

# How to use this deck

**Name:**

Windows Automation workshop

**Purpose:**

These additional slides are used in conjunction with the windows automation workshop as provisioned from:  
<https://github.com/ansible/workshops>

**Last updated:**

Jan 19, 2022

**What this deck is for?****What this deck is not for?****Google Slides source link (Red Hat internal):**

[https://docs.google.com/presentation/d/1R05CQiCoqLDES1vTI\\_1fQrRoWM1NuW-uBOJRvtJzE/edit#slide=id.g10efc4a0549\\_0\\_2429](https://docs.google.com/presentation/d/1R05CQiCoqLDES1vTI_1fQrRoWM1NuW-uBOJRvtJzE/edit#slide=id.g10efc4a0549_0_2429)

**Owner:**

Ansible MBU, ansible-pmm-tmm@redhat.com

**List of all official Ansible content:**

Red Hat Ansible Automation Platform One Stop:

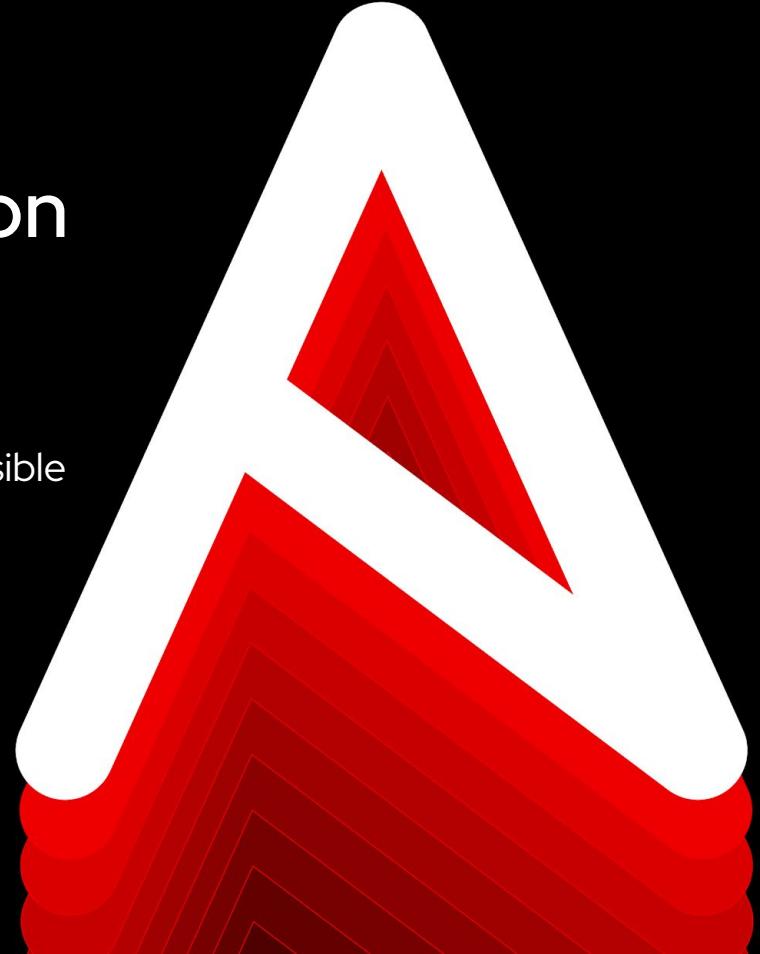
[https://redhat.hightspot.com/items/5966647572ad8e20778bc270?lfr\\_m=srp.10](https://redhat.hightspot.com/items/5966647572ad8e20778bc270?lfr_m=srp.10)

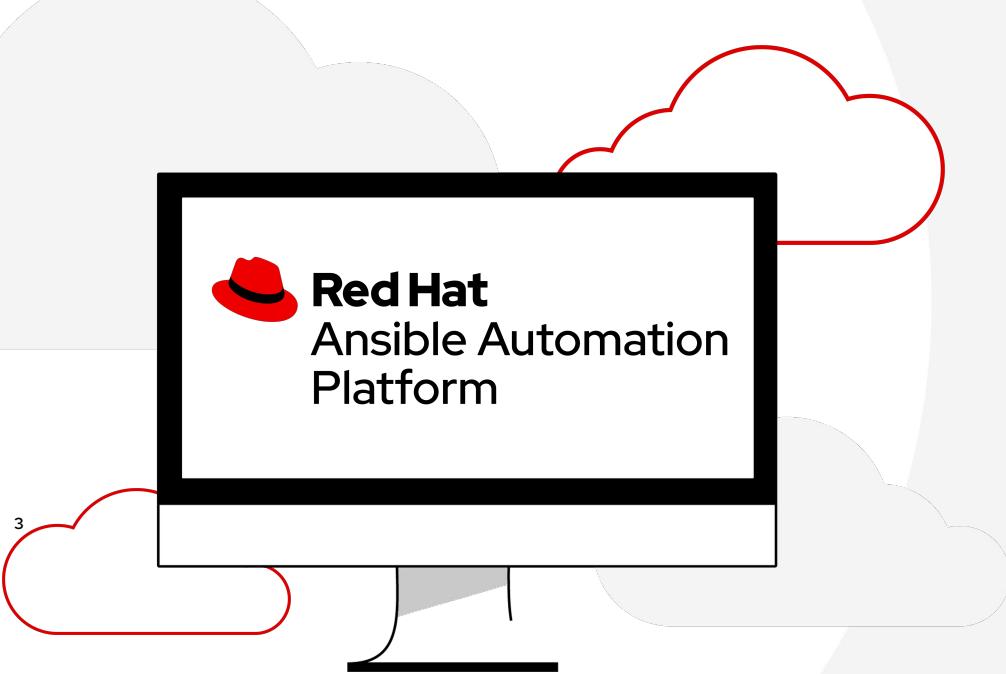


Ansible Automation  
Platform

# Ansible Windows automation workshop

Introduction to automating Microsoft Windows with Ansible  
Automation Platform 2



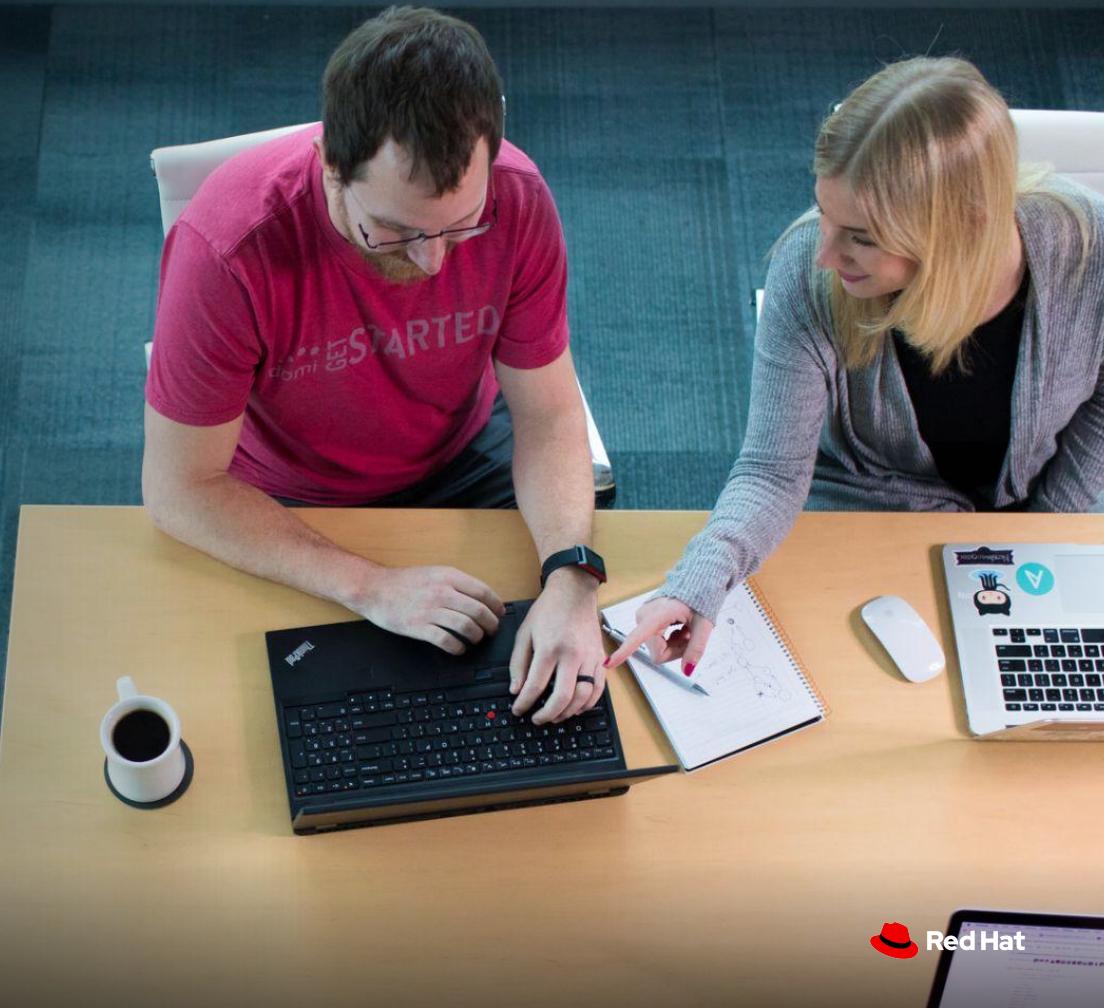


# What you will learn

- 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

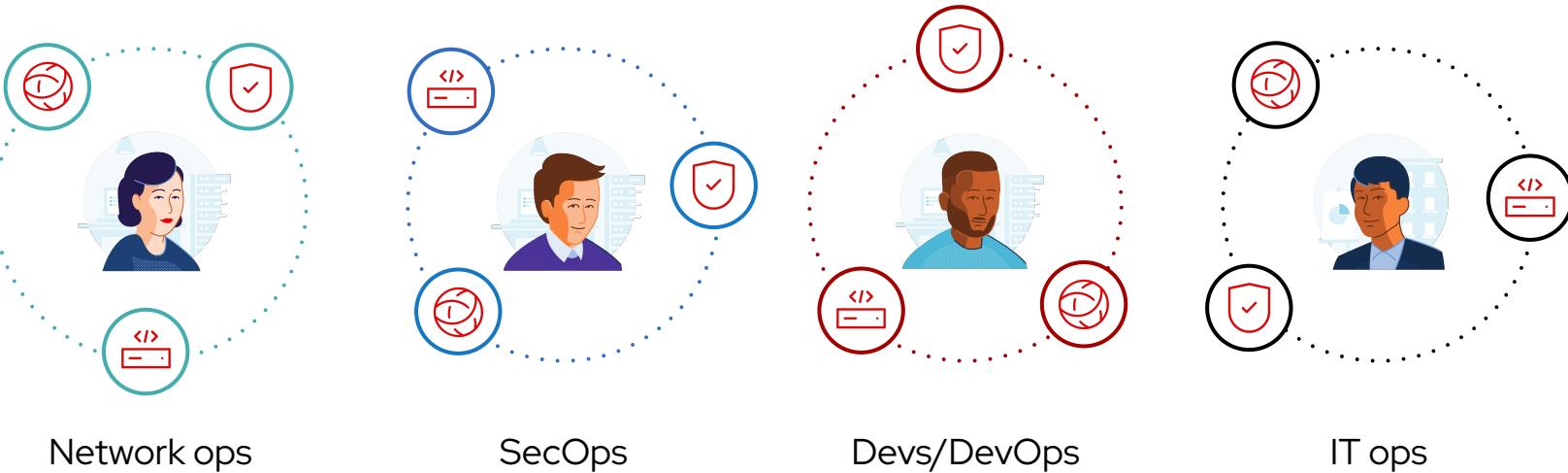


Anyone can automate...  
but an enterprise needs  
to coordinate and scale



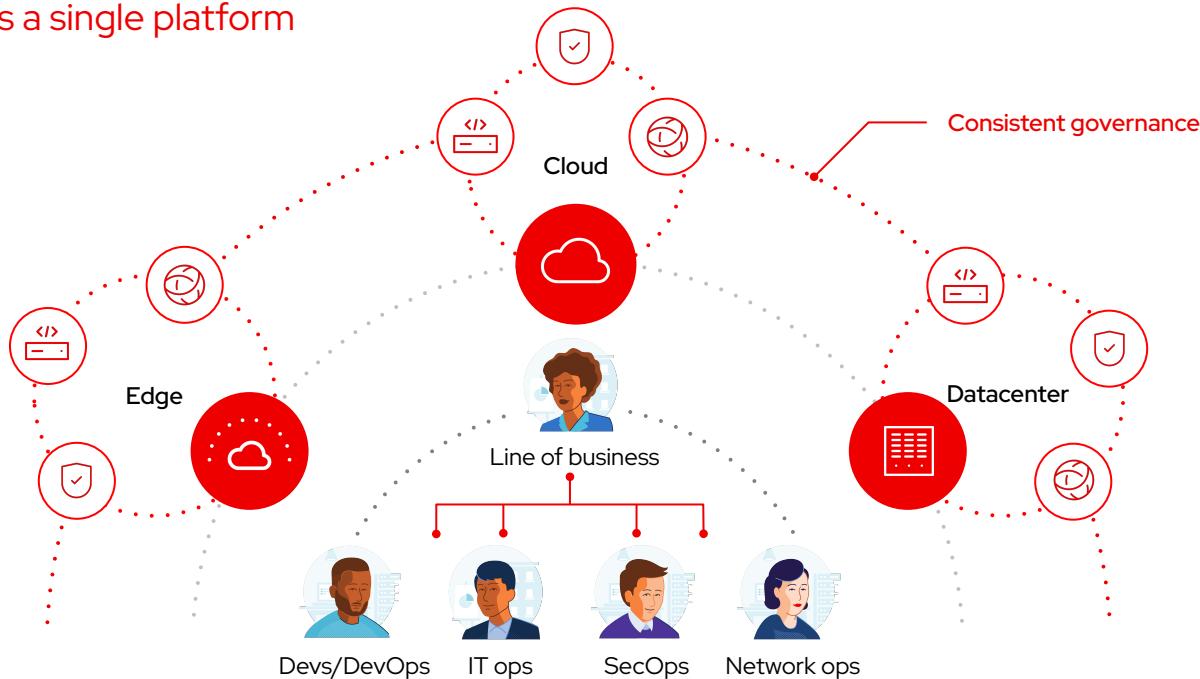
# Many organizations share the same challenge

Too many unintegrated, domain-specific tools



# Break down silos

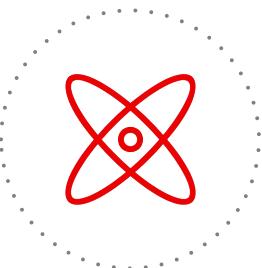
Different teams a single platform



---

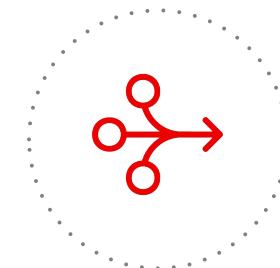
# Why the Red Hat® Ansible® Automation Platform?

# 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 ...

100+  
certified platforms



Infrastructure



Cloud



Network



Security



ARISTA



Check Point  
SOFTWARE TECHNOLOGIES LTD



CYBERARK®



FORTINET®

---

# What makes a platform?



## Red Hat Ansible Automation Platform

Combining the universal automation language with cloud services and certified content for automating, deploying, and operating applications, infrastructure and services securely at enterprise scale.



### Ansible automation

Providing scalable, secure implementation for describing, building, and managing the deployment of enterprise IT applications across diverse enterprise architectures.

### Cloud services

Cloud services that facilitate team collaboration and provide operational analytics for automating heterogeneous, hybrid environments.

### Certified content

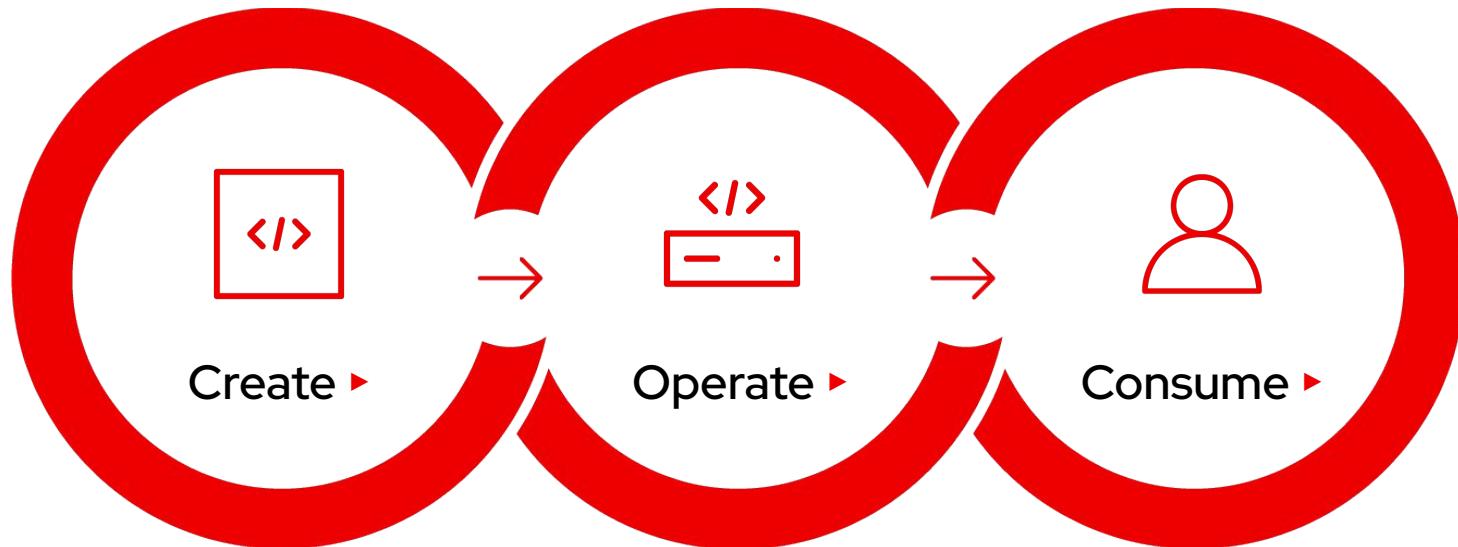
Extends native platform capabilities with certified, supported content designed to expand the automation domain and accelerate adoption for enterprise customers.



**Red Hat**

Ansible Automation  
Platform

Holistic automation for your enterprise .....





## Red Hat Ansible Automation Platform



Content creators



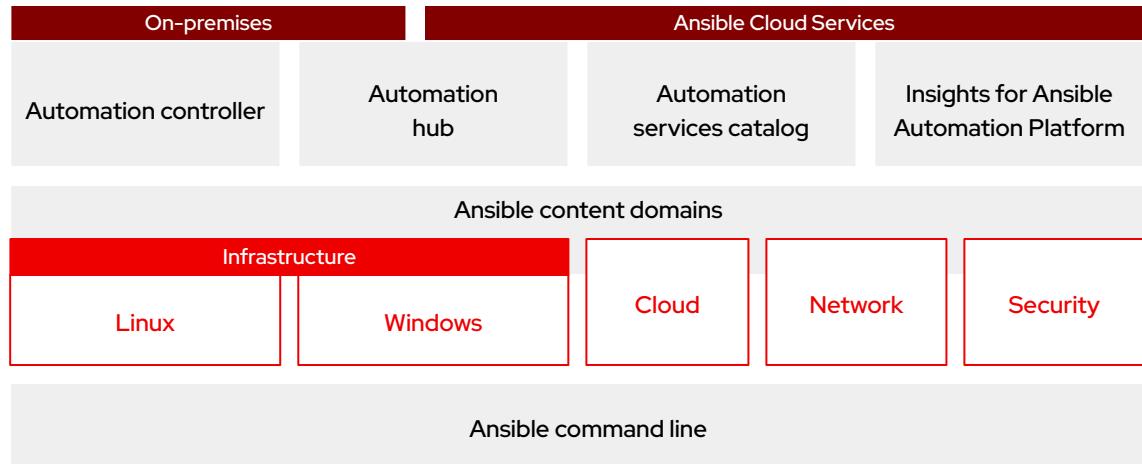
Operators



Domain experts



Users



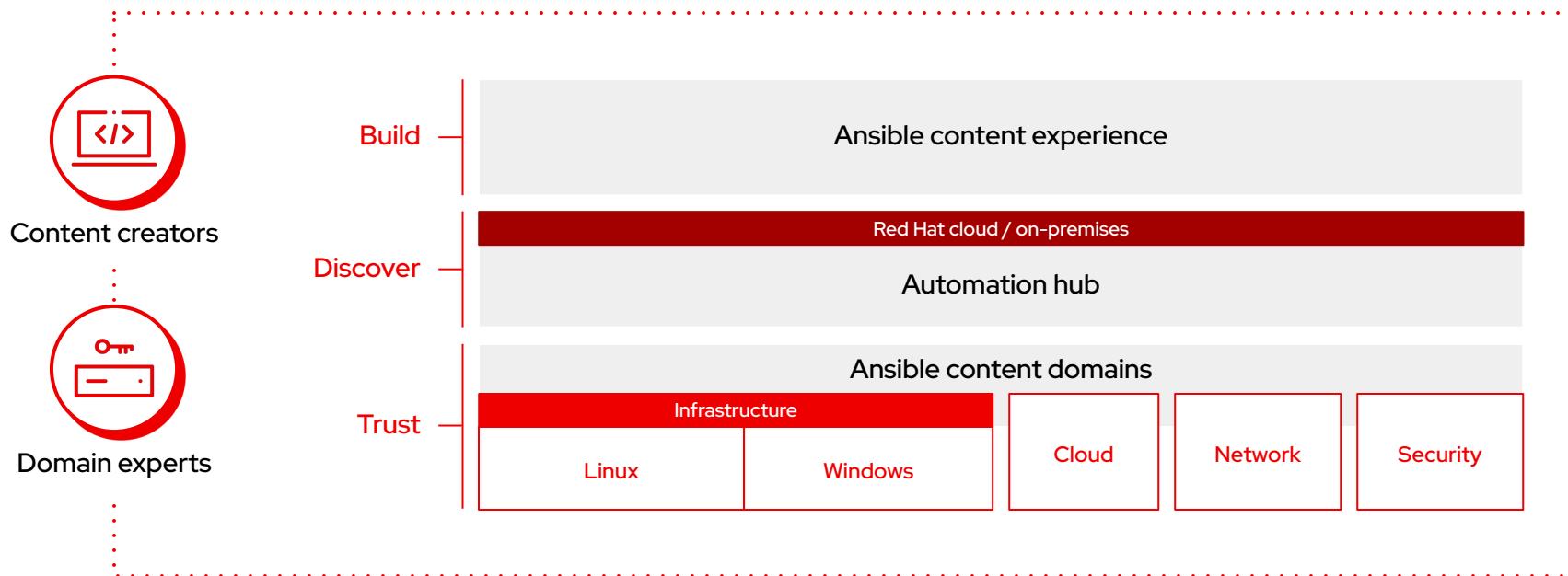
Fueled by an  
open source community



# Create

# Create

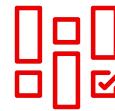
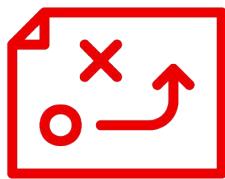
## The automation lifecycle





```
---  
- 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
```

## 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”



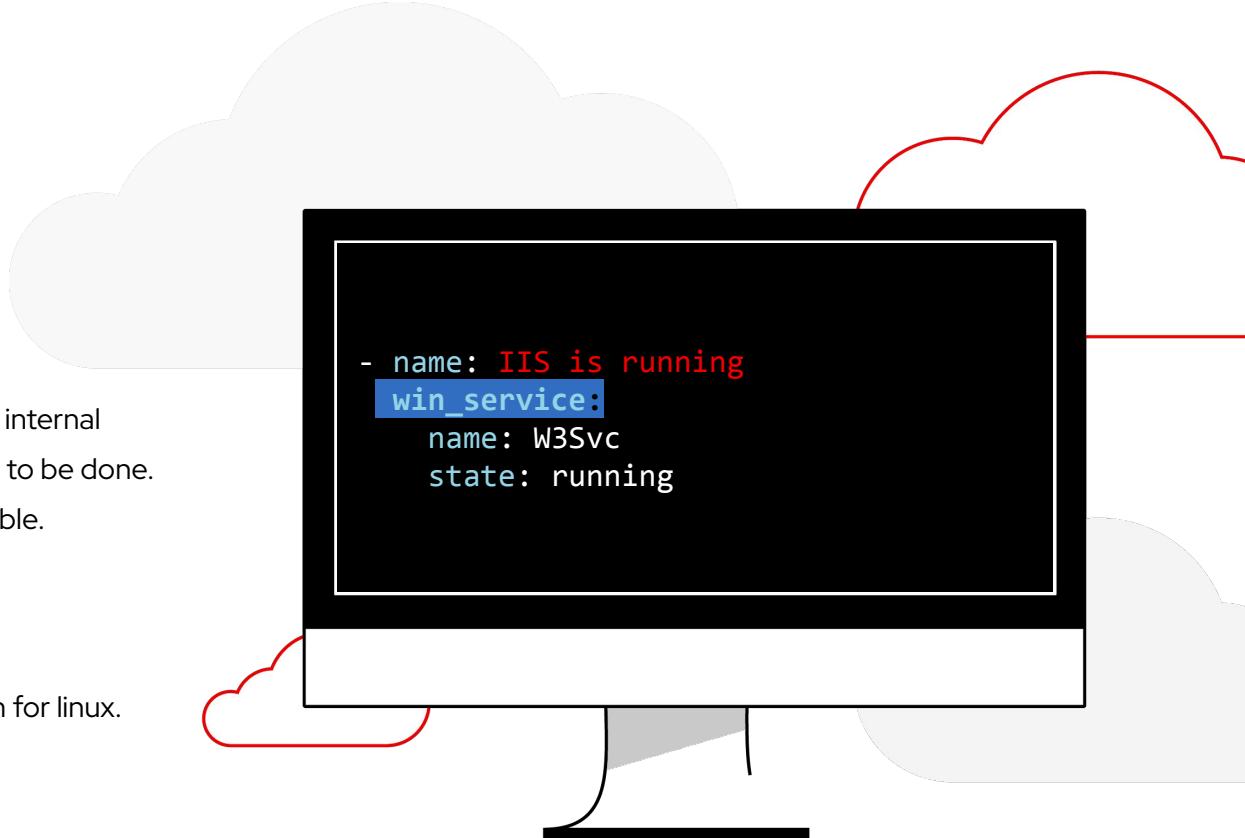
## What are they?

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



## Language

Powershell for Windows, python for linux.  
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: start IIS  
hosts: windows-web  
become: yes  
  
Example filter plugins:  
{{ some_variable | to_nice_json }}  
{{ some_variable | to_nice_yaml }}
```

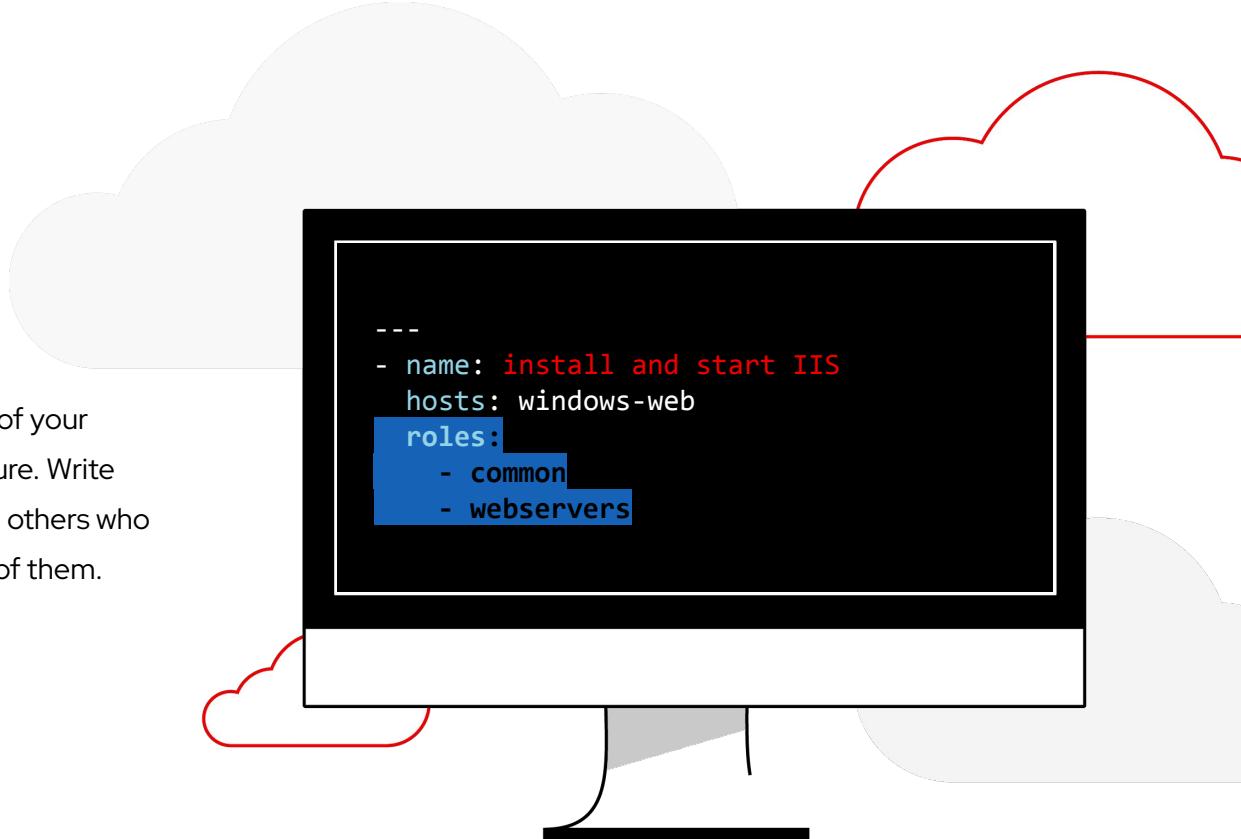
# Ansible roles

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.



# Collections

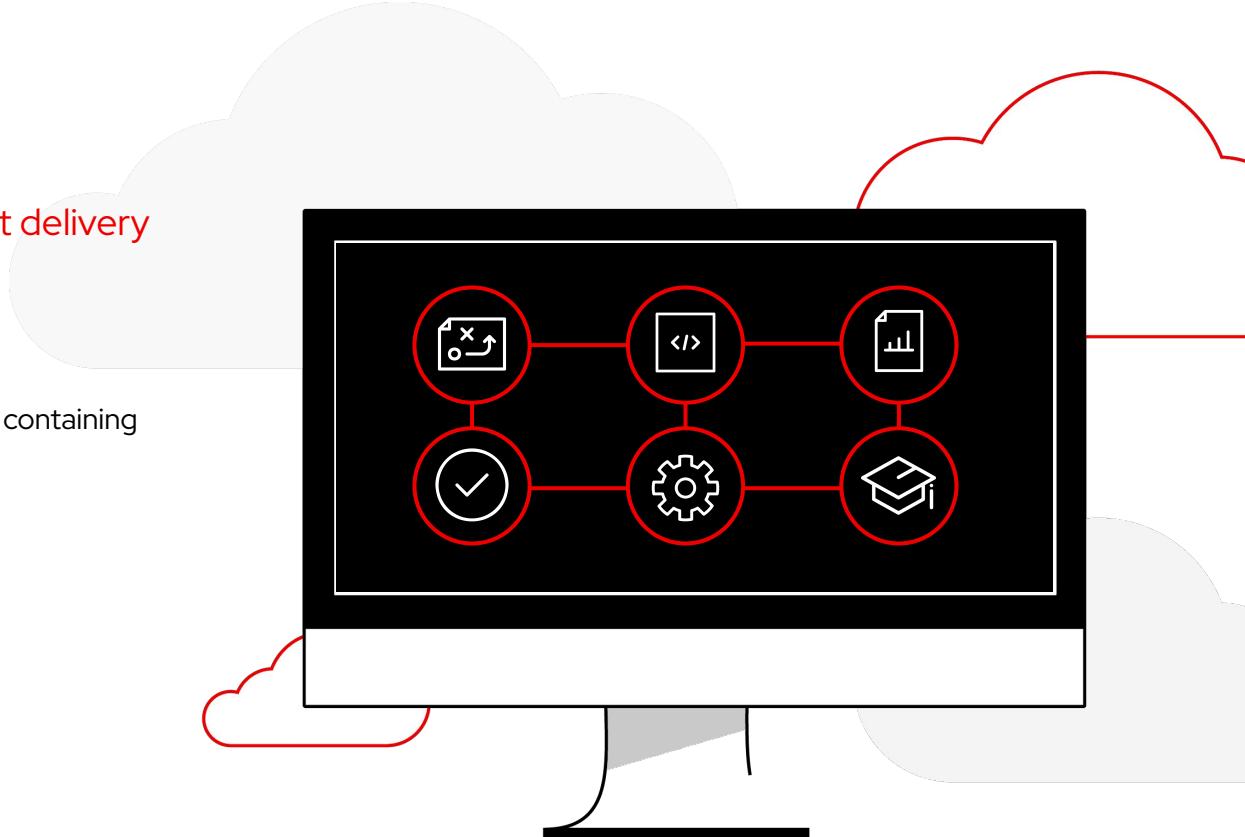
Simplified and consistent content delivery



## What are they?

Collections are a data structure containing automation content:

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





```
nginx_core
├── MANIFEST.json
├── playbooks
│   └── deploy-nginx.yml
├── ...
├── plugins
└── README.md
```

roles

```
    └── nginx
        ├── defaults
        ├── files
        │   └── ...
        ├── tasks
        └── templates
            └── ...
```

```
    └── nginx_app_protect
    └── nginx_config
```

### deploy-nginx.yml

```
---
```

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

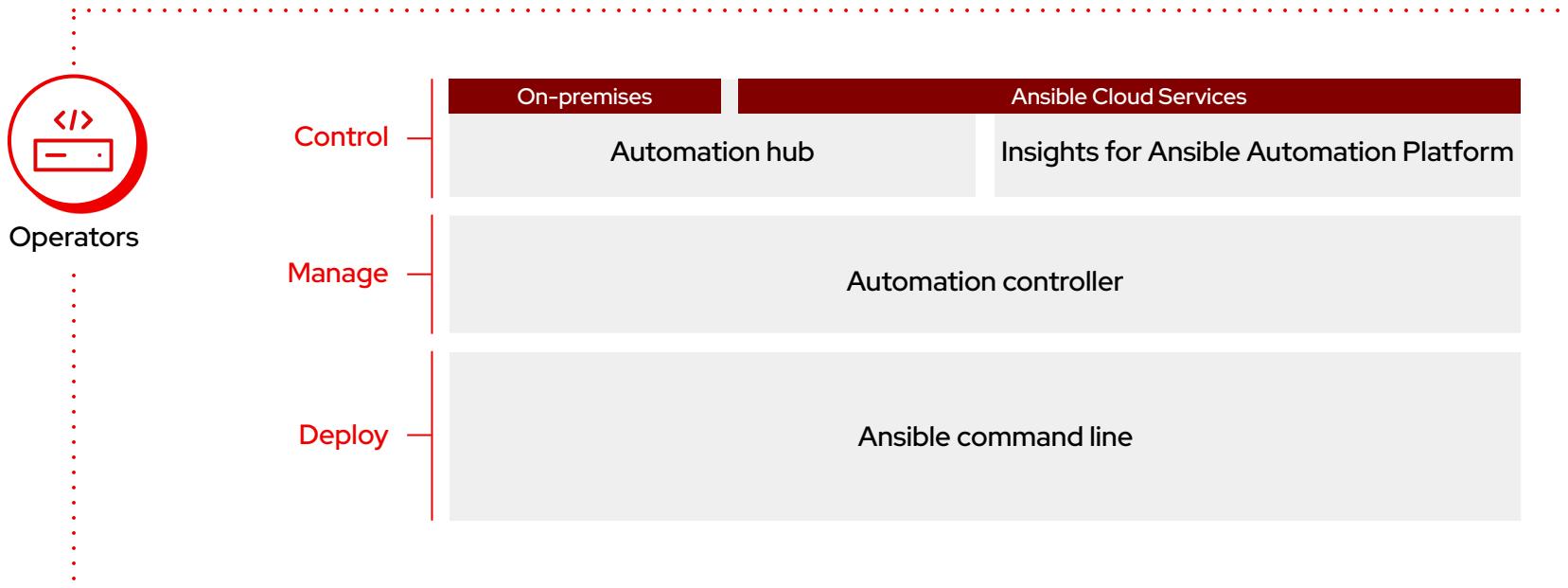
```
- name: Install NGINX App Protect
  include_role:
    name: nginxinc.nginx_app_protect
  vars:
    nginx_app_protect_setup_license: false
    nginx_app_protect_remove_license: false
    nginx_app_protect_install_signatures: false
```

---

# Automation Controller

# Operate

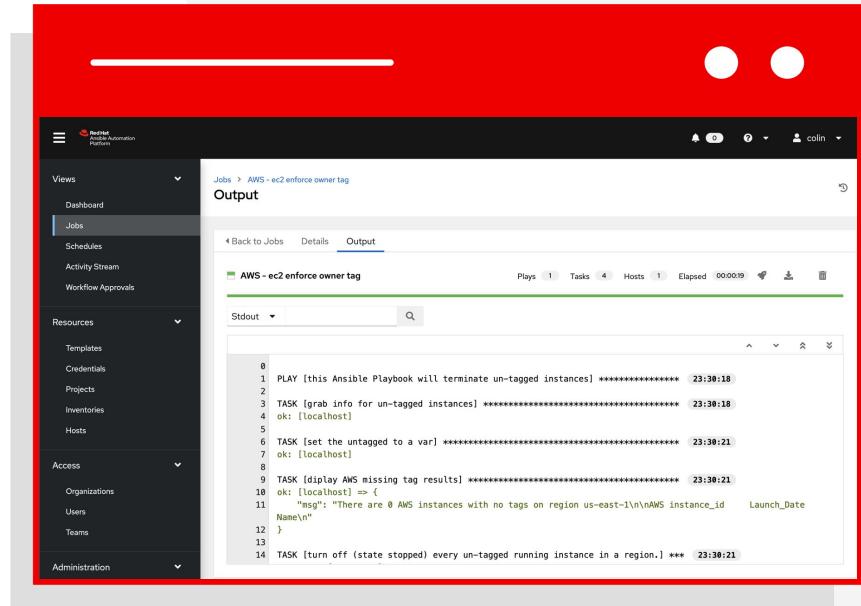
## The automation lifecycle



# A playbook run

## Where it all starts

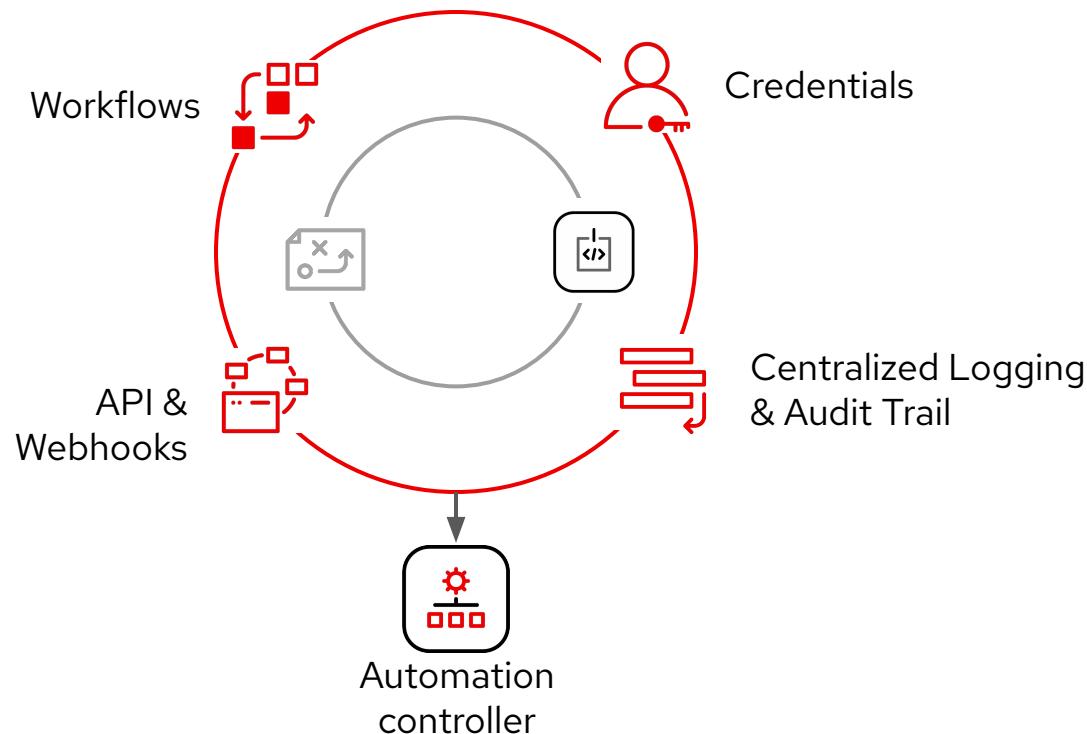
- ▶ A playbook is interpreted and run against one or multiple hosts - task by task. The order of the tasks defines the execution.
- ▶ In each task, the module does the actual work.



The screenshot shows the Red Hat Ansible Automation Platform web interface. On the left, a dark sidebar menu lists various sections: Views (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), Resources (Templates, Credentials, Projects, Inventories, Hosts), Access (Organizations, Users, Teams), and Administration. The main content area has a red header bar. Below the header, the URL is "Jobs > AWS - ec2 enforce owner tag" and the tab selected is "Output". The output pane displays the command-line logs of a playbook run titled "AWS - ec2 enforce owner tag". The logs show the following tasks:

```
0 PLAY [this Ansible Playbook will terminate un-tagged instances] ****
1 TASK [grab info for un-tagged instances] ****
2 ok: [localhost]
3 TASK [set the untagged to a var] ****
4 ok: [localhost]
5 TASK [display AWS missing tag results] ****
6 ok: [localhost]
7 ok: [localhost]
8
9 TASK [display AWS missing tag results] ****
10 ok: [localhost] => {
11     "msg": "There are 0 AWS instances with no tags on region us-east-1\n\nAWS instance_id Launch_Date
Name\n"
12 }
13
14 TASK [turn off (state stopped) every un-tagged running instance in a region.] ***
```

# Anatomy of Automation Operation



# Execution of content

## Running at the core

- ▶ The central execution of automation content is managed and done either via central cluster..
- ▶ Can also sync git repositories, takes care of execution environments, collections, credentials, inventory and logging.
- ▶ Full audit trail of the execution, including what version of content was executed, what variable values were provided, etc.

Templates > AWS - ec2 enforce owner tag

Details

Name	AWS - ec2 enforce owner tag	Description	stop all instances without an owner tag
Job Type	run	Organization	Public Cloud Group
Inventory	no inventory - API	Project	aws ec2 operational project
Execution Environment	aws_ee	Playbook	playbooks/no_tags.yaml
Forks	0	Verbosity	0 (Normal)
Timeout	0	Show Changes	Off
Job Slicing	1	Created	4/15/2021, 9:59:03 AM by seanc
Last Modified	1/5/2022, 4:25:40 PM by admin		
Enabled Options	Concurrent Jobs		
Credentials	Cloud: Sean AWS		

# Inventories and credentials

## How to talk to others

- ▶ An inventory is a collection of hosts (nodes) with associated data and groupings that the automation platform can connect to and manage:
  - Nodes
  - Groups
  - Can be static or dynamic
  - Smart inventories possible
- ▶ And what usernames and passwords do you use during connection? That is kept in the credentials.

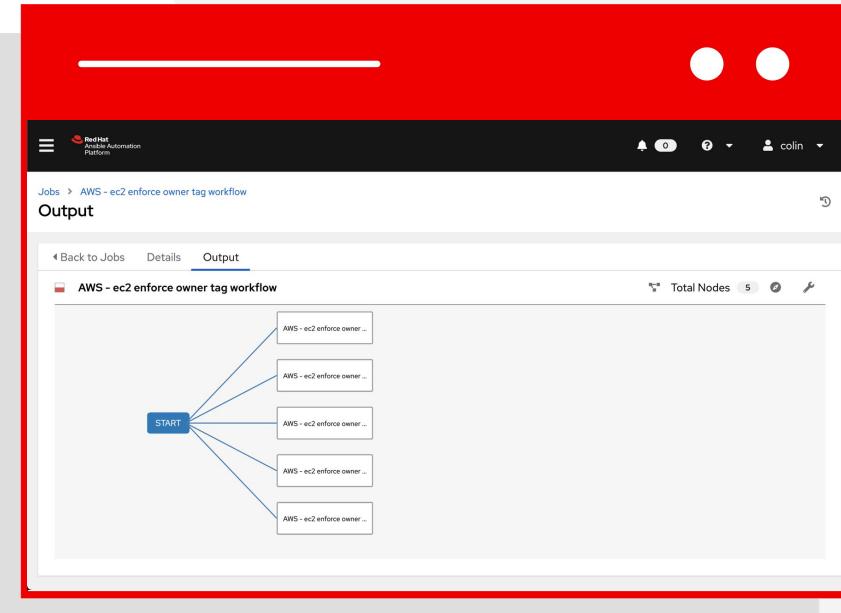
The screenshot shows the 'Hosts' page within the Red Hat Ansible Automation Platform. The top navigation bar includes 'Inventories > ServiceNow' and 'Hosts'. Below the navigation is a search bar and buttons for 'Add' and 'Delete'. The main area lists four hosts:

Name	Action
ApplicationServerHelpdesk	On (Edit)
ApplicationServerPeopleSoft	On (Edit)
Car-1	On (Edit)
Car-3	On (Edit)

# 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.



# 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.

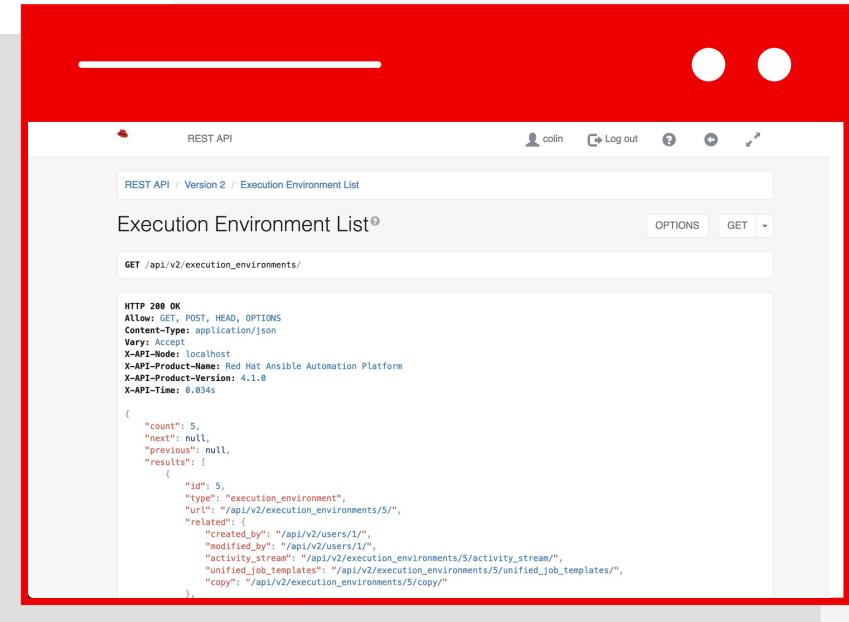
The screenshot shows the Red Hat Ansible Platform web interface. On the left, a dark sidebar menu lists various navigation options: Views (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), Resources (Templates, Credentials, Projects, Inventories, Hosts), Access (Organizations, Users, Teams), and Administration. The 'Inventories' option under Resources is currently selected. The main content area has a red header bar. Below it, the title 'Access' is displayed above a table. The table has columns for Username, First name, Last name, and User Roles. It lists five users: admin, ajay, andriusb, colin, and brandt, each assigned the 'System Administrator' role. Navigation controls at the bottom indicate there are 9 items in total, with 1-5 visible and 1 of 2 pages shown.

Username	First name	Last name	User Roles
admin			System Administrator
ajay	Ajay	Chenampara	System Administrator
andriusb	Andrius	Benokraitis	System Administrator
colin	Colin	McNaughton	System Administrator
brandt	Craig	Brandt	System Administrator

# API

## Integration of automation into larger workflows

- ▶ The API provides programmatic access to the automation via a defined interface.
- ▶ Underneath it is still powered by the same bits and pieces which are at the core: workflows, inventories, etc.
- ▶ It offers simple integration into other tools like ITSM, SOAR, etc.



The screenshot shows a web browser window with a red header bar. The header bar contains the text "REST API", a user profile icon with the name "colin", and a "Log out" button. Below the header, the main content area has a title "Execution Environment List®". To the right of the title are two buttons: "OPTIONS" and "GET". Below these buttons is a code block containing a JSON response from a GET request to "/api/v2/execution\_environments". The JSON response includes headers and a detailed list of execution environments.

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
X-API-Node: localhost
X-API-Product-Name: Red Hat Ansible Automation Platform
X-API-Product-Version: 4.1.0
X-API-Time: 0.034s

{
  "count": 5,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 5,
      "type": "execution_environment",
      "url": "/api/v2/execution_environments/5/",
      "related": {
        "created_by": "/api/v2/users/1/",
        "modified_by": "/api/v2/users/1/",
        "activity_stream": "/api/v2/execution_environments/5/activity_stream/",
        "unified_job_templates": "/api/v2/execution_environments/5/unified_job_templates/",
        "copy": "/api/v2/execution_environments/5/copy/"
      }
    }
]
```

# Lab Time

## Exercise 1 - Configure Automation Controller

This lab is all about exploring the environment and configuring Automation Controller to import project code from source control



Approximate time: 15 mins

---

# Ad-hoc command execution

# 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 win_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 Automation Controller

The screenshot shows the Red Hat Ansible Automation Platform interface. The top navigation bar includes the Red Hat logo, the platform name, a notification bell with 0 notifications, a help icon, and a user account for 'admin'. The left sidebar has sections for 'Views' (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals) and 'Resources' (Templates, Credentials, Projects, Inventories, Hosts). The 'Inventories' section is currently selected. The main content area shows the 'Inventories > Demo Inventory' view, specifically the 'Hosts' tab. The host list displays one entry: 'localhost'. Above the host list, there is a search bar with a dropdown set to 'Name', a magnifying glass icon, an 'Add' button, a prominent blue 'Run Command' button, and a 'Delete' button. To the right of the host list, there is an 'Actions' column with a toggle switch labeled 'On' and an edit icon. At the bottom of the host list, there is a pagination area showing '1-1 of 1 items' and navigation icons.

# Lab Time

## Exercise 2 - Ad-hoc commands

This lab guides you through executing ad-hoc commands from Automation Controller



Approximate time: [15 mins](#)

---

# Playbooks

# 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: "{{ service\_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: "{{ service_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: "{{ service\_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: "{{ service\_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: "{{ service\_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: "{{ service\_name }}"
      - state: started

# Lab Time

## Exercise 3 - Intro to playbooks

In this lab you'll author your first playbook

## Exercise 4 - Configure a job template

This lab guides you through creating a job template from an existing project



Approximate time: 25 mins

---

# Advanced playbooks

# 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"
```

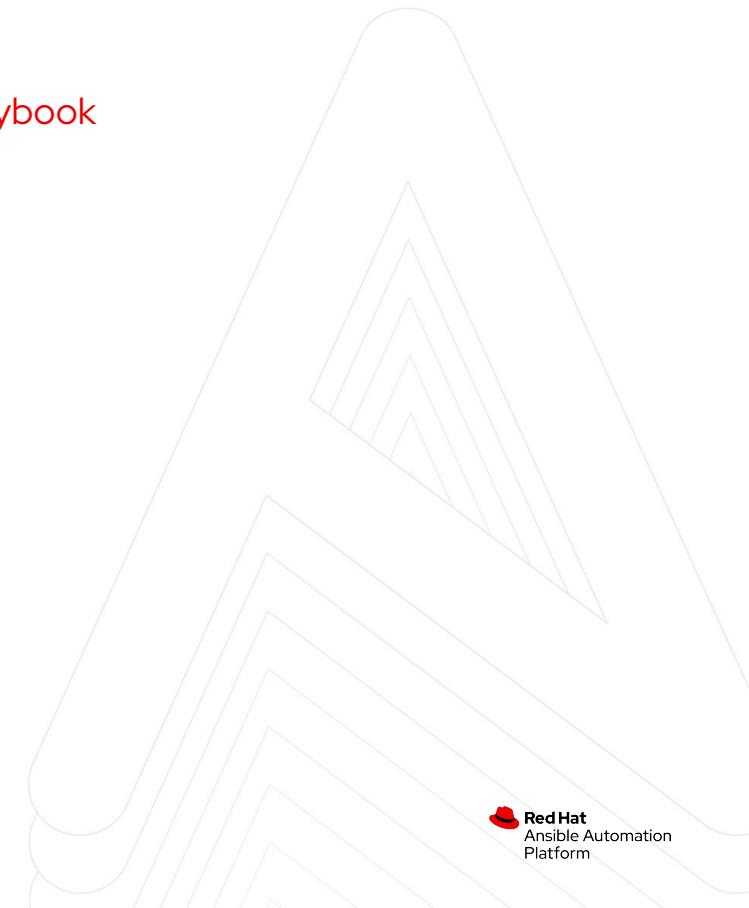
# Lab Time

## Exercise 5 - More advanced playbook

This lab expands on the existing playbook



Approximate time: 15 mins



---

# Sharing automation

# 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.

# Lab Time

## Exercise 6 - Ansible roles

In this lab you will convert your existing automation into roles that can be reused as a part of larger automated workflows



Approximate time: 15 mins



# Where to go next

## Learn more

- ▶ [Workshops](#)
- ▶ [Documents](#)
- ▶ [Youtube](#)
- ▶ [Twitter](#)

## Get started

- ▶ [Evals](#)
- ▶ [cloud.redhat.com](#)

## Get serious

- ▶ [Red Hat Automation Adoption Journey](#)
- ▶ [Red Hat Training](#)
- ▶ [Red Hat Consulting](#)

# Thank you

Red Hat is the world's leading provider of enterprise open source software solutions.

Award-winning support, training, and consulting services make

Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



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



[twitter.com/RedHat](https://twitter.com/RedHat)