

Lab 3 - Manage AWS Infrastructure with Terraform (IaC) and Ansible (CM)

Contents

Lab Evidence	2
Learning Goals	2
Pre-requisite	2
Introduction	2
Demonstrate Steps for managing AWS Infrastructure with Terraform (IaC) and Ansible (CM)	2
Step 1: Generate SSH Keys	4
Step 3: Download PuTTYgen and Generate .ppk key	5
Step 4: Terraform Structure	9
Step 5: Terraform Code	10
Step 6: Ansible Playbook	13
Step 7: Terraform and Ansible in action	16
Step 8: Connect to EC2 instance	18
Using Mac OS – Terminal	18
Using PuTTY	18
Step 9: Terraform destroy	22
Note	22
Conclusion	22

Lab Evidence

Your report on this lab will be the basis for your grade. The purpose of the report is to demonstrate to your instructor that you have completed the lab successfully and understood the material. Please include the following screenshots as part of your evidence.

- `webserver-Public-URL` and the URL assigned from the end of the Terraform apply output. Please do not include all of the output from apply, just the `webserver-Public-URL` value
- The Cloud Computing Course webpage in your browser, including the address bar showing the URL
- Provisioned Infrastructure as viewed through your AWS account, including the public IP of your webserver, under Instances
- `ls` command output from putty session to your EC2 instance to show folder contents.

A portion of the grade will be allocated to the quality of the report. The report should be well formatted, concise and to the point.

Learning Goals

Demonstrate and execute steps for managing AWS Infrastructure with Terraform (IaC) and Ansible (CM).

Pre-requisite

- Created AWS free tier accounts account (Check provided AWS instructions in Moodle - 02.Create AWS Free Tier Account.docx)
- Completed the following in Lab2: Manage AWS Infrastructure with Terraform
 - Downloaded and installed Terraform on your local machine link
 - Created AWS free tier accounts account <https://github.com/>
 - Downloaded and installed AWS CLI on your local machine link
 - Configured AWS Access Keys

Introduction

You study during this week how Configuration Management (CM) and how you can use Ansible as CM tool. Let's work through a scenario which combine what you have learned in the previous week and this week to manage AWS infrastructure with Terraform (IaC) and Ansible (CM).

Demonstrate Steps for managing AWS Infrastructure with Terraform (IaC) and Ansible (CM)

To accomplish this, you are going to follow these steps:

- Step 1: Generate SSH Keys
- Step 2: Download PuTTY
- Step 3: Download PuTTYgen and Generate .ppk key
- Step 4: Terraform Structure

- Step 5: Terraform Code
- Step 6: Ansible Play
- Step 7: Terraform and Ansible in action
- Step 8: Connect to EC2 instance using PuTTY
- Step 9: Terraform destroy

Step 1: Generate SSH Keys

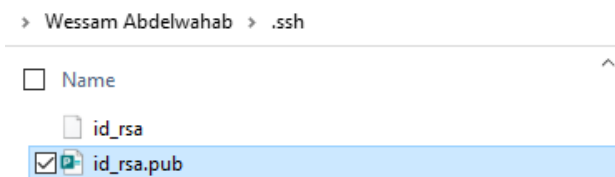
This will be used later to connect to EC2 instances in your code and using ssh terminal

- Open a terminal window. At the shell prompt, type the following command:
 - `ssh-keygen -t rsa`
 - The `ssh-keygen` program will prompt you for the location of the key file. Press Return to accept the defaults.
 - Press enters twice for no passphrase
 - Note the location of your public and private keys were saved, they will be required in a subsequent steps

```
C:\Users\wessa>ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\wessa\.ssh\id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\wessa\.ssh\id_rsa.
Your public key has been saved in C:\Users\wessa\.ssh\id_rsa.pub.
The key fingerprint is:

The key's randomart image is:
+---[RSA 3072]----+
|
|*
|
|*
|
|
|C
|
|*
+---[SHA256]-----+
```

- Here are the generated keys
 - **id_rsa** is your private key
 - **id_rsa.pub** is your public key



Steps 2 & 3 provide instructions for downloading and installing Putty, which allows you to connect to a remote server, an AWS EC2 instance in the case of this lab.

If you are using Mac OS, then you can connect to a remote server directly from the terminal window – you do not need to complete steps 2 & 3.

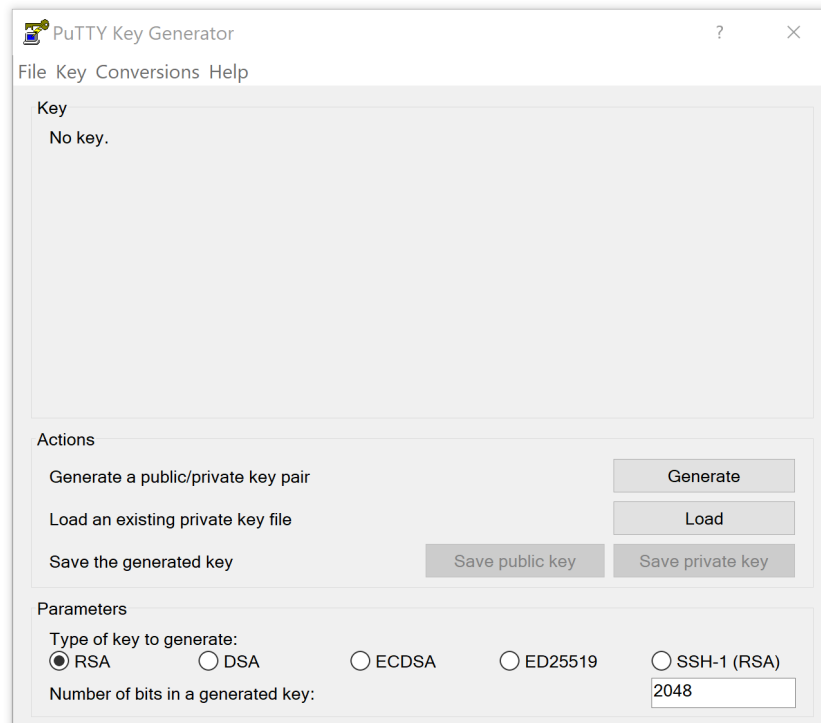
Step 2: Download PuTTY

- PuTTY website [link](#)
 - Download PuTTY [link](#)
- More details on that later in the document when you connect to EC2 instance. For now, just download it.

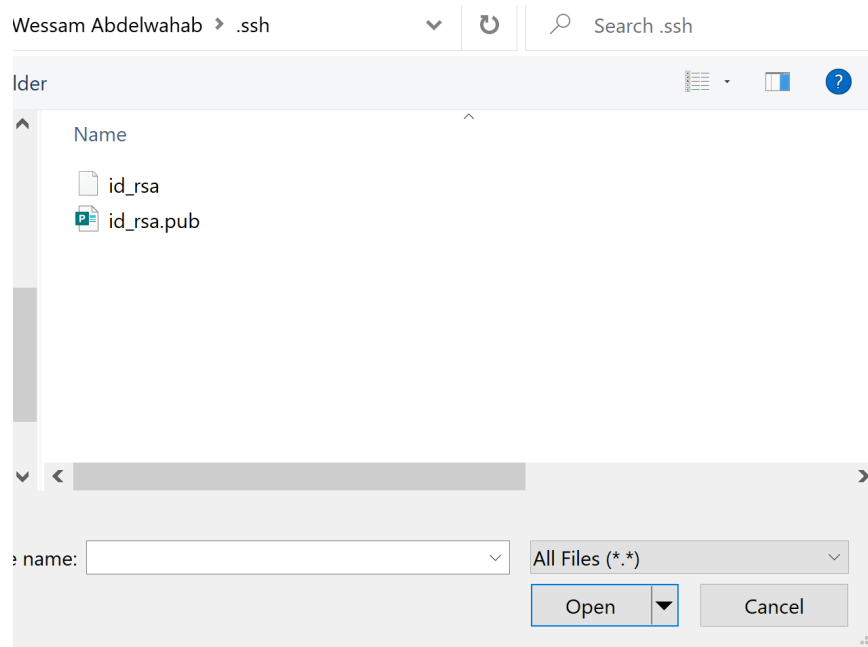
Step 3: Download PuTTYgen and Generate .ppk key

PuTTYgen will be used to convert .pem file into a .ppk to connect to EC2 instance using PuTTY

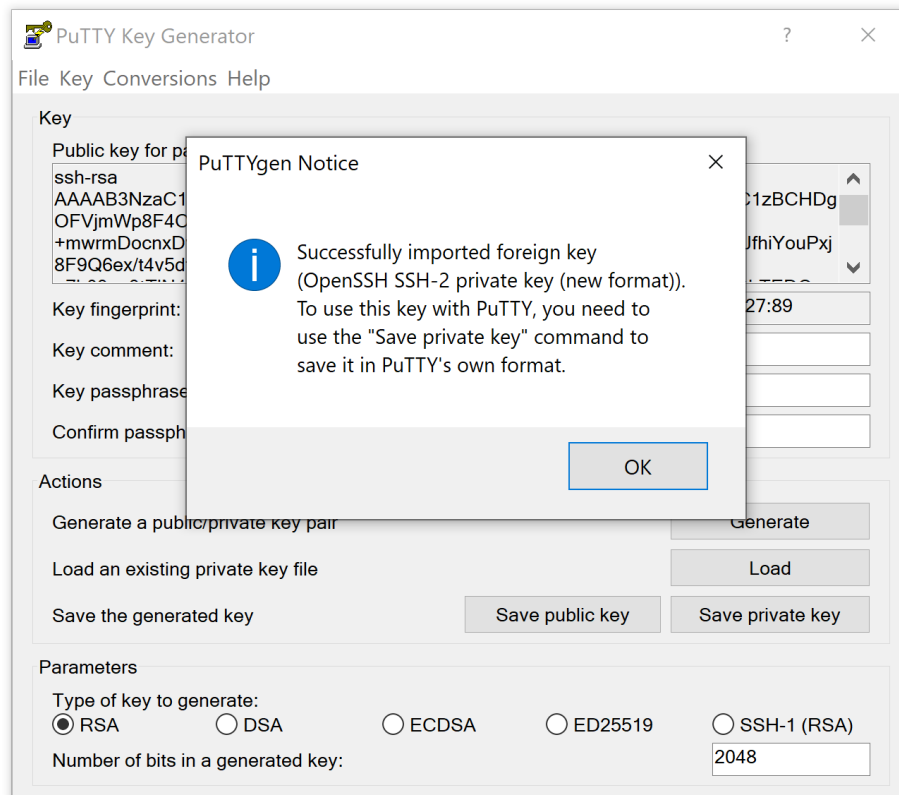
- PuTTYgen website [link](#) (Check the version for your OS)
 - For more information on related subject you may refer to **How do I convert a .pem file into a .ppk, and vice versa, on Windows and Linux?** [link](#)
- Here is a demonstration for Windows OS (64-bit x86)
 - Download (a RSA and DSA key generation utility) - 64-bit x86: [puttygen.exe](#)



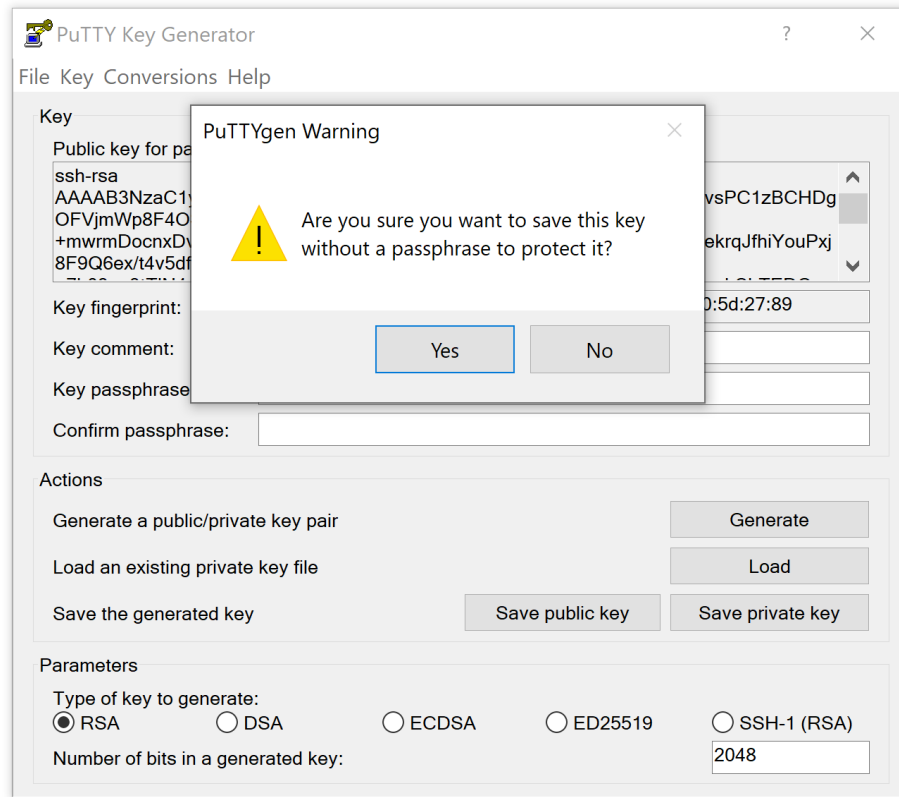
- Click on Load and select all files



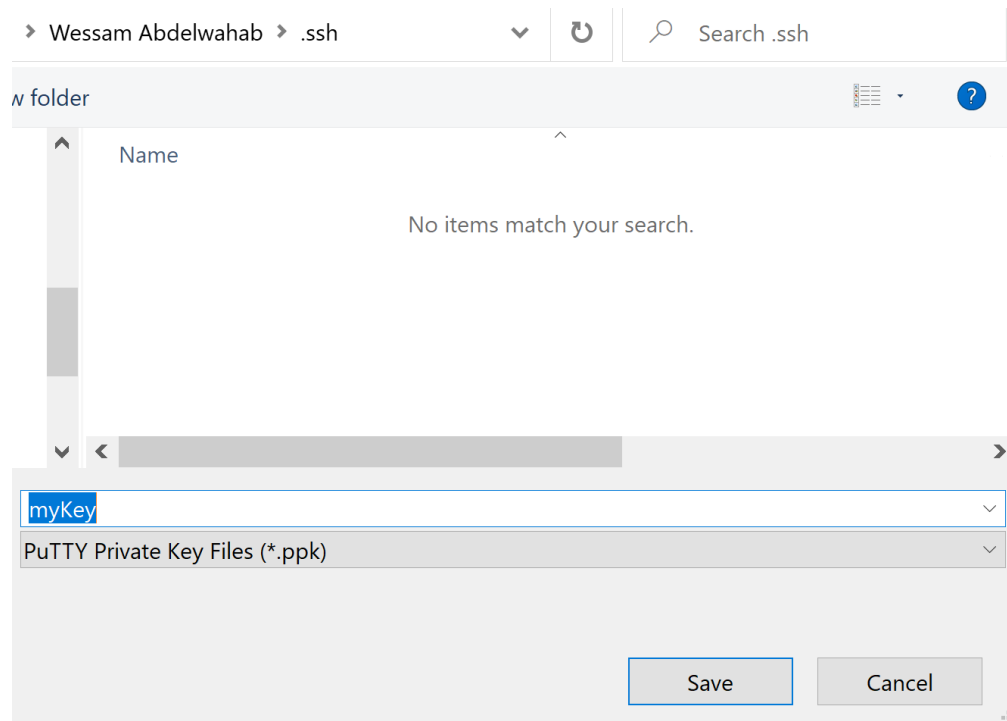
- Select your private key **id_rsa**



- Now click save private key -> click Yes

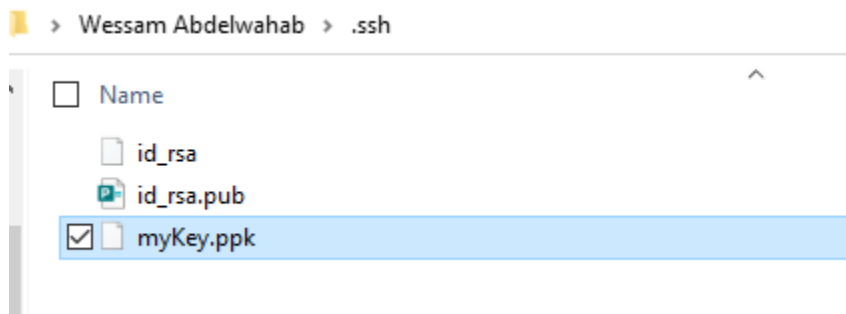


- Name your key and note where you will save it



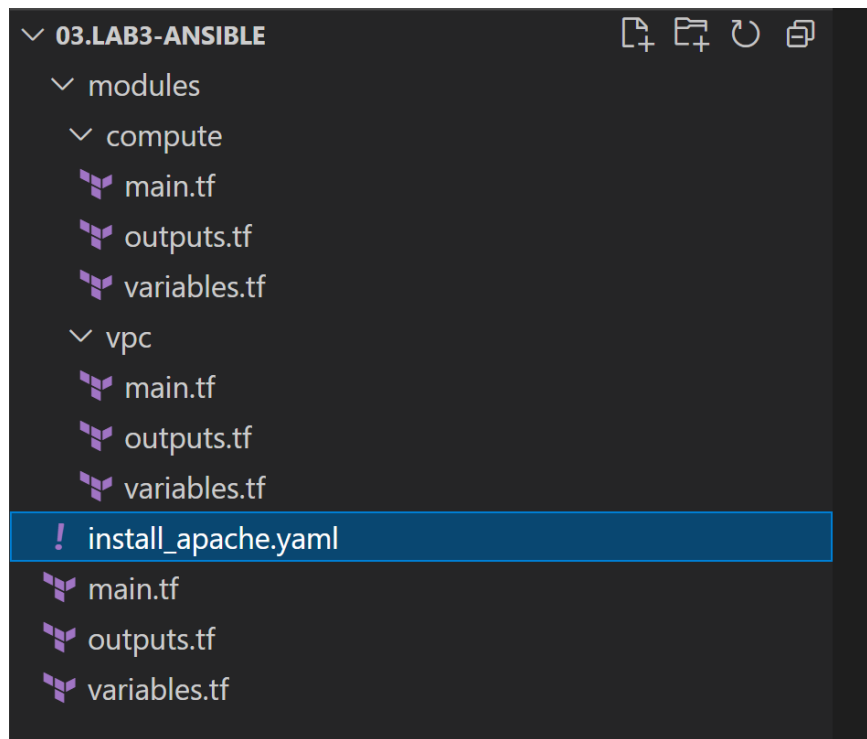
- You should end up with three generated keys
 - **id_rsa** is your private key

- **id_rsa.pub** is your public key
- **myKey.ppk** is your PuTTY private key



Step 4: Terraform Structure

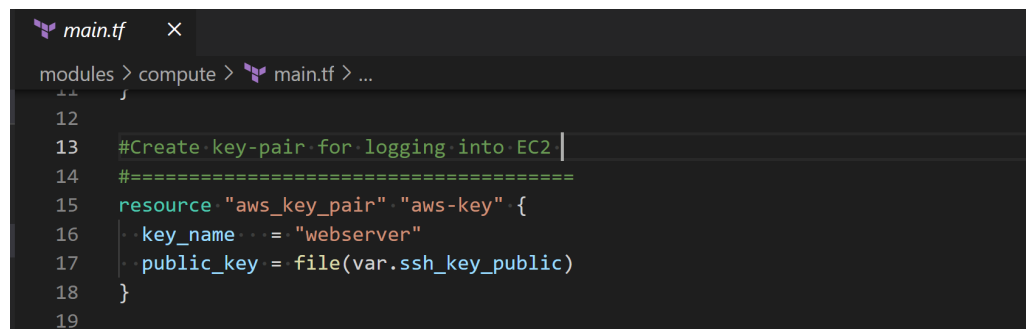
The same structure as the previous module, with one addition the Ansible play (install_apache.yaml)



Step 5: Terraform Code

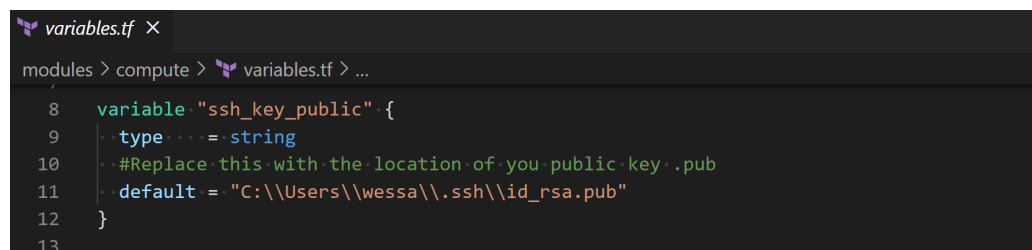
In this part will go only through the modifications of Terraform Code from the previous module

- **Terraform Main**
 - Same code from the previous module, no changes
- **Terraform Virtual Private Cloud (VPC) Module**
 - Same code from the previous module, no changes
- **Terraform Compute (EC2) Module**
 - outputs.tf same code from the previous module, no changes
 - main.tf & variables.tf modification (Understand Terraform Code)
 - Here we create a key pair in AWS using the public key we generated in a previous step
 - **file** is a function that reads the content in a file at the given path and returns them as a string, we are passing it a variable which declared in variables.tf



```
main.tf X
modules > compute > main.tf > ...
12
13 #Create key-pair for logging into EC2 |
14 #=====
15 resource "aws_key_pair" "aws-key" {
16   key_name    = "webserver"
17   public_key = file(var.ssh_key_public)
18 }
19
```

- Here is the ssh_key_public variable. You must change the path of the public key file (**id_rsa.pub**) to the location you noted earlier



```
variables.tf X
modules > compute > variables.tf > ...
8 variable "ssh_key_public" {
9   type    = string
10   #Replace this with the location of your public key .pub
11   default = "C:\\Users\\wessa\\.ssh\\id_rsa.pub"
12 }
13
```

- **key_name = aws_key_pair.aws-key.key_name** this will add the public key inside aws_instance resource

```
main.tf x
modules > compute > main.tf > ...
19
20 #Create and bootstrap webserver
21 #=====
22 resource "aws_instance" "webserver" {
23   instance_type = "t2.micro"
24   ami           = data.aws_ssm_parameter.webserver-ami.value
25   tags = {
26     Name = "webserver_tf"
27   }
28   key_name = aws_key_pair.aws-key.key_name
29   associate_public_ip_address = true
30   vpc_security_group_ids = [var.security_group]
31   subnet_id = var.subnets
32 }
```

- Then we are connecting to the create EC2 instance using SSH,
 - The default user for connecting to EC2 instance is ec2-user
 - **private_key = file(var.ssh_key_private)** here we are connecting to the EC2 instance using the private key (Never share your private key with anyone - These keys in a real-world scenario will be stored in a key vault)
 - Here is the ssh_key_private variable. You must change the path of the private key file (**id_rsa**) to the location you noted earlier
 - **host = self.public_ip** interpolate that resource's public IP address.

```
main.tf x
modules > compute > main.tf > resource "aws_instance" "webserver" > connection
32
33 connection {
34   type = "ssh"
35   user = "ec2-user"
36   private_key = file(var.ssh_key_private)
37   host = self.public_ip
38 }
39
```

- Here is the ssh_key_private variable. You must change the path of the public key file (**id_rsa**) to the location you noted earlier

```
variables.tf x
modules > compute > variables.tf > ...
14 variable "ssh_key_private" {
15   type = string
16   #Replace this with the location of your private key
17   default = "C:\\Users\\wessa\\.ssh\\id_rsa"
18 }
19
```

- The file provisioner used to copy the ansible yaml file (install_apache.yaml) from your local machine to the newly create resource in that case EC2 instance

```

main.tf X
modules > compute > main.tf > HashiCorp Terraform > resource "aws_instance" "webserver"
39
40 # Copy the file from local machine to EC2
41 provisioner "file" {
42     source = "install_apache.yaml"
43     destination = "install_apache.yaml"
44 }
45

```

- Then we are executing a script on the created resource remotely (EC2 instance)
 - The **remote-exec provisioner** invokes a script on a remote resource after it is created.
 - The **inline** execute a list of command strings in the provided order. Basically, here we are updating the linux distro, installing ansible in order to be able to execute the ansible play on the EC2 instance. Then wait for 60 seconds and then execute the ansible playbook by running the command **ansible-playbook install_apache.yaml**

```

main.tf X
modules > compute > main.tf > HashiCorp Terraform > resource "aws_instance" "webserver"
45
46 # Execute a script on a remote resource
47 provisioner "remote-exec" {
48     inline = [
49         "sudo yum update -y && sudo amazon-linux-extras install ansible2 -y",
50         "sleep 60s",
51         "ansible-playbook install_apache.yaml"
52     ]
53 }

```

Step 6: Ansible Playbook

- Understanding Ansible
 - The three `---` tell ansible this is the start of a playbook
 - **name** is the name of your play; you can put the name you want
 - **hosts: localhost** specify which host you want this play on, in this case it is the localhost
 - **remote_user** is the instance user in case of EC2 it is ec2-user
 - **become: true** allows you to escalate privileges and become another user other than the user logged into the machine (ec2-user). The purpose of this in our case to be able to install Apache.

```
! install_apache.yaml X
! install_apache.yaml > {} 0 > [ ] tasks
1  ---
2  - name: Web Server
3    hosts: localhost
4    remote_user: ec2-user
5    become: true
6
```

- We will break the construction of our playbook into three parts package management, configuration, and service handlers
- Package Management

Here we are installing the packages required by our system

- **tasks** a list of task, a task is the smallest unit of action you can automate using ansible playbook. Ansible executes tasks in the same order they are defined inside a playbook
- Then you **name** your tasks whatever you want to name it. It is optional but make it easier to read the outputs of playbook
- We are installing the Apache package that we will need to run the website using the built in module offered by ansible. The version we are installing it is the latest version of Apache
 - yum: (package manager module)
 - name: httpd (Apache)
 - state: latest (latest version)

```
! install_apache.yaml X
! install_apache.yaml > {} 0 > [ ] tasks > {} 2 > {} ansible.builtin.copy
7  tasks:
8  - name: Install apache packages
9    yum:
10     name: httpd
11     state: latest
12
```

- Infrastructure Configuration

Here we are configuring our system with the necessary application or configuration files that we need to configure the environment

- The unarchive module unpacks an archive. It will copy the files from the source to the destination before unpacking it
- **remote_src: yes** it will go to the remote/target machine for the source

```
! install_apache.yaml ✕
! install_apache.yaml > {} 0 > [ ] tasks > {} 3 > {} service
13  - name: Extract master.zip into /tmp
14  - ansible.builtin.unarchive:
15    - src: https://github.com/wessamabdelwahab/CSCC1030/archive/master.zip
16    - dest: /tmp
17    - remote_src: yes
18
```

- The copy module copies a file from the local or remote machine (source) to a location on the remote machines (destination). Here we copying the unpacked archive contents from the tmp to Apache location to run the website

```
! install_apache.yaml ✕
! install_apache.yaml > {} 0 > [ ] tasks
19  - name: Copy CSCC1030-master directory contents into /var/www/html
20  - ansible.builtin.copy:
21    - src: /tmp/CSCC1030-master/
22    - dest: /var/www/html/
23
```

- Ansible modules check whether the desired final state has already been achieved. This ensure that Apache is running and it is state started

```
! install_apache.yaml ✕
! install_apache.yaml > {} 0 > [ ] tasks
25  - name: Ensure httpd is running
26  - service:
27    - name: httpd
28    - state: started
29
```

```
! install_apache.yaml ✕
! install_apache.yaml > {} 0 > [ ] tasks
24  - name: Ensure httpd is running
25  - service:
26    - name: httpd
27    - state: started
28
```

- Service Handlers

- Create service handlers to start, stop, or restart our system. Sometimes you want a task to run only when a change is made on a machine. For example,

you may want to restart a service if a task updates the configuration of that service, but not if the configuration is unchanged. Ansible uses handlers to address this use case.

```
! install_apache.yaml X
! install_apache.yaml > {} 0 > [ ] handlers
30   - handlers:
31     - name: Restart apache
32       ansible.builtin.service:
33         name: httpd
34         state: restarted
35
```

```
! install_apache.yaml X
! install_apache.yaml > {} 0 > [ ] handlers
29   - handlers:
30     - name: Restart apache
31       ansible.builtin.service:
32         name: httpd
33         state: restarted
34
```

Step 7: Terraform and Ansible in action

- **terraform init:** Locate your working directory (Lab3-Ansible) and initialize terraform **terraform init**

```
Command Prompt
\03.Lab3-Ansible>terraform init

Initializing modules...
- compute in modules\compute
- vpc in modules\vpc

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.44.0"...
- Installing hashicorp/aws v3.44.0...
- Installed hashicorp/aws v3.44.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

- **terraform validate:** Validate your terraform

```
\03.Lab3-Ansible>terraform validate

Success! The configuration is valid.
```

- **terraform apply:** Provision terraform managed infrastructure. You must confirm by typing **yes** if you would like to continue and perform the actions described to provision your infrastructure resources

```
Enter a value: yes

module.vpc.aws_vpc.tf_vpc: Creating...
module.compute.aws_key_pair.aws-key: Creating...
module.compute.aws_key_pair.aws-key: Creation complete after 1s [id=webserver]
module.vpc.aws_vpc.tf_vpc: Still creating... [10s elapsed]
module.vpc.aws_vpc.tf_vpc: Creation complete after 16s [id=vpc-019d76a55e4d068b2]
module.vpc.aws_internet_gateway.tf_igw: Creating...
module.vpc.aws_subnet.tf_public_subnet: Creating...
module.vpc.aws_security_group.tf_public_sg: Creating...
module.vpc.aws_subnet.tf_public_subnet: Creation complete after 2s [id=subnet-07c91a13f756cbb38]
module.vpc.aws_internet_gateway.tf_igw: Creation complete after 3s [id=igw-04f75abaa90e4aca8]
module.vpc.aws_route_table.tf_public_route: Creating...
module.vpc.aws_route_table.tf_public_route: Creation complete after 2s [id=rtb-09a51e4f7d9dcb708]
module.vpc.aws_route_table_association.tf_public_assoc: Creating...
module.vpc.aws_route_table_association.tf_public_assoc: Creation complete after 1s [id=rtbassoc-0097e4d8ed3eb6b17]
module.vpc.aws_security_group.tf_public_sg: Creation complete after 6s [id=sg-08b1569fc61916f2a]
module.compute.aws_instance.webserver: Creating...
module.compute.aws_instance.webserver: Still creating... [10s elapsed]
module.compute.aws_instance.webserver: Still creating... [20s elapsed]
```

- Watch the ansible playbook execution


```

module.compute.aws_instance.websERVER: Still creating... [2m10s elapsed]
module.compute.aws_instance.websERVER (remote-exec): [WARNING]: provided hosts list is empty, only localhost is available. Note that
module.compute.aws_instance.websERVER (remote-exec): the implicit localhost does not match 'all'

module.compute.aws_instance.websERVER (remote-exec): PLAY [Web Server] *****
module.compute.aws_instance.websERVER (remote-exec): TASK [Gathering Facts] *****
module.compute.aws_instance.websERVER: Still creating... [2m20s elapsed]
module.compute.aws_instance.websERVER (remote-exec): ok: [localhost]

module.compute.aws_instance.websERVER (remote-exec): TASK [Install apache packages] *****
module.compute.aws_instance.websERVER (remote-exec): changed: [localhost]

module.compute.aws_instance.websERVER (remote-exec): TASK [Extract master.zip into /tmp] *****
module.compute.aws_instance.websERVER (remote-exec): changed: [localhost]

module.compute.aws_instance.websERVER (remote-exec): TASK [Copy CSCC1030-master directory contents into /var/www/html] *****
module.compute.aws_instance.websERVER: Still creating... [2m30s elapsed]
module.compute.aws_instance.websERVER (remote-exec): changed: [localhost]

module.compute.aws_instance.websERVER (remote-exec): TASK [Ensure httpd is running] *****
module.compute.aws_instance.websERVER (remote-exec): changed: [localhost]

module.compute.aws_instance.websERVER (remote-exec): PLAY RECAP *****
module.compute.aws_instance.websERVER (remote-exec): localhost : ok=5 changed=4 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

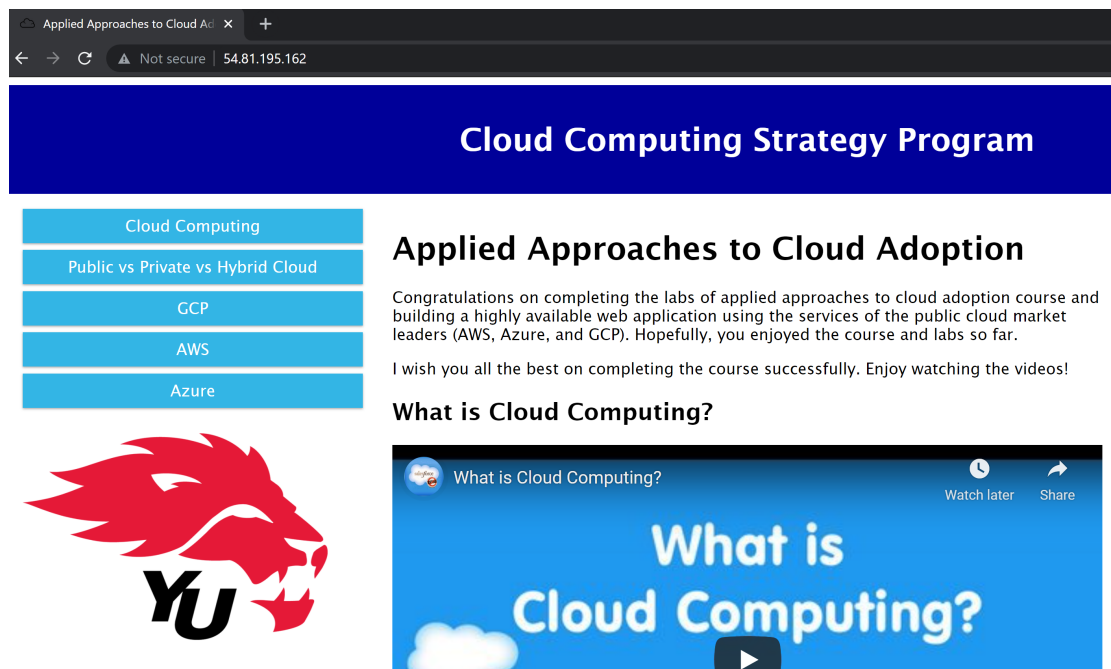
module.compute.aws_instance.websERVER: Creation complete after 2m38s [id=1-069314bf276de2297]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:
Apache-Webserver-Public-URL = "http://54.81.195.162"

```

- Now you can access the website from the Cloud Computing course using the URL provided in the output



Applied Approaches to Cloud Adoption

Cloud Computing Strategy Program

- Cloud Computing
- Public vs Private vs Hybrid Cloud
- GCP
- AWS
- Azure

Applied Approaches to Cloud Adoption

Congratulations on completing the labs of applied approaches to cloud adoption course and building a highly available web application using the services of the public cloud market leaders (AWS, Azure, and GCP). Hopefully, you enjoyed the course and labs so far.

I wish you all the best on completing the course successfully. Enjoy watching the videos!

What is Cloud Computing?

What is Cloud Computing?

Watch later Share

- You can also login to your AWS account and see the provisioned infrastructure resources. You must select N. Virginia which is us-east-1 from the top right corner

N. Virginia ▼ Support ▼

Step 8: Connect to EC2 instance

Using Mac OS - Terminal

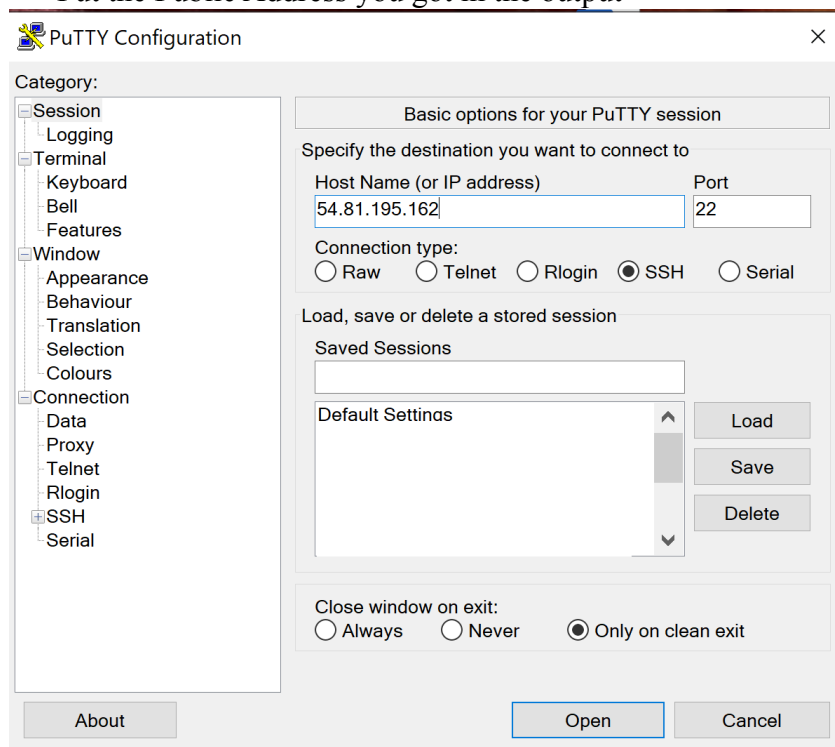
Using the Public Address you got in the output, enter the following command from a terminal window.

```
ssh ec2-user@184.72.67.219
```

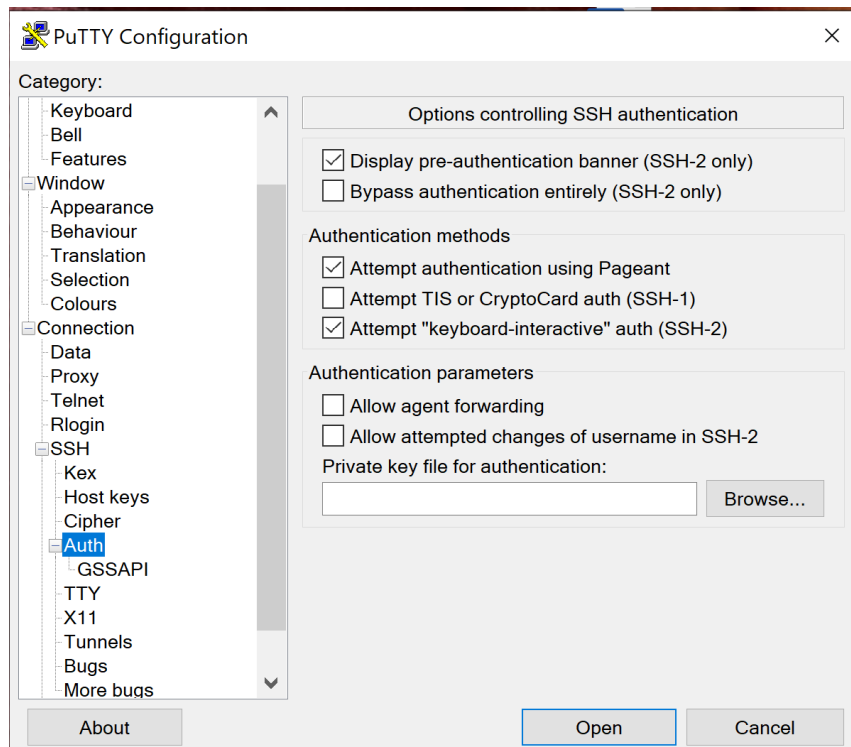
Using PuTTY

Here is a demonstration for connecting to the EC2 instance from Windows OS machine

- Download putty.exe (the SSH and Telnet client itself) - 64-bit x86: [putty.exe](#)
- Open PuTTY
 - Put the Public Address you got in the output



- Then go to Auth under SSH

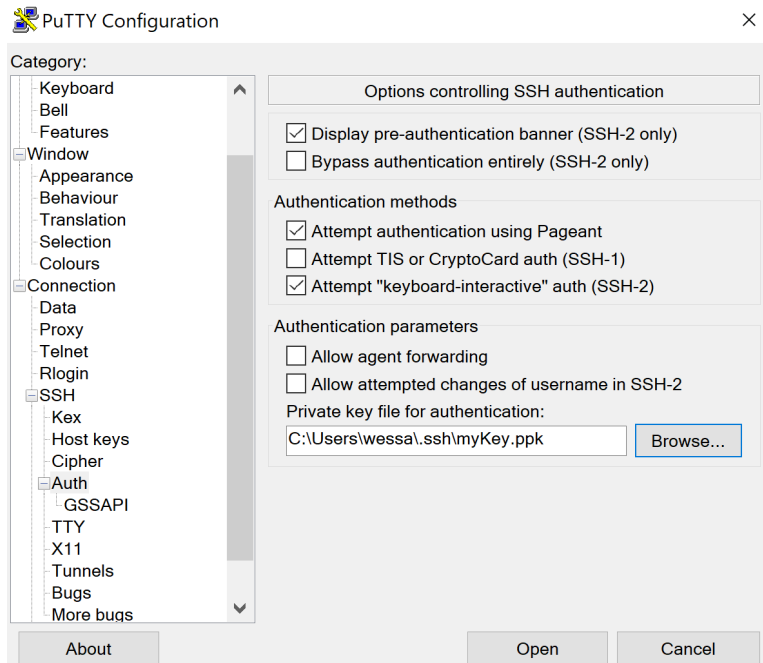


- Browse to the location of your myKey.ppk generated early using PuTTYgen

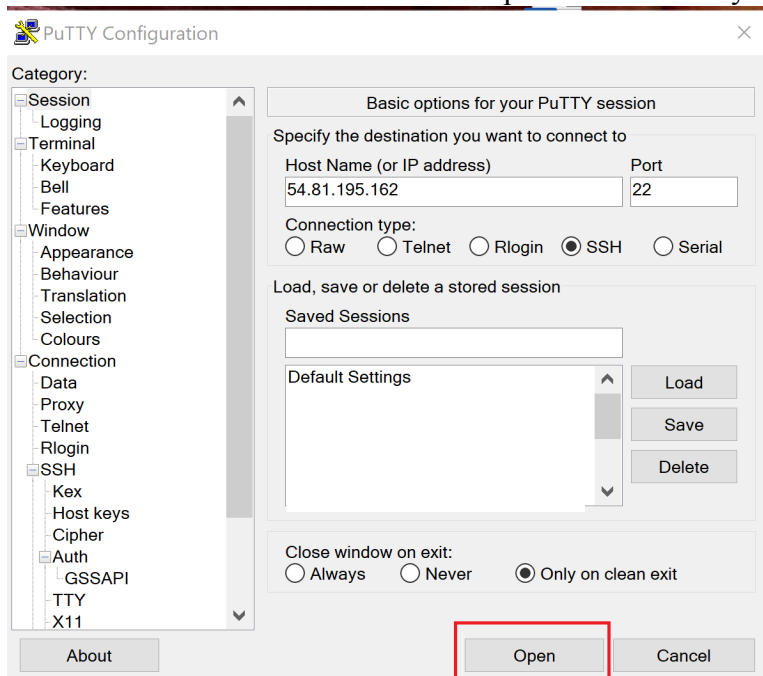
📁 > Wessam Abdelwahab > .ssh

New folder

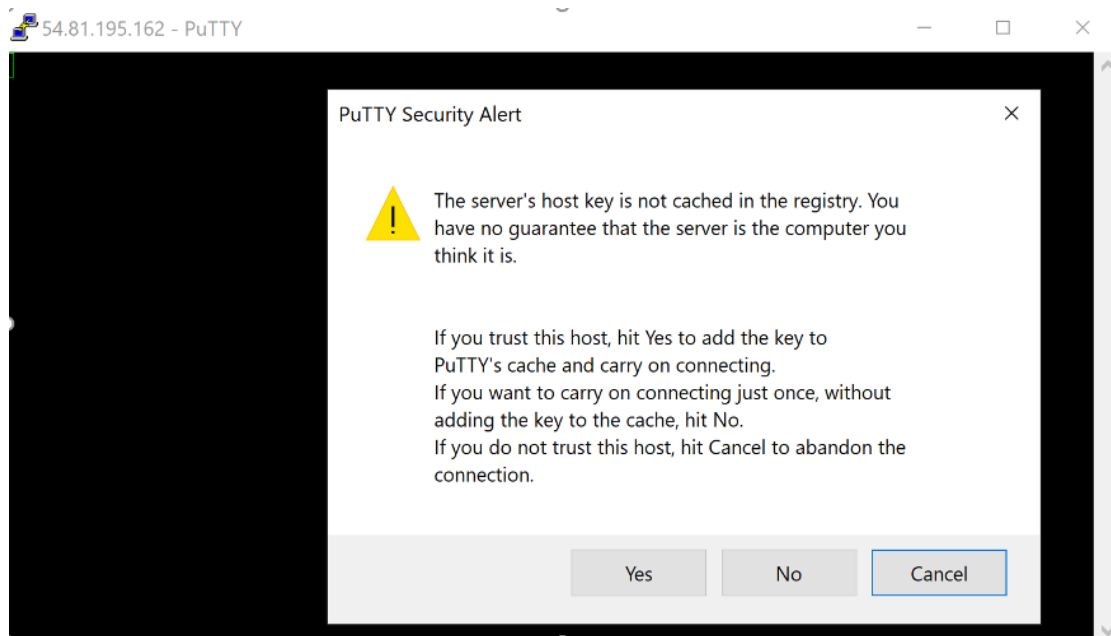
Name	
s	myKey.ppk



- Go back to the Session and click Open to connect to your EC2 instance



- Click yes



○ User to login ec2-user.



Step 9: Terraform destroy

- Once you are done with the lab you must destroy the provisioned infrastructure resources to avoid any risks of encountering cost in the future.
- **terraform destroy**: Destroy terraform managed infrastructure. You must confirm by typing yes

```
Changes to Outputs:
- Apache-Webserver-Public-URL = "http://54.81.195.162" -> null

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

module.vpc.aws_route_table_association.tf_public_assoc: Destroying... [id=rtbassoc-0097e4d8ed3eb6b17]
module.compute.aws_instance.webserver: Destroying... [id=i-069314bf276de2297]
module.vpc.aws_route_table_association.tf_public_assoc: Destruction complete after 0s
module.vpc.aws_route_table.tf_public_route: Destroying... [id=rtb-09a51e4f7d9dcb708]
module.vpc.aws_route_table.tf_public_route: Destruction complete after 2s
module.vpc.aws_internet_gateway.tf_igw: Destroying... [id=igw-04f75abaa90e4aca8]
module.compute.aws_instance.webserver: Still destroying... [id=i-069314bf276de2297, 10s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Still destroying... [id=igw-04f75abaa90e4aca8, 10s elapsed]
module.compute.aws_instance.webserver: Still destroying... [id=i-069314bf276de2297, 20s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Still destroying... [id=igw-04f75abaa90e4aca8, 20s elapsed]
module.compute.aws_instance.webserver: Still destroying... [id=i-069314bf276de2297, 30s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Still destroying... [id=igw-04f75abaa90e4aca8, 30s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Destruction complete after 31s
module.compute.aws_instance.webserver: Still destroying... [id=i-069314bf276de2297, 40s elapsed]
module.compute.aws_instance.webserver: Destruction complete after 43s
module.compute.aws_key_pair.aws-key: Destroying... [id=webserver]
module.vpc.aws_subnet.tf_public_subnet: Destroying... [id=subnet-07c91a13f756cbb38]
module.vpc.aws_security_group.tf_public_sg: Destroying... [id=sg-08b1569fc61916f2a]
module.compute.aws_key_pair.aws-key: Destruction complete after 0s
module.vpc.aws_security_group.tf_public_sg: Destruction complete after 1s
module.vpc.aws_subnet.tf_public_subnet: Destruction complete after 1s
module.vpc.aws_vpc.tf_vpc: Destroying... [id=vpc-019d76a55e4d068b2]
module.vpc.aws_vpc.tf_vpc: Destruction complete after 1s

Destroy complete! Resources: 8 destroyed.
```

Note

Do not delete the keys you generated on your local machines. You will need it for subsequent labs.

Conclusion

Congrats on creating your Infrastructure using Terraform (IaC) and Ansible (CM)!