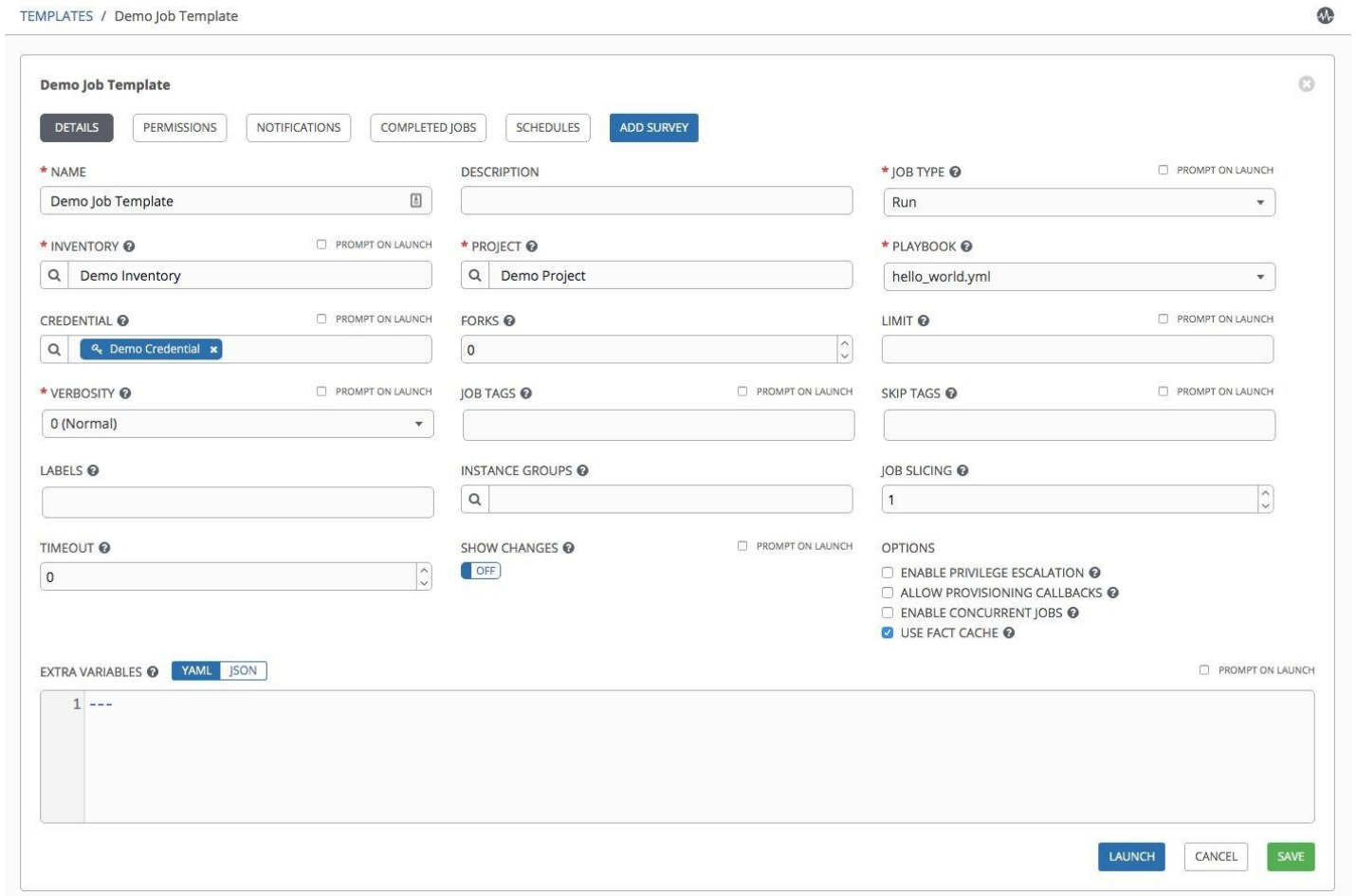
A **Job Template** requires:

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



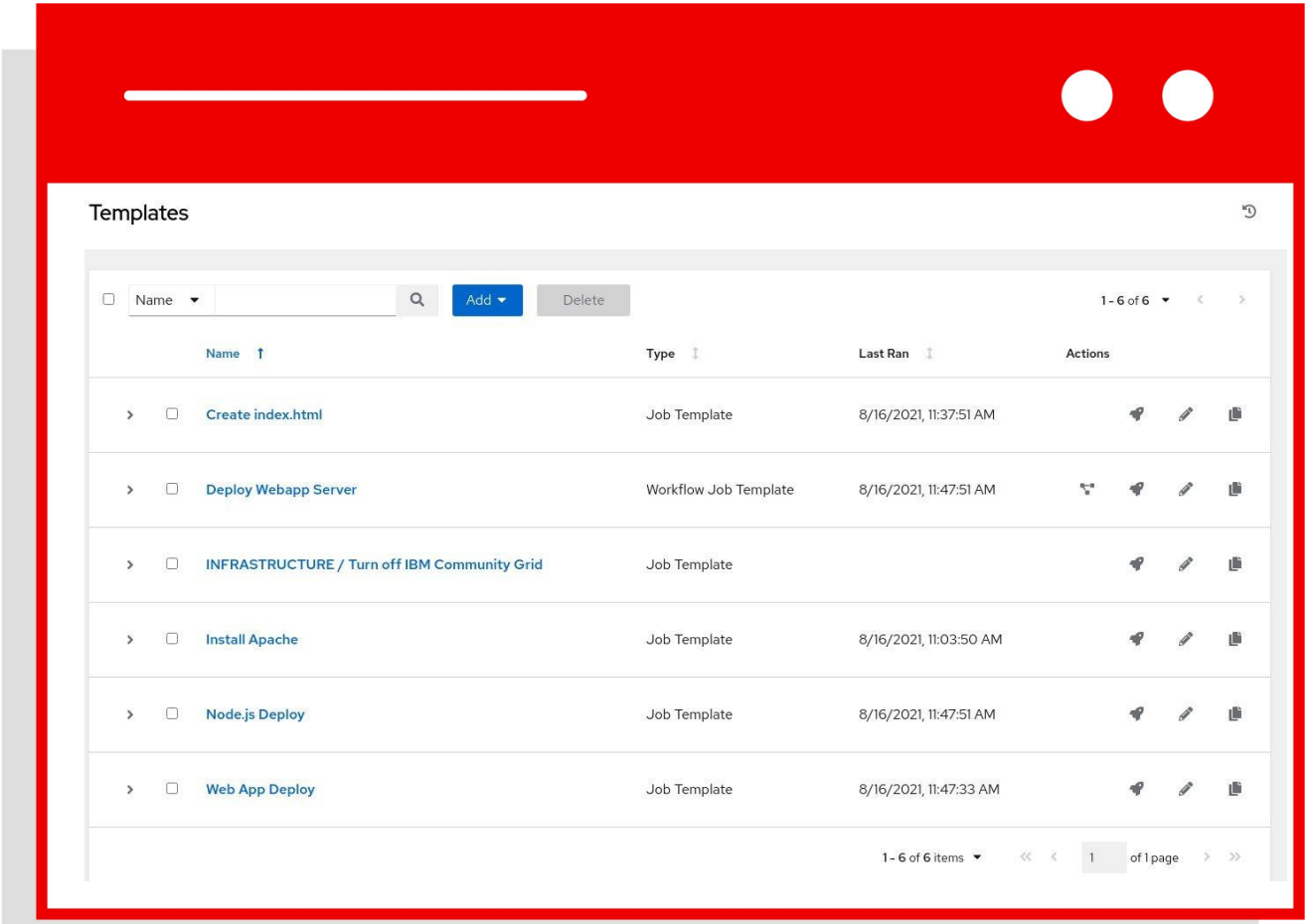
JOB TEMPLATES

- Under **Templates** in the left navigation bar, an example template called **Demo Job Template** is displayed.
- This job template runs the hello_world.yml playbook from **Demo Project** on the hosts in **Demo Inventory**, using **Demo Credential** to authenticate access.
- This initial job template can be used to test Ansible Tower.



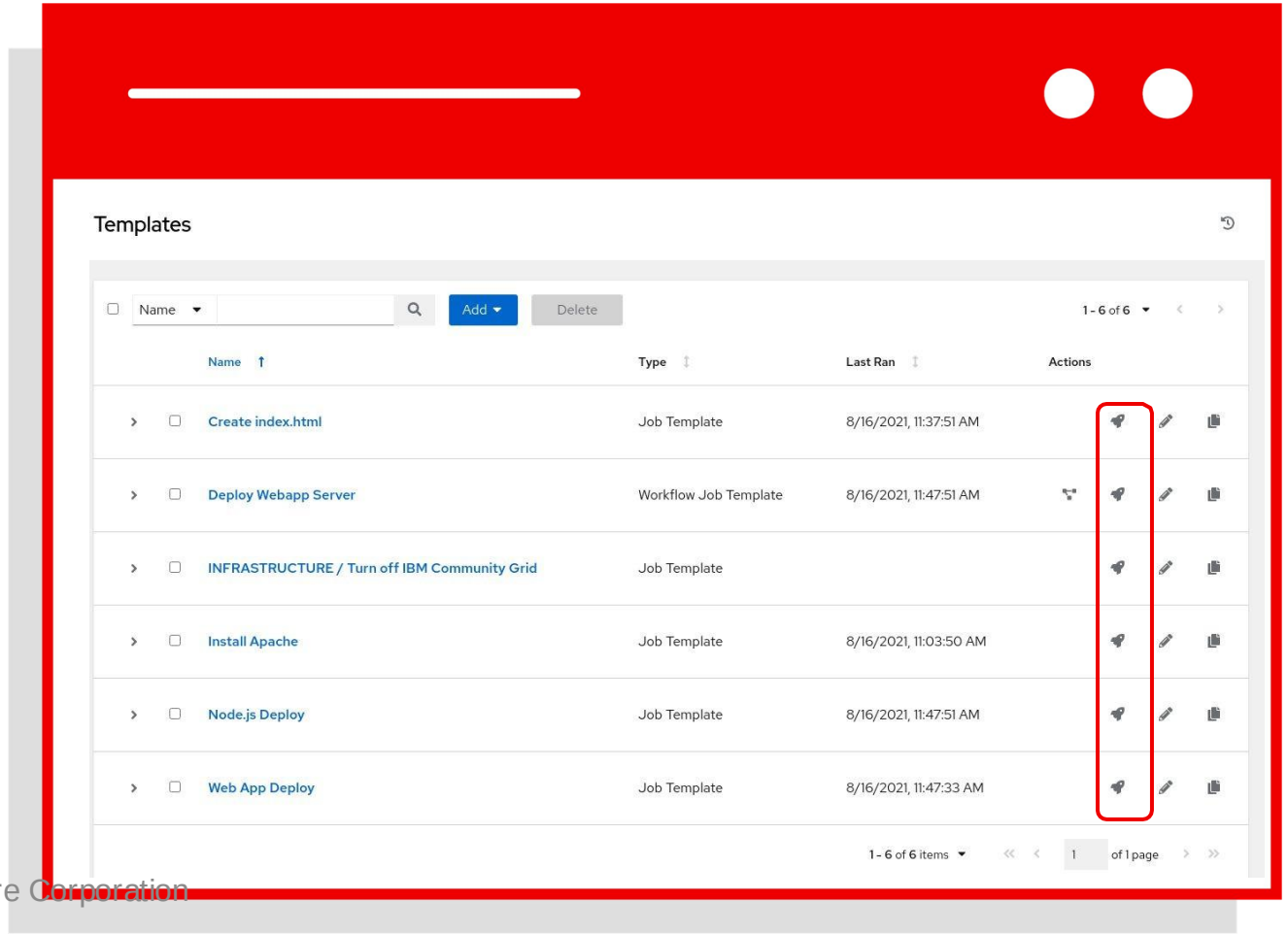
EXPANDING JOB TEMPLATES

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



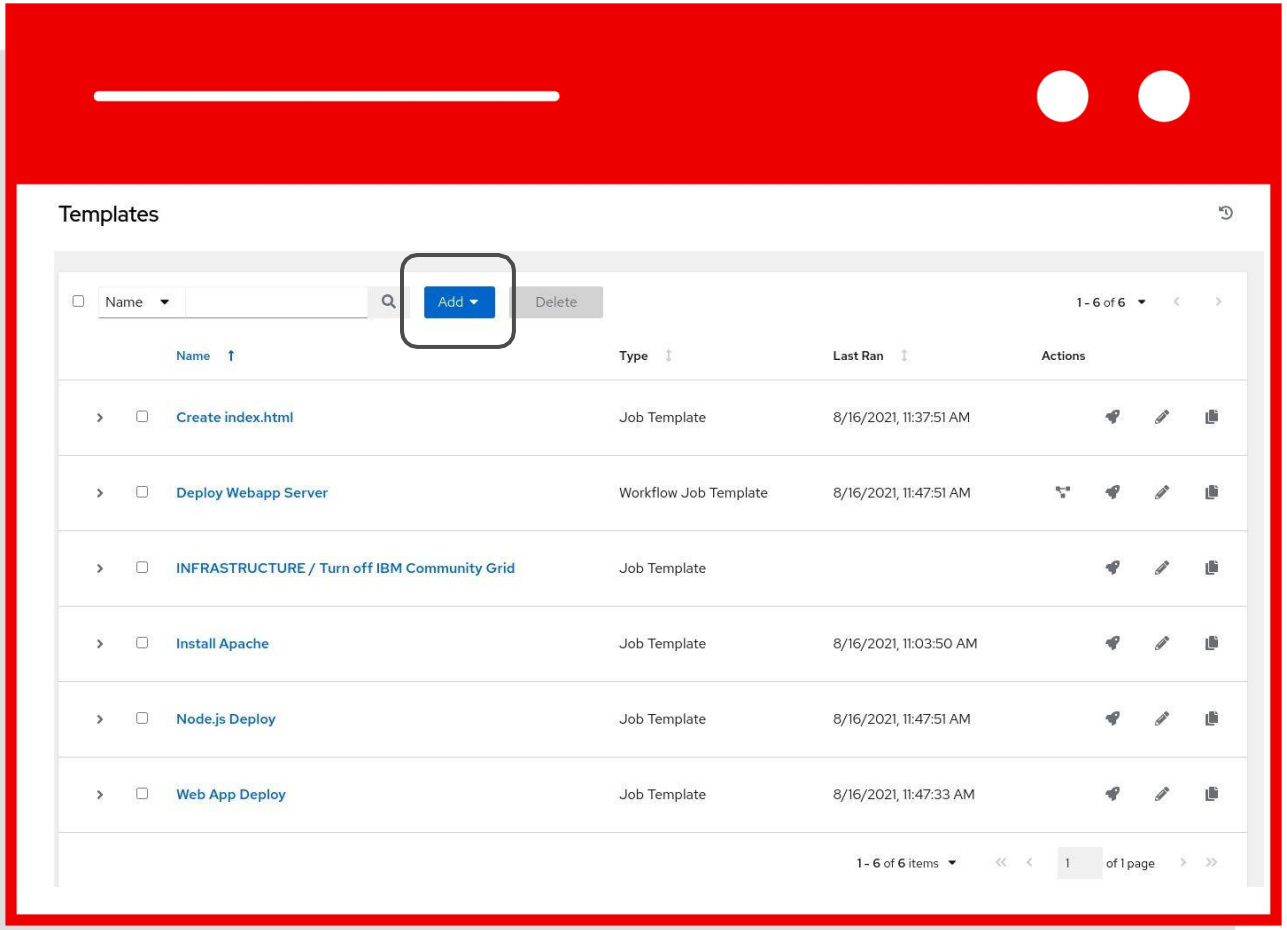
EXECUTING AN EXISTING JOB

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



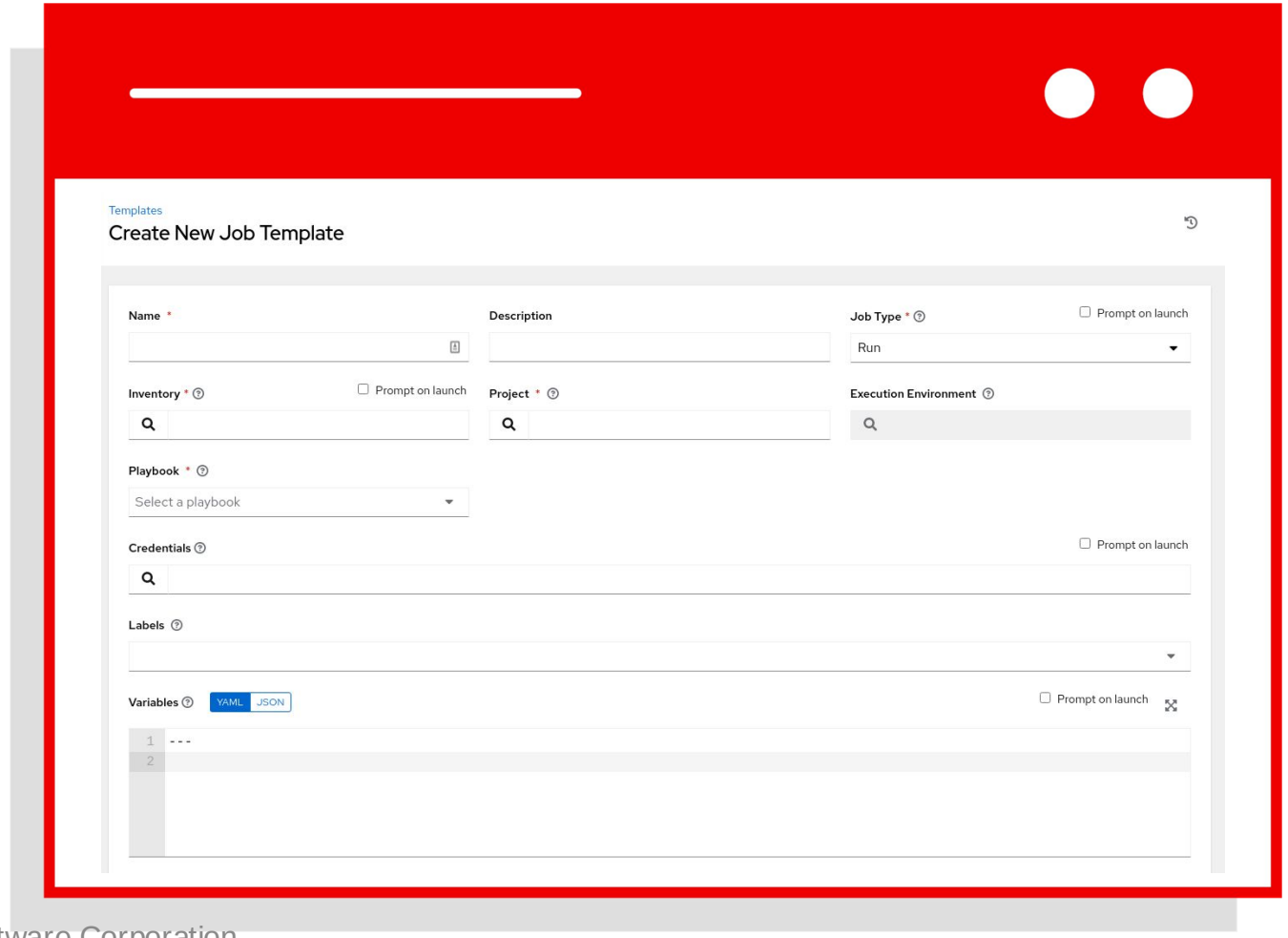
CREATING A NEW JOB TEMPLATE (1/2)

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



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 .

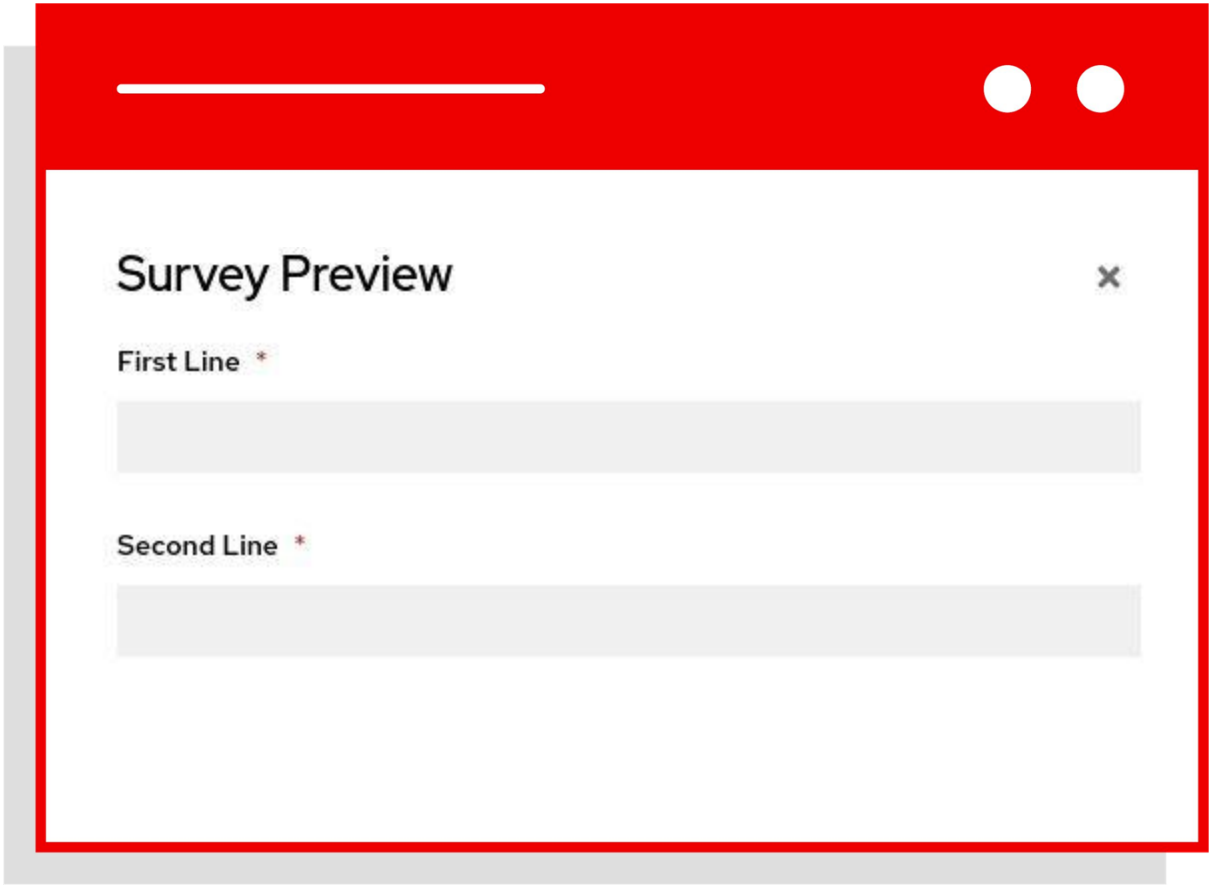


SURVEYS

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

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

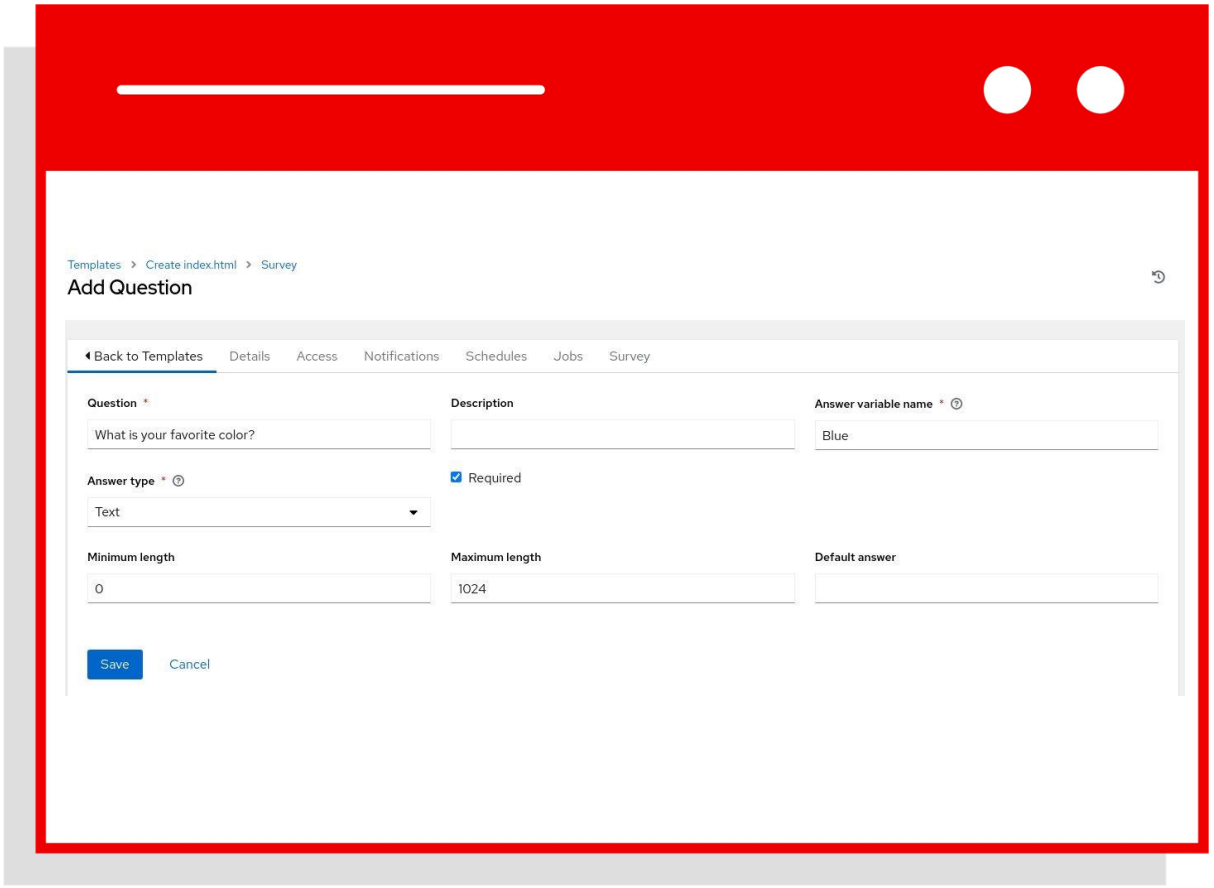
Combine that with Controller's role-based access control, and you can build simple, easy self-service for your users.



CREATING A SURVEY (1/2)

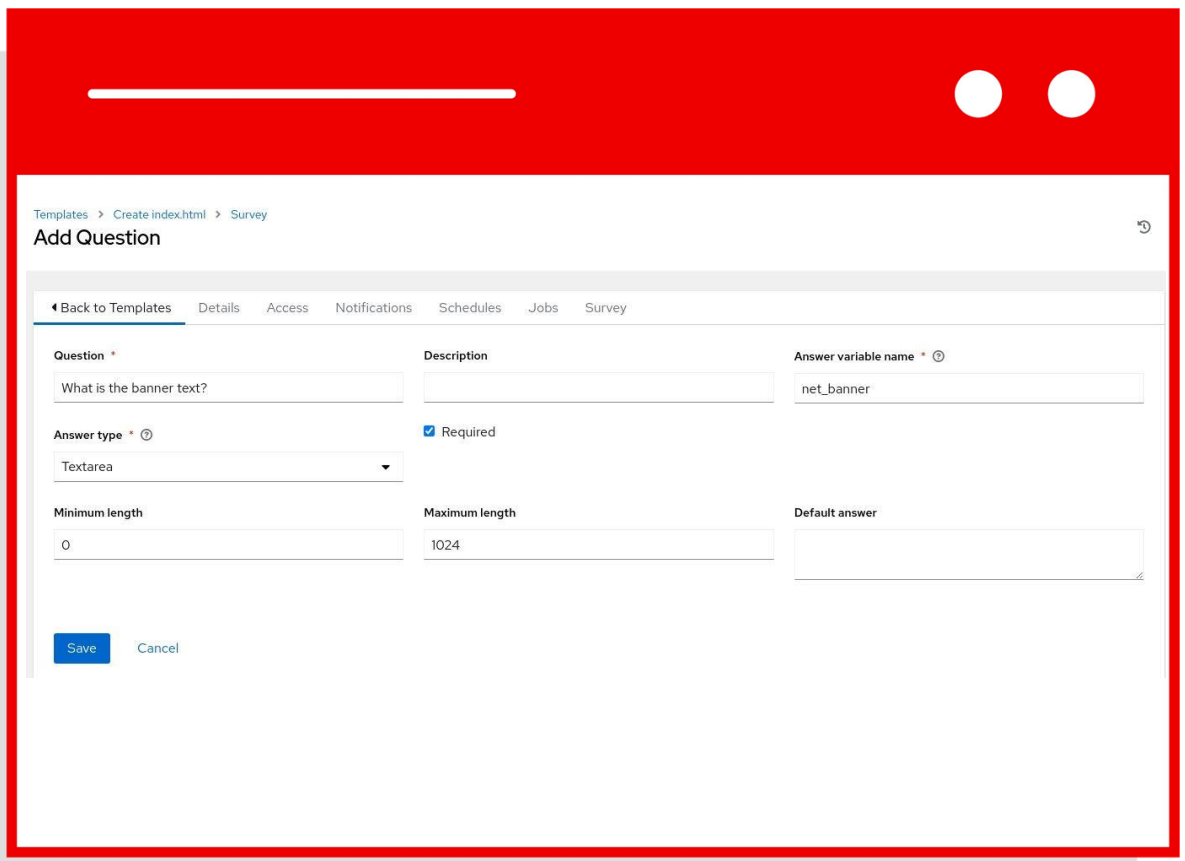
Once a Job Template is saved, the Survey menu will have an **Add Button**

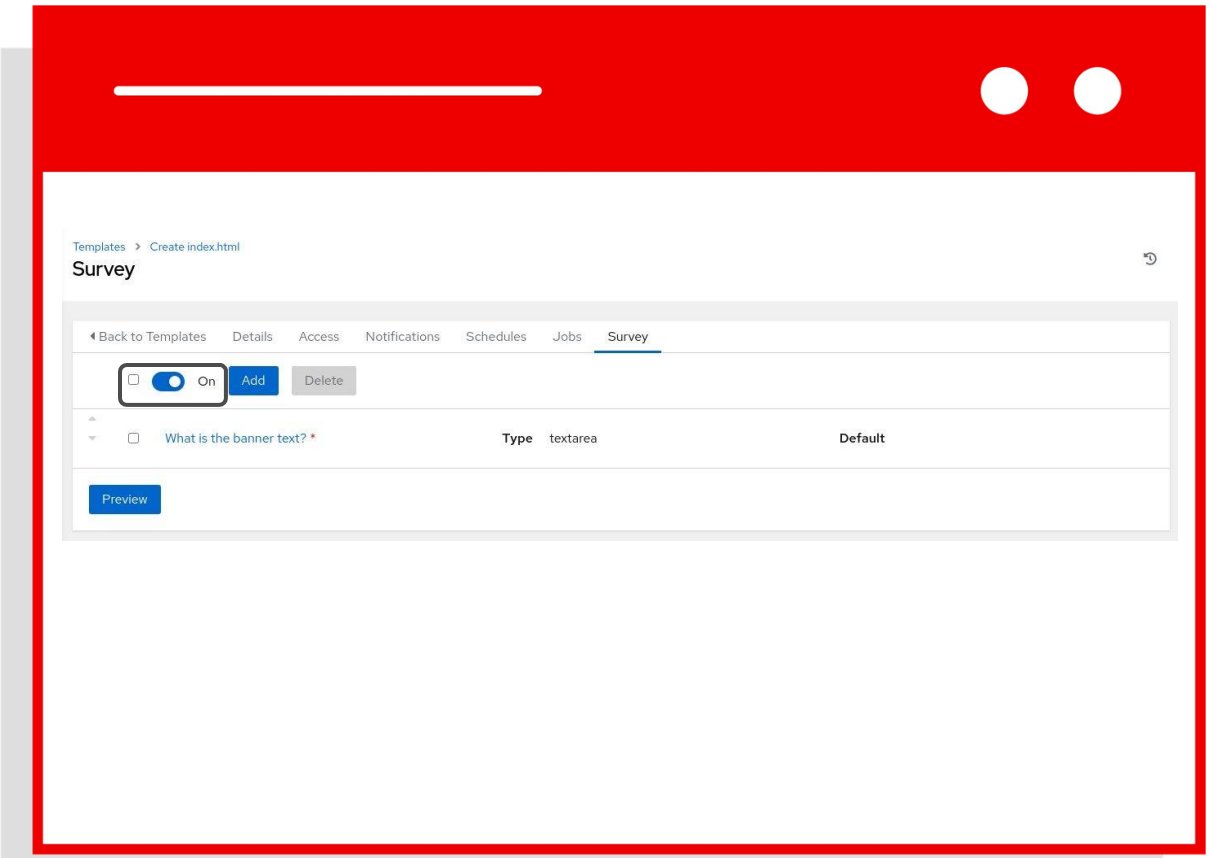
Click the button to open the Add Survey window.



CREATING A SURVEY (2/2)

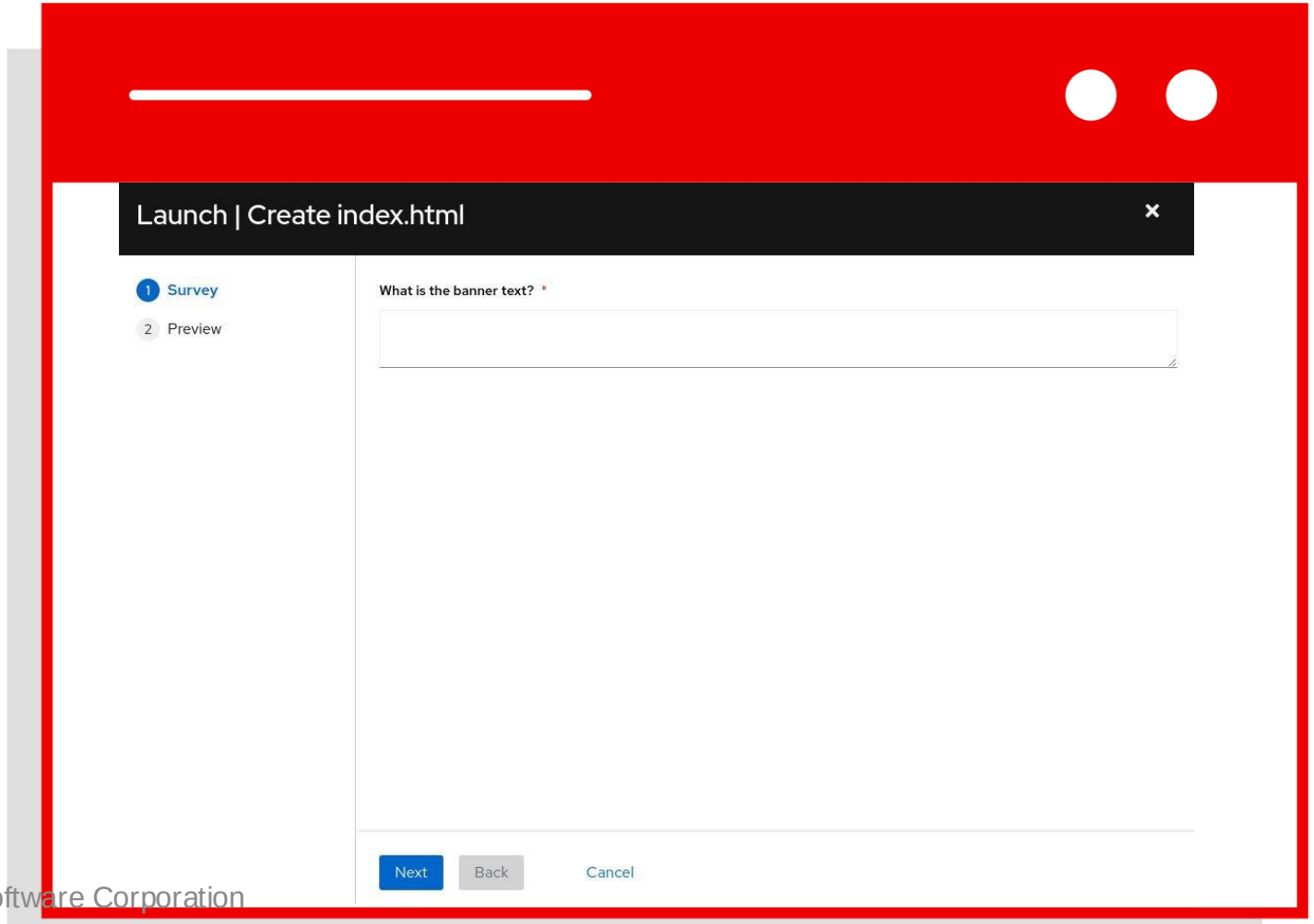
The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.





USING A SURVEY

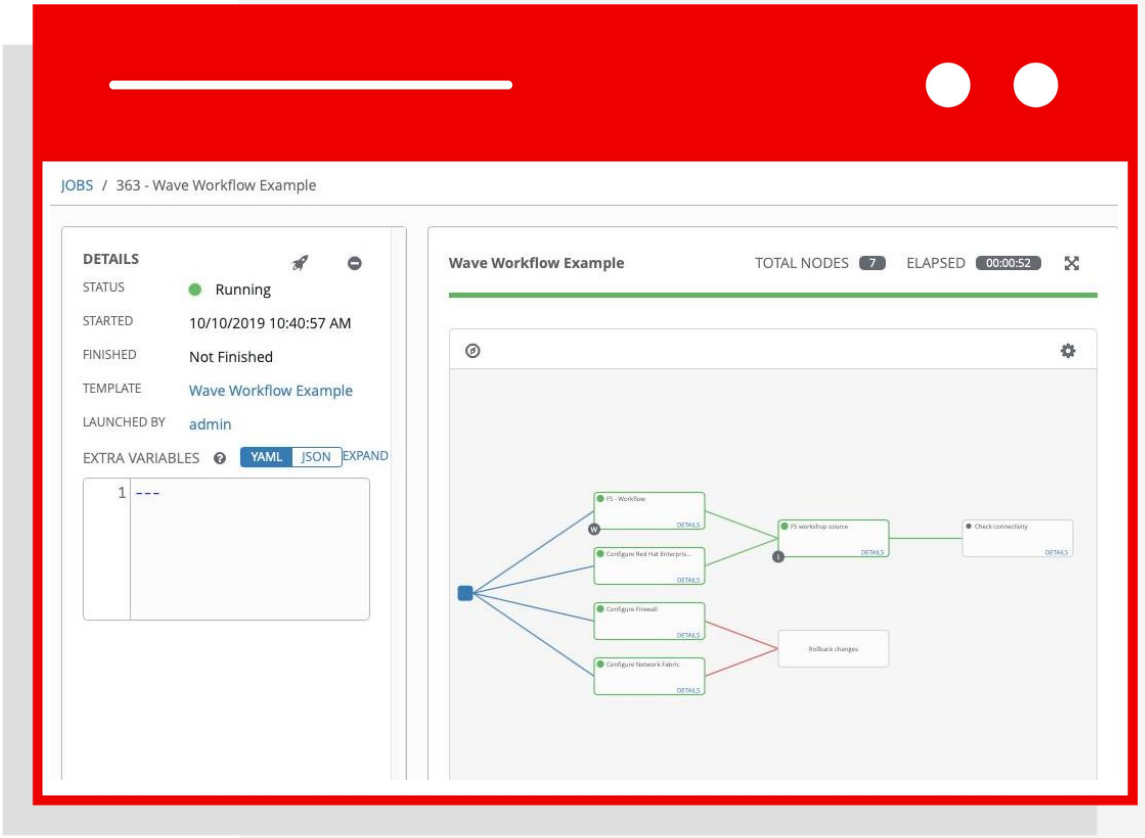
When launching a job, the user will now be prompted with the Survey. The user can be required to fill out the Survey before the Job Template will execute.



WORKFLOWS

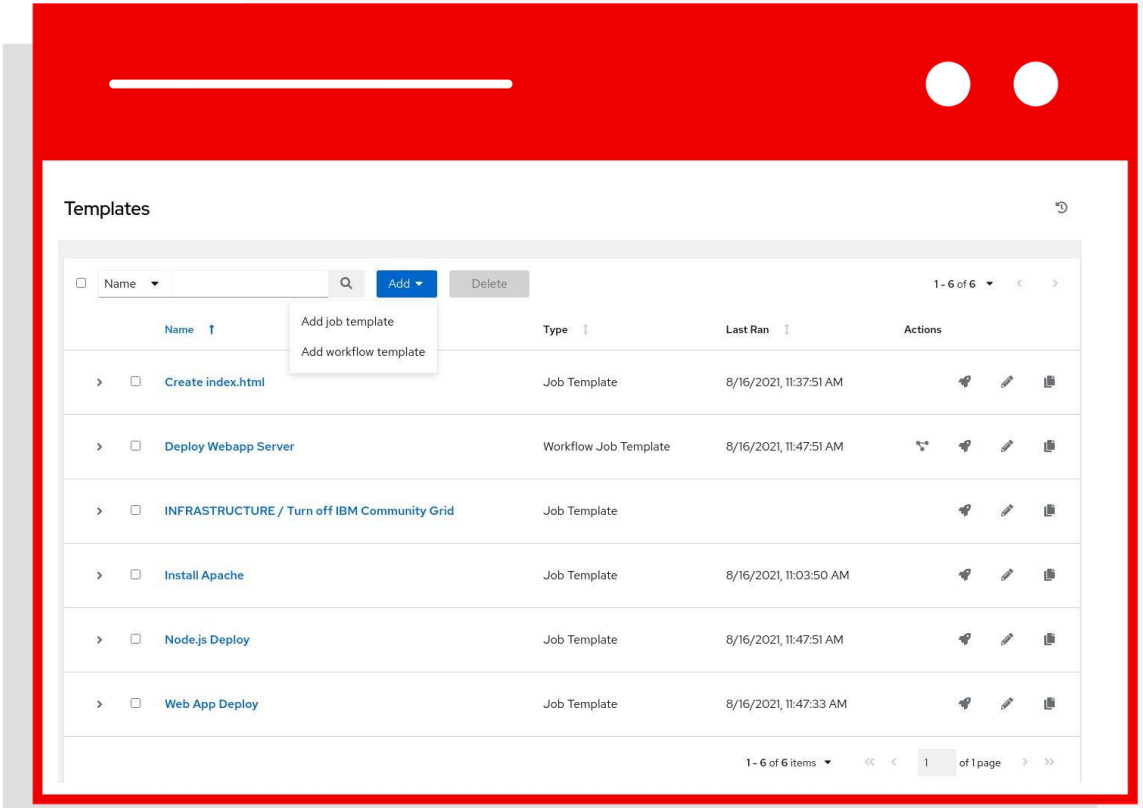
Combine automation to create something bigger

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



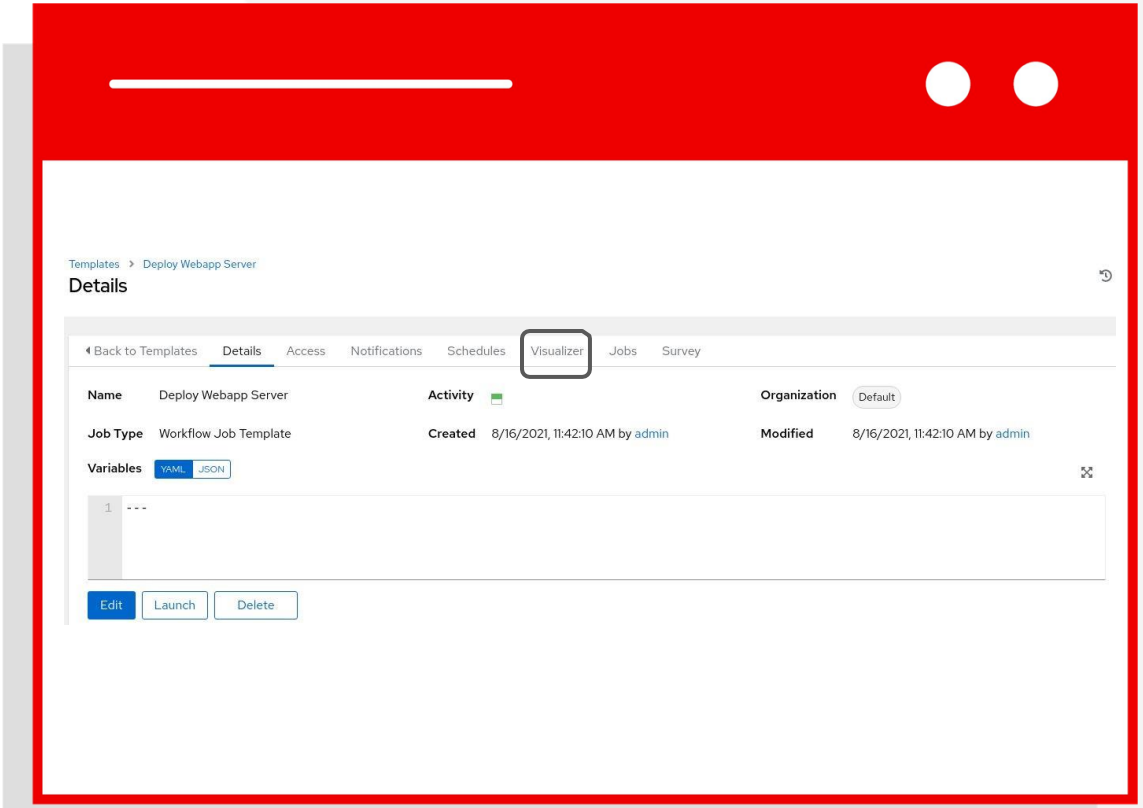
ADDING A NEW TEMPLATE

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



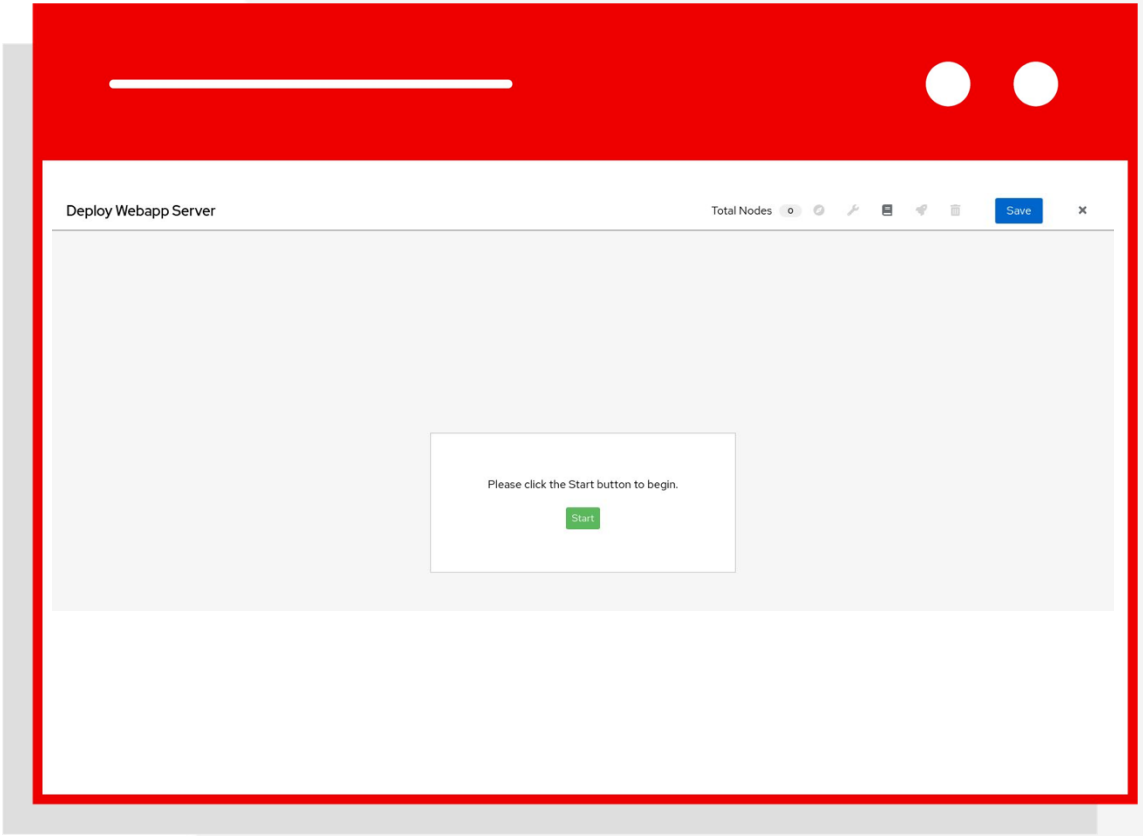
CREATING THE WORKFLOW

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



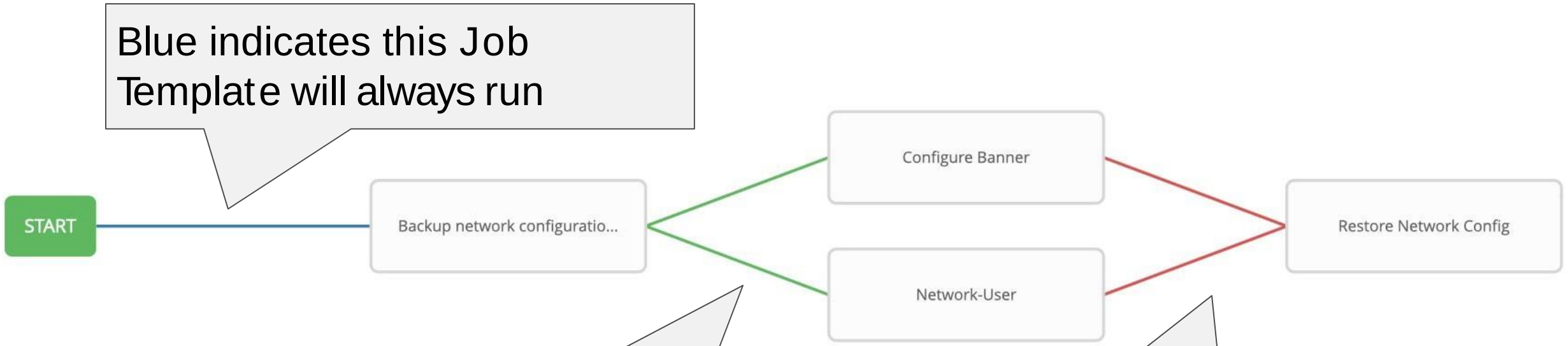
WORKFLOW VISUALIZER

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



VISUALIZING A WORKFLOW

Workflows can branch out, or converge in.



Lab: AAP Projects and job templates