Ansible Basic

An Ansible Training Course

DevOps Artisan

2. Running Ad Hoc Commands

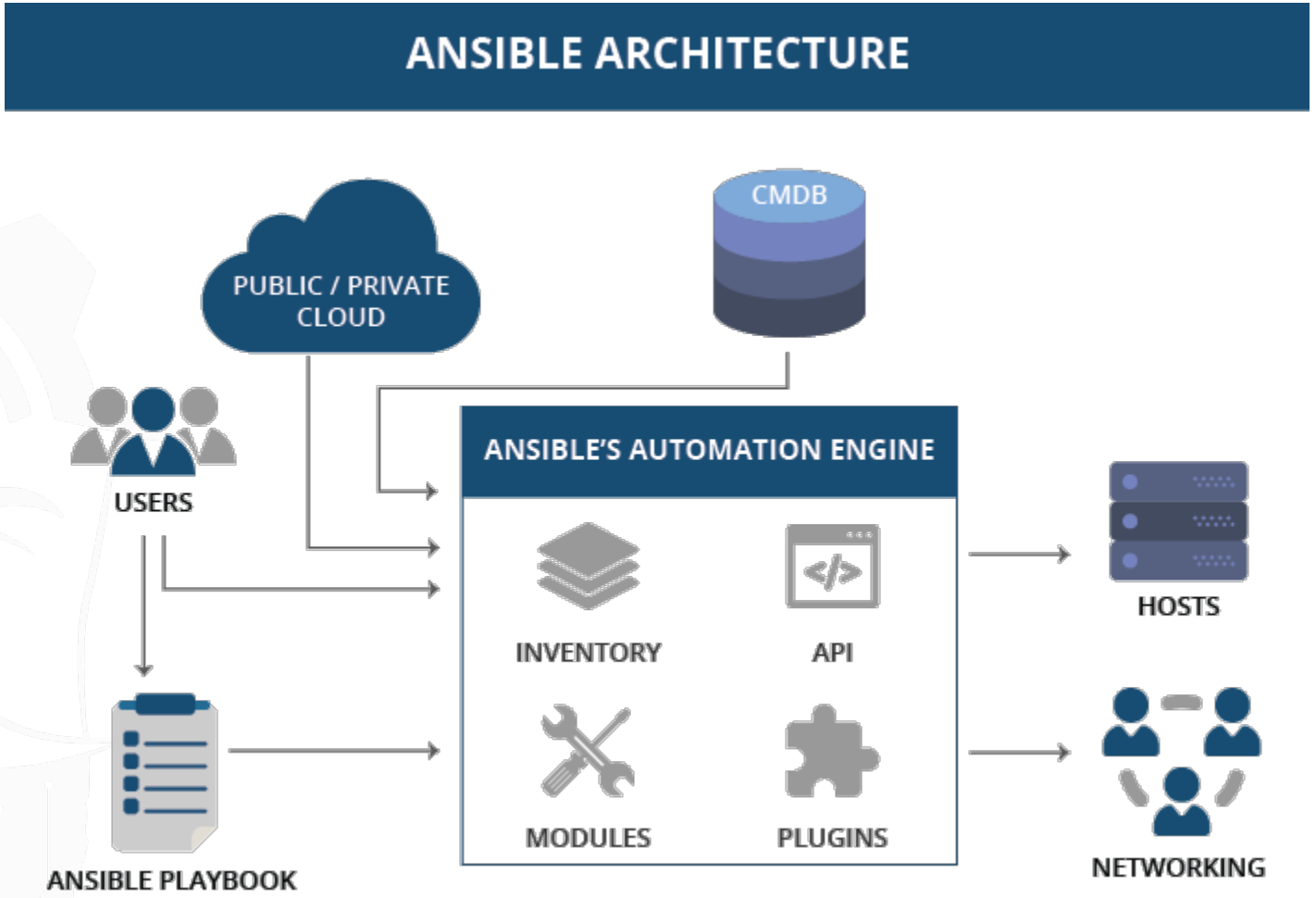
DevOps Artisan

Topics covered

- How Ansible works
- Authenticating Ansible Connections

DevOps Artisan

How Ansible Works



DevOps Artisan

How Ansible Works (Automation Engine)

- Inventories are **lists of hosts** (also called nodes) - servers that **need to be managed**.

- There are two types:
  - Static inventory
  - Dynamic inventory

```
[webservers]
ubuntu
centos

[dbservers]
ubuntu
hivemaster

[datacenter:children]
webservers
dbservers
```

DevOps Artisan

How Ansible Works (Automation Engine)

- **Modules** are executed directly on remote hosts through playbooks.
- Modules can control system resources, like services, packages, or files, or execute system commands.
- They enable you to manage virtually *everything* that has an API, CLI, or configuration file you can interact with, including network devices like load balancers, switches, firewalls, container orchestrators, containers themselves.

How Ansible Works

ANSIBLE PLAYBOOK

- Playbooks are simple files written in YAML format which describe the configuration tasks to be applied

- Playbooks can declare configurations, but they can also orchestrate the steps of any manual ordered process.

DevOps Artisan

How Ansible Works

- Ansible can also be used to automate different networks.

- It uses a data model that is separate from the Ansible automation engine that easily spans different network hardware.

How Ansible Works

- The hosts in the Ansible architecture are just node systems which are being configured.

- They can be any kind of machine – Windows, Linux

How Ansible Works

- CMDB - a repository that acts as a data warehouse for IT installations.

- It holds data relating to a collection of IT assets (configuration items (CI)), as well as relationships between such assets.

Ansible Configuration File

- Ansible has a number of options that can be adjusted
- The default configuration file: **`/etc/ansible/ansible.cfg`**
- Notable configuration options include:
  - default inventory file location
  - default remote user
  - default authentication settings

Ansible Configuration File

- The configuration properties may be overridden using local files.

- The first configuration file found is used **and later files are ignored**.

- Configuration file search order:
  - **ANSIBLE_CONFIG** (environment variable if set)
  - **ansible.cfg** (in the current directory)
  - **~/.ansible.cfg** (in the home directory)
  - **/etc/ansible/ansible.cfg**

Ansible Configuration File

- As of version 2.4, Ansible has a new utility called **`ansible-config`**

- This utility allows users to see:
  - All the configuration setting available
  - Their defaults
  - How to set them
  - Where their current value comes from

Ansible inventory

- An inventory is a list of hosts that Ansible manages.

- The default Ansible Inventory File is **`/etc/ansible/hosts`**

- Inventory location may be specified as follows:
  - Default: **`/etc/ansible/hosts`**
  - On the command line: **`ansible -i <filename>`**
  - Can be set in **`ansible.cfg`**

Ansible inventory

- Inventory files can also use YAML format
- You may want to keep separate inventories for staging and production.

DevOps Artisan

Ansible inventory file example

```
mail.example.com ansible_port=5556 ansible_host=192.168.0.100

[webservers]
httpd1.example.com
httpd2.example.com


[labservers]
lab[02:05]
```

DevOps Artisan

Ansible inventory file example

this line defines a host

```
mail.example.com ansible_port=5556 ansible_host=192.168.0.100

[webservers]
httpd1.example.com
httpd2.example.com


[labservers]
lab[02:05]
```

DevOps Artisan

Ansible inventory file example

```
mail.example.com ansible_port=5556 ansible_host=192.168.0.100

[webservers]
httpd1.example.com
httpd2.example.com


[labservers]
lab[02:05]
```

Two variables are affiliated with this host

DevOps Artisan

Ansible inventory file example

```
mail.example.com ansible_port=5556 ansible_host=192.168.0.100

[webservers]
httpd1.example.com
httpd2.example.com


[labservers]
lab[02:05]
```

DevOps Artisan

Ansible inventory file example

```
mail.example.com ansible_port=5556 ansible_host=192.168.0.100

[webservers]
httpd1.example.com
httpd2.example.com

[labservers]
lab[02:05]
```

labservers group has 4 hosts defined

the expression lab[02:05] is the same as lab02, lab03, lab04, lab05

Ansible command

- Ansible ad-hoc commands allow us to quickly run a task against a remote system

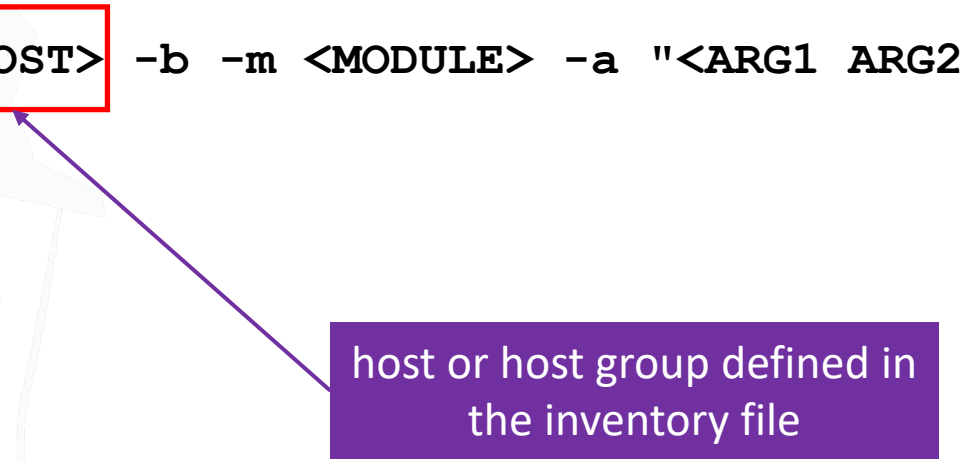
- Syntax:
  - `ansible <HOST> -b -m <MODULE> -a "<ARG1 ARG2 ARGN>" -f <NUM_FORKS>`

DevOps Artisan

Ansible command

- Ansible ad-hoc commands analogous to bash commands.

- Syntax:
  - `ansible <HOST> -b -m <MODULE> -a "<ARG1 ARG2 ARGN>" -f <NUM_FORKS>`



Ansible command

- Ansible ad-hoc commands analogous to bash commands.

- Syntax:
  - **`ansible <HOST> -b -m <MODULE> -a "<ARG1 ARG2 ARGN>" -f <NUM_FORKS>`**

- "become"
- Ansible escalates permission to `--become-user` using the method defined by `--become-method`

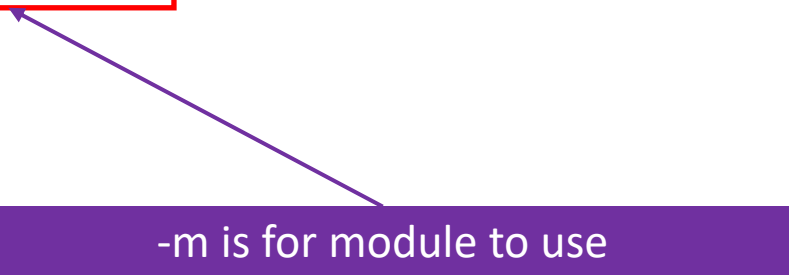
Default become-user is `root`.
Default become-method is `sudo`

DevOps Artisan

Ansible command

- Ansible ad-hoc commands analogous to bash commands.

- Syntax:
  - `ansible <HOST> -b -m <MODULE> -a "<ARG1 ARG2 ARGN>" -f <NUM_FORKS>`

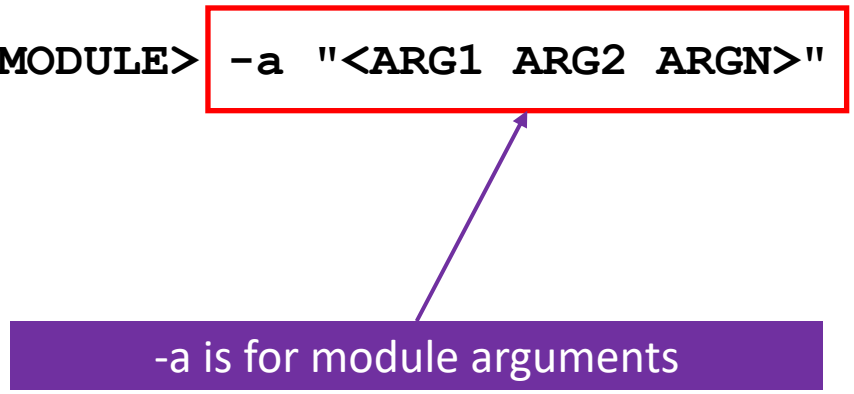


DevOps Artisan

Ansible command

- Ansible ad-hoc commands analogous to bash commands.

- Syntax:
  - `ansible <HOST> -b -m <MODULE> -a "<ARG1 ARG2 ARGN>" -f <NUM_FORKS>`



Note: If you do not specify "-m", the default module used will be "command"

DevOps Artisan

Ansible command

- Ansible ad-hoc commands analogous to bash commands.

- Syntax:
  - `ansible <HOST> -b -m <MODULE> -a "<ARG1 ARG2 ARGN>" -f <NUM_FORKS>`

-f is used to set forks for parallelism, which is how you can have Ansible execute tasks simultaneously on many hosts

Authenticating Ansible Connections

- You can run ad-hoc commands using:
  - password authentication
  - key-based authentication

DevOps Artisan

Authenticating Ansible Connections (password)

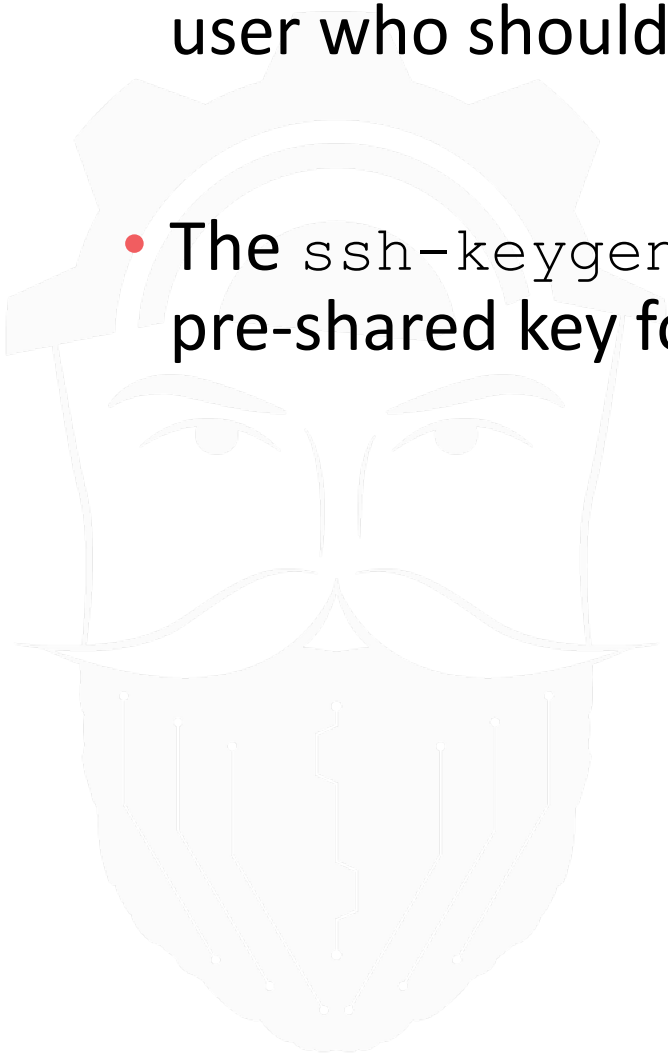
- To run ansible using password authentication:
  - `-u` for username
  - `--ask-pass` for prompting us to enter the account password

```
student:~$ ansible -i hosts all -m ping -u student --ask-pass
SSH password:
ansible-00-02-ubuntu | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Authenticating Ansible Connections (key-based)

- In everyday use, the best practice is to have a public key installed on the remote hosts and it is also recommended to create a dedicated user who should have access using that key.

- The `ssh-keygen` and `ssh-copy-id` command can facilitate creating a pre-shared key for user authentication.



DevOps Artisan

Authenticating Ansible Connections (key-based)

1. Generate a SSH keypair

```
student:~$ ssh-keygen -t rsa -b 2048 -f ansible_key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ansible_key.
Your public key has been saved in ansible_key.pub.
The key fingerprint is:
SHA256:gm+P8xHI5tl55qCJF88jmDj/9C4Z25rzLTzQSrZmO/4 student@ansible-00-01-hivemaster
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|                 |
|     o .         |
|     . =.S       |
|      +*+.o      |
|    . =*&= o     |
|   o o+#BOB      |
|    oo*X&E+o     |
+----[SHA256]-----+
```

DevOps Artisan

Authenticating Ansible Connections (key-based)

1. Generate a SSH keypair
   a. The previous command will generate 2 new files in the current directory

```
student:~$ ls | grep ansible
ansible_key
ansible_key.pub
```

   b. Add the newly-generated public key (ansible_key.pub) to the remote hosts

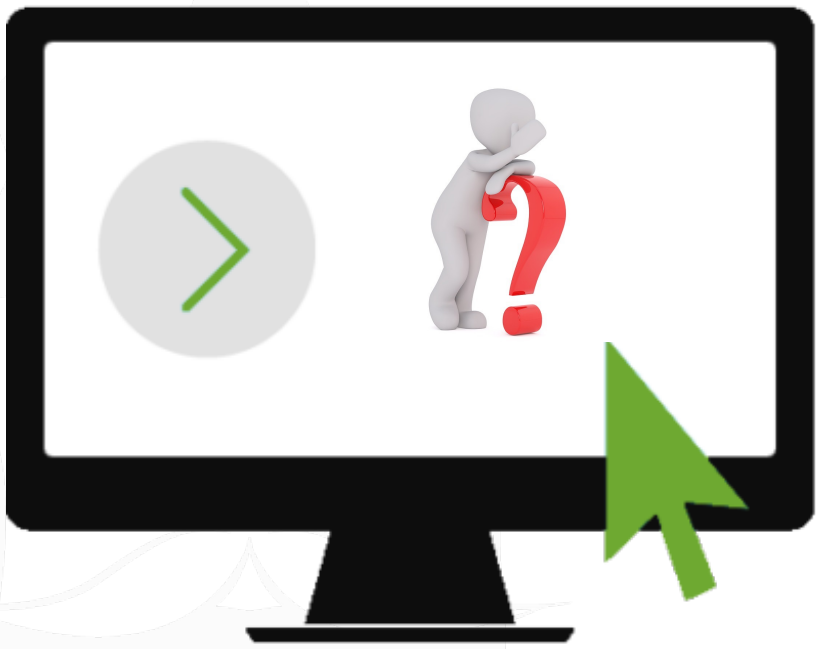
```
student:~$ cat ansible_key.pub >> /home/student/.ssh/authorized_keys
```

Authenticating Ansible Connections (key-based)

3. Ping all hosts using key-based auth

```
student:~$ ansible -i hosts all -m ping -u student --private-key
/home/student/ansible_key
ansible-00-02-ubuntu | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

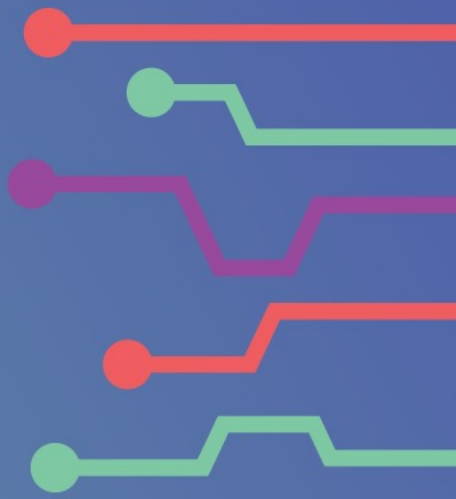
DevOps Artisan



DevOps Artisan

Lab 2: *Ad-Hoc commands*

DevOps Artisan



More practice, less theory

askformore@devopsartisan.com

DevOps Artisan