



# Writing your first Ansible playbook

Roger Lopez  
Principal Tech Marketing  
Manager

Anshul Behl  
Principal Tech Marketing  
Manager



## **Red Hat** **Ansible Automation** **Platform**

# What you will learn

- ▶ Basics & Creating an inventory
- ▶ Writing your First Playbook
- ▶ How to run an Ansible Playbook
- ▶ How to extend an Ansible Playbook
- ▶ Running Ansible playbooks on multiple hosts
- ▶ Using variables within your Ansible Playbook
- ▶ How to use Loops
- ▶ How to incorporate Templates
- ▶ How to create a Role

# Ansible Inventory

The systems that a playbook runs against



## What are they?

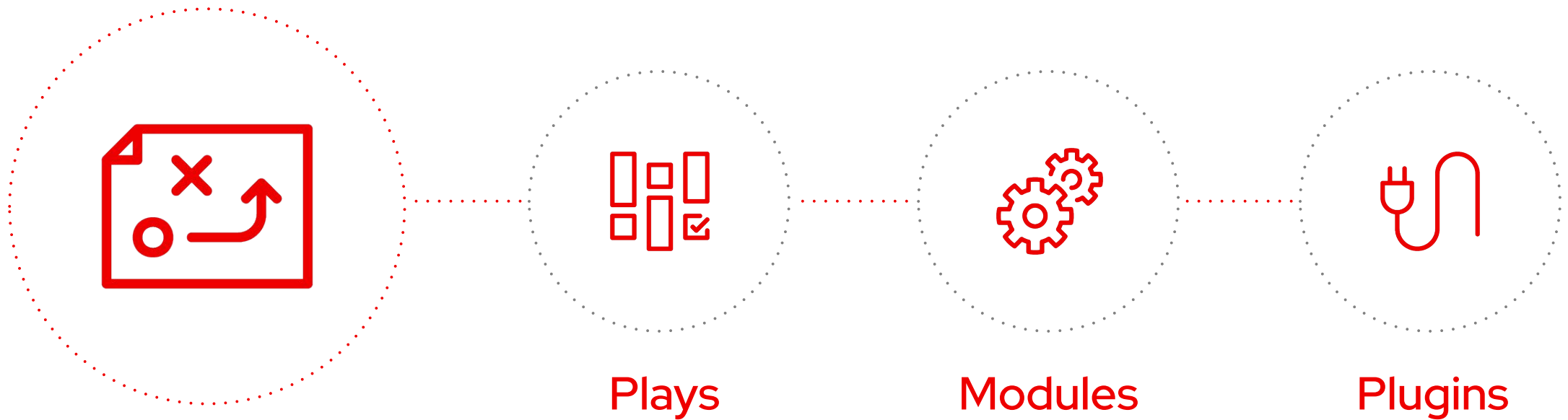
List of systems in your infrastructure that automation is executed against

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

[db]
dbserver1.example.com

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

# 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.

```
---  
- name: install and start apache  
  hosts: web  
  become: yes
```

# 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.



```
- name: latest index.html file ...  
  template:  
    src: files/index.html  
    dest: /var/www/html/
```



```
---
- 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
```

# Running an Ansible Playbook

Using the latest ansible-navigator command



## What is ansible-navigator?

ansible-navigator command line utility and text-based user interface (TUI) for running and developing Ansible automation content.

It replaces the previous command used to run playbooks "ansible-playbook".

A stylized illustration of a computer monitor. The screen is black with a white border and contains the command '\$ ansible-navigator run playbook.yml' in white text. The monitor has a white base and is set against a background of light gray clouds with red outlines.

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



# ansible-navigator

Bye ansible-playbook, Hello ansible-navigator




## How do I use ansible-navigator?

As previously mentioned, it replaces the ansible-playbook command.

As such it brings two methods of running playbooks:

- ▶ Direct command-line interface
- ▶ Text-based User Interface (TUI)



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

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

# Working with Variables



```
---  
- name: variable playbook test  
  hosts: localhost  
  
  vars:  
    var_one: awesome  
    var_two: ansible is  
    var_three: "{{ var_two }}"  
  
  tasks:  
    - name: print out var_three  
      debug:  
        msg: "{{ var_three }}"
```



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

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }}" "{{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{ var_three }}"
```

ansible is awesome

# Conditionals via VARS

Example of using a variable labeled *my\_mood* and using it as a conditional on a particular task.

```
vars:
  my_mood: happy

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



```
---  
- name: variable playbook test  
  hosts: localhost  
  
  vars:  
    my_mood: happy  
  
  tasks:  
    - name: task, based on my_mood var  
      debug:  
        msg: "Yay! I am {{ my_mood }}!"  
      when: my_mood == "happy"
```

## Alternatively

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



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

  tasks:
  - name: Install httpd
    yum:
      name: httpd
      state: latest
      when: ansible_distribution == 'RedHat'

  - name: Install apache
    apt:
      name: apache2
      state: latest
      when: ansible_distribution == 'Debian' or
            ansible_distribution == 'Ubuntu'
```



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

  tasks:
    - name: Ensure httpd package is present
      yum:
        name: httpd
        state: latest
        register: httpd_results

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





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

  tasks:
    - name: Ensure httpd package is present
      yum:
        name: httpd
        state: latest
      notify: restart_httpd

  handlers:
    - name: restart_httpd
      service:
        name: httpd
        state: restarted
```

**tasks:**

- name: Ensure httpd package is present
  - yum:
    - name: httpd
    - state: latest
    - notify: restart httpd
- name: Standardized index.html file
  - copy:
    - content: "This is my index.html file for {{ ansible\_host }}"
    - dest: /var/www/html/index.html
    - notify: restart httpd

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

TASK [Ensure httpd package is present] \*\*\*\*\*

ok: [web2] **unchanged**  
ok: [web1]

TASK [Standardized index.html file] \*\*\*\*\*

changed: [web2] **changed**  
changed: [web1]

NOTIFIED: [restart\_httpd] \*\*\*\*\*

changed: [web2]  
changed: [web1]

**handler runs once**

**tasks:**

- name: Ensure httpd package is present
  - yum:
    - name: httpd
    - state: latest
    - notify: restart httpd
- name: Standardized index.html file
  - copy:
    - content: "This is my index.html file for {{ ansible\_host }}"
    - dest: /var/www/html/index.html
    - notify: restart httpd

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

TASK [Ensure httpd package is present] \*\*\*\*\*

changed: [web2] **changed**  
 changed: [web1]

TASK [Standardized index.html file] \*\*\*\*\*

changed: [web2] **changed**  
 changed: [web1]

NOTIFIED: [restart\_httpd] \*\*\*\*\*

changed: [web2]  
 changed: [web1]

**handler runs once**

# Ansible roles

## Reusable automation actions












### What are they?

Group your tasks and variables of your automation in a reusable structure. Write roles once, and share them with others who have similar challenges in front of them.



```
---  
- name: install and start apache  
  hosts: web  
  roles:  
    - common  
    - webserver
```

# Visit [summit.demo.redhat.com](https://summit.demo.redhat.com)

|   |  |   |
|---|--|---|
| <br>Using GitOps to control and customize your Red Hat OpenShift clusters and applications<br>May 24, 10:30 AM<br><a href="#">Access this lab →</a>                      | <br>Red Hat OpenShift Virtualization: Migrate and integrate with front-end cloud-native applications<br>May 24, 10:30 AM<br><a href="#">Access this lab →</a> | <br>Hands-On Lab: Building secure containerized applications with Azure Red Hat OpenShift<br>May 24, 10:30 AM<br><a href="#">Access this lab →</a> |
| <br>Red Hat OpenShift and OpenShift application services: the ultimate combination for cloud-native development<br>May 24, 10:30 AM<br><a href="#">Access this lab →</a> | <br>Introduction to Event-Driven Ansible and GitOps automation<br>May 24, 11:00 AM<br><a href="#">Access this lab →</a>                                       | <br>Getting started with Red Hat Ansible Automation Platform and ServiceNow<br>May 24, 11:00 AM<br><a href="#">Access this lab →</a>               |
| <br>Enter serverless functions journey with Quarkus<br>May 24, 11:00 AM<br><a href="#">Access this lab →</a>   | <br>Writing your first Ansible Playbook<br>May 24, 1:00 PM<br><a href="#">Access this lab →</a>   | <br>Accelerate new application development with Red Hat OpenShift<br>May 24, 1:00 PM<br><a href="#">Access this lab →</a>                        |

# Learning resources

Continue your automation journey with Red Hat Ansible for public cloud automation



## Ansible Automation Labs

[red.ht/ansible\\_labs](https://red.ht/ansible_labs)

## E-book:

### An IT executive's guide to automation

[red.ht/automate\\_guide](https://red.ht/automate_guide)

## Ansible Basics:

### Automation Technical Overview

[red.ht/automation\\_basics](https://red.ht/automation_basics)



# 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.



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



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



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



[youtube.com/AnsibleAutomation](https://youtube.com/AnsibleAutomation)