



**Red Hat**

Ansible Automation  
Platform

# Ansible Windows Workshop

Introduction to Ansible Automation for Windows



**Red Hat**

# Housekeeping

- Timing
- Breaks
- Takeaways

# What you will learn

- Introduction to Ansible automation
- How Ansible works for Windows automation
- Understanding Ansible modules and playbooks
- Using Ansible Tower to scale automation to the enterprise
- Reusing automation with Ansible Roles

# Introduction

Topics Covered:

- Why Automate?
- How Ansible Windows Automation works
- Understanding Inventory
- An example Ansible Playbook



**Red Hat**  
Ansible Automation  
Platform



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

# Teams are automating...



Lines Of Business



Network



Security



Operations

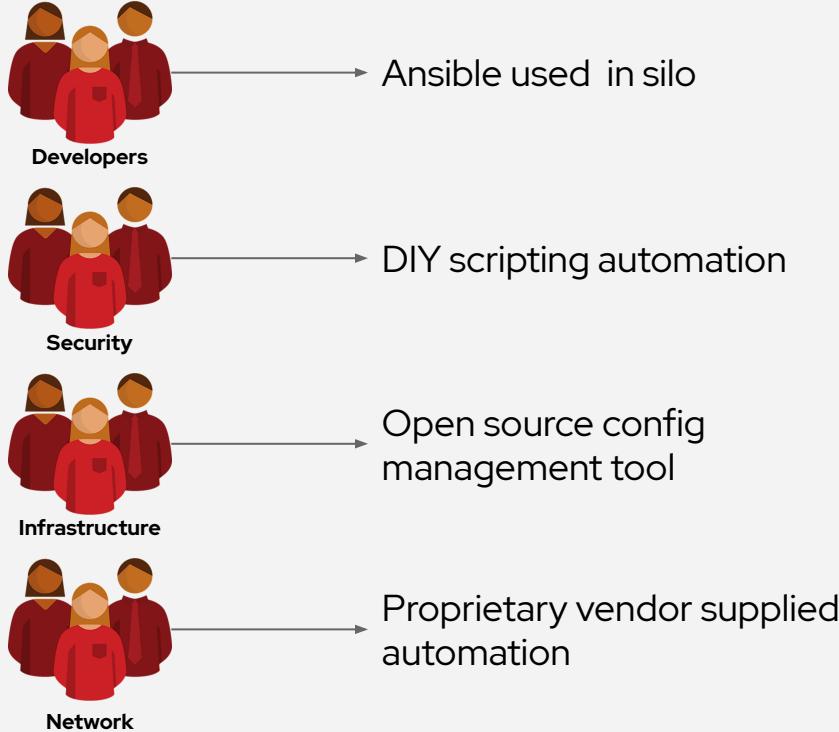


Developers



Infrastructure

# Ad-hoc Automation is happening in silos



Is organic  
automation enough?

# Why Ansible?



## Simple

Human readable automation

No special coding skills needed

Tasks executed in order

Usable by every team

**Get productive quickly**



## Powerful

App deployment

Configuration management

Workflow orchestration

Network automation

**Orchestrate the app lifecycle**



## Agentless

Agentless architecture

Uses OpenSSH & WinRM

No agents to exploit or update

Get started immediately

**More efficient & more secure**

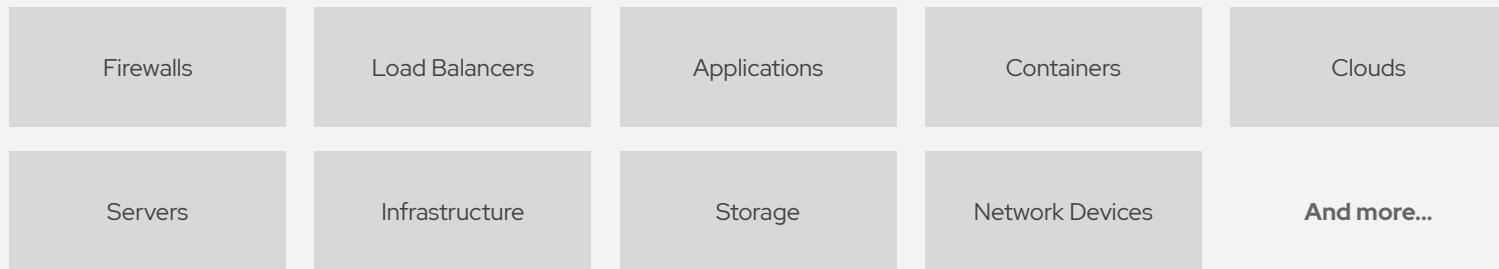
# What can I do using Ansible?

Automate the deployment and management of your entire IT footprint.

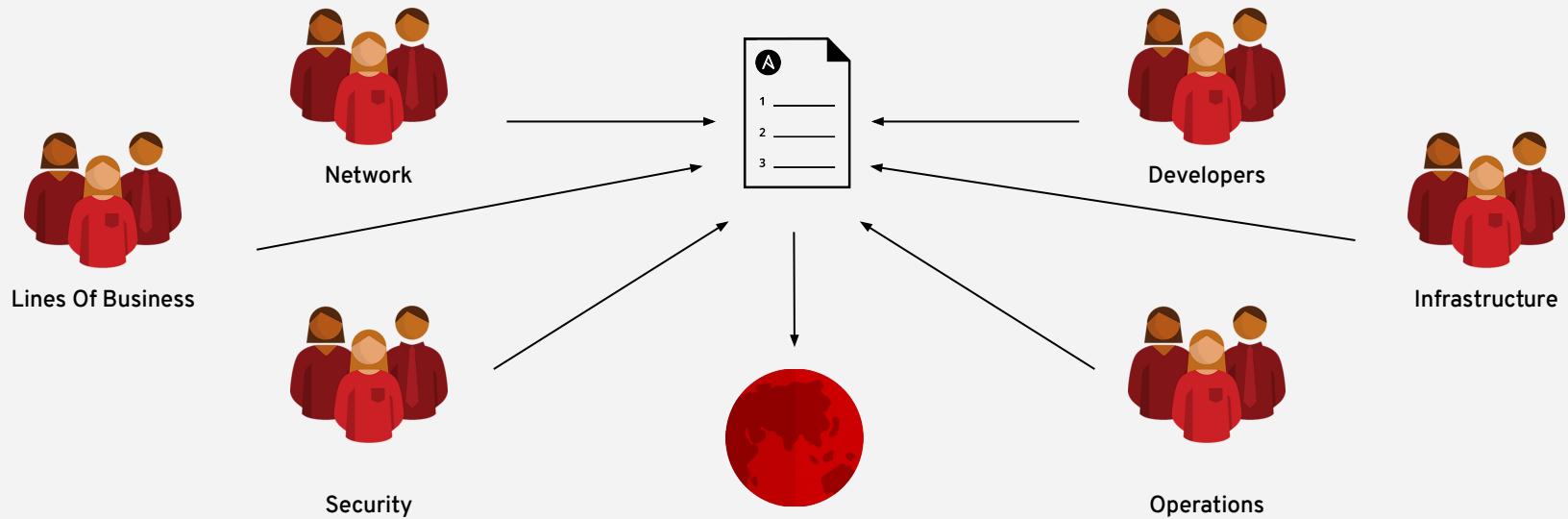
**Do this...**



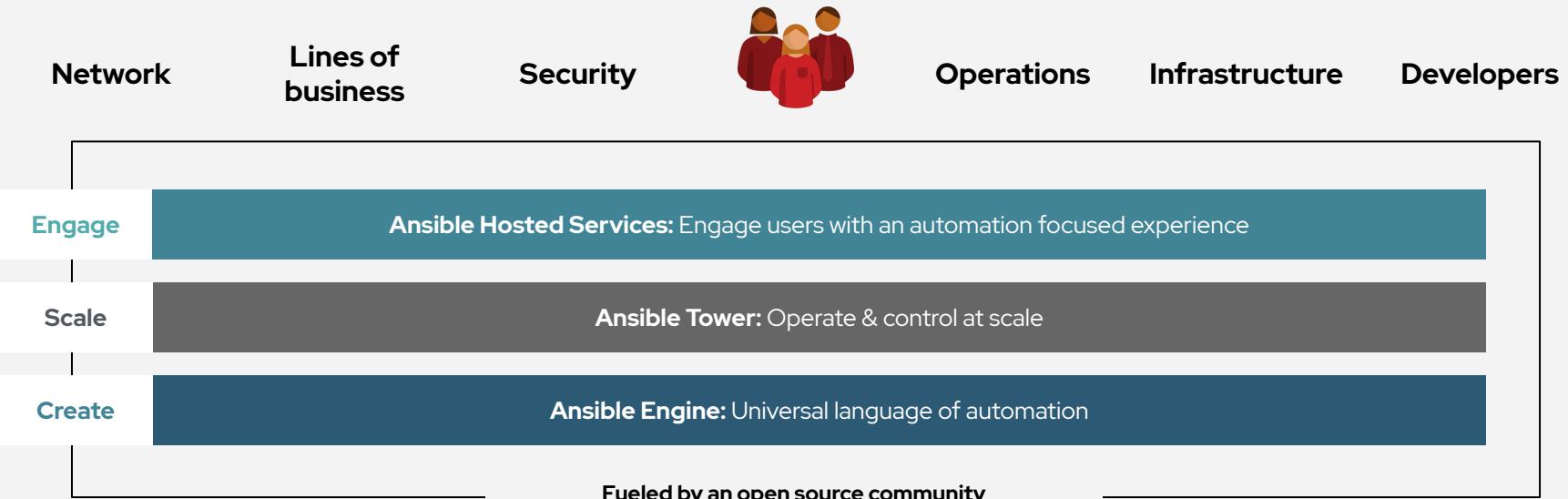
**On these...**



# When automation crosses teams, you need an automation platform



# Red Hat Ansible Automation Platform



# Red Hat Ansible Tower

## by the numbers:

**94%**

Reduction in recovery time following  
a security incident

**84%**

Savings by deploying workloads  
to generic systems appliances  
using Ansible Tower

**67%**

Reduction in man hours required  
for customer deliveries

Financial summary:

**146%**

**ROI on Ansible Tower**

<3 MONTHS

**Payback on Ansible Tower**

# WINDOWS AUTOMATION

**90+**

Windows  
Modules

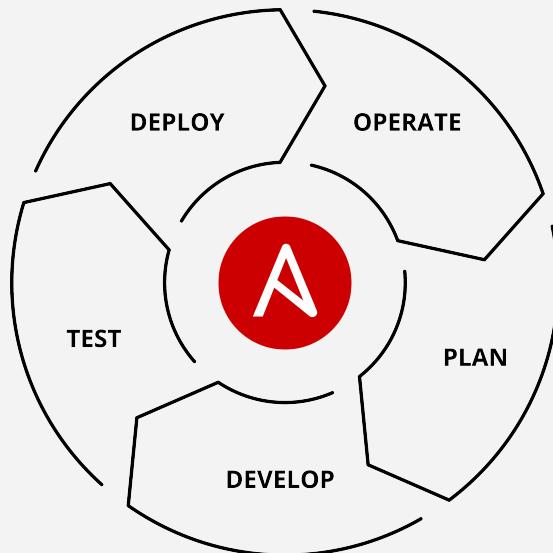
**1,300+**

Powershell DSC  
resources

[ansible.com/windows](http://ansible.com/windows)

# WHAT CAN I DO USING ANSIBLE FOR WINDOWS

Native Windows support uses PowerShell remoting to manage Windows in the same Ansible agentless way

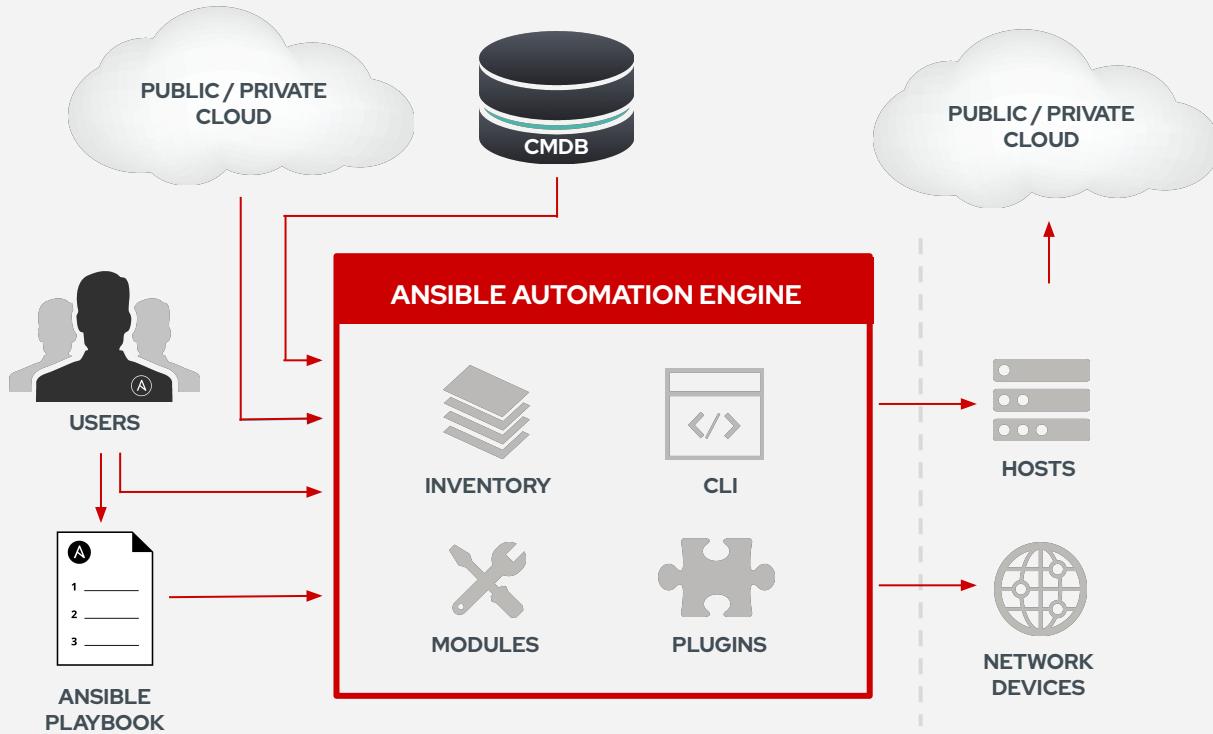


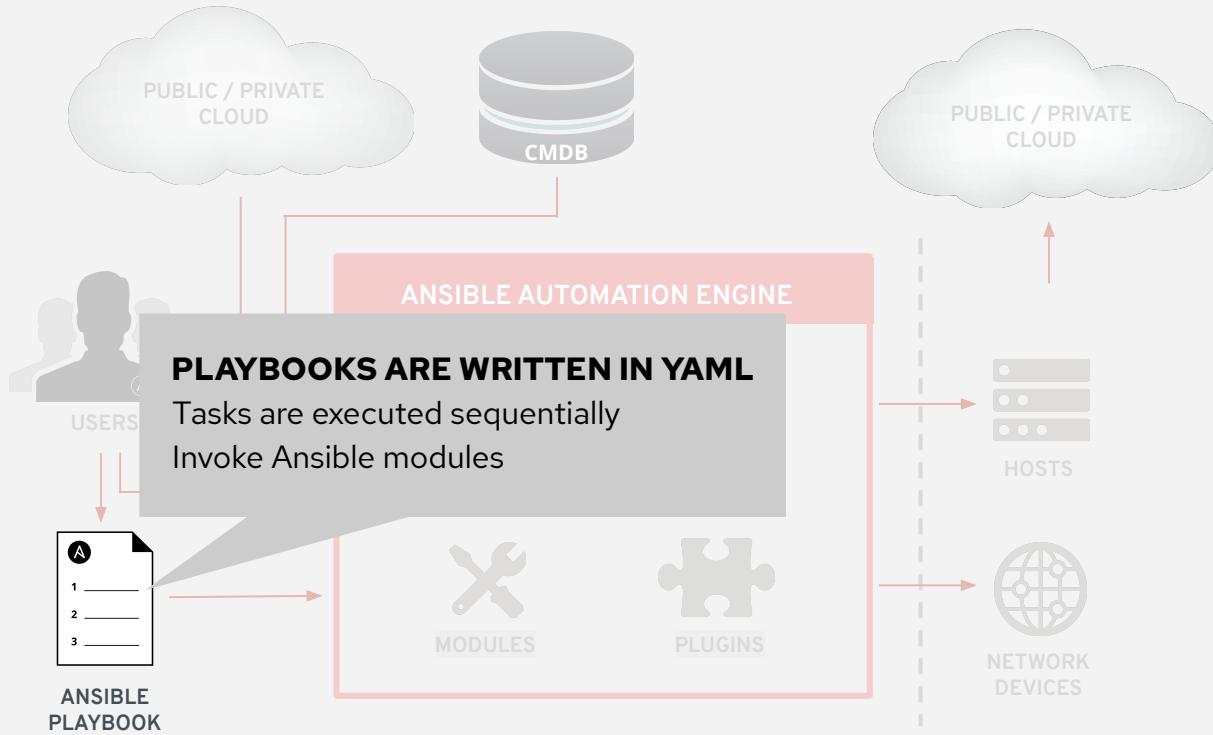
- Install and uninstall MSIs
- Gather facts on Windows hosts
- Enable and disable Windows features
- Start, stop, and manage Windows Services
- Create and Manage local users and groups
- Manage Windows packages via [Chocolatey package manager](#)
- Manage and install Windows updates
- Fetch files from remote sites
- Push and execute any Powershell scripts

# Ansible automates technologies you use

Time to automate is measured in minutes

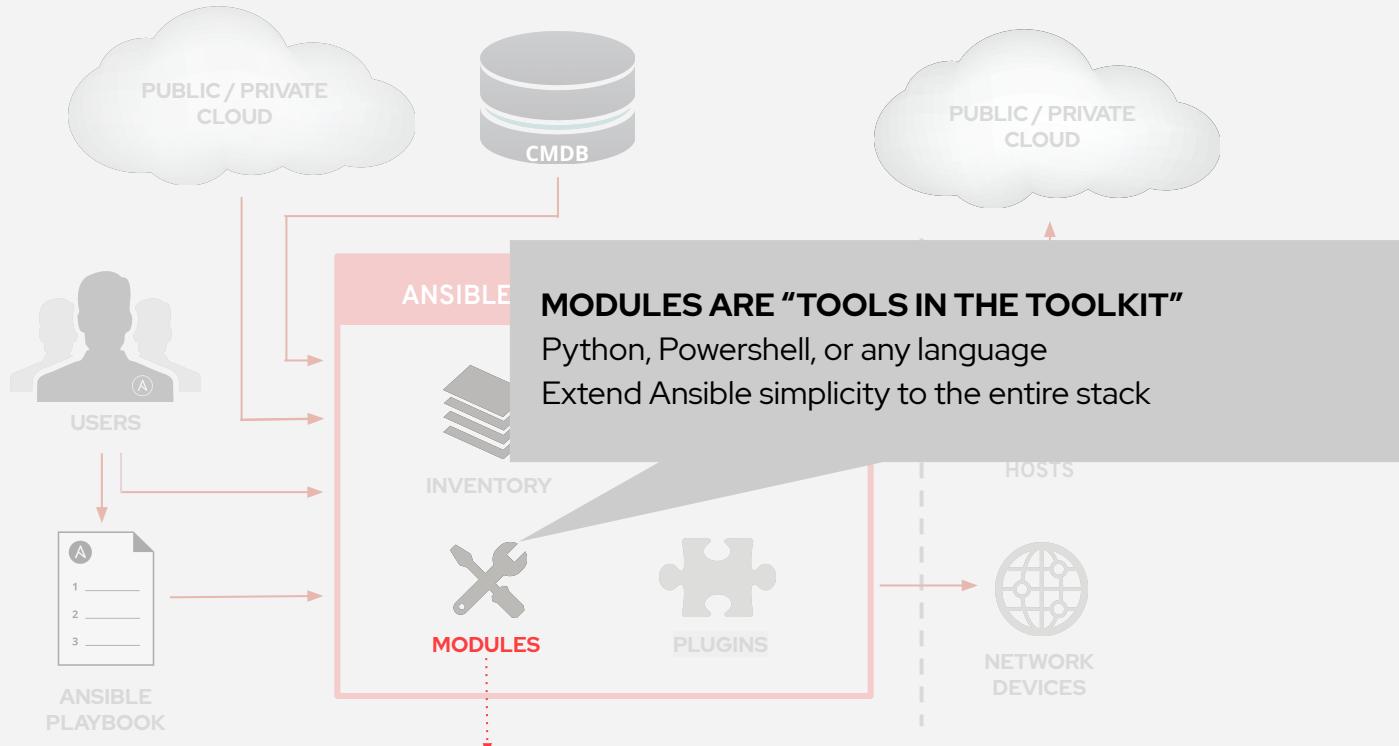
Cloud	Virt & Container	Windows	Network	Security	Monitoring
AWS	Docker	ACLs	A10	Checkpoint	Dynatrace
Azure	VMware	Files	Arista	Cisco	Datadog
Digital Ocean	RHV	Packages	Aruba	CyberArk	LogicMonitor
Google	OpenStack	IIS	Cumulus	F5	New Relic
OpenStack	OpenShift	Regedits	Bigswitch	Fortinet	Sensu
Rackspace	<b>+more</b>	Shares	Cisco	Juniper	<b>+more</b>
<b>+more</b>		Services	Dell	IBM	
Operating Systems	Storage	Configs	Extreme	Palo Alto	Devops
RHEL	Netapp	Users	F5	Snort	Jira
Linux	Red Hat Storage	Domains	Lenovo	<b>+more</b>	GitHub
Windows	Infinidat	<b>+more</b>	MikroTik		Vagrant
<b>+more</b>	<b>+more</b>		Juniper		Jenkins
			OpenSwitch		Slack
			<b>+more</b>		<b>+more</b>





```
---
```

- **name: start IIS/stop firewall**  
  **hosts:** windows-web  
  **become:** yes  
  **tasks:**
  - **name: IIS is running**  
    **win\_service:**
    - name:** W3Svc
    - state:** running
  - **name: firewall service is stopped/disabled**  
    **win\_service:**
    - name:** MpsSvc
    - state:** stopped
    - start\_mode:** disabled



```
- name: Start the SNMP service
  win_service:
    name: SNMP
    state: started
```

# Modules

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

- Written in Powershell
- Modules can be idempotent
- Modules take user input in the form of parameters

```
tasks:  
  - name: start IIS  
    win_service:  
      name: W3Svc  
      state: running
```

# Windows modules

Ansible modules for Windows automation typically begin with `win_*`

**win\_copy** - Copies files to remote locations on windows hosts

**win\_service** - Manage and query Windows services

**win\_domain** - Ensures the existence of a Windows domain

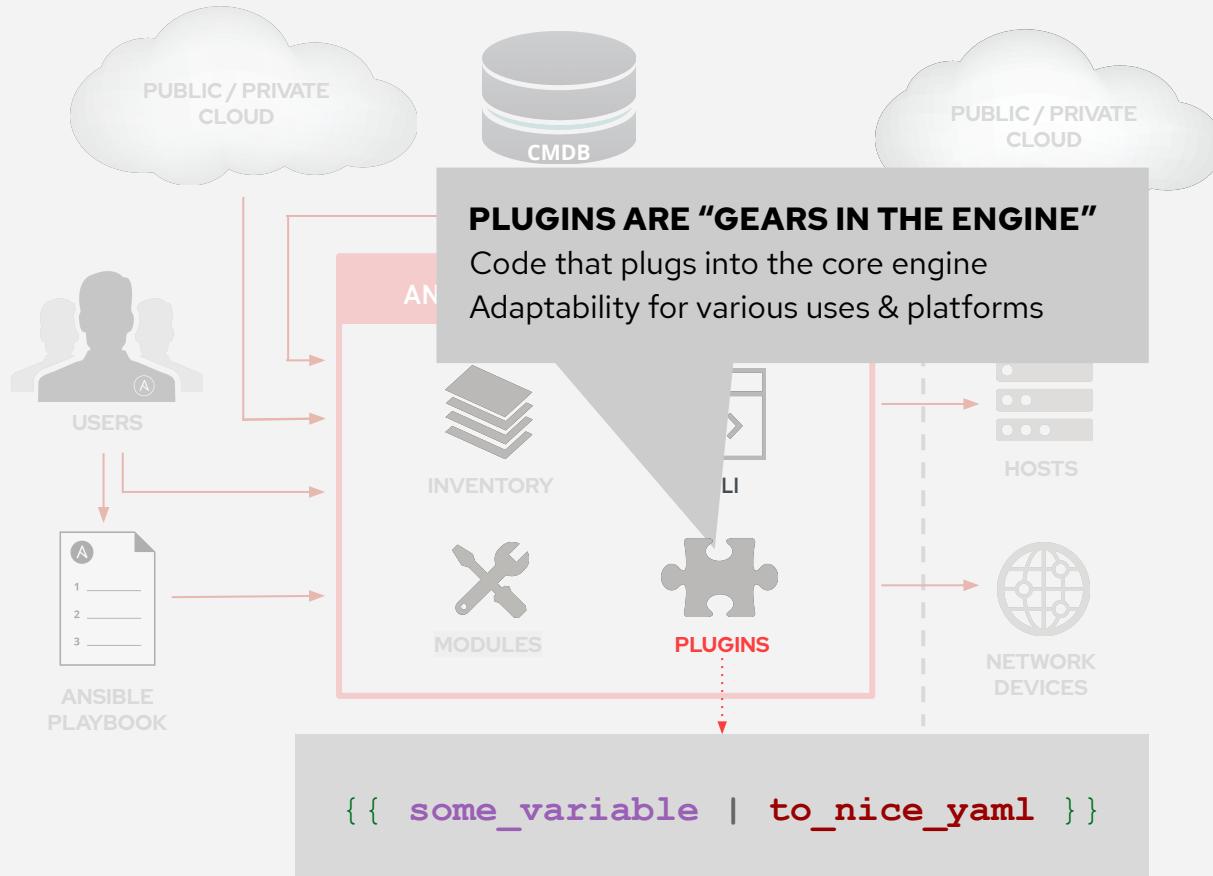
**win\_reboot** - Reboot a windows machine

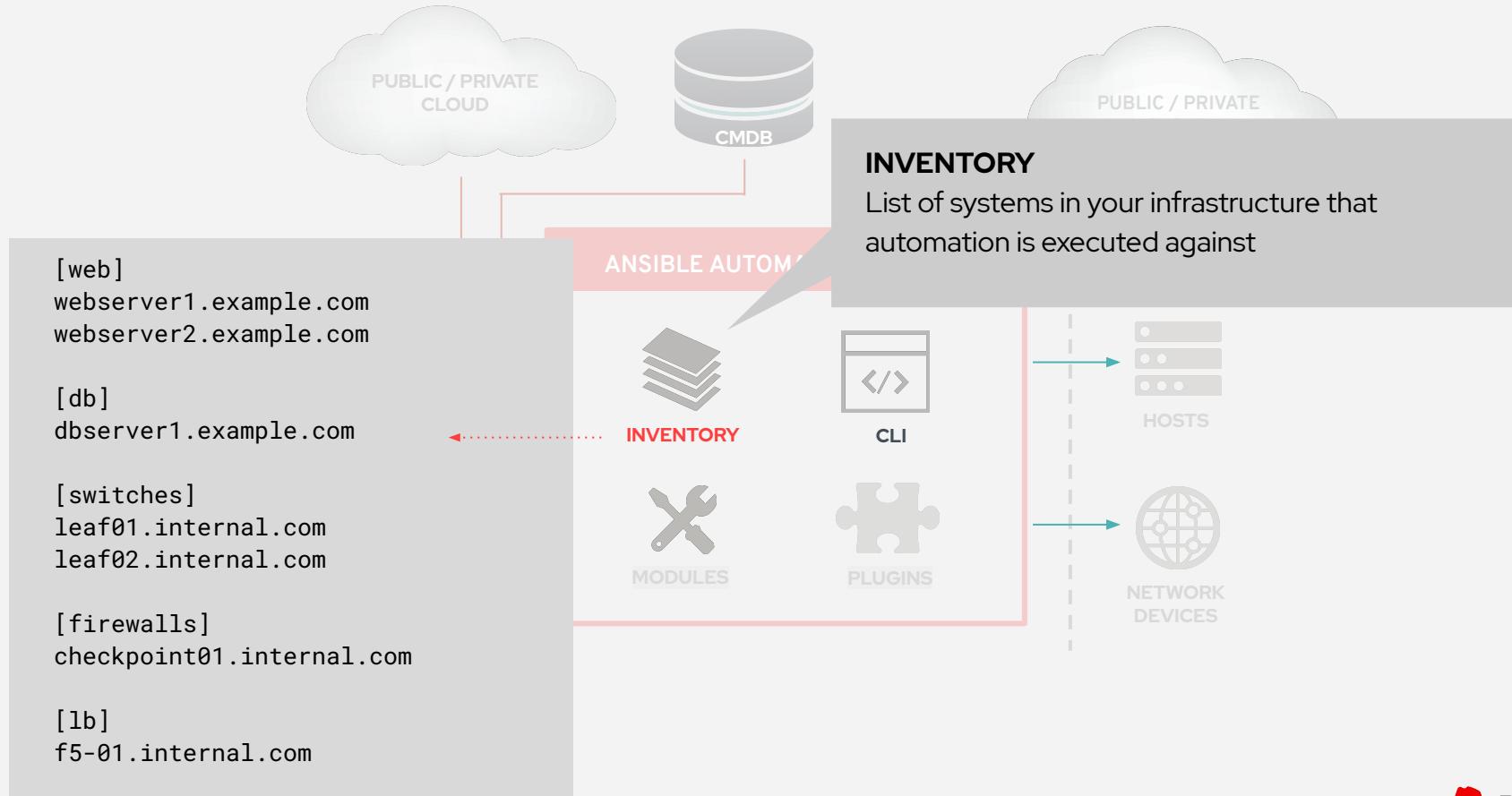
**win\_regedit** - `win_regedit` – Add, change, or remove registry keys and values

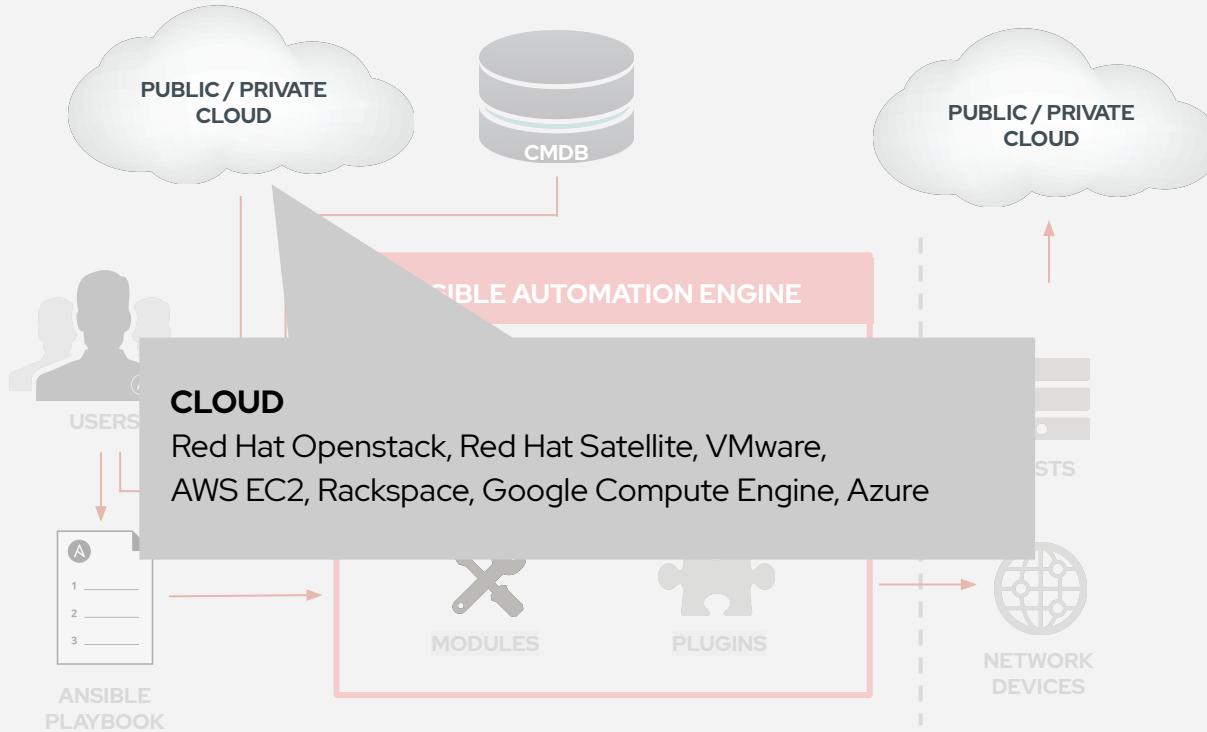
**win\_ping** - A windows version of the classic ping module

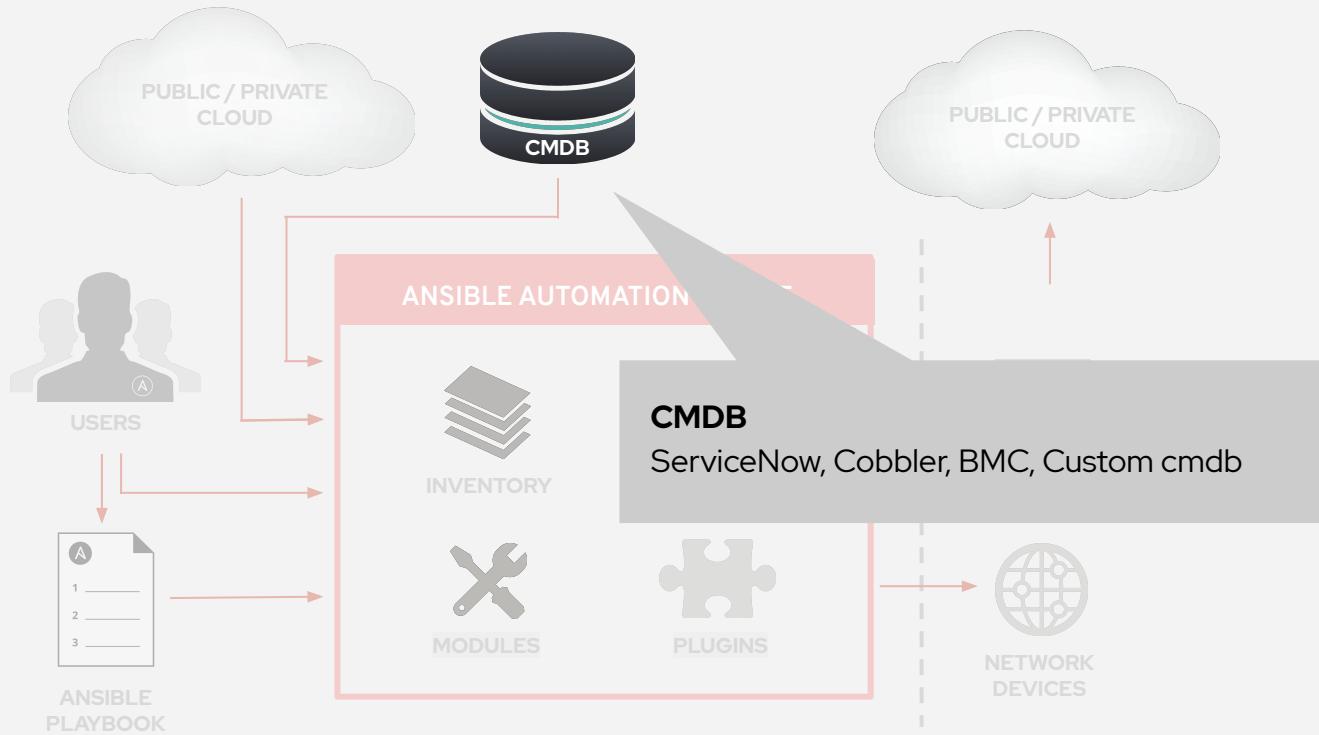
**win\_dsc** - Invokes a PowerShell DSC configuration

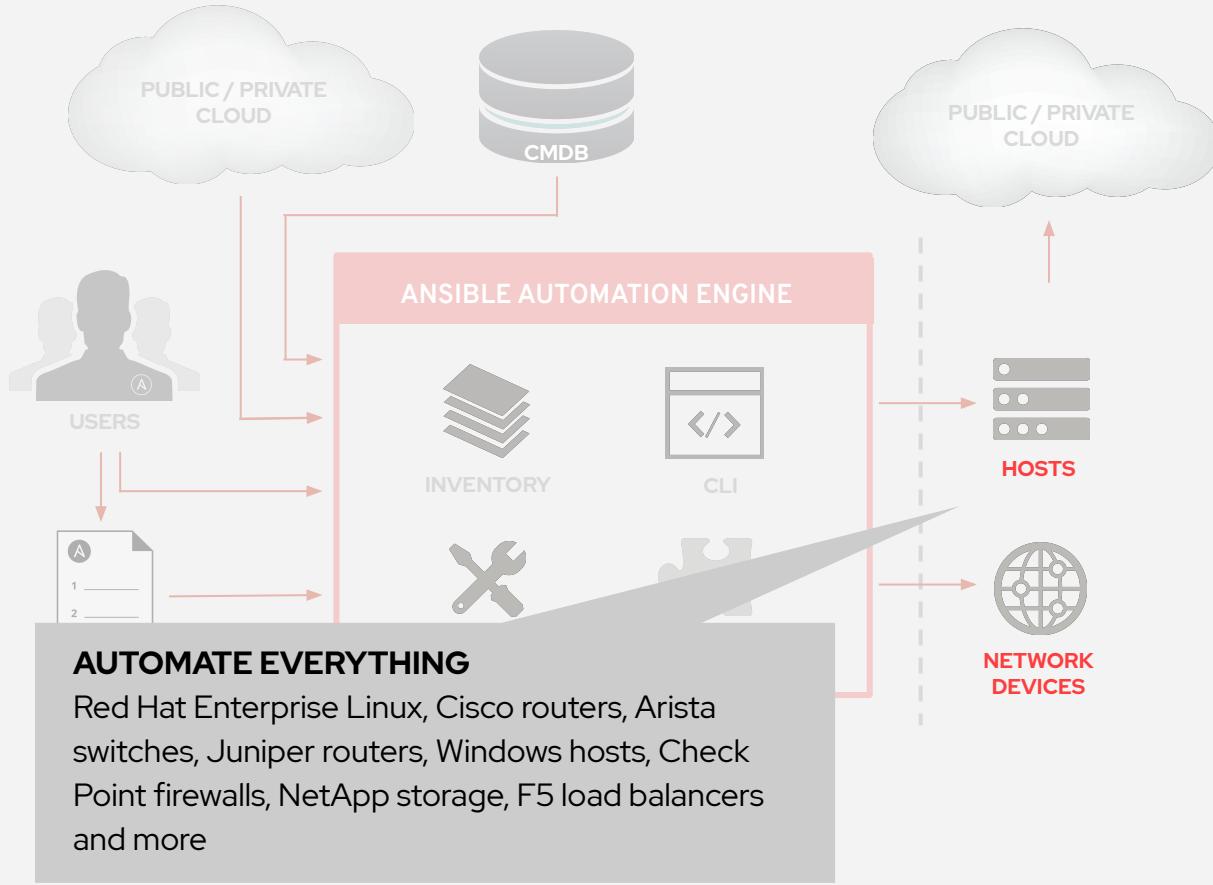
**win\_acl** - Set file/directory/registry permissions for a system user or group











# Tower Introduction

Topics Covered:

- What is Ansible Tower?
- Job Templates
  - Inventory
  - Credentials
  - Projects

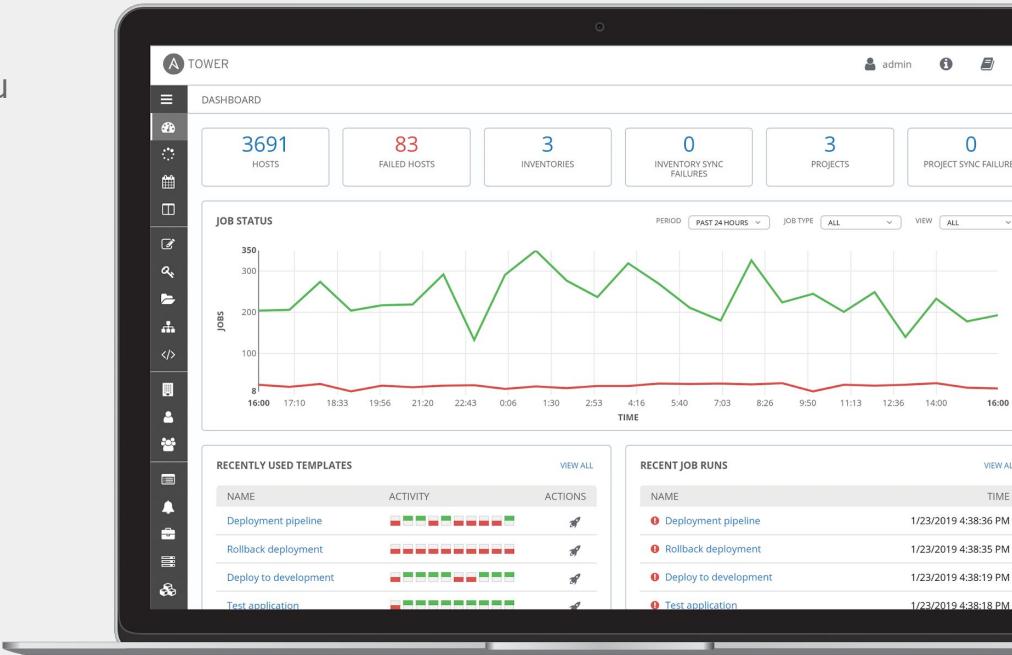


**Red Hat**  
Ansible Automation  
Platform

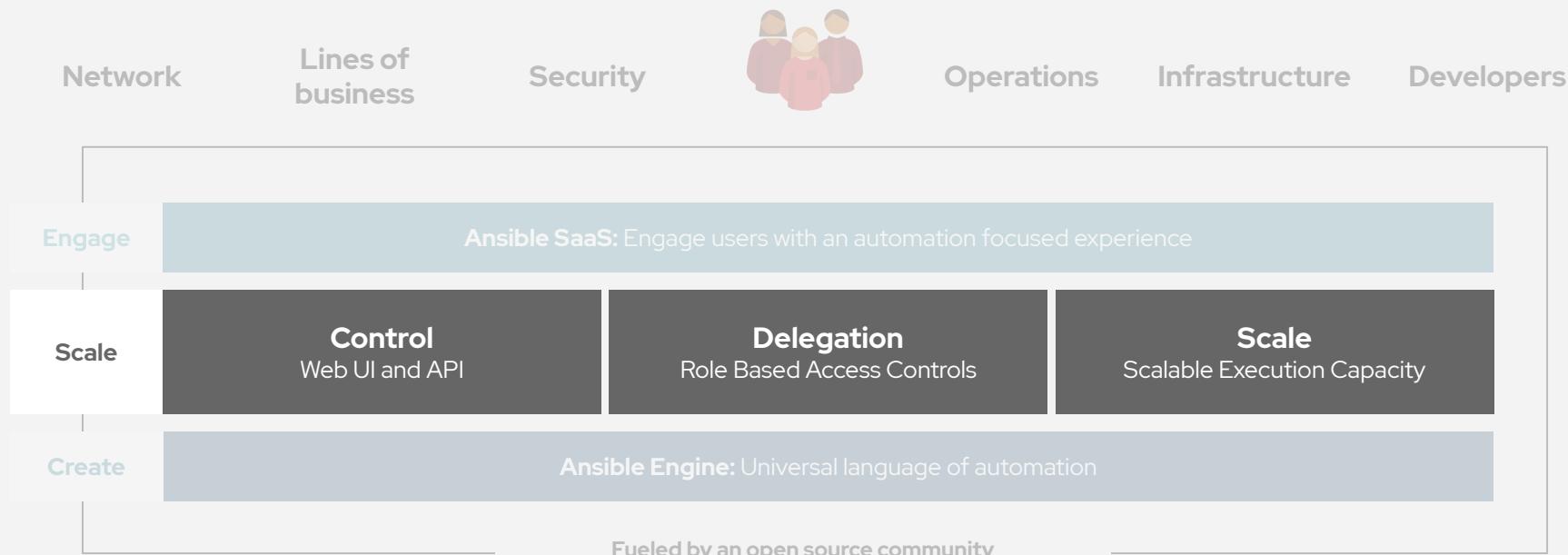
# What is Ansible Tower?

Ansible Tower 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



# Red Hat Ansible Automation Platform



# Red Hat Ansible Tower

## Push button

An intuitive user interface experience makes it easy for novice users to execute playbooks you allow them access to.

## RESTful API

With an API first mentality every feature and function of Tower can be API driven. Allow seamless integration with other tools like ServiceNow and Infoblox.

## RBAC

Allow restricting playbook access to authorized users. One team can use playbooks in check mode (read-only) while others have full administrative abilities.

## Enterprise integrations

Integrate with enterprise authentication like TACACS+, RADIUS, Azure AD. Setup token authentication with OAuth 2. Setup notifications with PagerDuty, Slack and Twilio.

## Centralized logging

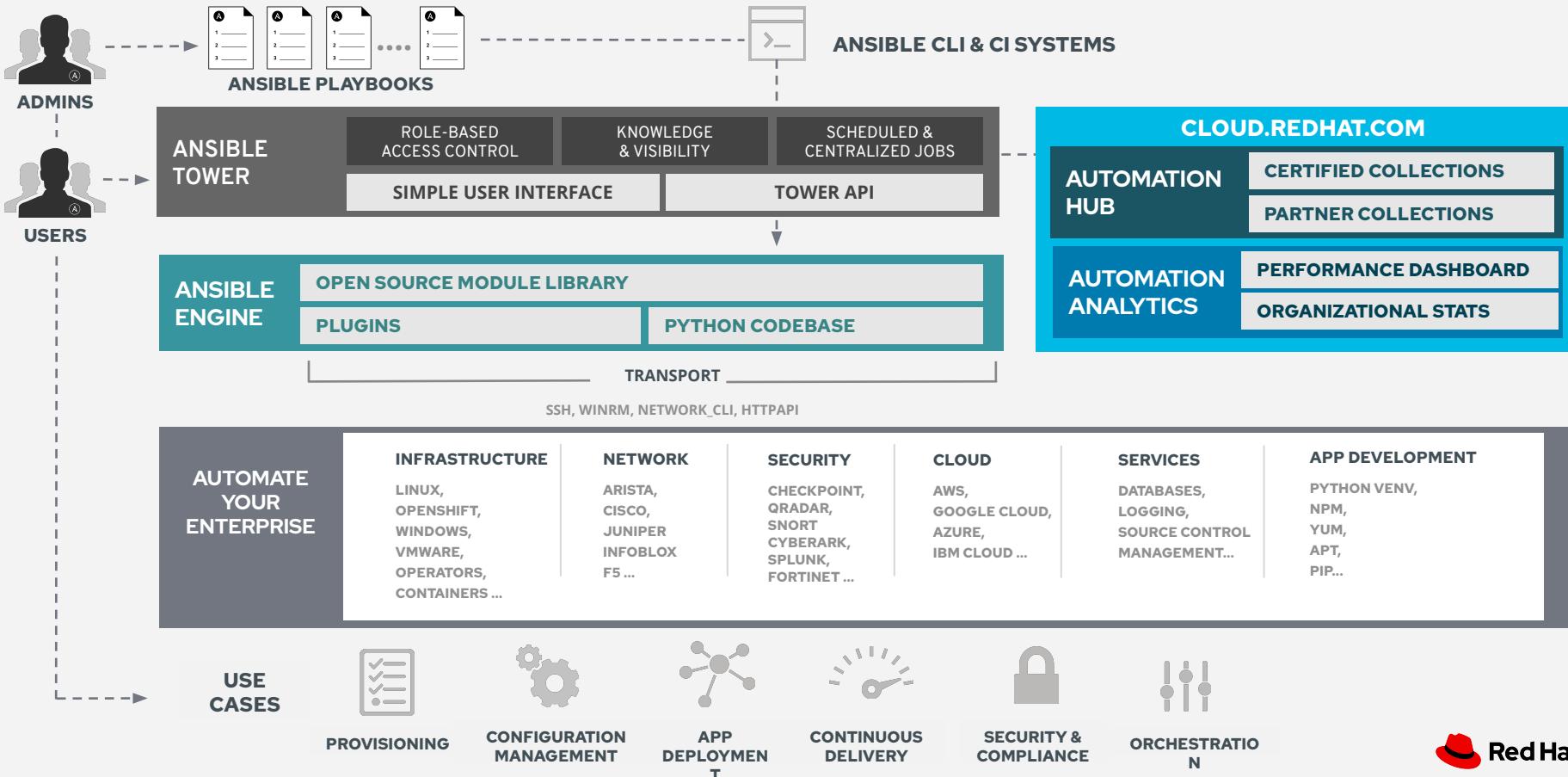
All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Ansible Tower's API.

## Workflows

Ansible Tower's multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.



# Ansible Automation Platform



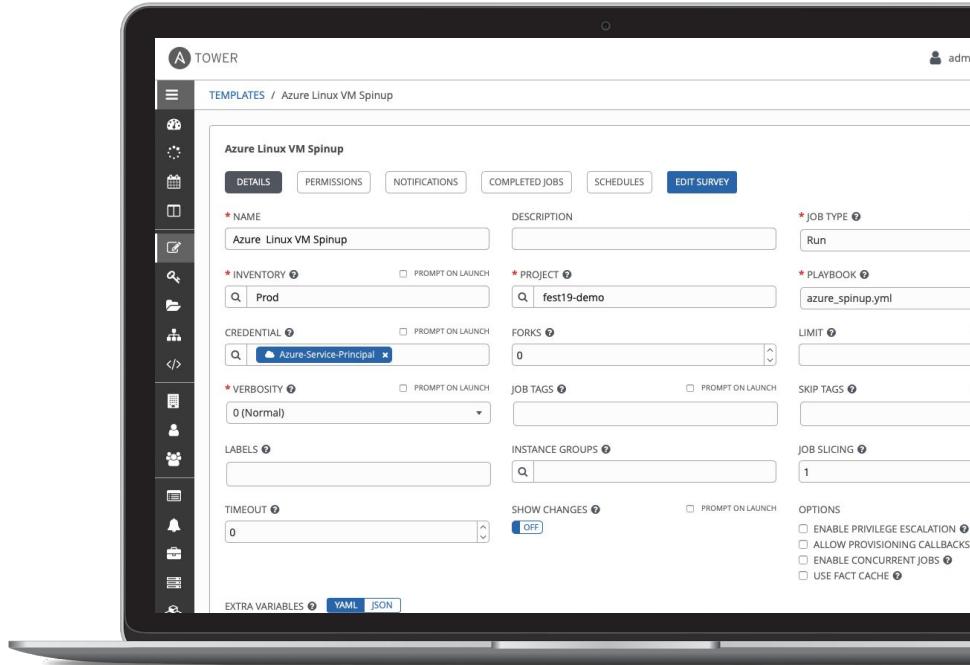
# Job Templates

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



# Inventory

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

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

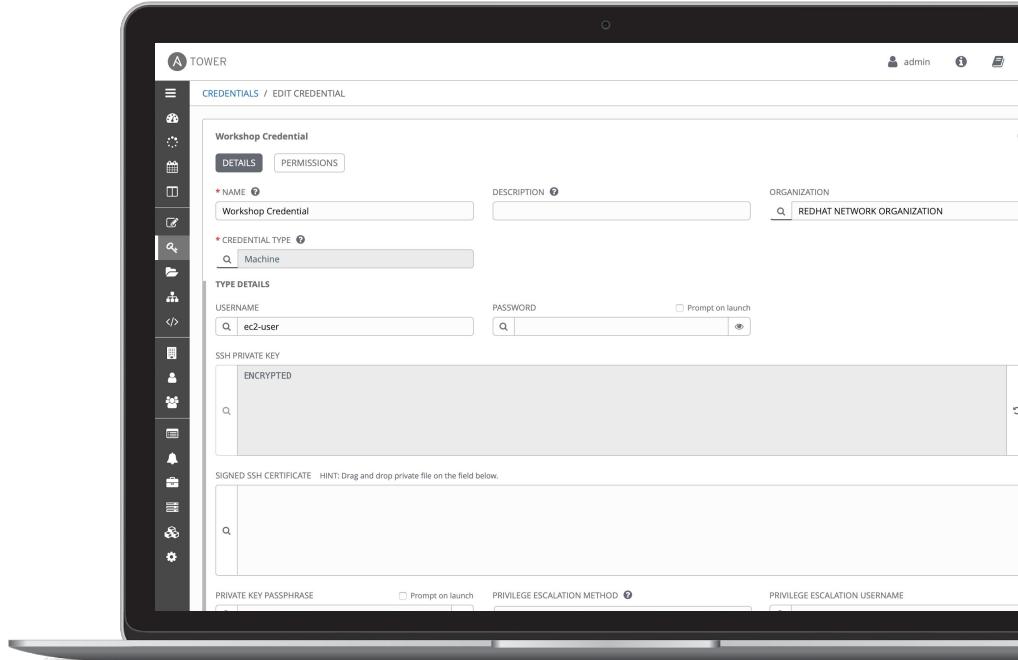
The screenshot shows the Ansible Tower web interface on a laptop screen. The main title bar says 'A TOWER'. Below it, the path 'INVENTORIES / Workshop Inventory / HOSTS' is visible. On the left, there's a vertical sidebar with various icons for navigation. The main content area has tabs at the top: 'DETAILS', 'PERMISSIONS', 'GROUPS', 'HOSTS' (which is selected), 'SOURCES', and 'COMPLETED JOBS'. Below these tabs is a search bar and a 'KEY' button. The main list is titled 'Workshop Inventory' and contains five entries: 'ansible', 'rtr1', 'rtr2', 'rtr3', and 'rtr4'. Each entry has an 'ON' switch (set to 'ON') and a radio button next to it. To the right of the host list, there's a 'RELATED GROUPS' section with several groups listed: 'control', 'cisco', 'arista', 'dc1', 'dc2', 'juniper', and 'arista' again. At the bottom of the page, there are tabs for 'INVENTORIES' and 'HOSTS', a search bar, and filters for 'NAME', 'TYPE', and 'ORGANIZATION'.

# Credentials

Credentials are utilized by Ansible Tower 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

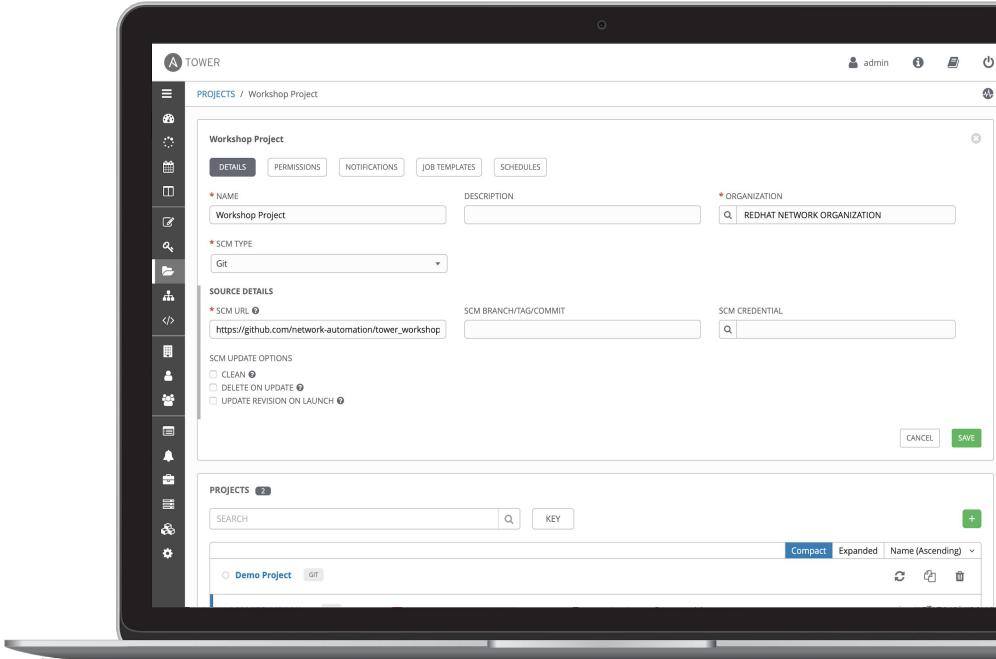
Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.



# Project

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

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.



# Exercise 1

- Configuring Ansible Tower



**Red Hat**  
Ansible Automation  
Platform

# Ad-hoc Commands

Topics Covered:

- What are ad-hoc commands
- Common options
- Run from
  - Command line
  - Ansible Tower



**Red Hat**  
Ansible Automation  
Platform

# Ad-hoc Commands

An ad-hoc command is a single Ansible task to perform quickly, but don't want to save for later.

# Ad-hoc Commands: Common Options

- **-m MODULE\_NAME, --module-name=MODULE\_NAME**  
Module name to execute the ad-hoc command
- **-a MODULE\_ARGS, --args=MODULE\_ARGS**  
Module arguments for the ad-hoc command
- **-b, --become**  
Run ad-hoc command with elevated rights such as sudo, the default method
- **-e EXTRA\_VARS, --extra-vars=EXTRA\_VARS**  
Set additional variables as key=value or YAML/JSON
- **--version**  
Display the version of Ansible
- **--help**  
Display the MAN page for the Ansible tool

# Ad-hoc Commands

```
# check all my inventory hosts are ready to be
# managed by Ansible
$ ansible all -m ping

# collect and display the discovered facts
# for the localhost
$ ansible localhost -m setup

# run the uptime command on all hosts in the
# web group
$ ansible web -m command -a "uptime"
```

# Ad-hoc Commands from Tower

**SNOW DYNAMIC INVENTORY**

DETAILS    PERMISSIONS    GROUPS    **HOSTS**    Select an inventory source by clicking the check box beside it. The inventory source can be a single host or a selection of multiple hosts.

SEARCH        KEY    RUN COMMANDS    

HOSTS	ACTIONS
<input checked="" type="checkbox"/>  server01.rhdemo .io	 
<input checked="" type="checkbox"/>  server02.rhdemo .io	 

# Exercise 2

- Ad-hoc Commands



**Red Hat**  
Ansible Automation  
Platform

# Playbooks

Topics Covered:

- Variables
  - Facts
  - Precedence
- Tasks
  - Handlers



**Red Hat**  
Ansible Automation  
Platform

# Variables

Ansible can work with metadata from various sources and manage their context in the form of variables.

- Command line parameters
- Plays and tasks
- Files
- Inventory
- Discovered facts
- Roles

# Discovered facts

Facts are bits of information derived from examining a host systems that are stored as variables for later use in a play.

```
$ ansible localhost -m setup
localhost | success >> {
    "ansible_facts": {
        "ansible_default_ipv4": {
            "address": "192.168.1.37",
            "alias": "wlan0",
            "gateway": "192.168.1.1",
            "interface": "wlan0",
            "macaddress": "c4:85:08:3b:a9:16",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "192.168.1.0",
            "type": "ether"
        },
    }
}
```

# Variable Precedence

The order in which the same variable from different sources will override each other.

1. command line values (eg “-u user”)
2. role defaults [1]
3. inventory file or script group vars [2]
4. **inventory group\_vars/all** [3]
5. playbook group\_vars/all [3]
6. **inventory group\_vars/\*** [3]
7. playbook group\_vars/\* [3]
8. inventory file or script host vars [2]
9. **inventory host\_vars/\*** [3]
10. playbook host\_vars/\* [3]
11. host facts / cached set\_facts [4]
12. play vars
13. play vars\_prompt
14. play vars\_files
15. role vars (defined in role/vars/main.yml)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include\_vars
19. set\_facts / registered vars
20. role (and include\_role) params
21. include params
22. extra vars (**always win precedence**)

# Tasks

Tasks are the application of a module to perform a specific unit of work.

- **win\_file**: A directory should exist
- **win\_package**: A package should be installed
- **win\_service**: A service should be running
- **win\_template**: Render a configuration file from a template
- **win\_get\_url**: Fetch an archive file from a URL
- **win\_copy**: Copy a file from your repository or a remote source

# Tasks

```
tasks:
- name: Ensure IIS Server is present
  win_feature:
    name: Web-Server
    state: present

- name: Ensure latest index.html file is present
  win_copy:
    src: files/index.html
    dest: c:\www\

- name: Restart IIS
  win_service:
    name: IIS Admin Service
    state: restarted
```

# Handler Tasks

Handlers are special tasks that run at the end of a play if notified by another task when a change occurs.

*If a package gets installed or updated, notify a service restart task that it needs to run.*

# Handler Tasks

```
tasks:  
- name: Ensure IIS Server is present  
  win_feature:  
    name: Web-Server  
    state: present  
  notify: Restart IIS  
  
- name: Ensure latest index.html file is present  
  win_copy:  
    src: files/index.html  
    dest: c:\www\  
  
handlers:  
- name: Restart IIS  
  win_service:  
    name: IIS Admin Service  
    state: restarted
```

# Plays and playbooks

Plays are ordered sets of tasks to execute against host selections from your inventory. A playbook is a file containing one or more plays.

# Plays and playbooks

```
---
```

- name: Ensure IIS is installed and started
  - hosts: web
  - become: yes
  - vars:
    - service\_name: IIS Admin Service
- tasks:
  - name: Ensure IIS Server is present
    - win\_feature:
      - name: Web-Server
      - state: present
  - name: Ensure latest index.html file is present
    - win\_copy:
      - src: files/index.html
      - dest: c:\www\
  - name: Ensure IIS is started
    - win\_service:
      - name: "{{ server\_name }}"
      - state: started

# Meaningful names

```
---
```

- `name: Ensure IIS is installed and started`
  - `hosts: web`
  - `become: yes`
  - `vars:`
    - `service_name: IIS Admin Service`
- `tasks:`
  - `name: Ensure IIS Server is present`
    - `win_feature:`
      - `name: Web-Server`
      - `state: present`
  - `name: Ensure latest index.html file is present`
    - `win_copy:`
      - `src: files/index.html`
      - `dest: c:\www\`
  - `name: Ensure IIS is started`
    - `win_service:`
      - `name: "{{ server_name }}"`
      - `state: started`

# Host selector

```
---
```

- name: Ensure IIS is installed and started
  - hosts: web
  - become: yes
  - vars:
    - service\_name: IIS Admin Service
- tasks:
  - name: Ensure IIS Server is present
    - win\_feature:
      - name: Web-Server
      - state: present
  - name: Ensure latest index.html file is present
    - win\_copy:
      - src: files/index.html
      - dest: c:\www\
  - name: Ensure IIS is started
    - win\_service:
      - name: "{{ server\_name }}"
      - state: started

# Privilege escalation

```
---
```

```
- name: Ensure IIS is installed and started
  hosts: web
  become: yes
  vars:
    service_name: IIS Admin Service

  tasks:
    - name: Ensure IIS Server is present
      win_feature:
        name: Web-Server
        state: present

    - name: Ensure latest index.html file is present
      win_copy:
        src: files/index.html
        dest: c:\www\

    - name: Ensure IIS is started
      win_service:
        name: "{{ server_name }}"
        state: started
```

# Plays variables

```
---
```

- name: Ensure IIS is installed and started
  - hosts: web
  - become: yes
  - vars:
    - service\_name: IIS Admin Service
- tasks:
  - name: Ensure IIS Server is present
    - win\_feature:
      - name: Web-Server
      - state: present
  - name: Ensure latest index.html file is present
    - win\_copy:
      - src: files/index.html
      - dest: c:\www\
  - name: Ensure IIS is started
    - win\_service:
      - name: "{{ server\_name }}"
      - state: started

# Tasks

```
---
```

- name: Ensure IIS is installed and started
  - hosts: web
  - become: yes
  - vars:
    - service\_name: IIS Admin Service
- tasks:
  - name: Ensure IIS Server is present
    - win\_feature:
      - name: Web-Server
      - state: present
  - name: Ensure latest index.html file is present
    - win\_copy:
      - src: files/index.html
      - dest: c:\www\
  - name: Ensure IIS is started
    - win\_service:
      - name: "{{ server\_name }}"
      - state: started

# Exercise 3 & 4

- Your First Playbook



**Red Hat**  
Ansible Automation  
Platform

# Advanced playbooks

Topics Covered:

- Templates
- Loops
- Conditionals
- Tags
- Blocks



**Red Hat**  
Ansible Automation  
Platform

# Doing more with playbooks

Here are some more essential playbook features that you can apply:

- Templates
- Loops
- Conditionals
- Tags
- Blocks

# Doing more with playbooks: **Templates**

Ansible embeds the Jinja2 templating engine that can be used to dynamically:

- Set and modify play variables
- Conditional logic
- Generate files such as configurations from variables

# Doing more with playbooks: Loops

Loops can do one task on multiple things, such as create a lot of users, install a lot of packages, or repeat a polling step until a certain result is reached.

```
- name: Ensure IIS Server is present
  win_feature:
    name: "{{ item }}"
    state: present
  loop:
    - Web-Server
    - NET-Framework-Core
```

# Doing more with playbooks: **Conditionals**

Ansible supports the conditional execution of a task based on the run-time evaluation of variable, fact, or previous task result.

```
- name: Ensure IIS Server is present
  win_feature:
    name: Web-Server
    state: present
  when: ansible_os_family == "Windows"
```

# Doing more with playbooks: Tags

Tags are useful to be able to run a subset of a playbook on-demand.

```
- name: Ensure IIS Server is present
  win_feature:
    name: "{{ item }}"
    state: present
  with_items:
    - Web-Server
    - NET-Framework-Core
  tags:
    - packages

- name: Copy web.config template to Server
  win_template:
    src: templates/web.config.j2
    dest: C:\inetpub\wwwroot\web.config
  tags:
    - configuration
```

# Doing more with playbooks: **Blocks**

Blocks cut down on repetitive task directives, allow for logical grouping of tasks and even in play error handling.

```
- block:
  - name: Ensure IIS Server is present
    win_feature:
      name: "{{ item }}"
      state: present
    with_items:
      - Web-Server

  - name: Copy web.config template to Server
    win_template:
      src: templates/web.config.j2
      dest: C:\inetpub\wwwroot\web.config

when: ansible_os_family == "Windows"
```

# Exercise 5

- Practical Playbook Development



**Red Hat**  
Ansible Automation  
Platform

# Sharing automation

Topics Covered:

- Roles
- Galaxy



**Red Hat**  
Ansible Automation  
Platform

# Roles

Roles are packages of closely related Ansible content that can be shared more easily than plays alone.

- Improves readability and maintainability of complex plays
- Eases sharing, reuse and standardization of automation processes
- Enables Ansible content to exist independently of playbooks, projects -- even organizations
- Provides functional conveniences such as file path resolution and default values

# Roles

## Project with Embedded Roles Example

```
site.yml  
roles/  
  common/  
  files/  
  templates/  
  tasks/  
  handlers/  
  vars/  
  defaults/  
  meta/
```

```
iis/  
  files/  
  templates/  
  tasks/  
  handlers/  
  vars/  
  defaults/  
  meta/
```

# Roles

## Project with Embedded Roles Example

```
# site.yml
---
- name: Execute common and iis role
  hosts: web
  roles:
    - common
    - iis
```

# Roles

**<http://galaxy.ansible.com>**

Ansible Galaxy is a hub for finding, reusing and sharing Ansible content.

Jump-start your automation project with content contributed and reviewed by the Ansible community.

# Exercise 6

- A Playbook Using Roles



**Red Hat**  
Ansible Automation  
Platform

# Next Steps

## GET STARTED

[ansible.com/get-started](https://ansible.com/get-started)

[ansible.com/tower-trial](https://ansible.com/tower-trial)

---

## WORKSHOPS & TRAINING

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

[Red Hat Training](#)

## JOIN THE COMMUNITY

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

---

## SHARE YOUR STORY

[Follow us @Ansible](#)

[Friend us on Facebook](#)



# AnsibleFest

October 13-14, 2020 | Virtual Experience



# 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)