**Ansible Foundation Workshop Hands-On Guide**

This article is a step-by-step guide for the hands-on session included in the Ansible Foundation Workshop conducted by ESD PSS Infra department. This hands-on session serves to teach you how to write simple Ansible Playbooks. The knowledge gained shall build the foundation necessary to utilise Ansible's various features and modules for complex use cases.

**1. Setup Ansible**

Ansible can be used to configure any target system that supports the SSH protocol and Python 2/3 library. However, Ansible's control node can only run on Unix OS (Ubuntu, MacOS, Red Hat, etc.). For this workshop, an Ubuntu 18.04 VM with Ansible is already prepared for you.

**1.1 (Optional) Install Ansible**

To install Ansible on an Ubuntu 18.04 VM, run the following commands as a sudo user:

sudo apt-add-repository ppa:ansible/ansible

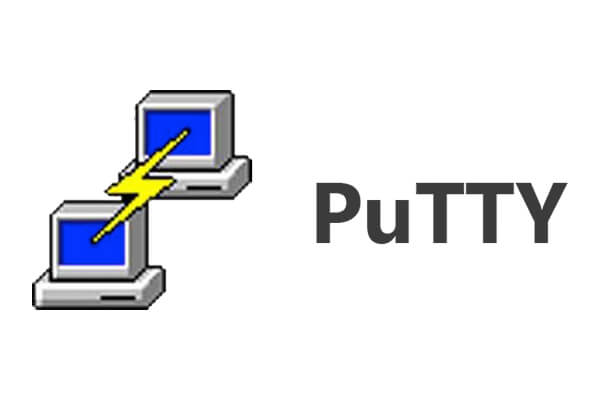
sudo apt update

sudo apt install ansible

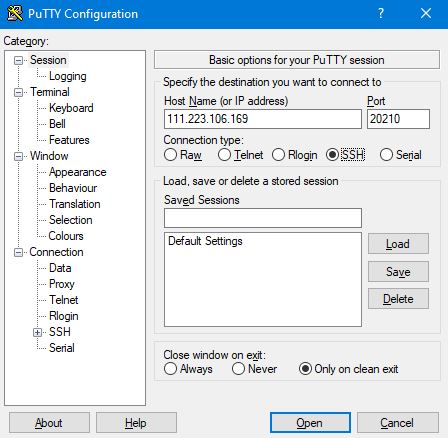
**2. Access Ansible Control Node Remotely**

**2.1 For Windows PC**

1. Install software PuTTY

[](https://github.com/CloudCommandos/shares/blob/master/Ansible%20Workshop/images/PuTTY.jpg)

1. Run PuTTY and start a session with the following settings:
   * Host Name: 111.223.106.169
   * Port: 20210
   * Connection type: SSH

[](https://github.com/CloudCommandos/shares/blob/master/Ansible%20Workshop/images/PuTTY_Interface.jpg)

1. In the Terminal that appears, login using the following credentials (replace 'X' with the number assigned to you):
   * Username: userX
   * Password: userX

**2.2 For Ubuntu/Debian PC**

1. Access the command console
2. Update Package list

sudo apt update

1. Install OpenSSH server

sudo apt install openssh-server

1. SSH to the target VM (replace 'X' with the number assigned to you). Password is the same as the username.

ssh userX@111.223.106.169 -p 20210

**3. Hands-On Exercises**

Before the start of the hands-on exercise, run the following command to create the ansible-workshop directory to work in (Replace 'X' with the number assigned to you):

mkdir /home/userX/ansible-workshop

cd /home/userX/ansible-workshop

**3.1 Exercise 1 - Introduction to Ansible**

**3.1.1 Part One - Basic Introduction**

Exercise 1 part 1 will guide the user through some basic scripting in the Ansible environment using the debug module which is similar to a printing module. The main intention is to get the user to familiarize with the Ansible environment before proceeding to the next exercise.

1. Create the playbooks directory and change your current directory to it.
2. mkdir playbooks

cd playbooks

1. Create exercise 1 part one task in main-script.yaml

Edit or create main-script.yaml file

nano main-script.yaml

Add the following task script into main-script.yaml (Replace 'X' with the number assigned to you).

- name: Exercise 1 Part One - Introduction to Ansible

hosts:

targetnodeX

tasks:

- name: Print Task 1

debug:

msg: "This is Task 1"

- name: Print Task 2

debug:

msg: "This is Task 2"

- name: Print Task 3

debug:

msg: "This is Task 3"

Once everything is filled up, save the file by pressing ctrl-o then enter. Exit the file by pressing ctrl-x.

1. Create the ansible host file for the task

nano inventory.yaml

Add your target node hostname/ip address into the file (Replace 'X' with the number assigned to you)

targetnodeX

or

10.0.1.21X

1. To run the ansible script, use the following command:

ansible-playbook -ki inventory.yaml main-script.yaml

-k : prompt user to input ssh password

-i : to pass in the inventory file

**3.1.2 Part Two - Introduction to Roles & Tags**

Exercise 1 part two will focus on the introduction of roles and tags module. One of the best practices is to keep the Ansible environment well organised. This can be achieved through the use of roles module and file directories to organise all the tasks instead of having all contents inside a single main script. If the project uses alot of tasks, the main script will look very messy and hard to maintain. Using the roles module, users can split up all the tasks in the main script into individual roles and store the actual tasks in the roles directory, thus keeping the main script neat and tidy.

The tags module can be used to provide tagging to each individual role task or the whole task. By default Ansible will run all the tasks inside the main script, however with the use of tags module the user can specify which tasks to run instead of all the tasks.

1. Create exercise 1 part two task in main-script.yaml

nano main-script.yaml

Add the following into main-script.yaml below exercise 1 part Once task (Replace 'X' with the number assigned to you)

- name: Exercise 1 Part two - Introduction to roles & tags

hosts:

targetnodeX

roles:

- role: task-1

- role: task-2

- role: task-3

tags: ex-1-2

Also add in the following tag at the end of exercise 1 part one task (take note of the spacing)

tags: ex-1-1

Save the main-script.yaml file

1. Create the roles directory

mkdir roles

1. Create the task folder in the roles directory
2. mkdir -p roles/task-1/tasks
3. mkdir -p roles/task-2/tasks

mkdir -p roles/task-3/tasks

1. Create the main.yaml file in the task folder

nano roles/task-1/tasks/main.yaml

Fill in the main.yaml file with the following script

- name: Print Task 1

debug:

msg: "This is Task 1"

Repeat this step for task-2 and task-3, replacing the number as required

1. To run exercise 1 part two task, use the following command:

ansible-playbook -ki inventory.yaml main-script.yaml --tags=ex-1-2

--tags: run the task with the tag specified

--skip-tags : skip the task with the tag specified

**3.2 Exercise 2 - Using Variables & Loop function**

Exercise 2 will focus on the introduction of variables in Ansible tasks and also the loop module. This exercise will guide the user through the creation of the variable directory and also the playbook syntax to call for the variables inside tasks. The exercise will also focus on the loop module, which is very useful when the user is required to pass in arrays of values into tasks.

1. Create exercise 2 task in main-script.yaml

nano main-script.yaml

Add the following into main-script.yaml

- name: Exercise 2 - Using Variables & Loop function

hosts:

managed\_node

roles:

- role: single-variable

- role: multiple-variable

tags: ex-2

1. Edit the ansible host file to use group names

nano inventory.yaml

Add your target node hostname/ip address into the file (Replace 'X' with the number assigned to you)  
The [managed\_node] is an arbitrary host group name that collectively represents all the hosts under it

[managed\_node]

targetnodeX

1. Create the variable file

First create the group variable directory

mkdir group\_vars

Next create the variable file

nano group\_vars/all.yaml

Fill in the all.yaml files with the following variables

required\_package: openssh-server

install\_package:

- vim

- python3

- nginx

1. Create exercise 2 task folders in the roles directory
2. mkdir -p roles/single-variable/tasks

mkdir -p roles/multiple-variable/tasks

1. Create the main.yaml file for both tasks

For single-variable task:

nano roles/single-variable/tasks/main.yaml

Fill in the main.yaml file of single-variable task with the following script

- name: Printing single variable from group\_vars folder

debug:

msg: "Required to install '{{required\_package}}' module"

For multiple-variable task:

nano roles/multiple-variable/tasks/main.yaml

Fill in the main.yaml file of multiple-variable task with the following script

- name: Printing single variable from group\_vars folder

debug:

msg: "Please install {{item}} module"

loop: "{{install\_package}}"

1. To run exercise 2 task, use the following command:

ansible-playbook -ki inventory.yaml main-script.yaml --tags=ex-2

**3.3 Exercise 3 - Privilege Escalation**

Exercise 3 will focus on privilege escalation, which is required to run specific tasks. One such particular task is to install modules/packages on the target node VM. This task will fail if it is run by a normal user with no privileges. This exercise will introduce the become module. This module allows the user to become another user of the target node and use that user account to run the tasks. Becoming a sudoer/root user will thus give the necessary privileges for tasks such as module/package installation.

1. Create exercise 3 task in main-script.yaml

nano main-script.yaml

Add the following into main-script.yaml

- name: Exercise 3 - Privilege Escalation

hosts:

managed\_node

become: yes

roles:

- role: installpackage

tags: ex-3

1. Create exercise 3 task folders in the roles directory

mkdir -p roles/installpackage/tasks

1. Create the main.yaml file

nano roles/installpackage/tasks/main.yaml

Fill in the main.yaml file of installpackage task with the following script

- name: Install nginx package

apt:

name: "{{item}}"

state: present

loop: "{{install\_package}}"

- name: Start nginx service

service:

name: nginx

state: started

1. To run exercise 3 task, use the following command:

Run the script to only target the managed node:

ansible-playbook -Kki inventory.yaml main-script.yaml --tags=ex-3

-K : prompt user to input become password

**3.4 Exercise 4 - Target Control**

Exercise 4 focuses on target control. This exercise will demonstrate how to specify target nodes for each Ansible task through the setup of the ansible host file and the hosts module.

1. Create exercise 4 task in main-script.yaml

nano main-script.yaml

Add the following into main-script.yaml

- name: Exercise 4 - Target Control, Target only one managed node

hosts:

managed\_node

roles:

- role: get-block-devices

tags:

- ex-4

- ex-4-managed\_node

- name: Exercise 4 - Target Control, Target both the managed node & localhost

hosts:

all\_node

roles:

- role: get-disk-usage

tags:

- ex-4

- ex-4-all\_node

1. Edit the ansible host file to add another group

nano inventory.yaml

Add your target node hostname/ip address into the file (Replace 'X' with the number assigned to you)

[managed\_node]

targetnodeX

[all\_node]

targetnodeX

localhost

1. Create the task folders in the roles directory
2. mkdir -p roles/get-disk-usage/tasks

mkdir -p roles/get-block-devices/tasks

1. Create the main.yaml file for both tasks

For get-disk-usage task:

nano roles/get-disk-usage/tasks/main.yaml

Fill in the main.yaml file of get-disk-usage task with the following script

- name: Get disk usage data

shell: "df -h"

register: DiskUsageData

- name: Print disk usage data

debug:

msg: "{{DiskUsageData.stdout\_lines}}"

For get-block-devices task:

nano roles/get-block-devices/tasks/main.yaml

Fill in the main.yaml file of get-block-devices task with the following script

- name: Get block devices data

shell: "lsblk"

register: BlockDeviceData

- name: Print block devices data

debug:

msg: "{{BlockDeviceData.stdout\_lines}}"

1. To run exercise 4 task, use the following command:

Run the task to get block devices data from the managed node:

ansible-playbook -ki inventory.yaml main-script.yaml --tags=ex-4-managed\_node

Run the task to get the disk usage data from both the managed node and control node:

ansible-playbook -ki inventory.yaml main-script.yaml --tags=ex-4-all\_node

Run both of the task:

ansible-playbook -ki inventory.yaml main-script.yaml --tags=ex-4

**Appendix A**

**Useful Ubuntu Commands**

Change directory to current user's home directory

cd ~

Change directory to a specific path

cd /path/to/directory

Change directory to a relative path

cd path/to/directory/from/current/directory

List down all files and folders in the current directory

ls -al

List down all files and folders in the specified directory

ls -al /path/to/directory

Edit file with nano text editor  
To save your file when inside the editor: CTRL + o, then ENTER  
To exit the editor: CTRL + x

nano your\_file\_name

Access target server remotely through SSH protocol

ssh <username>@<target\_ip> -p <port\_number(default is 22)>

Terminate your current user's session

exit