Automating Infrastructure using Terraform

Table of contents

Project Description

Organizational requirement is to automate the infrastructure using Terraform first and install other required automation tools in it.

Expected Deliverables

- Launch an EC2 instance using Terraform
- Connect to the instance
- Install Jenkins, Java and Python in the instance

Tools required

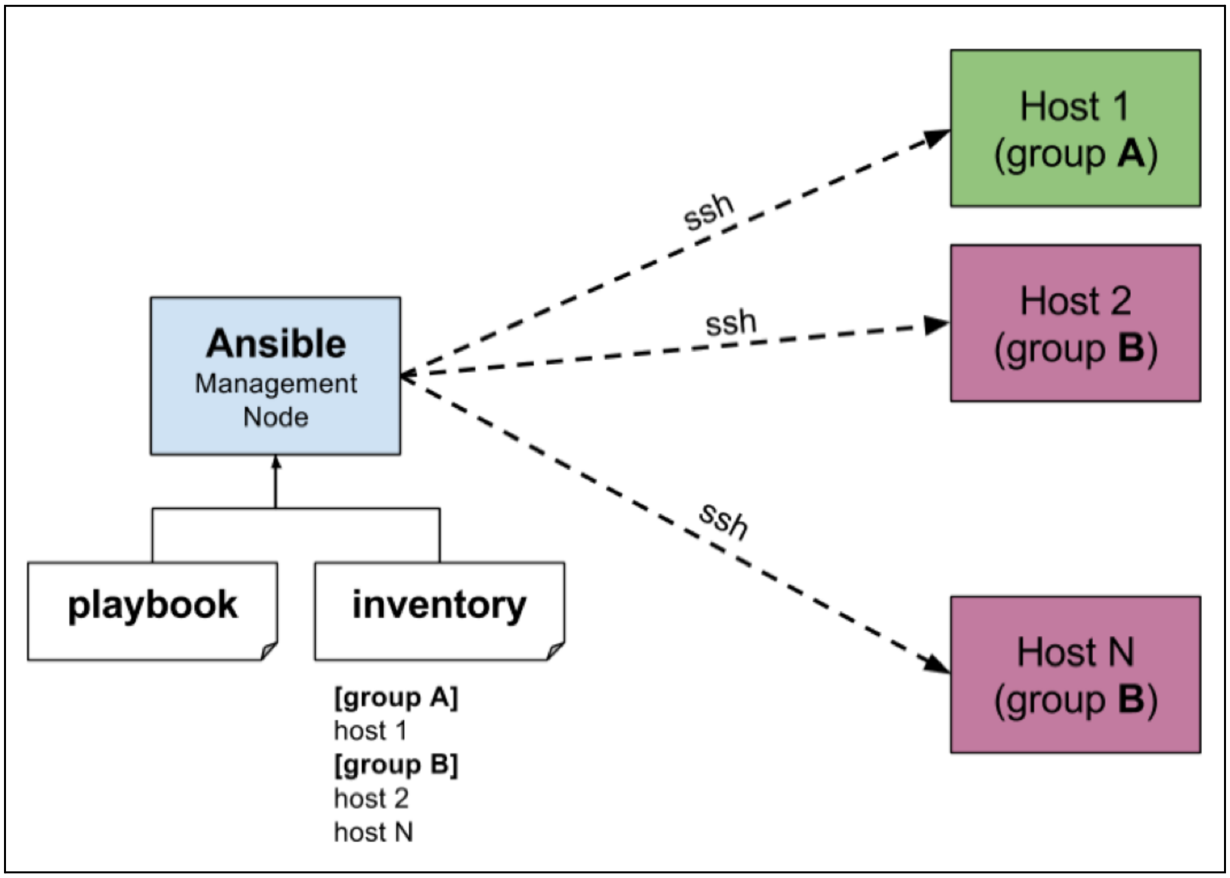
- Terraform
- AWS account with security credentials,
- Keypair
- Ansible

Terraform Overview

- **Infrastructure as code**
  Use infrastructure as code to automate the provisioning of your infrastructure including servers, databases, firewall policies, and almost any other resource.
- **Multi-cloud provisioning**
  Deploy serverless functions with AWS Lambda, manage Microsoft Azure Active Directory resources, provision a load balancer in Google Cloud, and more.
- **Network infrastructure automation**
  Automate key networking tasks such as updating load-balancer member pools or applying firewall policies.
- **Manage Kubernetes**
  Provision and manage Kubernetes clusters on AWS, Microsoft Azure, or Google Cloud, and interact with your cluster using the Kubernetes Terraform provider.
- **Manage virtual machine images**
  Create multi-cloud golden image pipelines with HCP Packer and Terraform Cloud
- **Integrate with existing workflows**
  Automate infrastructure deployments through existing CI/CD workflows.

Ansible Overview

- Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.
- Ansible is completely agentless and works by connecting nodes primarily through SSH.
- Ansible pushes small programs, called Ansible modules, on the nodes and removes them when finished.
- Ansible manages inventory in simple text files called hosts file.
- Ansible uses the hosts file to control the actions on a specific group in the playbooks.

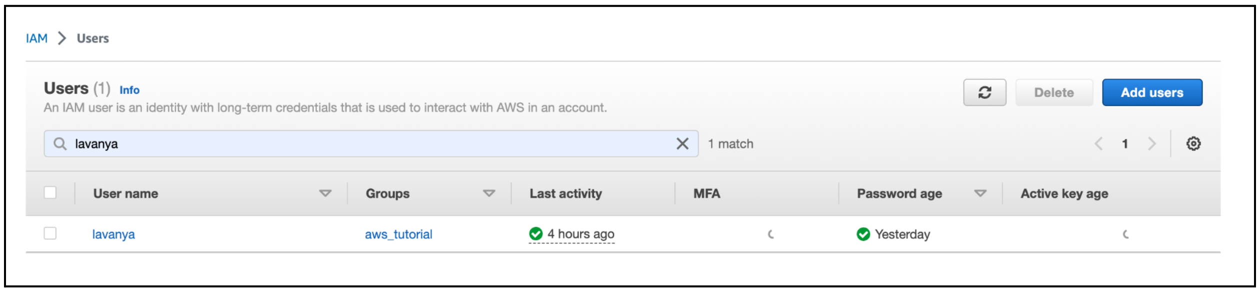


Tasks Lists

- Setup AWS account
- Launch an EC2 instance from AWS web console
- Connect to the EC2 Ubuntu instance from local machine
- Install Terraform
- Launch an Ubuntu EC2 instance using Terraform(Ansible node)
- Install Ansible
- Install Java, Maven, Jenkins in the ansible node using ansible-playbook

Setup AWS account

- Create an account in https://aws.amazon.com. Set that as basic free tier account



Users (1) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

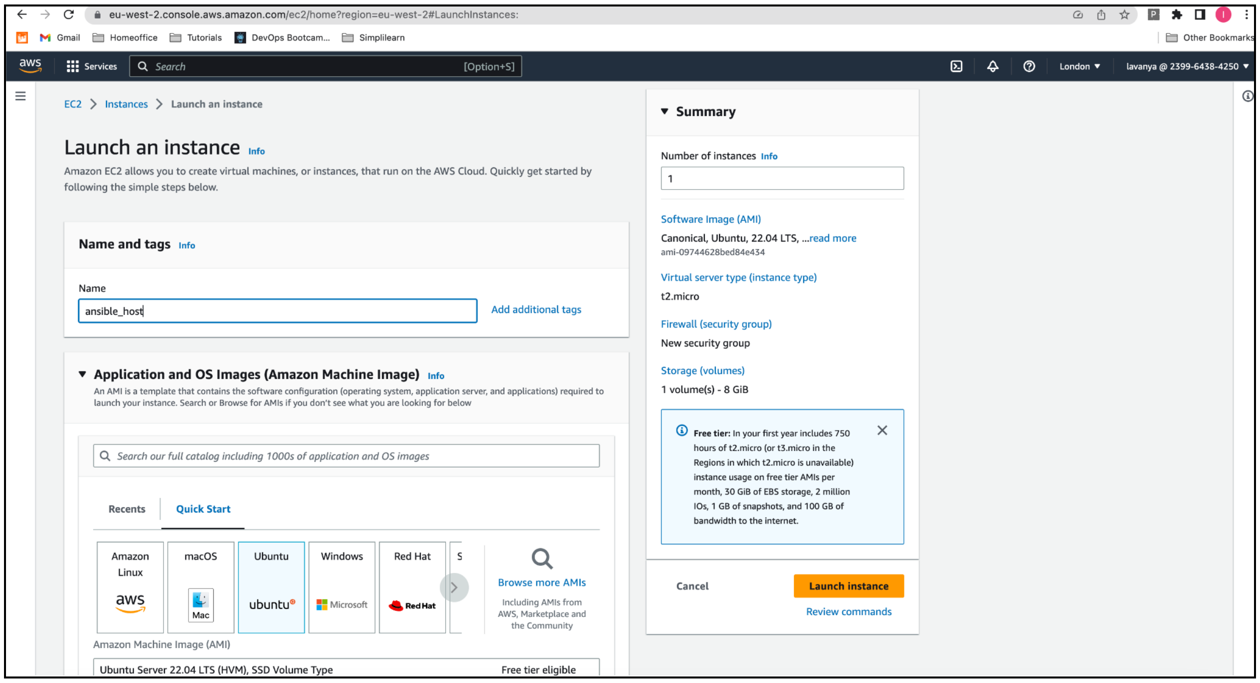
lavanya

1 match

User name	Groups	Last activity	MFA	Password age	Active key age
lavanya	aws_tutorial	✓ 4 hours ago		✓ Yesterday	

Launch an EC2 instance from AWS web console

- Create an EC2 instance using Ubuntu AMI from AWS web console. Terraform and Ansible will be installed in this node



eu-west-2.console.aws.amazon.com/ec2/home?region=eu-west-2#LaunchInstances:

EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: ansible_host

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | S

Browse more AMIs — Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI): Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible

Summary

Number of instances Info: 1

Software Image (AMI): Canonical, Ubuntu, 22.04 LTS, ...read more — ami-09744628bed84e434

Virtual server type (instance type): t2.micro

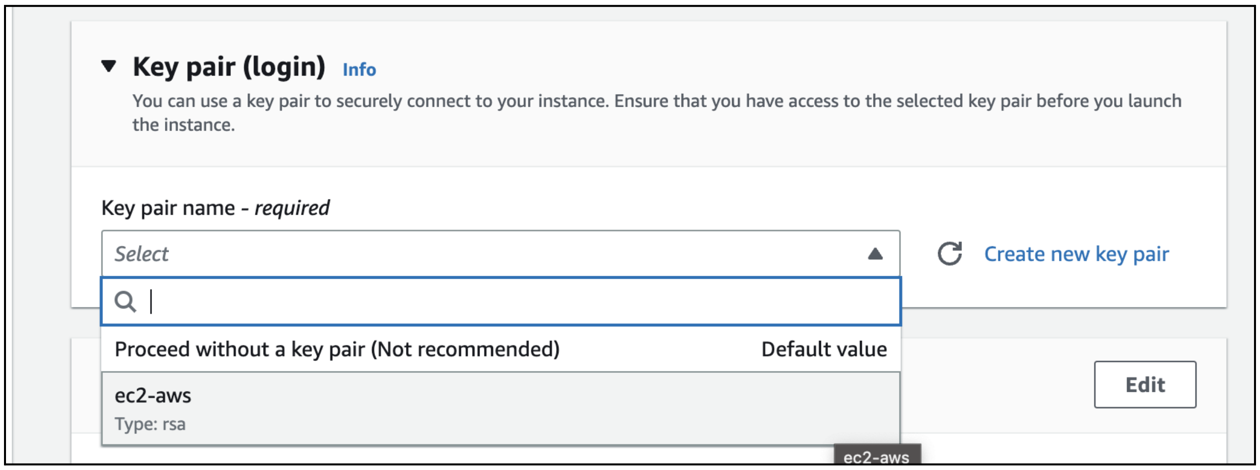
Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel | Launch instance | Review commands

- Create a new SSH keypair. This will be used to connect to this instance from local machine



Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Select

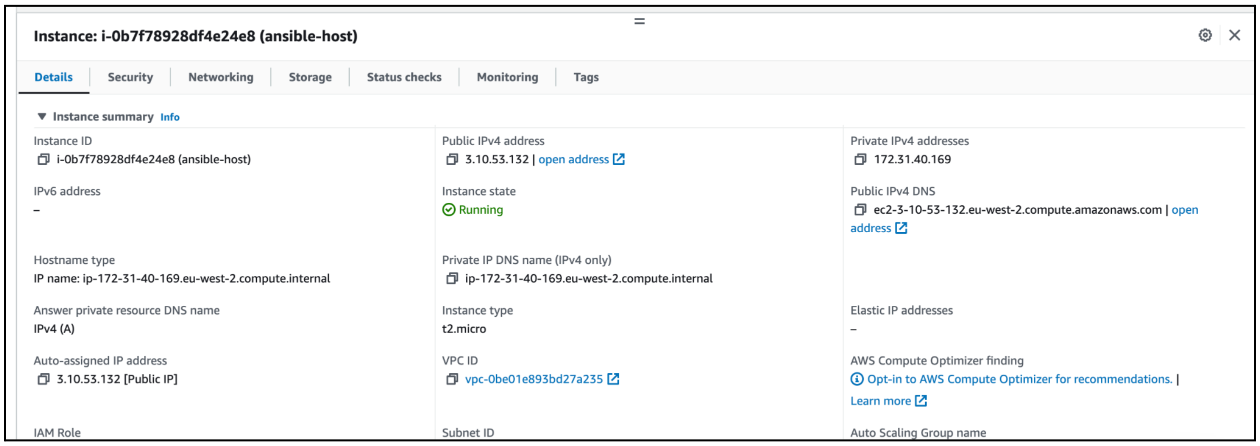
Create new key pair

Proceed without a key pair (Not recommended) — Default value

ec2-aws
Type: rsa

Edit

- Select the other default configuration.



Instance: i-0b7f78928df4e24e8 (ansible-host)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

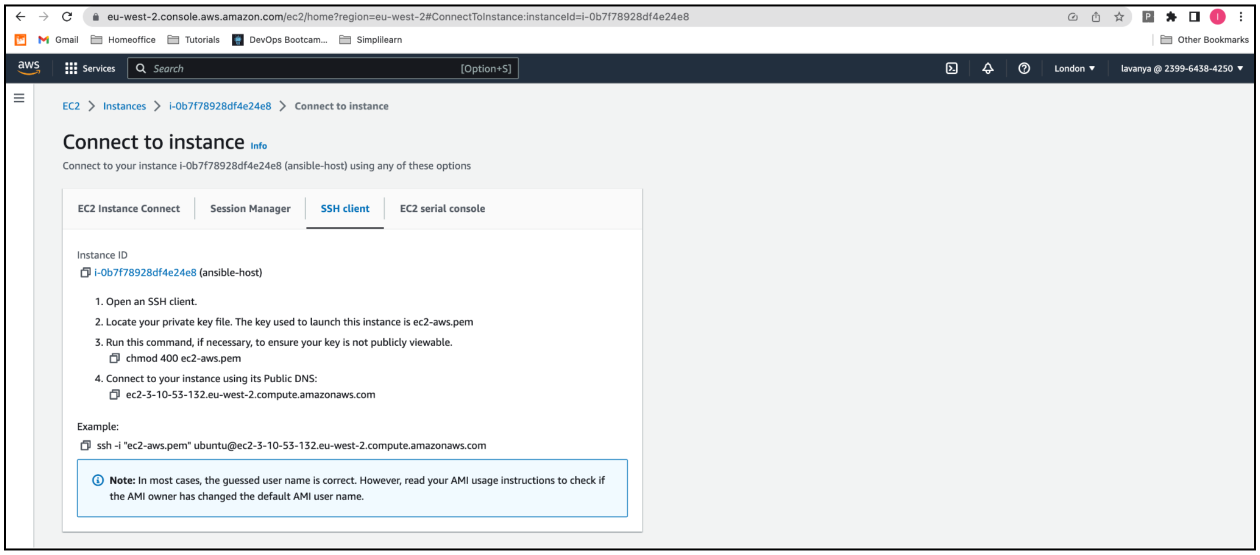
Instance summary Info

Instance ID i-0b7f78928df4e24e8 (ansible-host)	Public IPv4 address 3.10.53.132 | open address	Private IPv4 addresses 172.31.40.169
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-3-10-53-132.eu-west-2.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-40-169.eu-west-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-40-169.eu-west-2.compute.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 3.10.53.132 [Public IP]	VPC ID vpc-0be01e893bd27a235	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role	Subnet ID	Auto Scaling Group name

Connect to the EC2 Ubuntu instance from local machine

- Below command to be used to the connect to the instance created above. This command can be obtained from AWS console

```
sudo ssh -i "ec2-aws.pem" ubuntu@ec2-3-10-53-132.eu-west-2.compute.amazonaws.com
-oHostKeyAlgorithms=+rsa-sha2-512
```



Connect to instance Info

Connect to your instance i-0b7f78928df4e24e8 (ansible-host) using any of these options

Instance ID

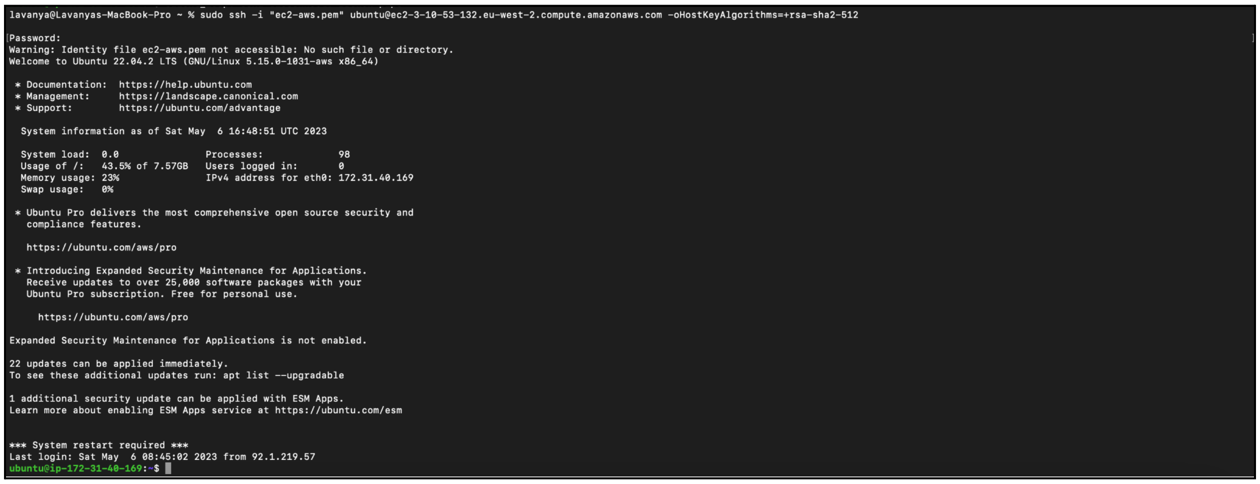
i-0b7f78928df4e24e8 (ansible-host)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is ec2-aws.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 ec2-aws.pem
4. Connect to your instance using its Public DNS:
   ec2-3-10-53-132.eu-west-2.compute.amazonaws.com

Example:

```
ssh -i "ec2-aws.pem" ubuntu@ec2-3-10-53-132.eu-west-2.compute.amazonaws.com
```

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

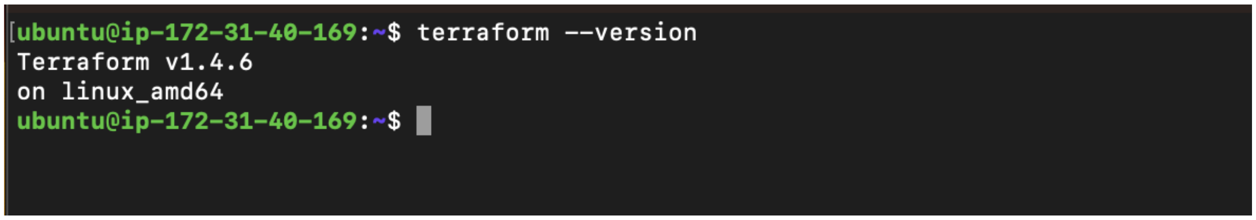


```
lavanya@Lavanyas-MacBook-Pro ~ % sudo ssh -i "ec2-aws.pem" ubuntu@ec2-3-10-53-132.eu-west-2.compute.amazonaws.com -oHostKeyAlgorithms=+rsa-sha2-512

Password:
Warning: Identity file ec2-aws.pem not accessible: No such file or directory.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-1031-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat May  6 16:48:51 UTC 2023

  System load:  0.0               Processes:             98
  Usage of /:   43.5% of 7.57GB   Users logged in:       0
  Memory usage: 23%               IPv4 address for eth0: 172.31.40.169
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

22 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


*** System restart required ***
Last login: Sat May  6 08:45:02 2023 from 92.1.219.57
ubuntu@ip-172-31-40-169:~$
```

Install Terraform

Below list of commands are used to install terraform on the EC2 Ubuntu instance. Obtained from https://developer.hashicorp.com/terraform/downloads?product_intent=terraform

- **wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg**
- **echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list**
- **sudo apt update && sudo apt install terraform**

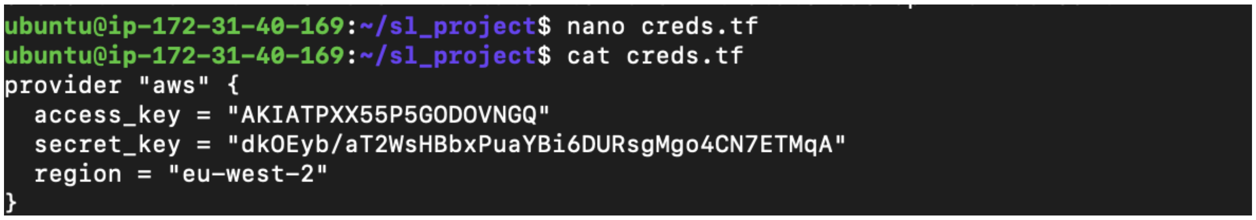


```
ubuntu@ip-172-31-40-169:~$ terraform --version
Terraform v1.4.6
on linux_amd64
ubuntu@ip-172-31-40-169:~$
```

Launch an Ubuntu EC2 instance using Terraform

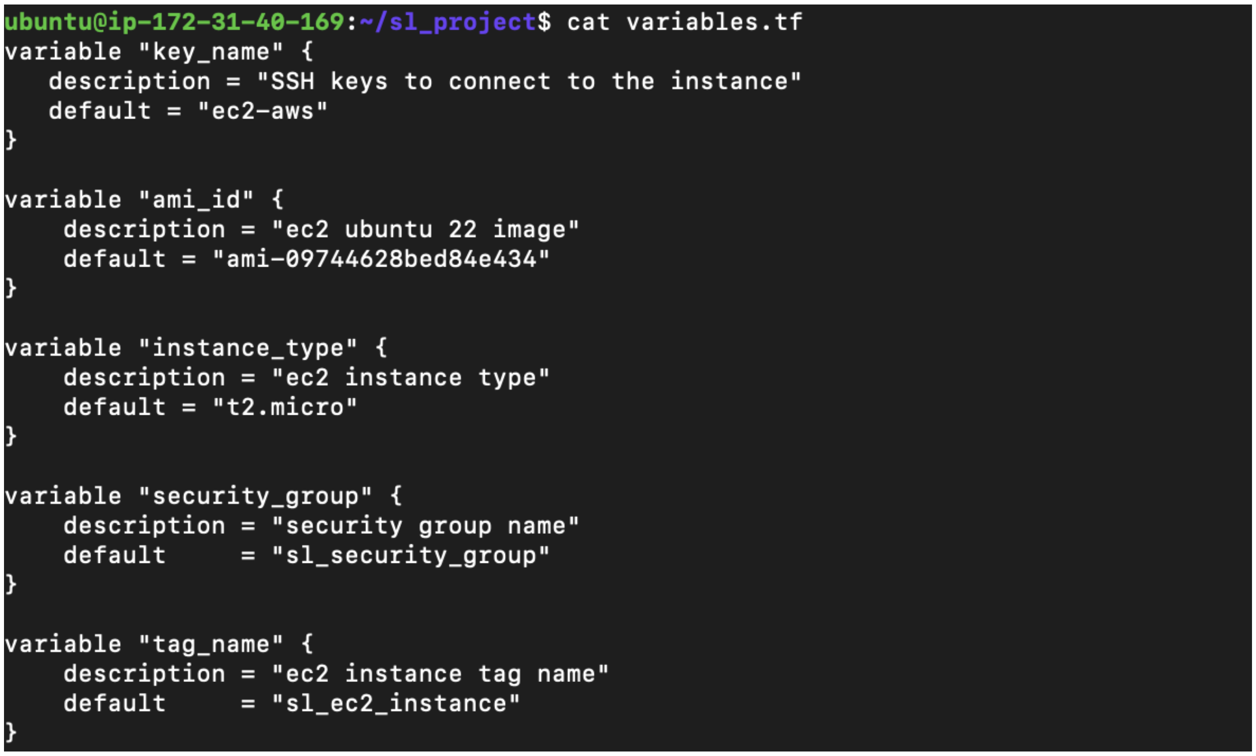
1. Create Terraform files

- Step 1 : **$ nano creds.tf**



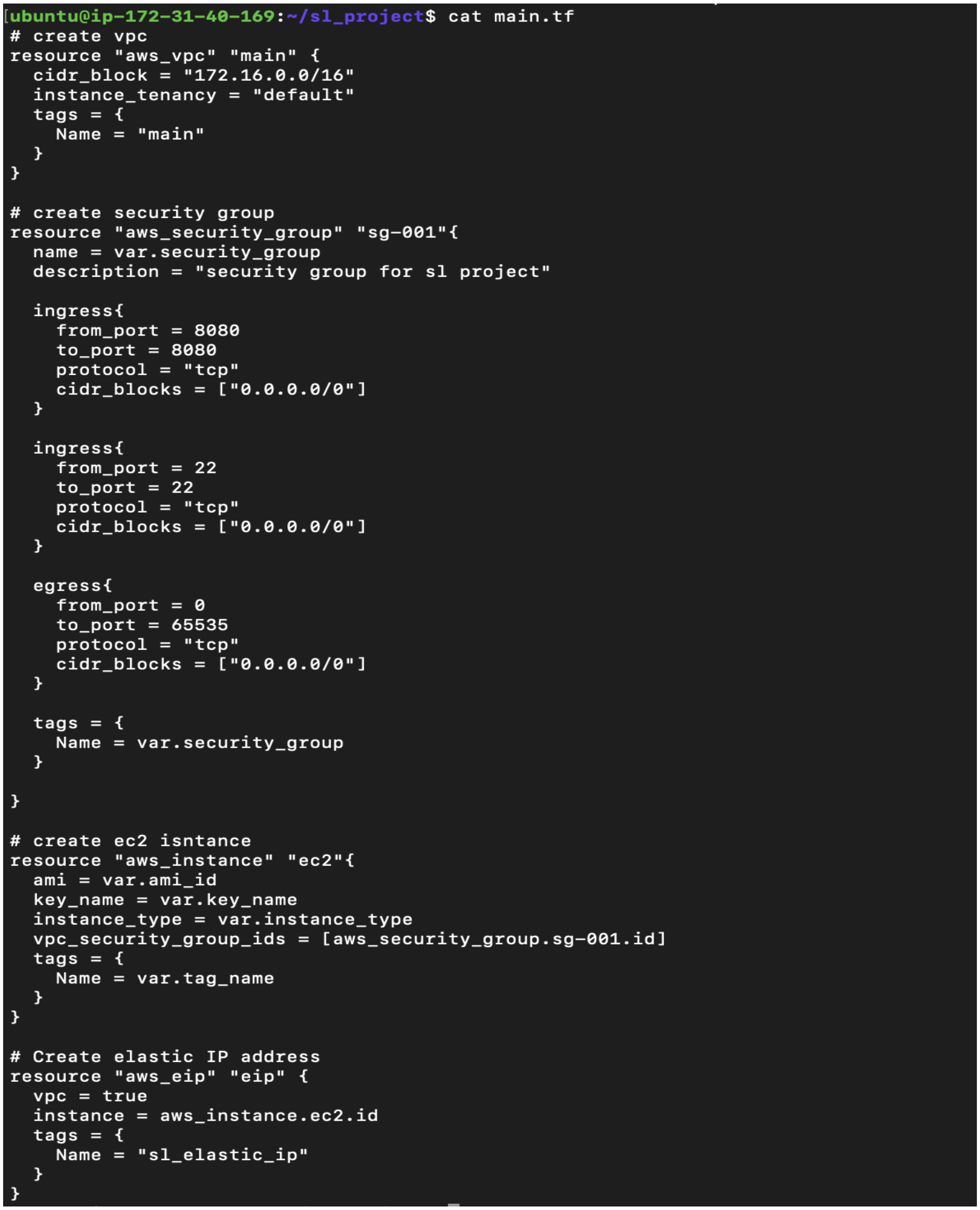
```
ubuntu@ip-172-31-40-169:~/sl_project$ nano creds.tf
ubuntu@ip-172-31-40-169:~/sl_project$ cat creds.tf
provider "aws" {
  access_key = "AKIATPXX55P5GODOVNGQ"
  secret_key = "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"
  region = "eu-west-2"
}
```

- Step 2: **$ nano variables.tf**



```
ubuntu@ip-172-31-40-169:~/sl_project$ cat variables.tf
variable "key_name" {
   description = "SSH keys to connect to the instance"
   default = "ec2-aws"
}

variable "ami_id" {
    description = "ec2 ubuntu 22 image"
    default = "ami-09744628bed84e434"
}

variable "instance_type" {
    description = "ec2 instance type"
    default = "t2.micro"
}

variable "security_group" {
    description = "security group name"
    default     = "sl_security_group"
}

variable "tag_name" {
    description = "ec2 instance tag name"
    default     = "sl_ec2_instance"
}
```

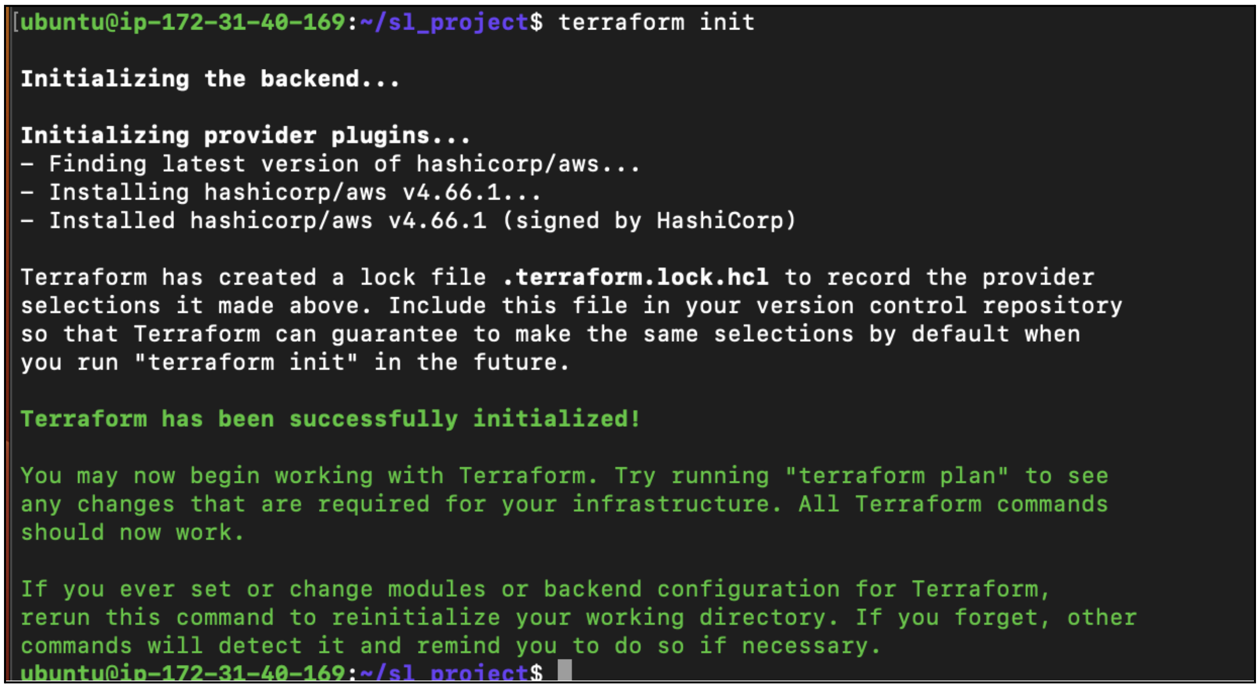
- Step 3: **$ nano main.tf**



```
ubuntu@ip-172-31-40-169:~/sl_project$ cat main.tf
# create vpc
resource "aws_vpc" "main" {
  cidr_block = "172.16.0.0/16"
  instance_tenancy = "default"
  tags = {
    Name = "main"
  }
}

# create security group
resource "aws_security_group" "sg-001"{
  name = var.security_group
  description = "security group for sl project"

  ingress{
    from_port = 8080
    to_port = 8080
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress{
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress{
    from_port = 0
    to_port = 65535
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = var.security_group
  }

}

# create ec2 isntance
resource "aws_instance" "ec2"{
  ami = var.ami_id
  key_name = var.key_name
  instance_type = var.instance_type
  vpc_security_group_ids = [aws_security_group.sg-001.id]
  tags = {
    Name = var.tag_name
  }
}

# Create elastic IP address
resource "aws_eip" "eip" {
  vpc = true
  instance = aws_instance.ec2.id
  tags = {
    Name = "sl_elastic_ip"
  }
}
```

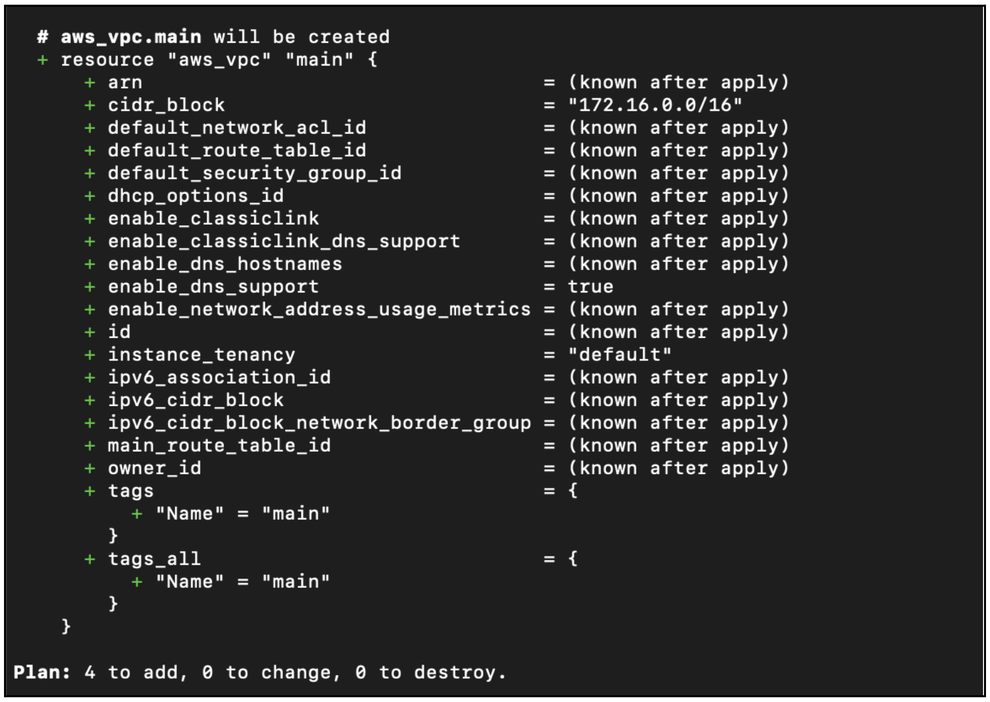
2. Execute Terraform Commands

- Step 1: $ terraform init



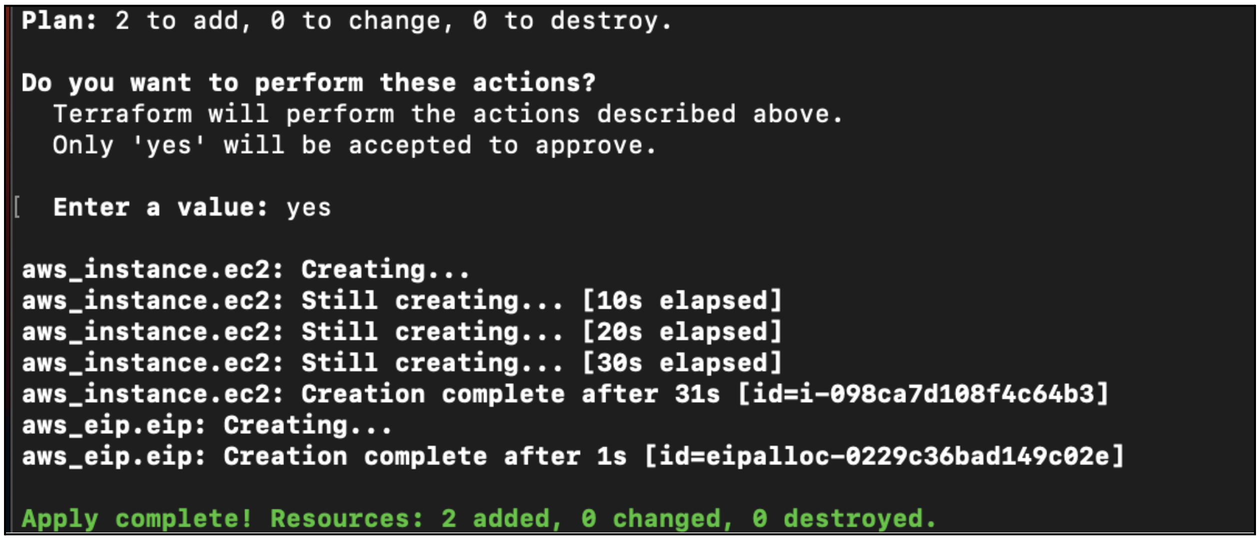
```
[ubuntu@ip-172-31-40-169:~/sl_project$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.66.1...
- Installed hashicorp/aws v4.66.1 (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.
ubuntu@ip-172-31-40-169:~/sl_project$
```

- Step 2: $ terraform plan



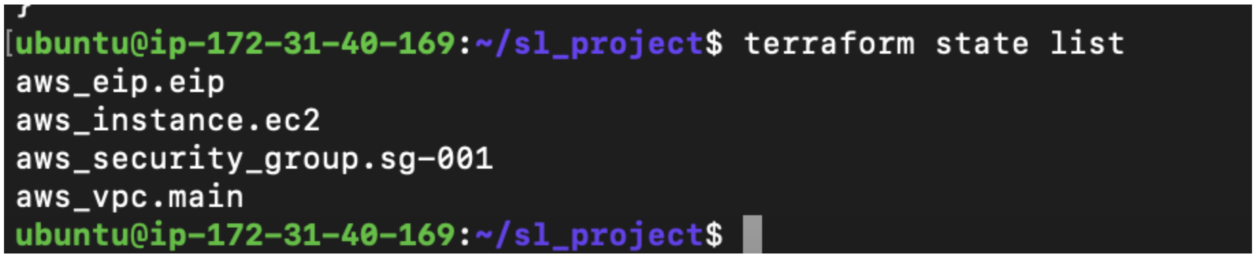
```
  # aws_vpc.main will be created
  + resource "aws_vpc" "main" {
      + arn                                  = (known after apply)
      + cidr_block                           = "172.16.0.0/16"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_classiclink                   = (known after apply)
      + enable_classiclink_dns_support       = (known after apply)
      + enable_dns_hostnames                 = (known after apply)
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "main"
        }
      + tags_all                             = {
          + "Name" = "main"
        }
    }

Plan: 4 to add, 0 to change, 0 to destroy.
```

- Step 3: $ terraform apply



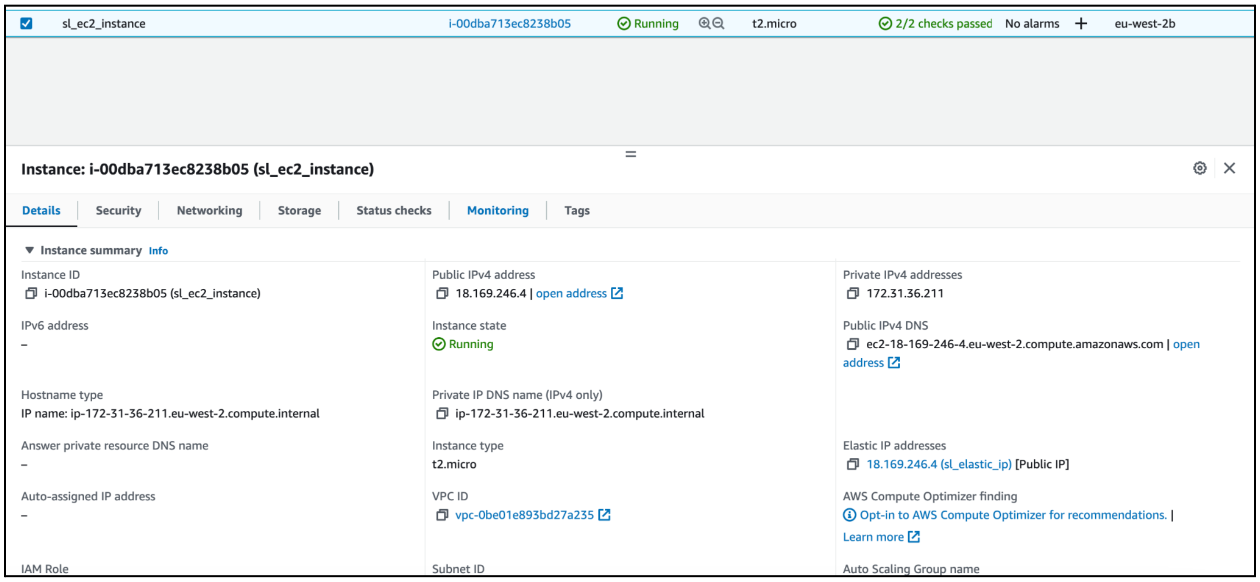
```
Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.ec2: Creating...
aws_instance.ec2: Still creating... [10s elapsed]
aws_instance.ec2: Still creating... [20s elapsed]
aws_instance.ec2: Still creating... [30s elapsed]
aws_instance.ec2: Creation complete after 31s [id=i-098ca7d108f4c64b3]
aws_eip.eip: Creating...
aws_eip.eip: Creation complete after 1s [id=eipalloc-0229c36bad149c02e]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

- Step 4: $ terraform state list



```
ubuntu@ip-172-31-40-169:~/sl_project$ terraform state list
aws_eip.eip
aws_instance.ec2
aws_security_group.sg-001
aws_vpc.main
ubuntu@ip-172-31-40-169:~/sl_project$
```

3. Check the new EC2 instance in AWS console



sl_ec2_instance | i-00dba713ec8238b05 | Running | t2.micro | 2/2 checks passed | No alarms | eu-west-2b

Instance: i-00dba713ec8238b05 (sl_ec2_instance)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

Instance summary Info

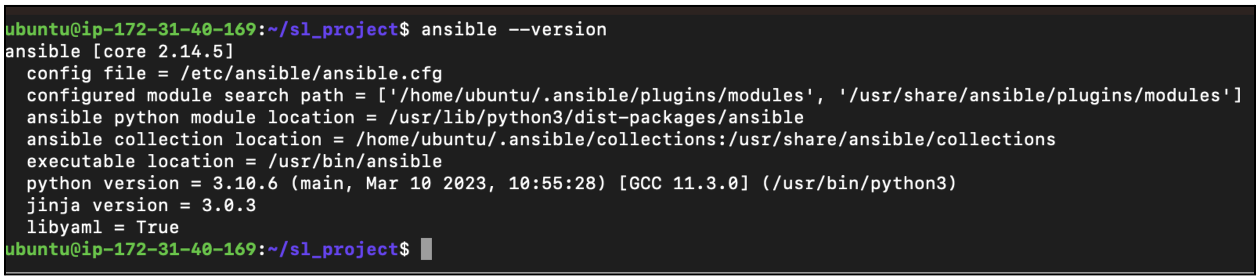
Instance ID i-00dba713ec8238b05 (sl_ec2_instance)	Public IPv4 address 18.169.246.4 | open address	Private IPv4 addresses 172.31.36.211
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-18-169-246-4.eu-west-2.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-36-211.eu-west-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-36-211.eu-west-2.compute.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses 18.169.246.4 (sl_elastic_ip) [Public IP]
Auto-assigned IP address –	VPC ID vpc-0be01e893bd27a235	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role	Subnet ID	Auto Scaling Group name

Setup Ansible Control Machine

1. Install ansible

   Configure the PPA on the control machine(Ansible for Ubuntu builds are available in a PPA here) and to install ansible run these commands:

   - **$ sudo apt-get install software-properties-common**
   - **$ sudo apt-add-repository ppa:ansible/ansible**
   - **$ sudo apt-get update**
   - **$ sudo apt-get install ansible**

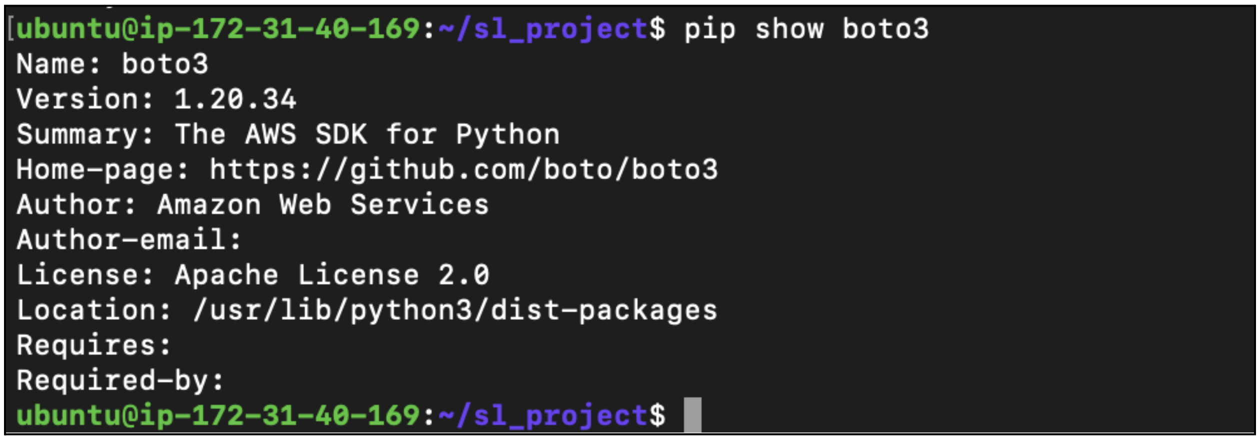


```
ubuntu@ip-172-31-40-169:~/sl_project$ ansible --version
ansible [core 2.14.5]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/ubuntu/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
ubuntu@ip-172-31-40-169:~/sl_project$
```

2. Install boto framework

   Boto is a software development kit (SDK) designed to improve the use of the Python programming language in Amazon Web Services.

   - **$ sudo apt update**
   - **$ sudo apt install python3-boto3**
   - **$ sudo apt install python3-pip**
   - **$ pip show boto3**



```
ubuntu@ip-172-31-40-169:~/sl_project$ pip show boto3
Name: boto3
Version: 1.20.34
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache License 2.0
Location: /usr/lib/python3/dist-packages
Requires:
Required-by:
ubuntu@ip-172-31-40-169:~/sl_project$
```

Install Java, Maven, Jenkins in the ansible node using ansible-playbook

1. Setup ansible node

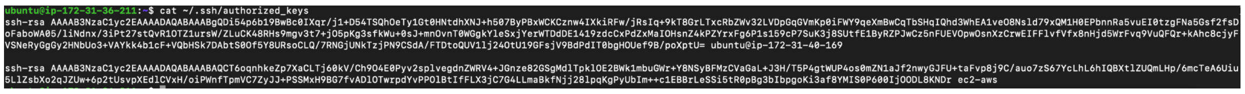
   In order to make EC2 instance created using terraform as ansible node, perform the following steps

   - Step 1: Create SSH keys in ansible host machine
     Use the below command to create SSH keypair to connect the ansible host machine with the EC2 instance(ansible node)
     - **$ ssh-keygen**

   - Step 2: Connect to the ansible node from local machine using the below command

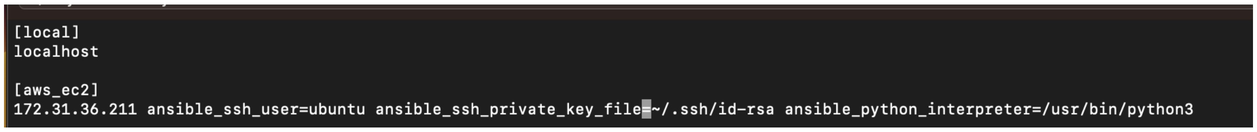
     ```
     sudo  ssh -i "ec2-aws.pem" ubuntu@ec2-18-169-246-4.eu-west-2.compute.amazonaws.com
     -oHostKeyAlgorithms=+rsa-sha2-512
     ```

   - Step 3: Copy the ansible_host public key to the ansible node ~/.ssh/authorised_keys file



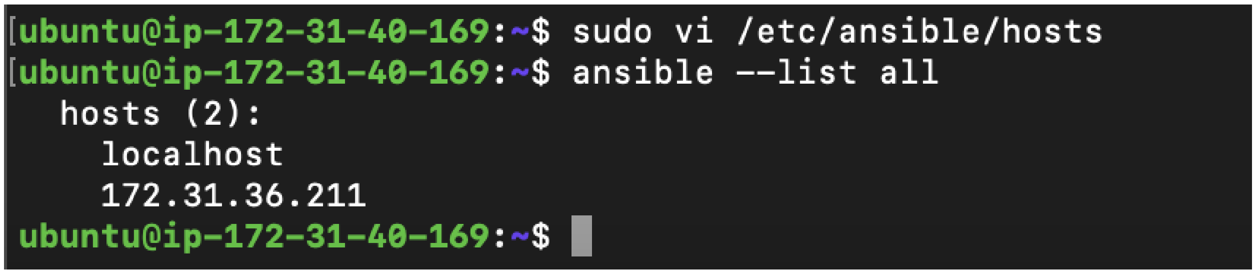
```
ubuntu@ip-172-31-36-211:~$ cat ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDi54p6b19BwBc0IXqr/j1+D54TSQhOeTy1Gt0HNtdhXNJ+h507ByPBxWCKCznw4IXkiRFw/jRsIq+9kT8GrLTxcRbZWv32LVDpGqGVmKp0iFWY9qeXmBwCqTbSHqIQhd3WhEA1veO8Nsld79xQM1H0EPbnnRa5vuEI0tzgFNa5Gsf2fsD
oFaboWA05/liNdnx/3iPt27stQvR1OTZ1ursW/ZLuCK48RHs9mgv3t7+jO5pKg3sfkWu+0sJ+mnOvnT0WGgkYleSxjYerWTDdDE1419zdcCxPdZxMaIOHsnZ4kPZYrxFg6P1s159cP7SuK3j8SUtfE1ByRZPJwCz5nFUEVOpwOsnXzCrwEIFFlvfVfx8nHjd5WrFvq9VuQFQr+kAhc8cjyF
VSNeRyGgGy2HNbUo3+VAYkk4b1cF+VQbHSk7DAbtS0Of5Y8URsoCLQ/7RNGjUNkTzjPN9CSdA/FTDtoQUV1lj24OtU19GFsjV9BdPdIT0bgHOUef9B/poXptU= ubuntu@ip-172-31-40-169

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCT6oqnhkeZp7XaCLTj60kV/Ch9O4E0Pyv2splvegdnZWRV4+JGnze82GSgMdlTpklOE2BWk1mbuGWr+Y8NSyBFMzCVaGaL+J3H/T5P4gtWUP4os0mZN1aJf2nwyGJFU+taFvp8j9C/auo7zS67YcLhL6hIQBXtlZUQmLHp/6mcTeA6Uiu
5LlZsbXo2qJZUw+6p2tUsvpXEdlCVxH/oiPWnfTpmVC7ZyJJ+PSSMxH9BG7fvADlOTwrpdYvPPOlBtIfFLX3jC7G4LLmaBkfNjj28lpqKgPyUbIm++c1EBBrLeSSi5tR0pBg3bIbpgoKi3af8YMIS0P600IjOODL8KNDr ec2-aws
```

   - Step 4: Make an entry about the ansible node in the ansible host /etc/ansible/hosts file



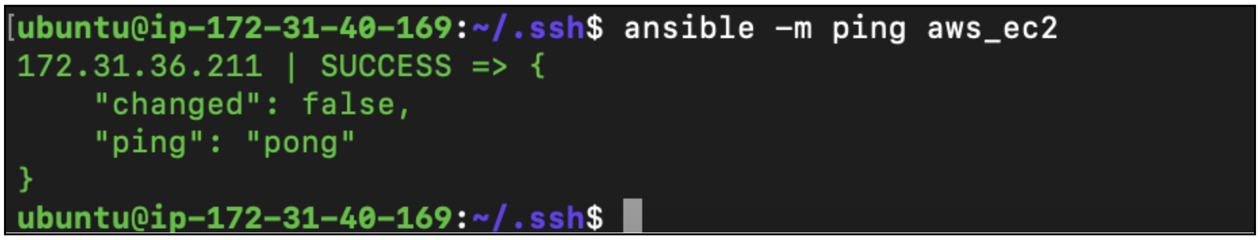
```
[local]
localhost

[aws_ec2]
172.31.36.211 ansible_ssh_user=ubuntu ansible_ssh_private_key_file=~/.ssh/id-rsa ansible_python_interpreter=/usr/bin/python3
```

   - Step 5: Verify the entries in ansible hosts file
     - **$ ansible –list all**



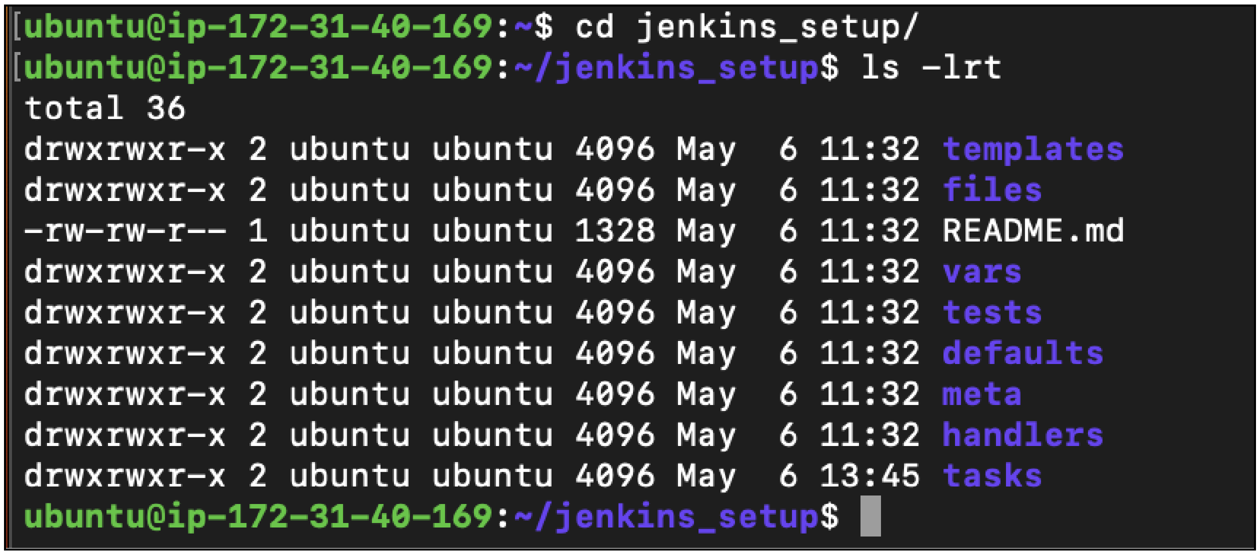
```
ubuntu@ip-172-31-40-169:~$ sudo vi /etc/ansible/hosts
ubuntu@ip-172-31-40-169:~$ ansible --list all
  hosts (2):
    localhost
    172.31.36.211
ubuntu@ip-172-31-40-169:~$
```

- Step 6: Verify ansible host is able to ping the ansible node



```
ubuntu@ip-172-31-40-169:~/.ssh$ ansible -m ping aws_ec2
172.31.36.211 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ubuntu@ip-172-31-40-169:~/.ssh$
```

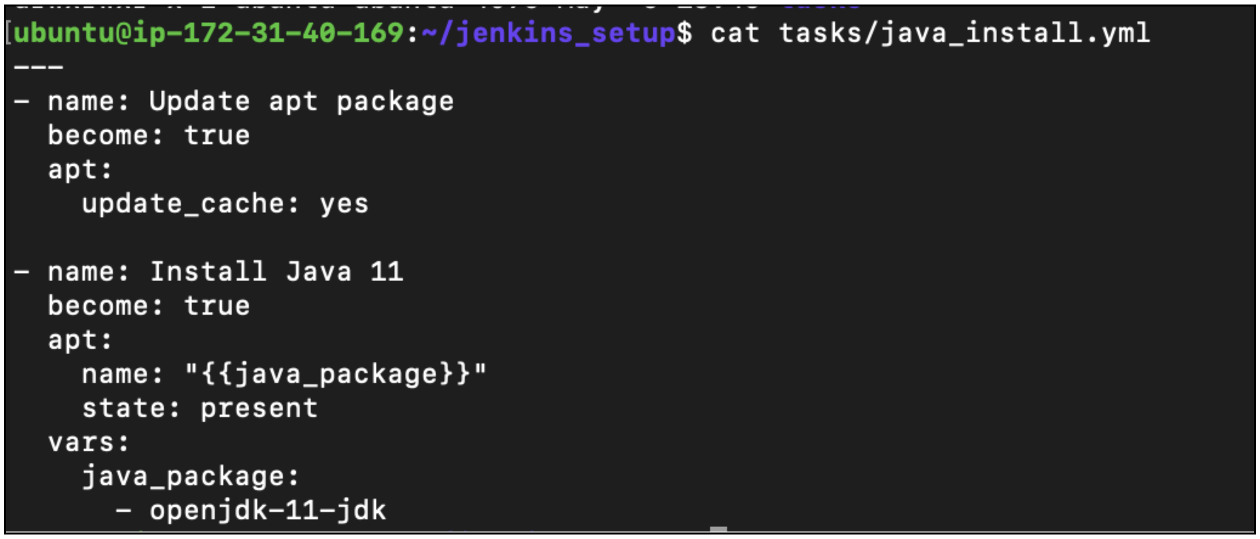
2. Create ansible files
    - Step 1: Setup a role called jenkins_setup using ansible-galaxy on the ansible host
        - **$ ansible-galaxy init jenkins_setup**



```
ubuntu@ip-172-31-40-169:~$ cd jenkins_setup/
ubuntu@ip-172-31-40-169:~/jenkins_setup$ ls -lrt
total 36
drwxrwxr-x 2 ubuntu ubuntu 4096 May  6 11:32 templates
drwxrwxr-x 2 ubuntu ubuntu 4096 May  6 11:32 files
-rw-rw-r-- 1 ubuntu ubuntu 1328 May  6 11:32 README.md
drwxrwxr-x 2 ubuntu ubuntu 4096 May  6 11:32 vars
drwxrwxr-x 2 ubuntu ubuntu 4096 May  6 11:32 tests
drwxrwxr-x 2 ubuntu ubuntu 4096 May  6 11:32 defaults
drwxrwxr-x 2 ubuntu ubuntu 4096 May  6 11:32 meta
drwxrwxr-x 2 ubuntu ubuntu 4096 May  6 11:32 handlers
drwxrwxr-x 2 ubuntu ubuntu 4096 May  6 13:45 tasks
ubuntu@ip-172-31-40-169:~/jenkins_setup$
```

- Step 2: Create a task to install Java by creating java_install.yml under jenkins_setup/tasks folder with the following contents

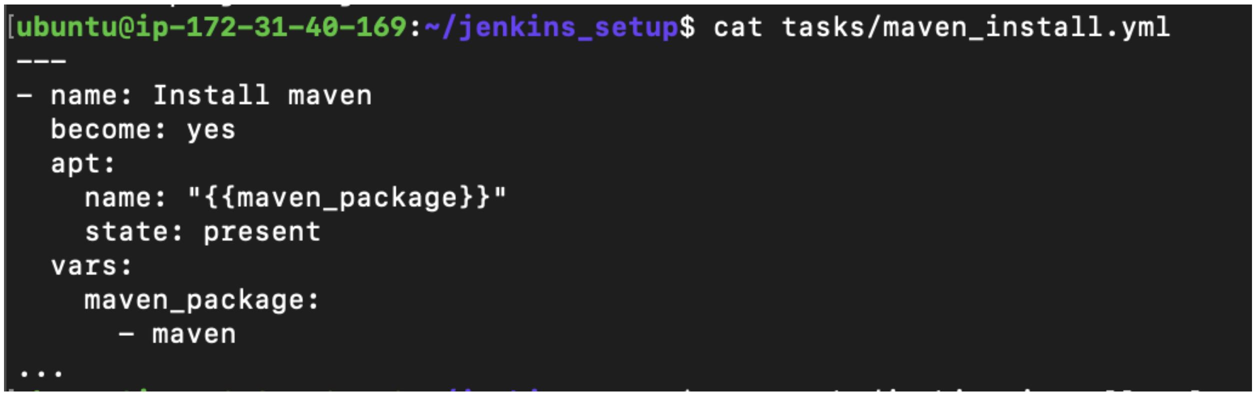
```
---
- name: Update apt package
  become: true
  apt:
    update_cache: yes

- name: Install Java 11
  become: true
  apt:
    name: "{{java_package}}"
    state: present
  vars:
    java_package:
      - openjdk-11-jdk
```



```
[ubuntu@ip-172-31-40-169:~/jenkins_setup$ cat tasks/java_install.yml
---
- name: Update apt package
  become: true
  apt:
    update_cache: yes

- name: Install Java 11
  become: true
  apt:
    name: "{{java_package}}"
    state: present
  vars:
    java_package:
      - openjdk-11-jdk
```

- Step 3: Create a task to install Maven by creating maven_install.yml under jenkins_setup/tasks folder with the following contents

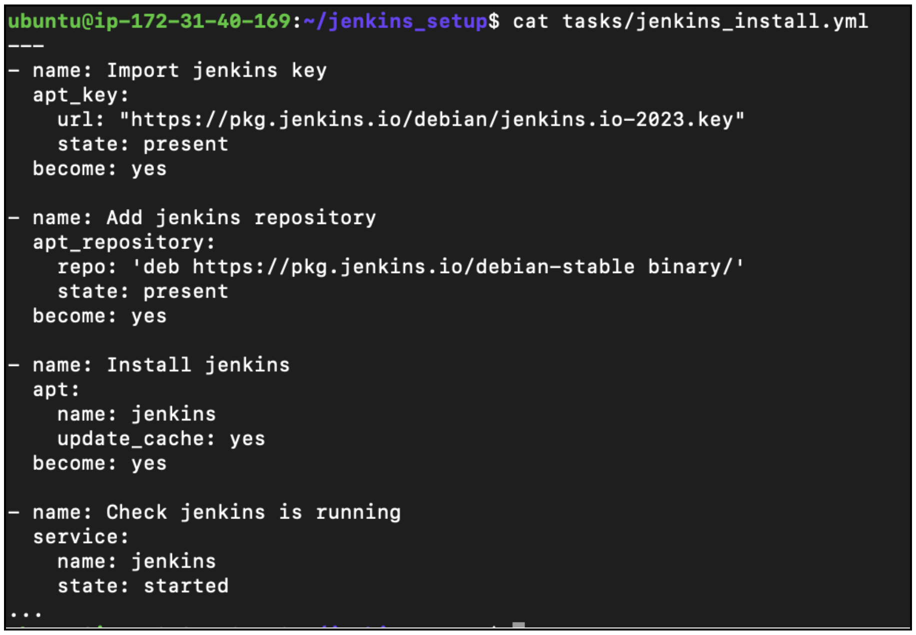
```
---
- name: Install maven
  become: yes
  apt:
    name: "{{maven_package}}"
    state: present
  vars:
    maven_package:
      - maven
...
```



```
[ubuntu@ip-172-31-40-169:~/jenkins_setup$ cat tasks/maven_install.yml
---
- name: Install maven
  become: yes
  apt:
    name: "{{maven_package}}"
    state: present
  vars:
    maven_package:
      - maven
...
```

- Step 4: Create a task to install jenkins by creating jenkins_install.yml under jenkins_setup/tasks folder with the following contents

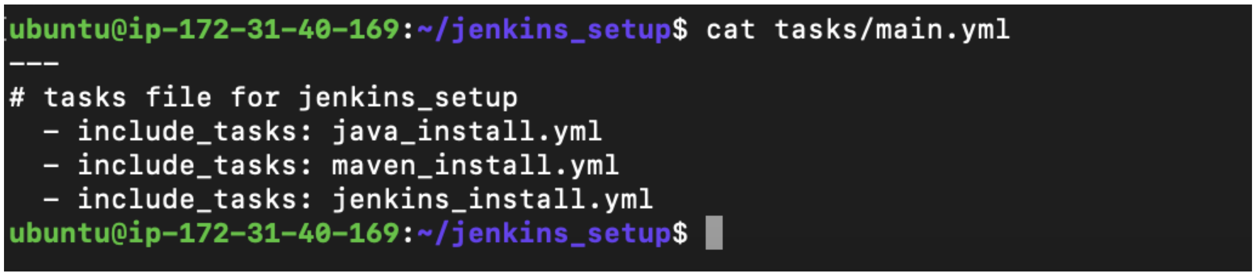
```
---
- name: Import jenkins key
  apt_key:
    url: "https://pkg.jenkins.io/debian/jenkins.io-2023.key"
    state: present
  become: yes

- name: Add jenkins repository
  apt_repository:
    repo: 'deb https://pkg.jenkins.io/debian-stable binary/'
    state: present
  become: yes

- name: Install jenkins
  apt:
    name: jenkins
    update_cache: yes
  become: yes

- name: Check jenkins is running
  service:
    name: jenkins
    state: started
...
```



```
ubuntu@ip-172-31-40-169:~/jenkins_setup$ cat tasks/jenkins_install.yml
---
- name: Import jenkins key
  apt_key:
    url: "https://pkg.jenkins.io/debian/jenkins.io-2023.key"
    state: present
  become: yes

- name: Add jenkins repository
  apt_repository:
    repo: 'deb https://pkg.jenkins.io/debian-stable binary/'
    state: present
  become: yes

- name: Install jenkins
  apt:
    name: jenkins
    update_cache: yes
  become: yes

- name: Check jenkins is running
  service:
    name: jenkins
    state: started
...
```

- Step 5: Include all the above tasks in the main.yaml under jenkins_setup/tasks folder with the following contents

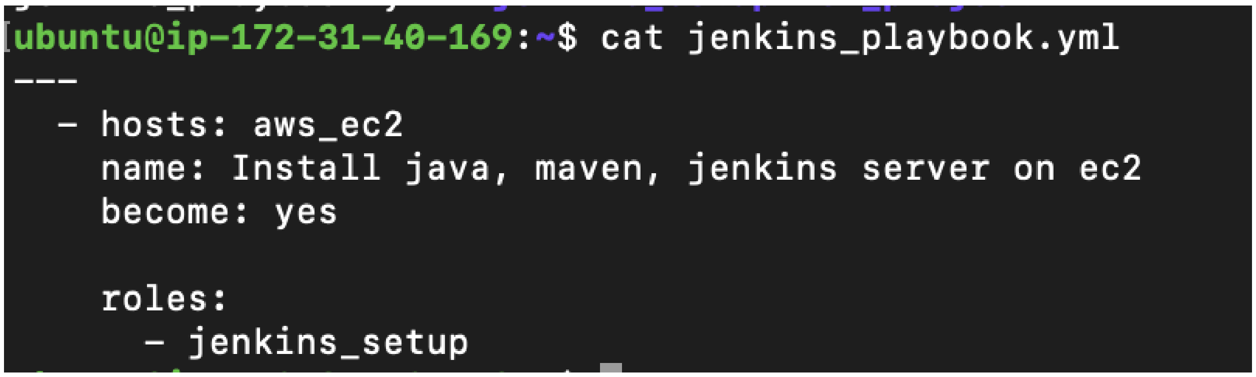
```
---
# tasks file for jenkins_setup
  - include_tasks: java_install.yml
  - include_tasks: maven_install.yml
  - include_tasks: jenkins_install.yml
```



```
ubuntu@ip-172-31-40-169:~/jenkins_setup$ cat tasks/main.yml
---
# tasks file for jenkins_setup
  - include_tasks: java_install.yml
  - include_tasks: maven_install.yml
  - include_tasks: jenkins_install.yml
ubuntu@ip-172-31-40-169:~/jenkins_setup$
```

- Step 6: Create the playbook and add the jenkins_setup role to it

```
---
  - hosts: aws_ec2
    name: Install java, maven, jenkins server on ec2
    become: yes

    roles:
      - jenkins_setup
```

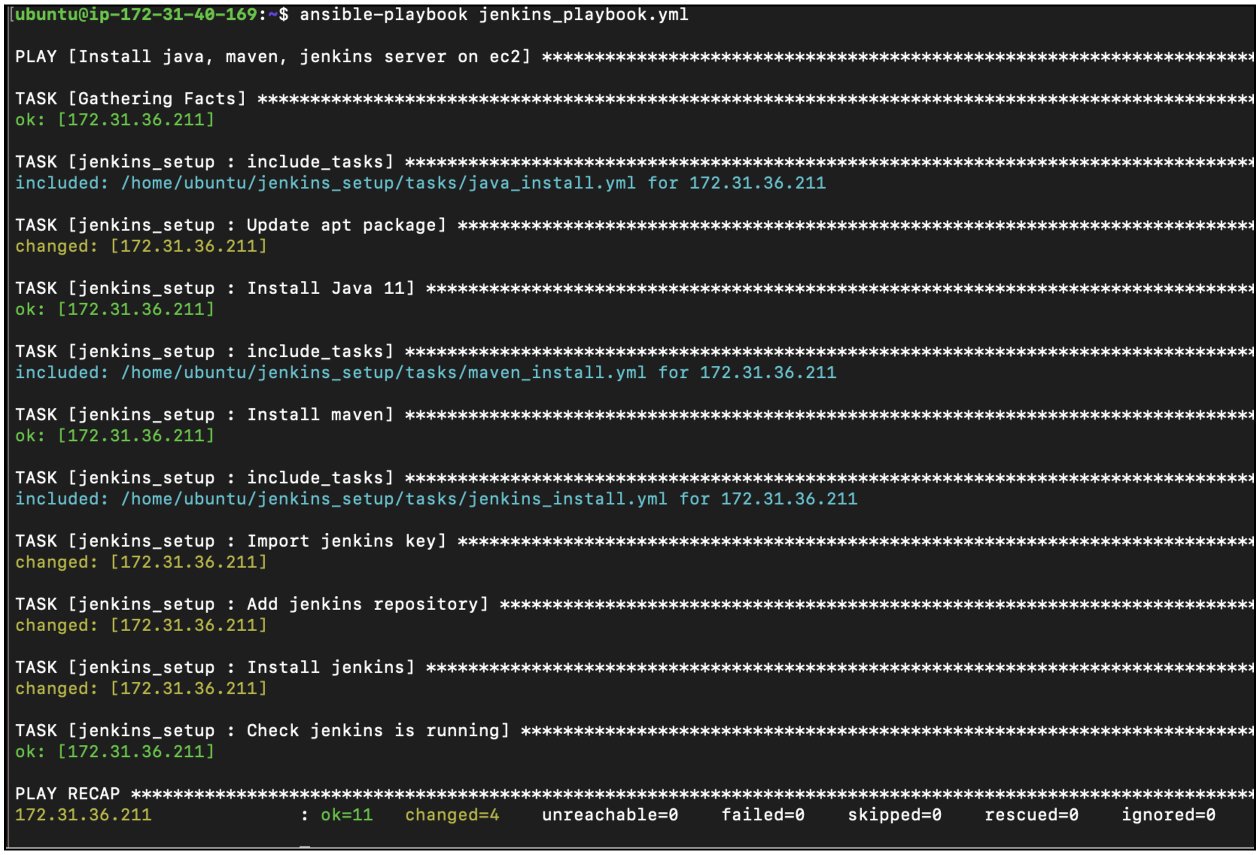


```
ubuntu@ip-172-31-40-169:~$ cat jenkins_playbook.yml
---
  - hosts: aws_ec2
    name: Install java, maven, jenkins server on ec2
    become: yes

    roles:
      - jenkins_setup
```

3. Run the playbook

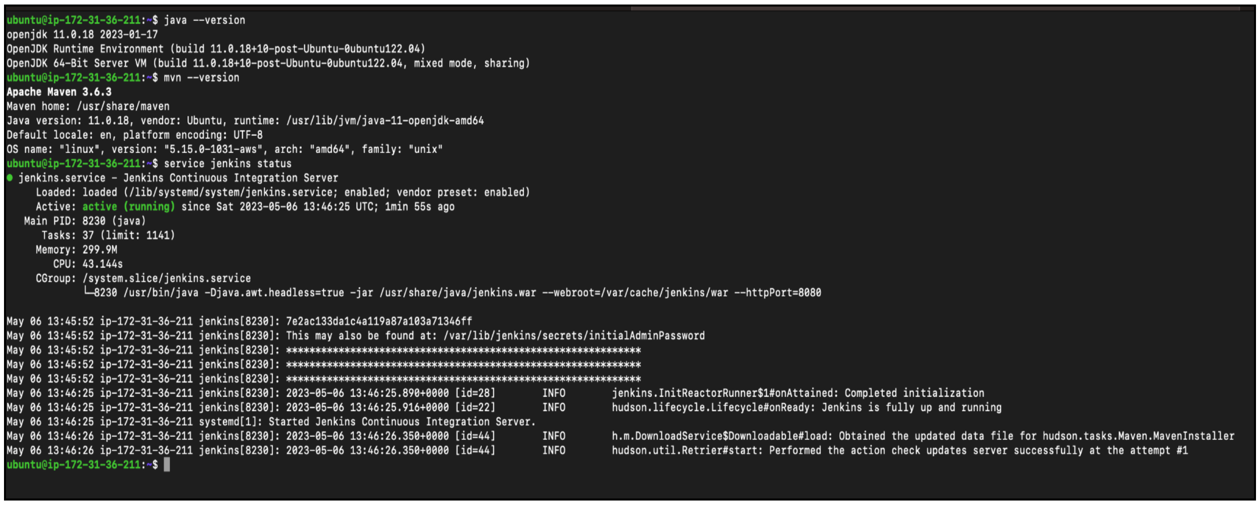
   Run the following command to run the jenkins_playbook.yml so that Java, Maven, Jenkins would be installed in the ansible node

   - **$ ansible-playbook jenkins_playbook.yml**



```
ubuntu@ip-172-31-40-169:~$ ansible-playbook jenkins_playbook.yml

PLAY [Install java, maven, jenkins server on ec2] ******************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [172.31.36.211]

TASK [jenkins_setup : include_tasks] *******************************************************************************
included: /home/ubuntu/jenkins_setup/tasks/java_install.yml for 172.31.36.211

TASK [jenkins_setup : Update apt package] **************************************************************************
changed: [172.31.36.211]

TASK [jenkins_setup : Install Java 11] *****************************************************************************
ok: [172.31.36.211]

TASK [jenkins_setup : include_tasks] *******************************************************************************
included: /home/ubuntu/jenkins_setup/tasks/maven_install.yml for 172.31.36.211

TASK [jenkins_setup : Install maven] *******************************************************************************
ok: [172.31.36.211]

TASK [jenkins_setup : include_tasks] *******************************************************************************
included: /home/ubuntu/jenkins_setup/tasks/jenkins_install.yml for 172.31.36.211

TASK [jenkins_setup : Import jenkins key] **************************************************************************
changed: [172.31.36.211]

TASK [jenkins_setup : Add jenkins repository] **********************************************************************
changed: [172.31.36.211]

TASK [jenkins_setup : Install jenkins] *****************************************************************************
changed: [172.31.36.211]

TASK [jenkins_setup : Check jenkins is running] ********************************************************************
ok: [172.31.36.211]

PLAY RECAP *********************************************************************************************************
172.31.36.211              : ok=11   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

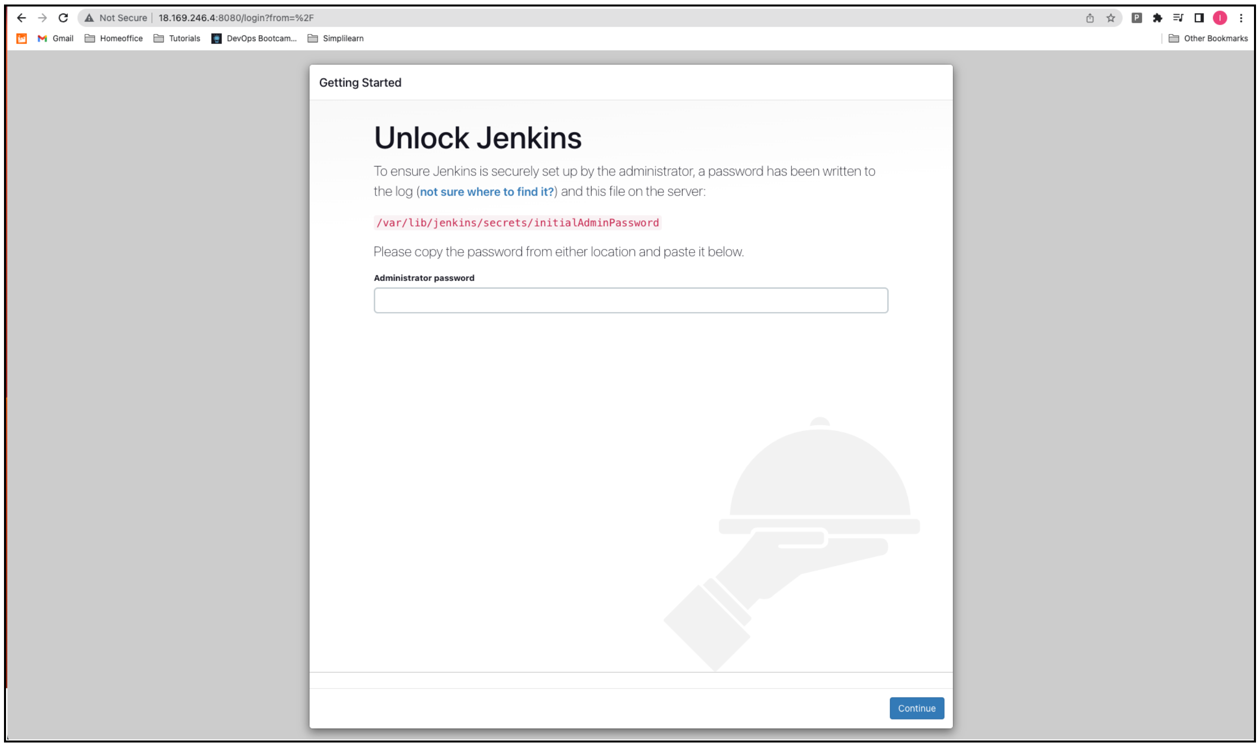
4. Check the installations in ansible node



```
ubuntu@ip-172-31-36-211:~$ java --version
openjdk 11.0.18 2023-01-17
OpenJDK Runtime Environment (build 11.0.18+10-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.18+10-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-36-211:~$ mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.18, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.15.0-1031-aws", arch: "amd64", family: "unix"
ubuntu@ip-172-31-36-211:~$ service jenkins status
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2023-05-06 13:46:25 UTC; 1min 55s ago
   Main PID: 8230 (java)
      Tasks: 37 (limit: 1141)
     Memory: 299.9M
        CPU: 43.144s
     CGroup: /system.slice/jenkins.service
             └─8230 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

May 06 13:45:52 ip-172-31-36-211 jenkins[8230]: 7e2ac133da1c4a119a87a103a71346ff
May 06 13:45:52 ip-172-31-36-211 jenkins[8230]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
May 06 13:45:52 ip-172-31-36-211 jenkins[8230]: *************************************************************
May 06 13:45:52 ip-172-31-36-211 jenkins[8230]: *************************************************************
May 06 13:45:52 ip-172-31-36-211 jenkins[8230]: *************************************************************
May 06 13:46:25 ip-172-31-36-211 jenkins[8230]: 2023-05-06 13:46:25.890+0000 [id=28]       INFO       jenkins.InitReactorRunner$1#onAttained: Completed initialization
May 06 13:46:25 ip-172-31-36-211 jenkins[8230]: 2023-05-06 13:46:25.916+0000 [id=22]       INFO       hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
May 06 13:46:25 ip-172-31-36-211 systemd[1]: Started Jenkins Continuous Integration Server.
May 06 13:46:26 ip-172-31-36-211 jenkins[8230]: 2023-05-06 13:46:26.350+0000 [id=44]       INFO       h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
May 06 13:46:26 ip-172-31-36-211 jenkins[8230]: 2023-05-06 13:46:26.350+0000 [id=44]       INFO       hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
ubuntu@ip-172-31-36-211:~$ 
```

Enter public ip address or public dns name with port no 8080 of ansible host in the browser to see Jenkins up and running.

- **http://18.169.246.4:8080/**



Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password