

# Ansible Basic

An Ansible Training Course

PRESENTED BY:

Bittnet DevOps Team

ANSIBLE

We Make Custom Trainings



FASTER  
SMARTER  
SAFER

# 3. Preparing Hosts for Ansible



# Topics covered:

- Creating a dedicated user
- Configuring key-based auth
- Default `remote_user` and `private_key` in Ansible
- Static host inventories (INI and YAML)



# Creating a dedicated user

- Ansible is best implemented using a common user across all Ansible controlled systems.
- Only a basic system user with ssh access is needed for Ansible to connect to a host.

# Creating a dedicated user

- The following commands (executed as root) will create a new user on a system called **ansible** and allow for the user password to be set:
  - `useradd ansible`
  - `passwd ansible`
    - Recommended: `passwd -l ansible`
- This step should be performed on every node that will be managed by Ansible
  - Repetitive and time-consuming, you say? Well... not necessarily... 😊

# Creating a dedicated user

- Or you can use Ansible to automate this for all hosts

```
student:~$ ansible -i hosts.ini all --ask-pass --become -u student -m user -a  
"name=ansible state=present"  
SSH password:  
BECOME password[defaults to SSH password]:  
ansible-00-02-ubuntu | CHANGED | rc=0 >>  
[...]
```

# Configuring key-based auth

- a. After you created the user, you have to setup SSH key-based authentication ( learned in previous lesson)
- b. Create `.ssh` folder in `/home/ansible`
  - owner/group **ansible:ansible** and **0700** permissions

```
sudo mkdir /home/ansible/.ssh
sudo chown ansible:ansible /home/ansible/.ssh
sudo chmod 700 /home/ansible/.ssh
```

# Configuring key-based auth

c. Copy public key to `/home/ansible/.ssh/authorized_keys`

```
student:~$ sudo -i
root:~# cat /home/student/ansible_key.pub >> /home/ansible/.ssh/authorized_keys
root:~# chown ansible:ansible /home/ansible/.ssh/authorized_keys
root:~# chmod 644 /home/ansible/.ssh/authorized_keys
```



# Configuring key-based auth

## d. Test key-based auth for 'ansible' user

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

## e. In order to make sure that the correct user was used, we can also run the "id" command

```
student:~$ ansible -i hosts all -a "id" -u ansible --private-key /home/student/ansible_key
ansible-00-02-ubuntu | CHANGED | rc=0 >>
uid=1004(ansible) gid=1005(ansible) groups=1005(ansible)
```

# Giving the User sudo permissions

- The '**ansible**' user should be able to run commands as root, without entering the password
- The recommended way to perform changes on the `sudoers` file is by using the `visudo` command
  - it performs a check before saving, so that you do not get locked out of the system

```
student:~$ sudo visudo /etc/sudoers.d/95-ansible
#add this line
ansible ALL=(ALL:ALL) NOPASSWD: ALL
```

# Default `remote_user` and `private_key` in Ansible

- When we tested the key-based authentication for our new '**ansible**' user, we specified the name and path to the private key.
- We can configure default values for these parameters in **`/etc/ansible/ansible.cfg`** in order to avoid writing them every time we run `ansible`.
- These settings will be applied both when running ad hoc commands, and when running playbooks (discussed later)

# Default remote\_user and private\_key in Ansible

```
student:~$ sudo vi /etc/ansible/ansible.cfg
```

```
#Uncomment and modify (or add) the following lines:  
remote_user = ansible  
private_key_file = /home/student/ansible_key
```



```
student:~$ ansible -i hosts all -m shell -a "whoami"  
ansible-00-02-ubuntu | CHANGED | rc=0 >>  
ansible
```

```
ansible-00-01-hivemaster | CHANGED | rc=0 >>  
ansible
```

```
10.142.15.213 | CHANGED | rc=0 >>  
ansible
```

# Default remote\_user and private\_key in Ansible

- In the same file, **ansible.cfg**, we can also set (enable) some other parameters like **become**, **become\_user**, **become\_method** in **privilege\_escalation** section

```
[privilege_escalation]
#become=True
#become_method=sudo
#become_user=root
#become_ask_pass=False
```

# Default remote\_user and private\_key in Ansible

- Notice that if we set **become=True** all the tasks will be executed as **become\_user** (root by default):

```
student@ansible-00-01-hivemaster:~$ ansible -i hosts all -m shell -a "whoami"  
ansible-00-02-ubuntu | CHANGED | rc=0 >>  
root
```

```
ansible-00-01-hivemaster | CHANGED | rc=0 >>  
root
```

```
10.142.15.213 | CHANGED | rc=0 >>  
root
```

# Static host inventories

- In a minimal form, a static inventory is a list of host names and IP addresses that can be managed by Ansible.
- Host can be placed into groups to make it easy to address multiple hosts at once.
- A host can be a member of multiple groups.
- Nested groups are also available.

# Static host inventories

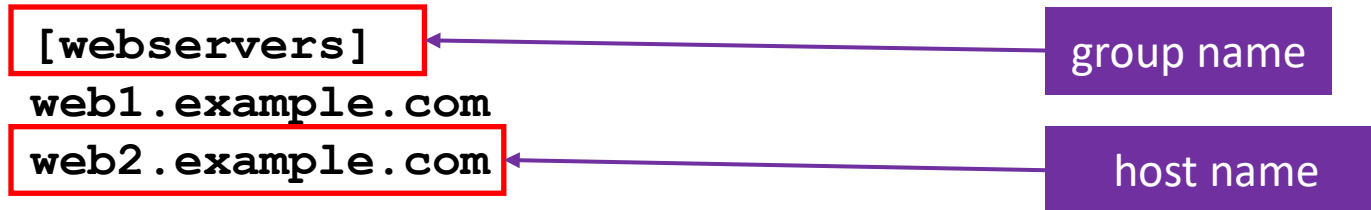
- It is common to work with project-based inventory files.
- Variables can be set from the inventory file – but this does not scale very well
- Ranges can be used:
  - `server[1:20]` matches server 1 up to server 20
  - `192.168.[4:5].[0:255]` matches two full class C subnets
- The most common formats are INI and YAML



# Inventory File Locations

- **/etc/ansible/hosts** is the default inventory.
- Alternative inventory location can be specified through the **ansible.cfg** configuration file
- Or use the **-i inventory** option to specify the location of the inventory file to use.
- It is common practice to put the inventory file in the current project directory.

# Static Inventory Example – INI format



```

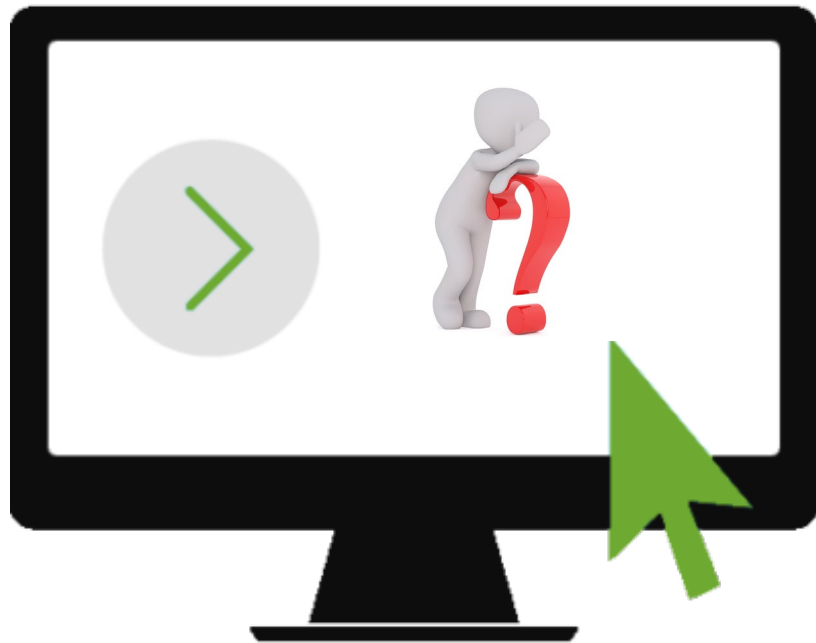
[webservers]
web1.example.com
web2.example.com

[fileservers]
file1.example.com
file2.example.com

[server:children]
webservers
fileservers
  
```

# Static Inventory Example – YAML format

```
webservers:
  hosts:
    ubuntu:
      ansible_host: 10.128.0.20
      # We have not yet created ansible user here
      ansible_user: student
    centos:
  vars:
    type: webserver
dbservers:
  hosts:
    ubuntu:
    hivemaster:
datacenter:
  children:
    webservers:
    dbservers:
```



# Lab 3: Preparing hosts for Ansible





Solutions for training the world.