PROJECT DETAILS

Project Title

Automating Infrastructure using Terraform

Problem statement

Nowadays, infrastructure automation is critical. We tend to put the most emphasis on software development processes, but infrastructure deployment strategy is just as important. Infrastructure automation not only aids disaster recovery, but it also facilitates testing and development.

Your organization is adopting the DevOps methodology and in order to automate provisioning of infrastructure there's a need to setup a centralised server for Jenkins.

Terraform is a tool that allows you to provision various infrastructure components. Ansible is a platform for managing configurations and deploying applications. It means you'll use Terraform to build a virtual machine, for example, and then use Ansible to install the necessary applications on that machine.

Considering the Organizational requirement, you are asked to automate the infrastructure using Terraform first and install other required automation tools in it.

Tool required

- Terraform (installed on a controller machine)
- AWS account with security credentials)
- For SSH I have created in terraform command code put ssh-key with instance that will create in future

Need configuration on instance after created

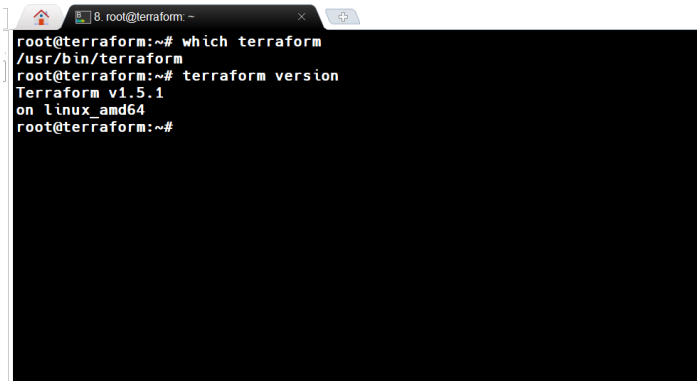
- Launch an EC2 instance using Terraform
- Connect to the instance
- Install Jenkins, Java and Python in the instance
- Validate that the packages are installed

Procedure

Step 1. Configure the controller node

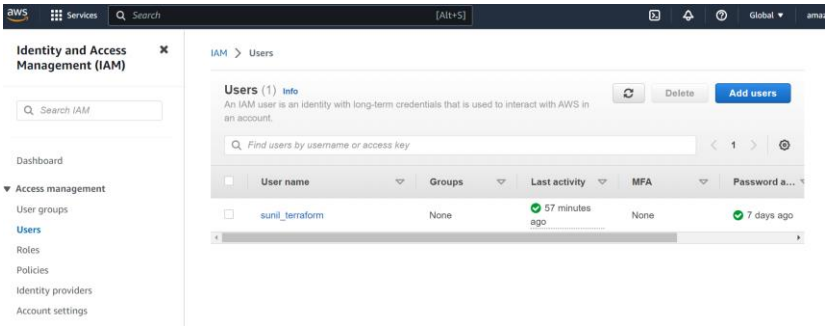
Installation steps:

- Procedure to install terraform in controller:
  - First need to visit https://developer.hashicorp.com/terraform/downloads?product_intent=terraform
  - Follow the below command to install terraform in linux os at controller node
    - 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
  - To validate:
    - Terraform version
    - Which terraform

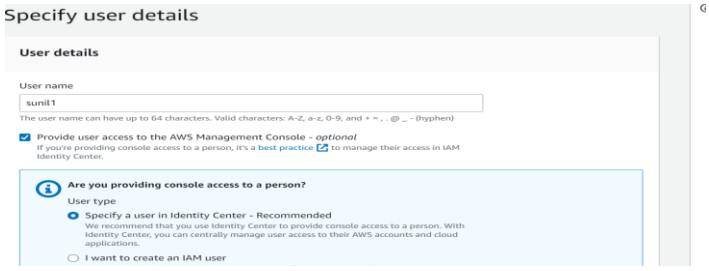


Step 2: Setup AWS "access key" && "secret key"

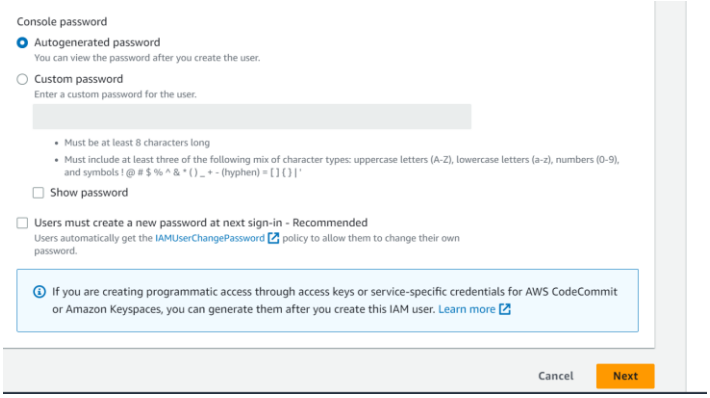
- Procedure to setup access key and secret key:
  - Visit AWS account > go to IAM service > user section > create user



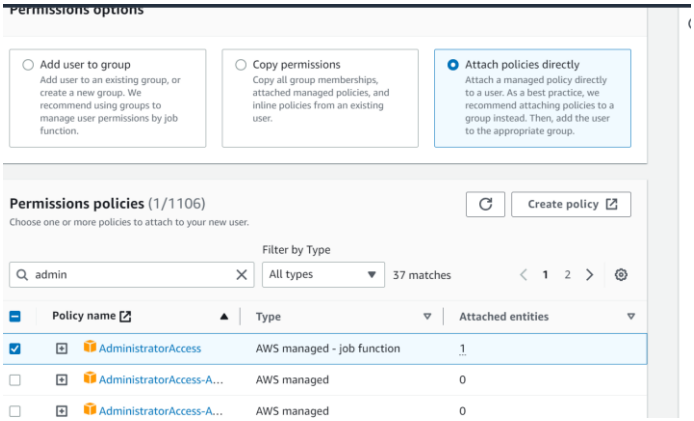
- Click “Add users” in case it not created in my case it is already created



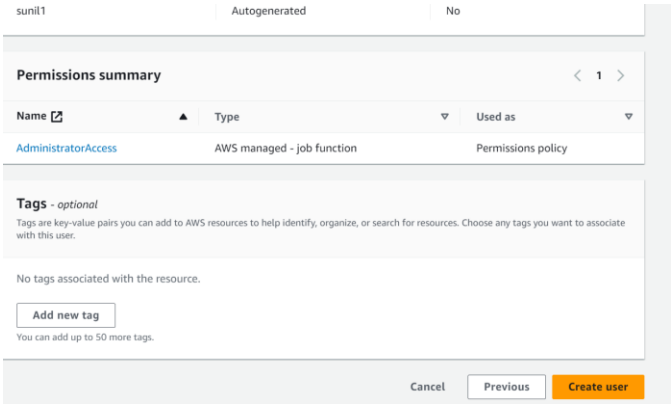
- Click “ I want to create an IAM user” and then then follow as below

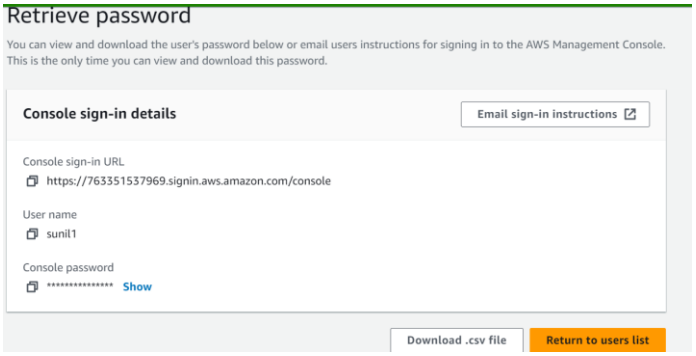


- Click “next” && attach policy as “ AdministratorAccess”

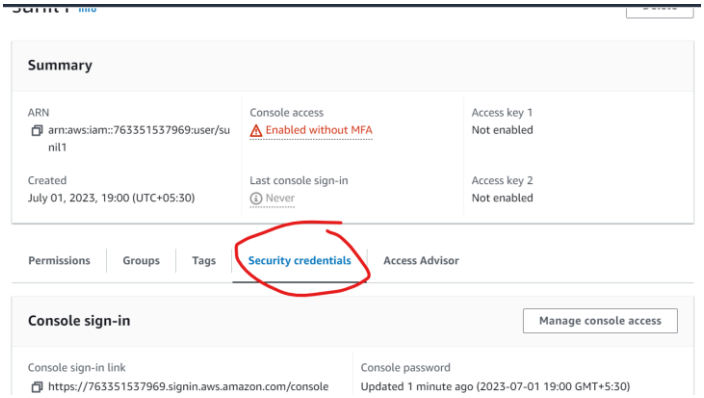


- Now create user

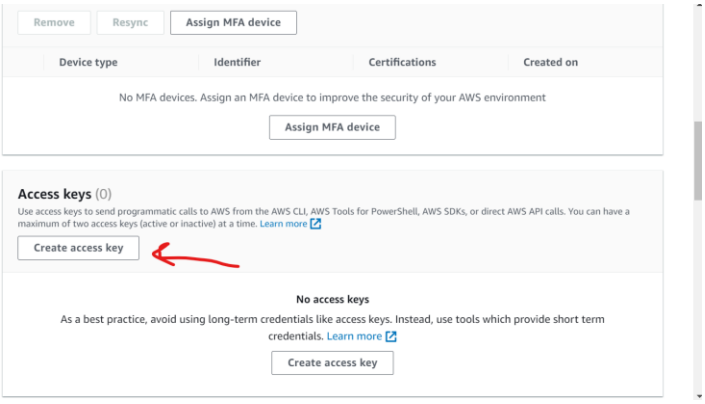




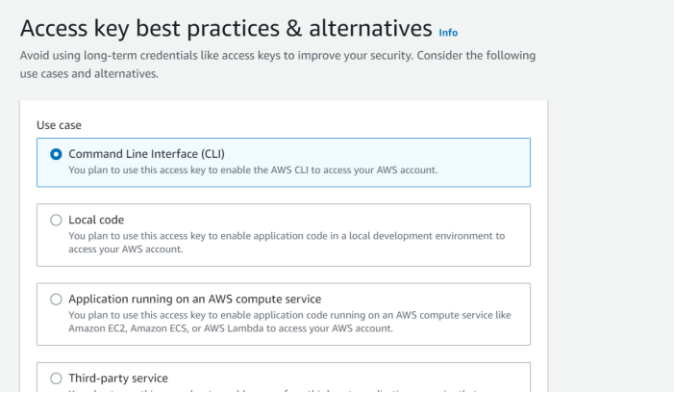
- Now create access and secret key for above user



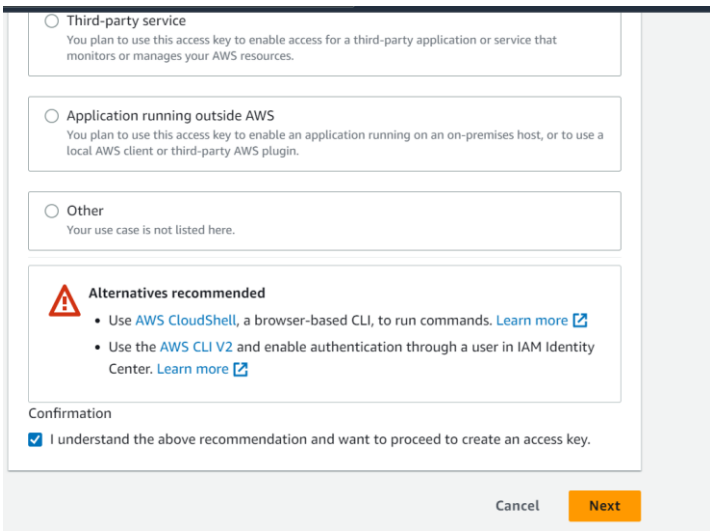
- Click create access key



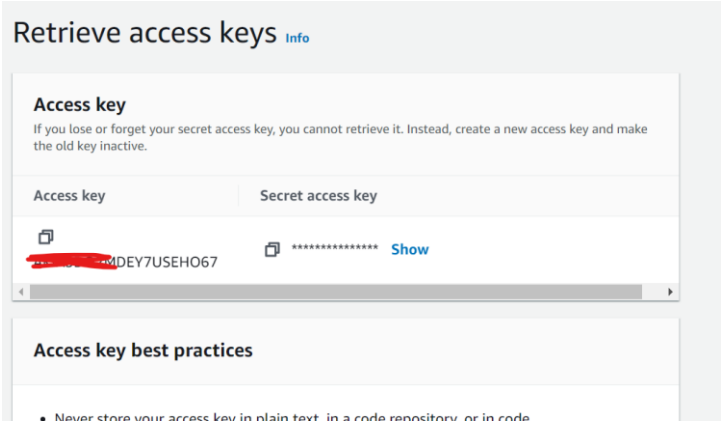
- Select for command line



- Check policy and click next



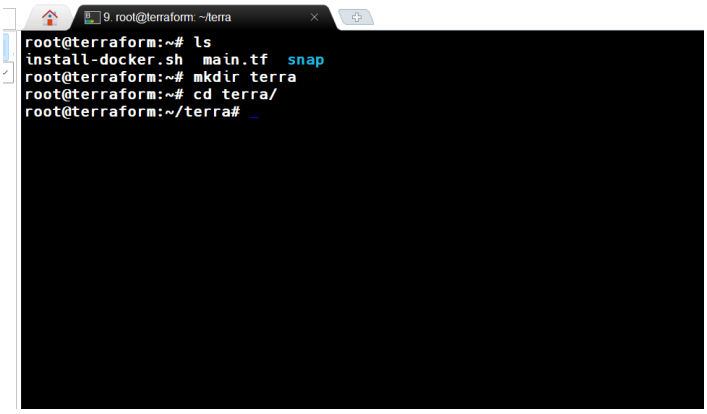
- Access and secret key generated and need to save for future used



Step3 Write terraform file

Create aws instance using terraform

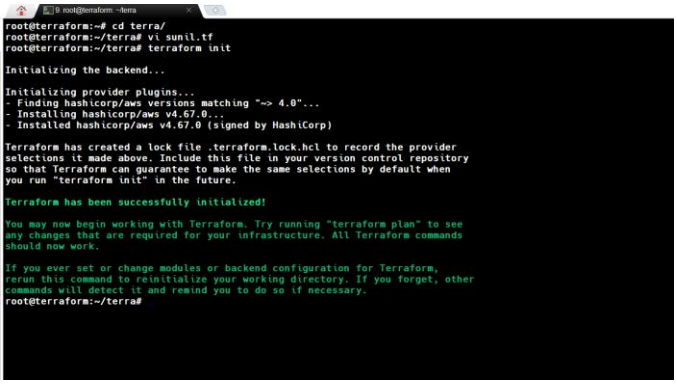
- **Procedure to follow as mentioned below:**
  - **First need to create directory as "terra" in my case**
  - **Use command "mkdir terra" && "cd terra"**



- **Make “sunil.tf” file and write code to add cloud providers (like aws, azure, gcp etc)**
- **Vi sunil.tf >> press “i” insert for write then write code as mention below**

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}
```

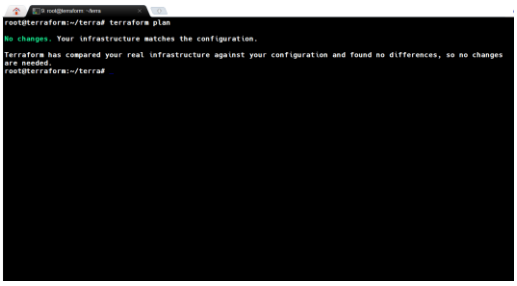
- **Then press “:wq!” to save**
- **Now initiate with terraform**
- **Command “terraform init”**



- **Authentication and configuration**
  - Vi sunil.tf
  - And write code for “aws authentication and configuration”

```
# Authentication and configuration
provider "aws" {
  region     = "ap-southeast-2"
  access_key = "Your_access_key"
  secret_key = "your_secure_key"
}
```

  - Command “terraform plan” to check plan



- **Create ssh-key on controller**

  - Run command " ssh-keygen -t rsa"
  - Copy public id from id_rsa.pub file
  - And paste in code as mention below

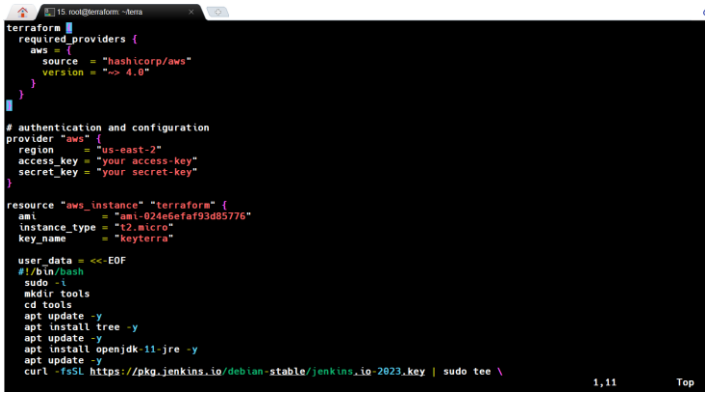

```
resource "aws_key_pair" "terraform" {
  key_name       = "keyterra"
  public_key     = "your ssh-key"
}
```

- **Create aws_instance using terraform**

  - Write a code for create instance in AWS

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}

# authentication and configuration
provider "aws" {
  region     = "us-east-1"
  access_key = "your access-key"
  secret_key = "your secret-key"
}

resource "aws_instance" "terraform" {
  ami           = "ami-024e6efaf93d85776"
  instance_type = "t2.micro"
```

```
  key_name      = "keyterra"

  user_data = <<-EOF
  #!/bin/bash
   apt update -y
   apt install tree -y
   apt update -y
   apt install openjdk-11-jre -y
   apt update -y
   curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-    2023.key | sudo tee \
   /usr/share/keyrings/jenkins-keyring.asc > /dev/null
   echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
   https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
   /etc/apt/sources.list.d/jenkins.list > /dev/null
   apt-get update -y
   apt-get install jenkins -y
   add-apt-repository ppa:deadsnakes/ppa
   apt update -y
   apt list --upgradable
   apt-get update -y
   apt install python -y
  EOF

  tags = {
    Name = "terraform"
  }
}
resource "aws_key_pair" "terraform" {
  key_name      = "keyterra"
  public_key    = "your ssh-key"
}
```



- User_data script for install tree, Java Python and Jenkins at created instance via terraform.

```
user_data = <<-EOF
  #!/bin/bash
   sudo -i
   mkdir tools
   cd tools
   apt update -y
   apt install tree -y
   apt update -y
   apt install openjdk-11-jre -y
   apt update -y
   curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo    tee \
   /usr/share/keyrings/jenkins-keyring.asc > /dev/null
   echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
   https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
   /etc/apt/sources.list.d/jenkins.list > /dev/null
   apt-get update -y
   apt-get install jenkins -y
   add-apt-repository ppa:deadsnakes/ppa
   apt update -y
   apt list --upgradable
   apt-get update -y
   apt install python -y
  EOF
```

- Code for terraform

sunil1.tf

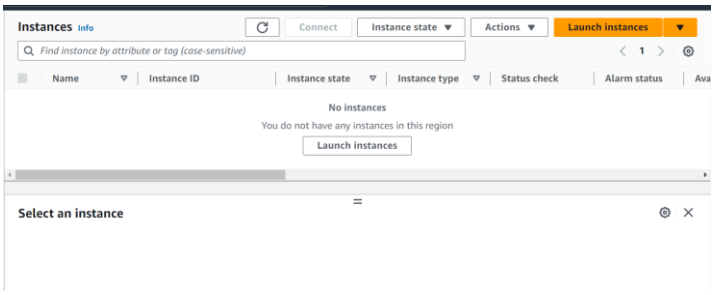
- Now execute terraform plan command to check plan for execution

```
15. root@terraform: ~/terra
      }
      + tenancy                              = (known after apply)
      + user_data                            = "0d90e05b0461714acedf2a976f2a3d9c462e7496"
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)
    }

  # aws_key_pair.terraform will be created
  + resource "aws_key_pair" "terraform" {
      + arn             = (known after apply)
      + fingerprint     = (known after apply)
      + id              = (known after apply)
      + key_name        = "keyterra"
      + key_name_prefix = (known after apply)
      + key_pair_id     = (known after apply)
      + key_type        = (known after apply)
      + public_key      = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCvDCDeEnUv5jkp4FE5HTiOsqGVOohIWh0IqTXgOnz/TdQUpruwD
inF9wtOtIcZJw0RLd4D6BM1sTNZbyXTNRgqs2kNkYpiPCGbAhqSdezO2MTjhuZ1qYOruRKFiQkN1WRWsVe1uy+e3ycYKFkBqIlrhvCqfNIZaeMUm0coG
vjYCzuWb/tOhOrWZDTYDXdYimRikma3abz1lqTtKaY4gUmONNN5RZeaazhXpYOTdV+EyI6UOIfWcLA3TAiHkJa2A1K4dzRiEPln2U81zpB784k0CZ9Ca9
pfPvoYUcsy/28OI2Ic7oX4pJ5Ak9hi6wzlcc/QUlUbS3knY67eUKSbXBHsdbcmwFJA6upIFIAMBBP7DgEUJAzGCmxrlX7iHMJUK/KwR/bY4Zd4fxE7Y
DJCbPhLasCYkh5V5ShDVMIRKYdpH+EclTookbpC9fbZ7ubR8WIxjsXpOPkc4+fBWVFTV66QHbORTh+cFJy3KTeKu7psMuHjdQSiTKXM0dtqTd+5fM=
root@terraform"
      + tags_all        = (known after apply)
    }

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

─────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions
if you run "terraform apply" now.
root@terraform:~/terra#
```

- To validate run command “terraform validate”

```
9. root@terraform: ~/terra
root@terraform:~/terra# terraform validate
Success! The configuration is valid.

root@terraform:~/terra# _
```

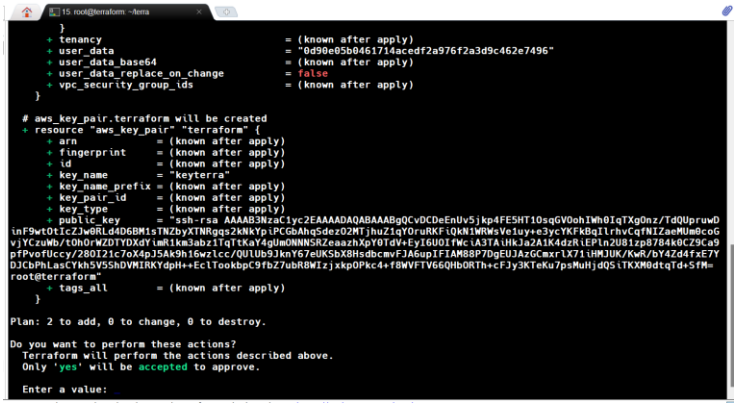
- Before go proceed for apply we need to validate instance in AWS account



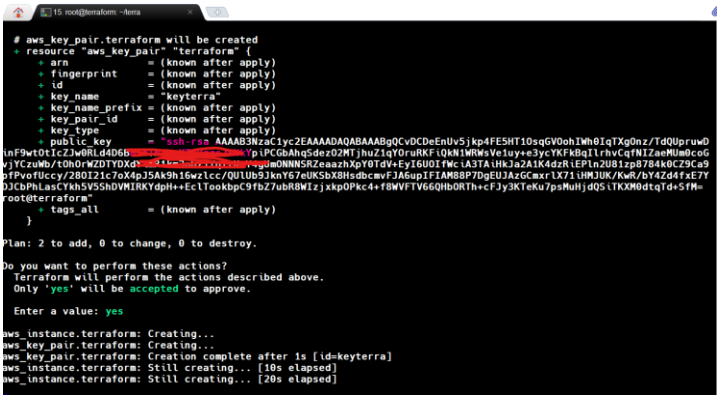
- **Final step to create instance via terraform**

  - To execute and create instance on AWS then run command “terraform apply”

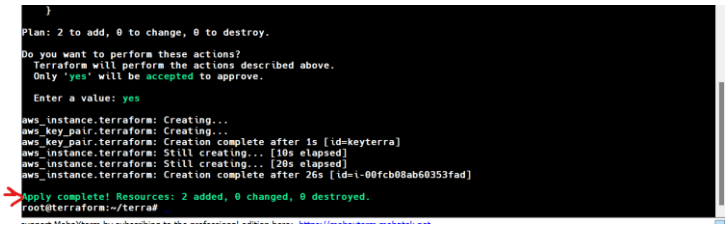
```
15. root@terraform: ~/terra
root@terraform:~/terra# terraform apply
```



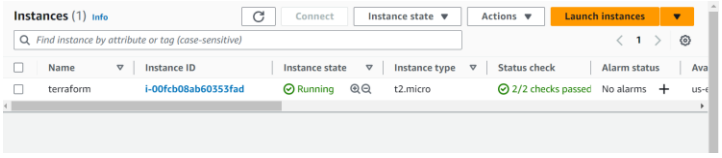
- Put “yes”



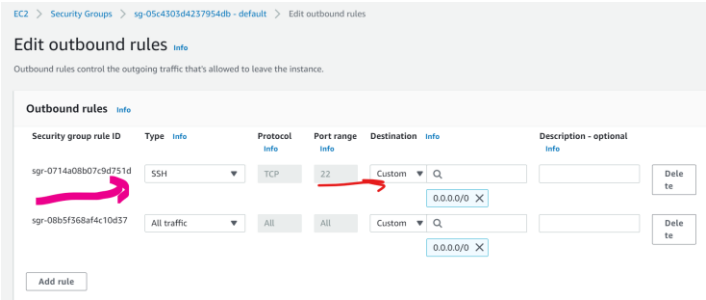
- Now the instance is created via terraform as shown below



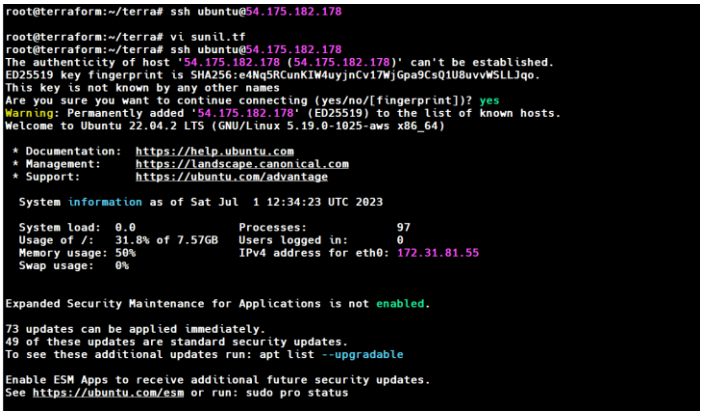
- Now validate instance at AWS account



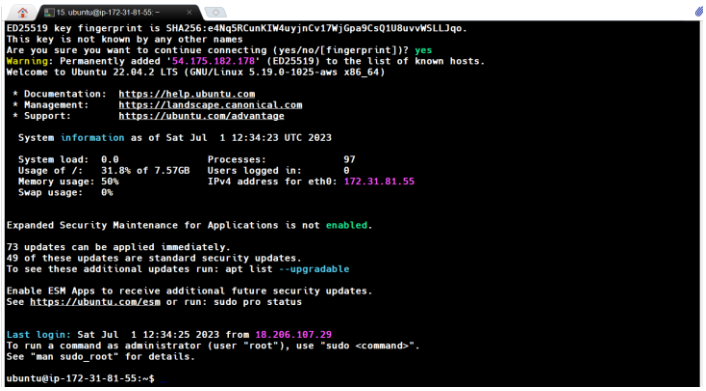
- **Login and validate the tools in created instance via terraform**
  - Before ssh need to check inbound and outbound in security group in AWS account.
  - Login AWS account and go to ec2 instance and add security inbound rule as ssh with 0.0.0.0/0



- **Login via controller**
  - Take public ip of newly created instance and do ssh from controller node as mention below
    - **Ssh -i ubuntu@public ip**
    - ssh ubuntu@54.175.182.178 (in my case)

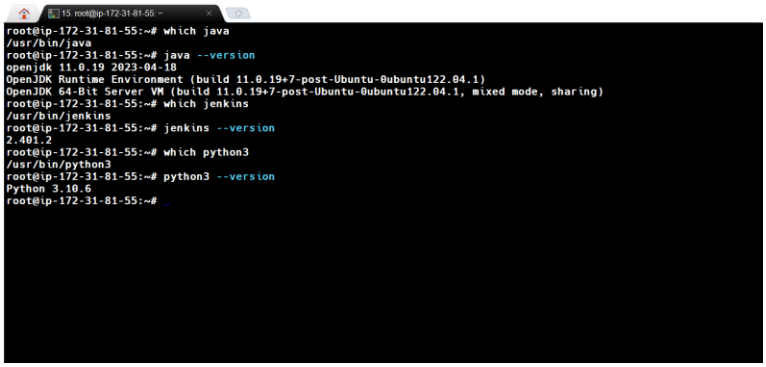


```
root@terraform:~/terra# ssh ubuntu@54.175.182.178

root@terraform:~/terra# vi sunil.tf
root@terraform:~/terra# ssh ubuntu@54.175.182.178
The authenticity of host '54.175.182.178 (54.175.182.178)' can't be established.
ED25519 key fingerprint is SHA256:e4Nq5RCunKIW4uyjnCv17WjGpa9CsQ1U8uvvWSLLJqo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.175.182.178' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

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

  System information as of Sat Jul  1 12:34:23 UTC 2023

  System load:  0.0               Processes:             97
  Usage of /:   31.8% of 7.57GB   Users logged in:       0
  Memory usage: 50%               IPv4 address for eth0: 172.31.81.55
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

73 updates can be applied immediately.
49 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```



```
ED25519 key fingerprint is SHA256:e4Nq5RCunKIW4uyjnCv17WjGpa9CsQ1U8uvvWSLLJqo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.175.182.178' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

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

  System information as of Sat Jul  1 12:34:23 UTC 2023

  System load:  0.0               Processes:             97
  Usage of /:   31.8% of 7.57GB   Users logged in:       0
  Memory usage: 50%               IPv4 address for eth0: 172.31.81.55
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

73 updates can be applied immediately.
49 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Sat Jul  1 12:34:25 2023 from 18.206.107.29
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-81-55:~$
```

- **validate**
  - Now validate tools like java, Jenkins and python3 output



```
root@ip-172-31-81-55:~# which java
/usr/bin/java
root@ip-172-31-81-55:~# java --version
openjdk 11.0.19 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu122.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu122.04.1, mixed mode, sharing)
root@ip-172-31-81-55:~# which jenkins
/usr/bin/jenkins
root@ip-172-31-81-55:~# jenkins --version
2.401.2
root@ip-172-31-81-55:~# which python3
/usr/bin/python3
root@ip-172-31-81-55:~# python3 --version
Python 3.10.6
root@ip-172-31-81-55:~#
```