One Command for All Server Environment Settings

Allen

Content

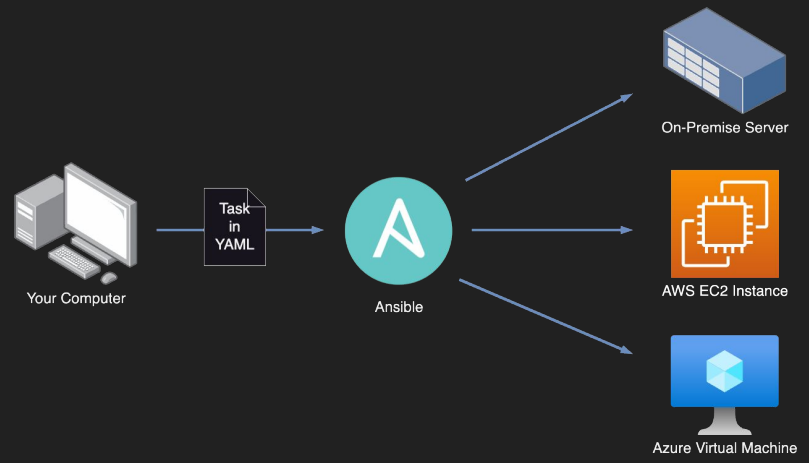
- What is Ansible?
- Inventory
- Modules
- Playbooks
- Idempotence
- Variables
- Conditionals
- Roles
- Demo

What is Ansible?

Ansible is a well-known IaC (Infrastructure as Code) tool

Unlike Terraform, which is used to build resources, Ansible is primarily used to configure the environment on the machine

Ansible is **Agentless**, you don't need to install any agent on the remote host to operate the remote host



Inventory

Inventory is a file used to define host connection information that Ansible refers to and connects to the remote host using SSH

Group

Host

```
# ansible inventory file in ini format
[web]
blog ansible_host=211.72.129.103 ansible_user=ubuntu
forum ansible_host=211.72.129.104 ansible_user=ubuntu

[database]
mysql ansible_host=211.72.129.105 ansible_user=ubuntu
```

P.S. Use snake_case for Group and Host names. ex. hello_world、web_server

Inventory in YAML

Besides INI, Inventory can also be written in YAML format

```yaml
# ansible inventory file in yaml format
web:
  hosts:
    blog:
      ansible_host: 211.72.129.103
      ansible_user: ubuntu
    forum:
      ansible_host: 211.72.129.104
      ansible_user: ubuntu
database:
  hosts:
    mysql:
      ansible_host: 211.72.129.105
      ansible_user: ubuntu
```

Using SSH Config with Inventory

Inventory can be used with `~/.ssh/config`

```
Host blog
    User ubuntu
    Hostname 211.72.129.103
    Port 22
    IdentityFile ~/.ssh/id_ed25519.pub

Host forum
    User ubuntu
    Hostname 211.72.129.103
    Port 22
    IdentityFile ~/.ssh/id_ed25519.pub

Host mysql
    User ubuntu
    Hostname 211.72.129.105
    Port 22
    IdentityFile ~/.ssh/id_ed25519.pub
```

```
# with ssh config file
web:
  hosts:
    blog:
    forum:
database:
  hosts:
    mysql:
```

Use Ansible to Do a Connection Test

Set up the Inventory and let's test it with the commands

- `all`: Specify the name of the target to connect to, either Group or Host. here `all` stands for all host
- `-i`: Specify the Inventory file to use
- `-m`: Specify the Ansible Module to use

```
ansible all -i inventory.yaml -m ping
```

Module

Ansible has a number of different modules that can be used, such as the ping module that we just used

And many other modules …

- apt : Managing apt packages on the host (for Ubuntu or Debian)
- command : Execute commands on the host
- file : Managing the contents of files on the host
- fetch : Pulling files from the host to the local machine

Playbooks

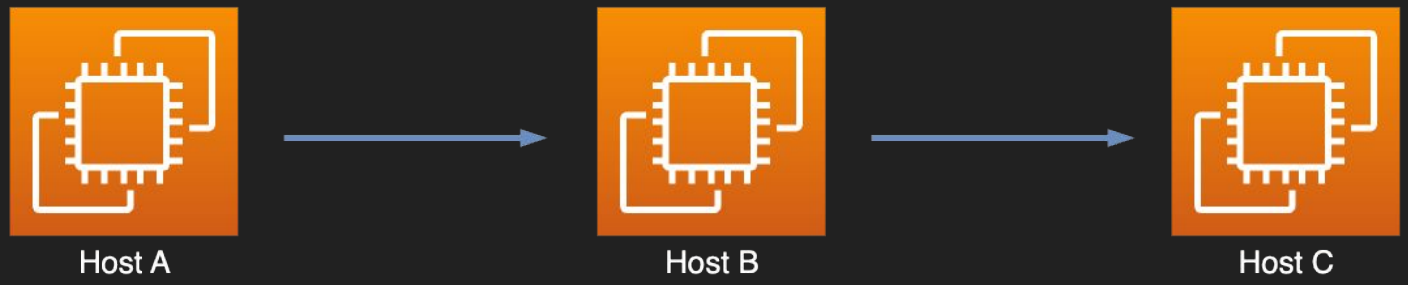
Playbooks allows you to run tasks on multiple hosts

You can even run **sequential tasks** on multiple hosts (A for B, B for C)

- Install Mysql
- Update the security settings
- Allow 3306 TCP Port in Firewall

- Install Laravel App
- Update Laravel settings
- Create the Database Schema

- Install Nginx
- Update the Nginx Config
- Allow 443 TCP Port in Firewall



Playbooks Example

Playbooks in YAML format

hosts : Host or Group as defined in Inventory

name : Task descriptions, Ansible Lint will require capitalization at the beginning.

module : For modules, Ansible Lint requires that they be written using fqcn (fully-qualified collection names).

parameter : Parameters of the module

```yaml
- name: Install nginx and enable it
  hosts: webservers
  become: true
  tasks:
    - name: Install nginx
      ansible.builtin.apt:
        name: nginx
        state: present
    - name: Start the nginx service ans enable it
      ansible.builtin.service:
        name: nginx
        state: started
        enabled: true
```

Execute Playbooks Task

Use ansible-playbook to run your written YAML file.

```
ansibe-playbook playbooks.yaml -i inventory.yaml
```

Don’t Specify the Inventory Every Time.

You can create an `ansible.cfg` file. This is the configuration file for Ansible, and there are a number of settings that you can set up

if you set the location of the Inventory file, you don't need to enter the command `-i inventory.yaml` afterward

```
[defaults]
inventory = ./inventory.yaml
```

Idempotence

Executing the same action one or more times will result in the same state for the object being executed

Idempotence is an important concept when writing tasks.

```yaml
# it's ok to execute it multiple times
# this task will make sure the file "./foo.txt" is absent on the machine
- name: Install nginx and enable it
  hosts: localhost
  tasks:
    - name: Remove file (delete file)
      ansible.builtin.file:
        path: ./foo.txt
        state: absent
```

```yaml
# each execute will append a line to the file
# so it's not an idempotent task
- name: Append a line to a file
  hosts: localhost
  tasks:
    - name: Add a line to a file
      ansible.builtin.shell:
        cmd: echo "hello world" >> ./foo.txt
      changed_when: true
```

Variables

You can declare a variable in playbooks so that subsequent tasks use it.

After defining the variable with `vars`, you can use the Jinja2 Filter `"{{ }}"` to convert the variable to the value you set.

```yaml
- name: Show how to use vars
  hosts: localhost
  vars:
    say_hello: 'hello'
  tasks:
    - name: Use debug module to show the vars
      ansible.builtin.debug:
        msg: "{{ say_hello }} Howard"
```

You Can Set Up Variables in Inventory.

Variables can be set in the Inventory, which is divided into Group Vars and Host Vars.

Group Vars will act on all Hosts under the genus.

```
[atlanta]
host1 http_port=80
host2 http_port=303

[atlanta:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com
```

Host Vars

Group Vars

```
atlanta:
  hosts:
    host1:
      http_port: 80
    host2:
      http_port: 303

  vars:
    ntp_server: ntp.atlanta.example.com
    proxy: proxy.atlanta.example.com
```

How to Use Variables in Inventory

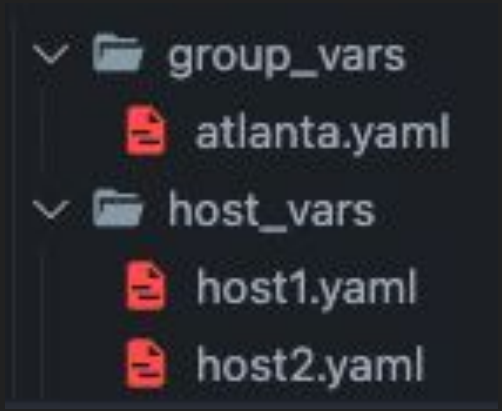
Accessing Group Vars and Host Vars with Ansible's `hostvars`

```yaml
- name: Show how to use vars in inventory
  hosts: host1
  tasks:
    - name: Print the host vars 'http_port'
      ansible.builtin.debug:
        msg: "{{ hostvars['host1']['http_port'] }}"

    - name: Print the group vars 'proxy'
      ansible.builtin.debug:
        msg: "{{ hostvars['host1']['proxy'] }}"
```

Organizing Your Variables with Folders

You can add a new `group_vars` and `host_vars` folder, and create a YAML file with the same name as Host or Group underneath it, and then write the variables in it.




```yaml
# host1.yaml
http_port: 80
```

Register Variables

You can register the output of a task as a new variable using `register`

```yaml
- name: Show how to use register
  hosts: localhost
  tasks:
    - name: Register a output of a command
      ansible.builtin.command:
        cmd: "echo 'Hello Tavia!'"
      changed_when: false
      register: result

    - name: Print the registered variable
      ansible.builtin.debug:
        msg: "{{ result.stdout }}"
```

Conditional

You can use the control flow in playbooks

`when` `loop`

when

When the given conditional equation holds, the task is executed.

```yaml
- name: Show how to use when
  hosts: proxy
  tasks:
    - name: Install Nginx on Debian
      ansible.builtin.apt:
        name: nginx
        state: present
      when: ansible_os_family == "Debian"

    - name: Install Nginx on RedHat
      ansible.builtin.yum:
        name: nginx
        state: present
      when: ansible_os_family == "RedHat"
```

loop

Bring in the values in the array and repeat the task

```yaml
- name: Show how to use loop
  hosts: localhost
  tasks:
    - name: Create users
      loop:
        - Johnson
        - Howard
        - Elizabeth
        - Peggy
        - Tavia
      ansible.builtin.user:
        name: "{{ item }}"
        state: present
```

```yaml
- name: Show how to use loop
  hosts: localhost
  tasks:
    - name: Create users
      loop:
        - name: Johnson
          uid: 1001
        - name: Howard
          uid: 1002
        - name: Elizabeth
          uid: 1003
      ansible.builtin.user:
        name: "{{ item.name }}"
        uid: "{{ item.uid }}"
        state: present
```

Role

You can modularize your tasks for reuse, similar to Terraform's Modules.

Assuming you have a playbooks installation of MySQL, you can package it as a Role, and when you later need to install MySQL on another machine, you can use this Role directly without writing it again

```
# create a new role 'say_hello'
ansible-galaxy init say_hello
```

Role's File Structure

You can place the Role under the roles folder in the same directory as the Playbooks file

You can also place the Role in the `/etc/ansible/roles` folder

```
.
├── playbooks.yaml
└── roles
    └── say_hello
        ├── README.md
        ├── defaults
        │   └── main.yml
        ├── files
        ├── handlers
        │   └── main.yml
        ├── meta
        │   └── main.yml
        ├── tasks
        │   └── main.yml
        ├── templates
        ├── tests
        │   ├── inventory
        │   └── test.yml
        └── vars
            └── main.yml
```

```yaml
---
# roles/say_hello/tasks/main.yml
# tasks file for say_hello
- name: Say hello
  ansible.builtin.debug:
    msg: "Hello!"

```

```yaml
# playbooks.yaml
- name: Show how to use role
  hosts: localhost
  roles:
    - say_hello
```

Collections

It's also a way to reuse code

Collections contain playbooks, roles, modules, and plugins

Unlike roles, which are tasks, collections describe a more complete IT infrastructure

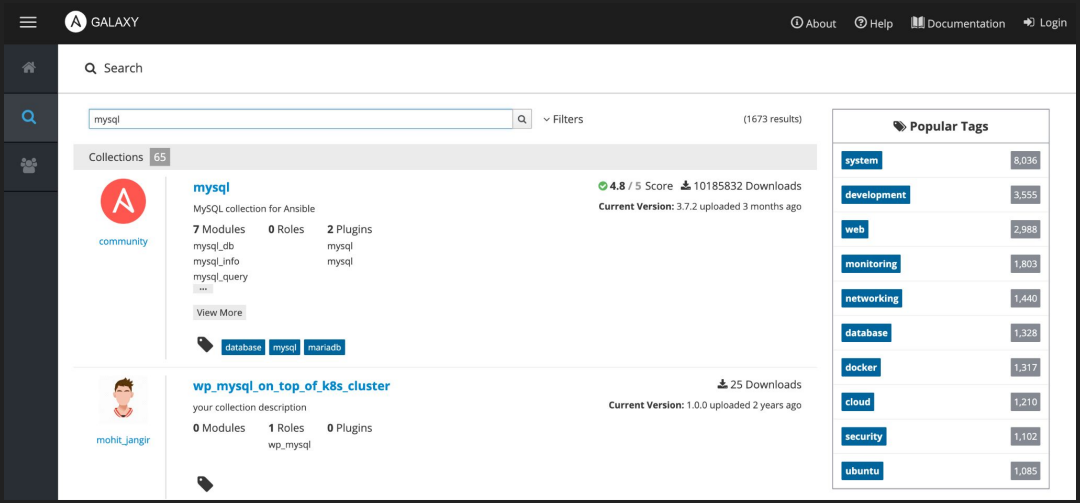
```yaml
- name: Reference a collection content using its FQCN
  hosts: all
  tasks:

    - name: Call a module using FQCN
      my_namespace.my_collection.my_module:
        option1: value
```

Ansible Galaxy

Before you start writing a new project, you can look on Ansible Galaxy to see if there is a role or collection you need.

You can also share your Role on the web for others to use.



Follow Ansible's Best Practice

The official documentation lists a number of recommendations for using Ansible and is highly recommended.

Best Practices

Here are some tips for making the most of Ansible and Ansible playbooks.

You can find some example playbooks illustrating these best practices in our ansible-examples repository. (NOTE: These may not use all of the features in the latest release, but are still an excellent reference!).

After We Talk About Ansible

I just want to say …

Long Live the IMMUTABLE !!!