Automating Infrastructure using Terraform

Created by: Mark Oliver Cañeda

Tools & Technologies Required:

- Terraform
- Ansible (optional setup for admin verification)
- AWS account with IAM credentials
- Keypair for SSH access
- Jenkins, Java, and Python

Prerequisites

Make sure the following prerequisites are in place before you begin:

- Install Terraform
  Download and install Terraform from https://www.terraform.io/downloads.html.

- Install Ansible
  Run the following command to install Ansible on your local machine:

```
sudo apt update && sudo apt install ansible -y
```

- AWS Account and Keypair Setup
  - Create an AWS account and generate security credentials (Access Key and Secret Key).
  - Create a keypair for SSH access (`my-terraform.pem`).

Step-by-Step Procedure

I. Install Terraform (Local Machine)

- Begin by updating the package list on your server.

```
sudo apt update
```

- Install dependencies for downloading and verifying Terraform.

```
sudo apt install -y gnupg software-properties-common curl
```

- Add the HashiCorp GPG Key
  HashiCorp, the maker of Terraform, provides an official GPG key to verify downloads.

```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/hashicorp-archive-keyring.gpg
```

- Add the HashiCorp Repository
  Now add the official HashiCorp Linux repository.

```
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
```

- After adding the repository, update the package list again to include the HashiCorp repository, then install Terraform.

```
sudo apt update
sudo apt install -y terraform
```

- Verify Installation
  Confirm that Terraform is installed and check its version.

```
terraform -version
```

```
markmcaneda@ip-172-31-43-226:~$ terraform -version
Terraform v1.9.8
on linux_amd64
markmcaneda@ip-172-31-43-226:~$
```

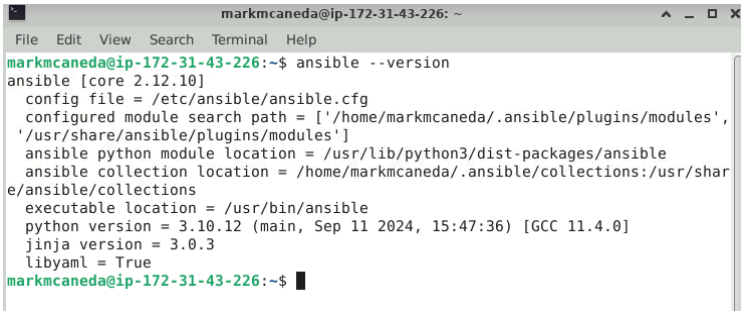
II. Install Ansible (Local Machine)

1. To configure install Ansible run these commands:

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

2. To check the version and it was installed. Run these commands:

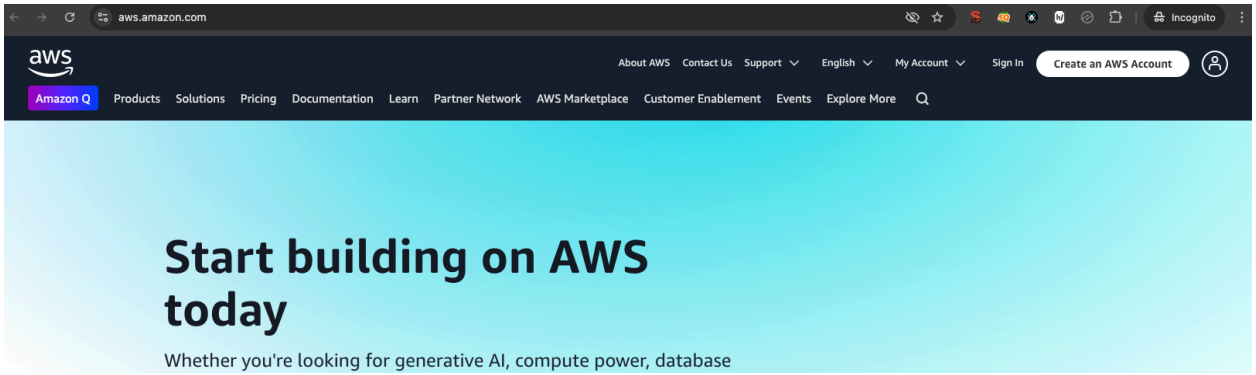
```
ansible --version
```



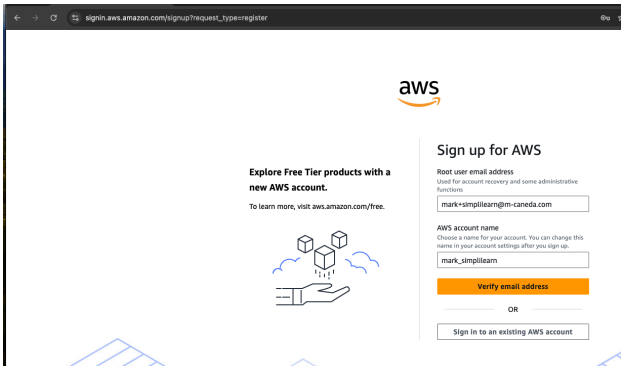
III. AWS Account and Keypair Setup (AWS Account)

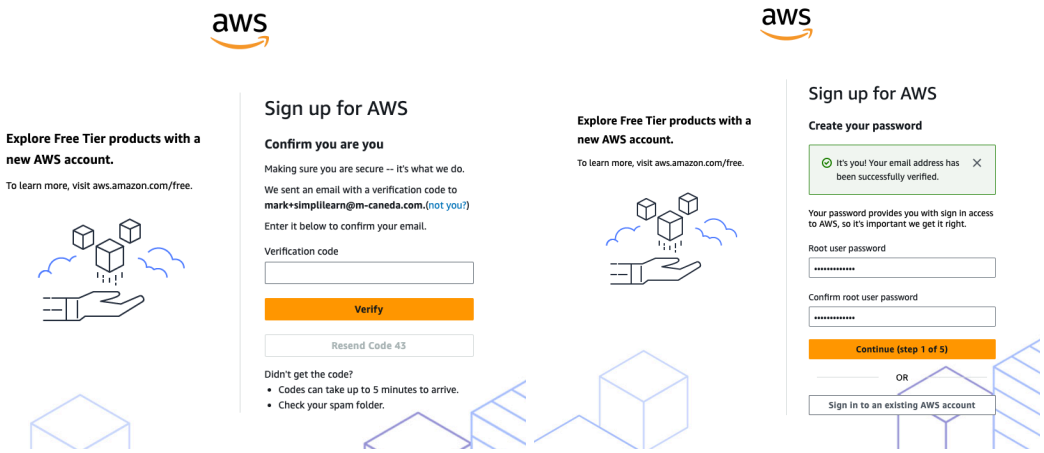
Guide on setting up an AWS account, generating security credentials, and creating a keypair for SSH access

1. **Go to the AWS Website**: Visit https://aws.amazon.com/ and click on "Create an AWS Account."

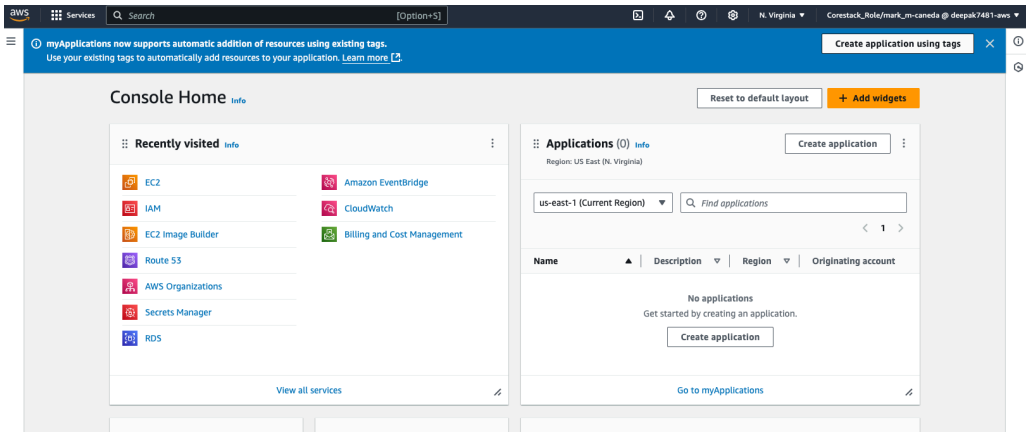


2. **Sign Up**: Enter your email address, create a password, and set an account name. Click "Verify email address" to continue and complete the registration steps.



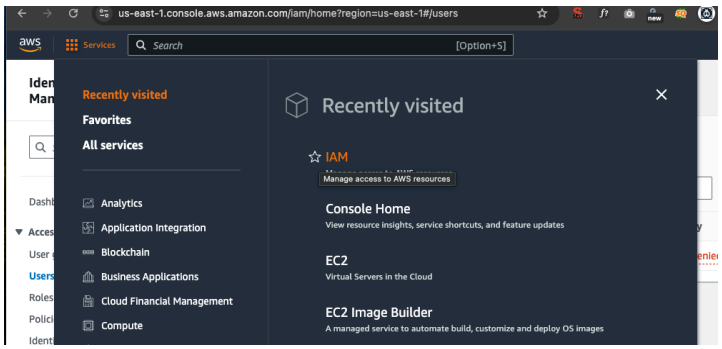


3. **Verify Your Identity**: Provide billing information and verify your identity by phone. AWS may offer a free tier, but entering a payment method is required.
4. **Log In**: Once your account is set up, log in to the **AWS Management Console**.

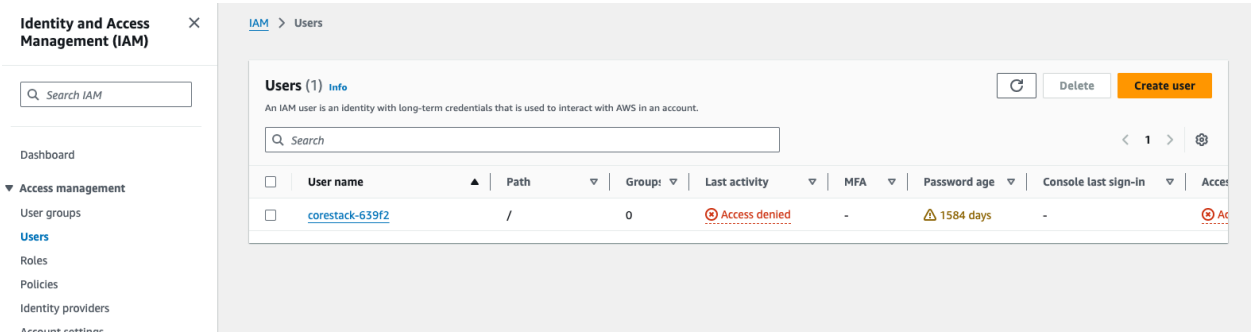


Generate IAM - Security Credentials

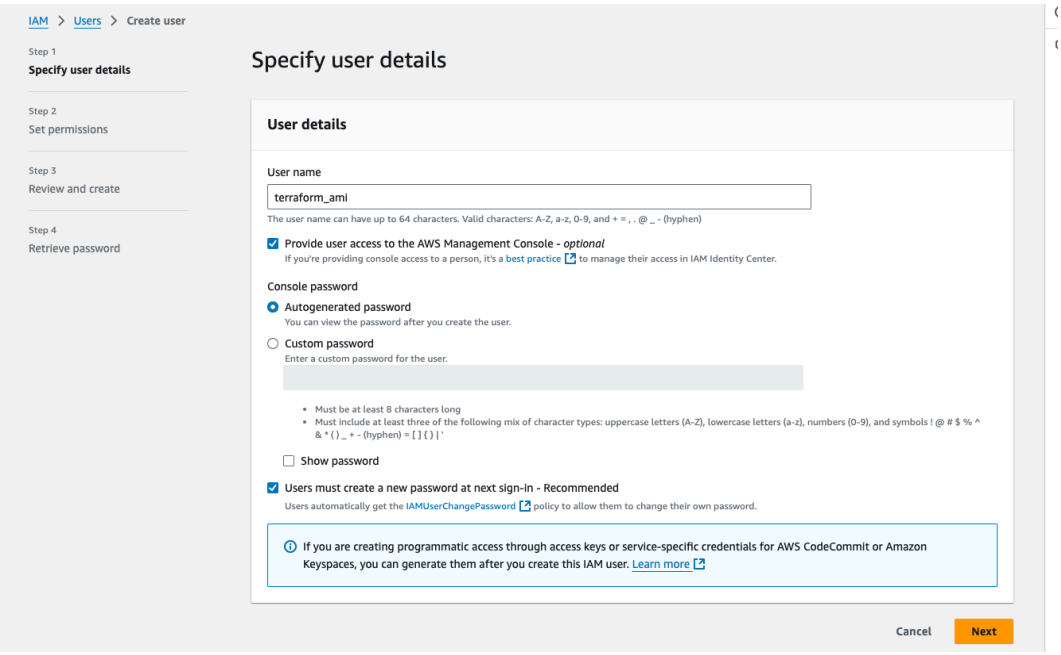
1. Go to IAM to manage access to AWS resources, type “IAM” at the search bar.



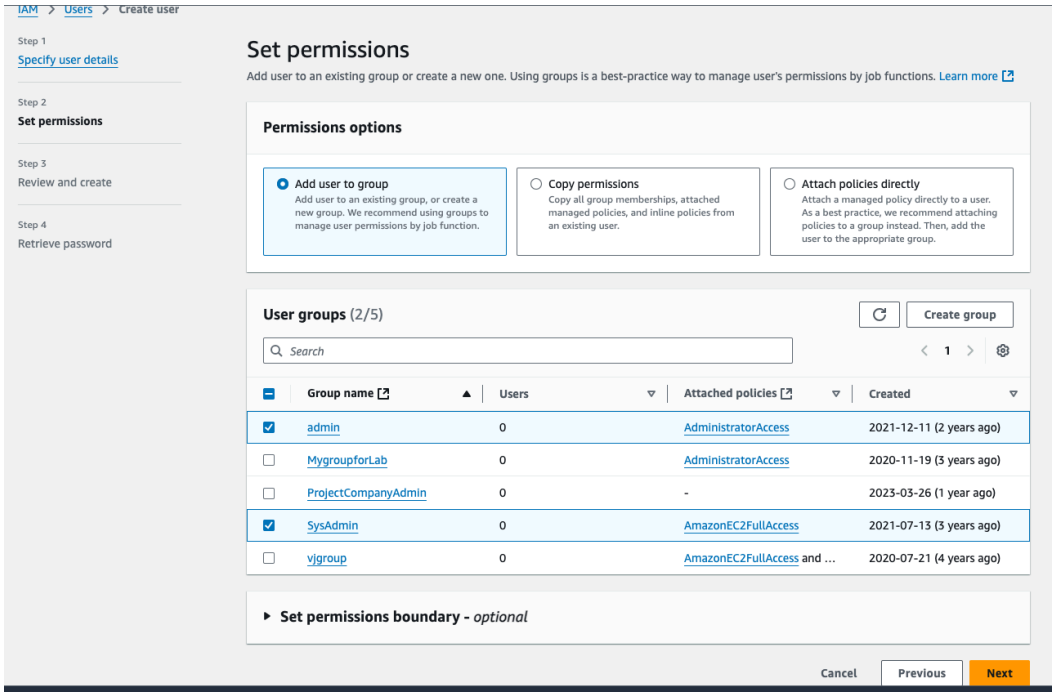
2. Click “Create User” to proceed



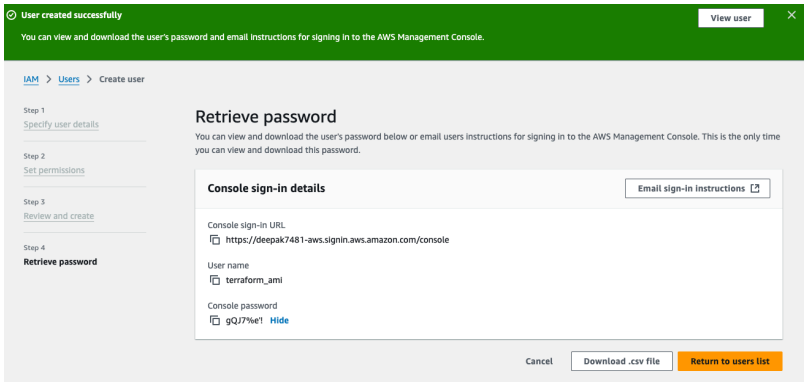
3. Fill out the necessary fields for User Details. Select “Provide user access to the AWS Management Console” to have a console access and a password to login. Click “Next” once done.



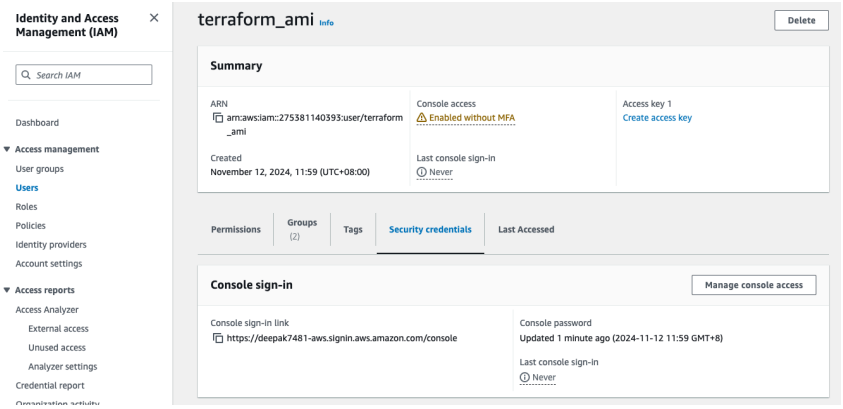
4. **Set permissions** for the user based on their job functions. In the meantime, set ‘admin’ related to the setup of EC2 Instances. Click “Next”, to proceed.

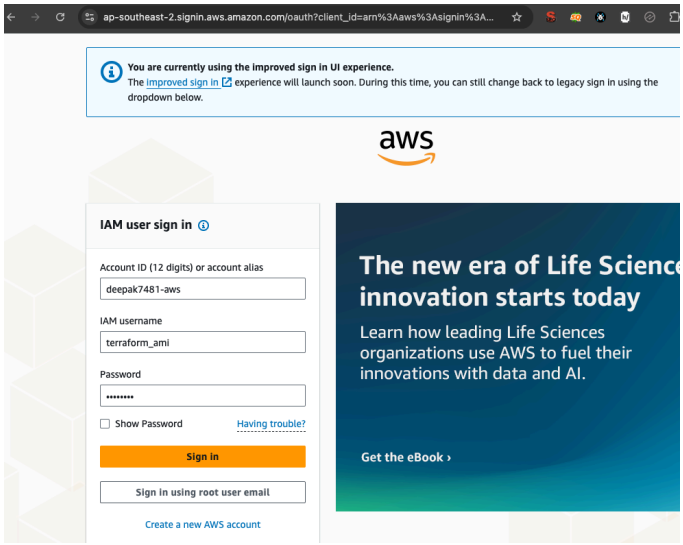


5. Next, Review your user details and Create the user. Once done and we are in Step 4, you can Retrieve the password and Console sign-in details. Click “Download .csv file” to back up the credentials.



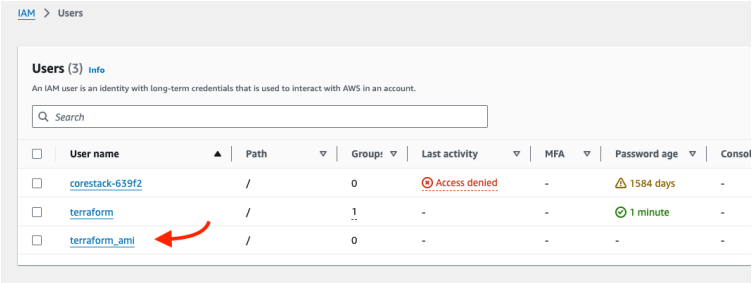
6. Now user can go to the console. Copy and Paste the Console sign-in link to log in. So we can proceed to the next step.



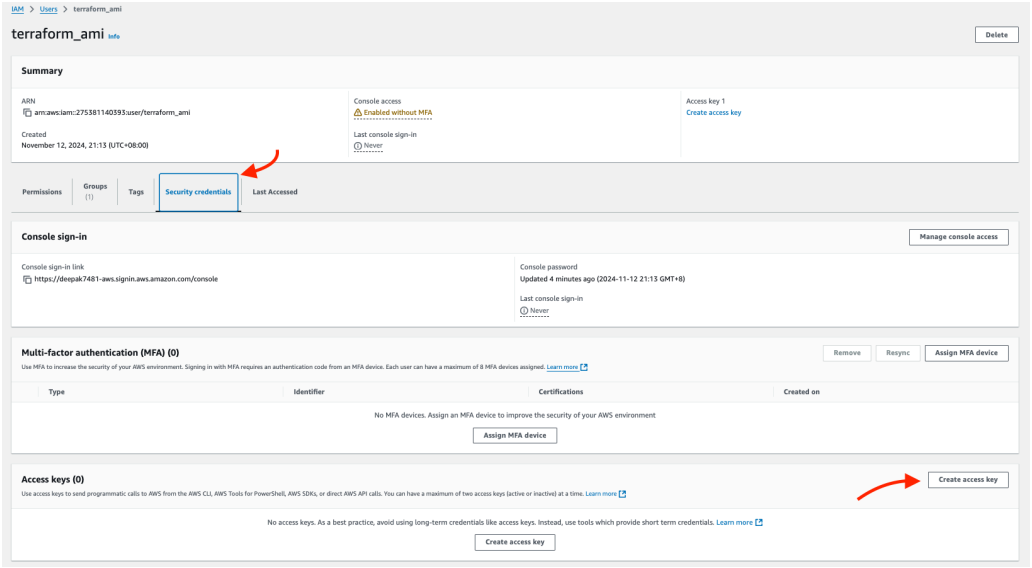


Generate AWS Access and Secret Keys for Terraform

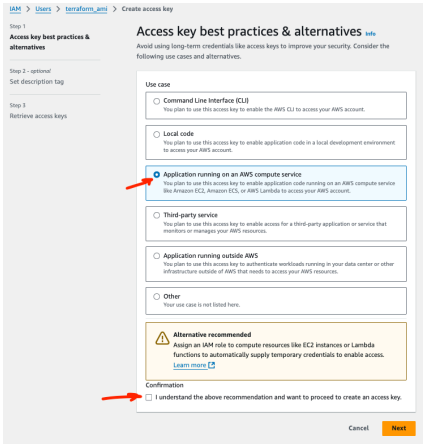
1. At the IAM Users, go back and select the created user.



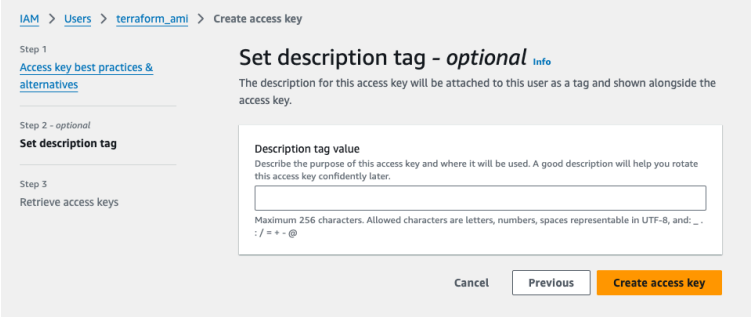
2. Select the “Security Credentials” tab and go to Access keys. Click “Create access key”.

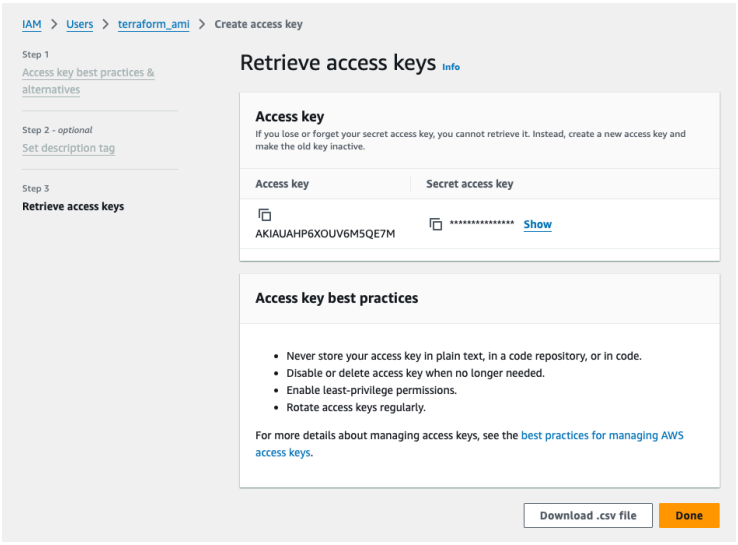


3. Select Use case “Application running on an AWS compute service”. Access keys to connect with the AWS feature like EC2 Instance.

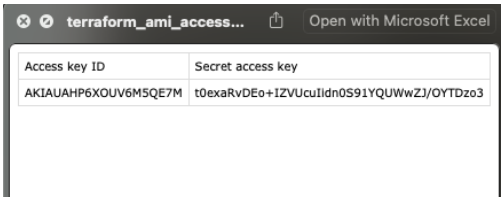


4. Under Step 2, which is optional for setting up a description tag. Click “Create access key” to retrieve the keys.



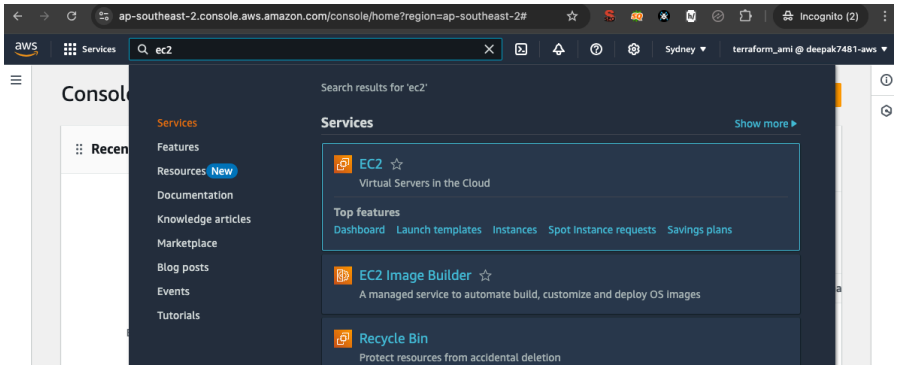


5. Once done, just click “Download .csv file” to safeguard account access and secret keys.

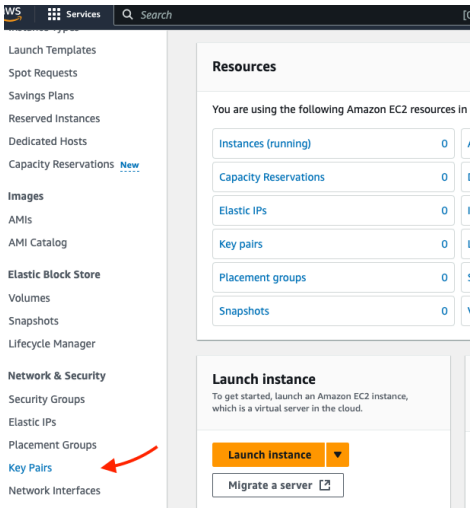


Generate Key Pairs of EC2 Instance

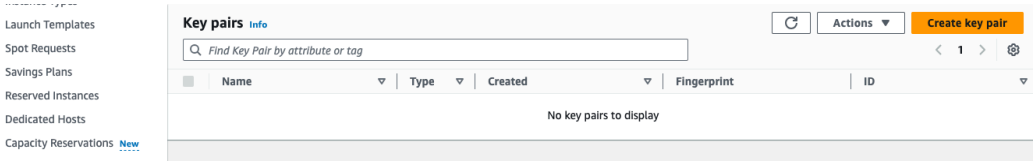
1. Once done above, log in to the AWS, after logging in to the AWS Console, Type “EC2” in the search bar to land in the EC2 Dashboard.



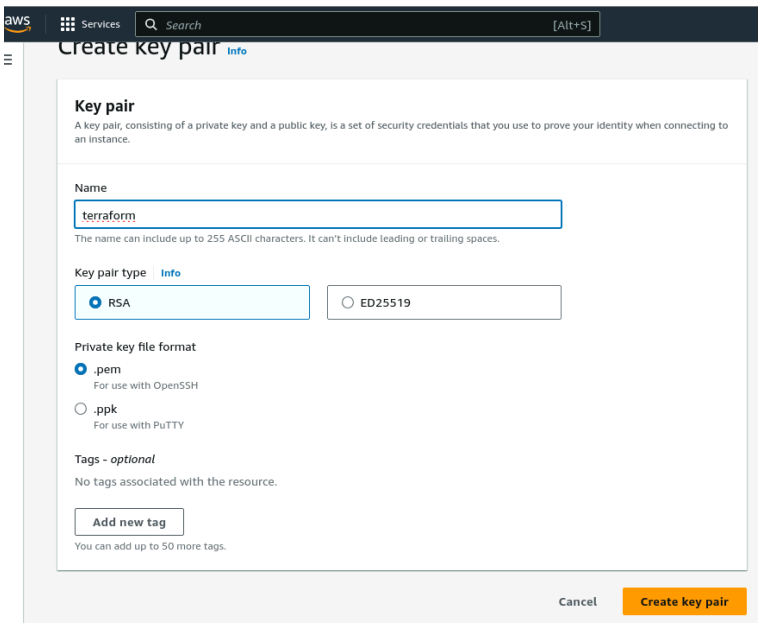
2. Search for “Key Pairs”, select and go to the page. This is another required keys that Terraform need for accessing EC2 Instance.



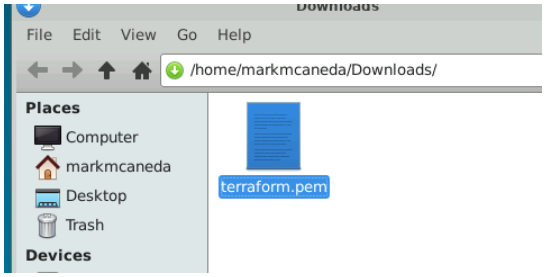
3. Click “Create key pair”.



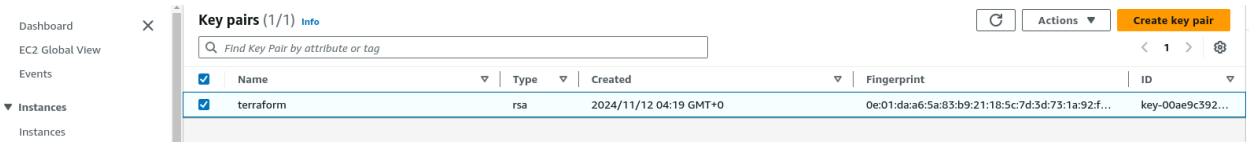
4. Fill out the Name of the Key Pair. Select by default the pair type to “RSA” and file format as “.pem”. Click “Create key pair” to finish the process and it will download the .pem file



Keep the file and place it in the folder of your choice for a later location path of Terraform.

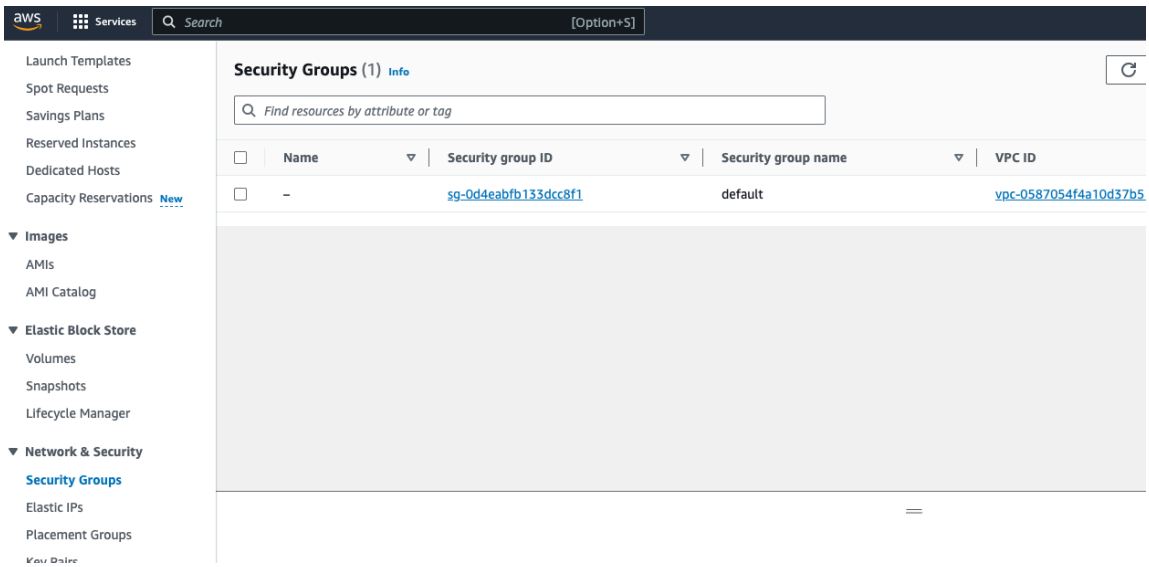


Key pair(s) have been created.

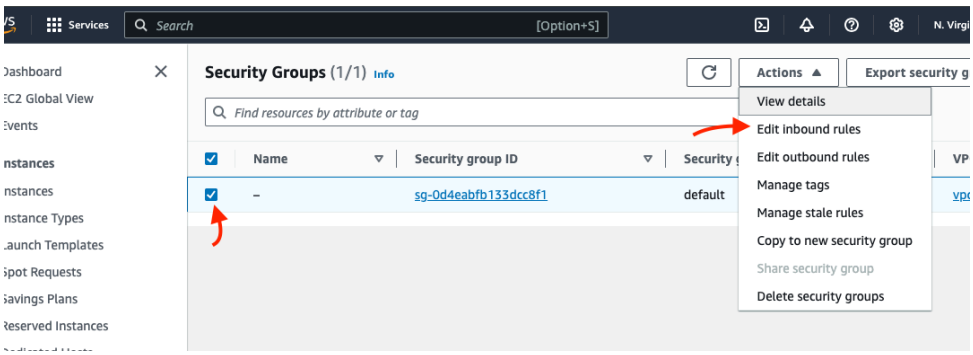


Update Security Group for the Key Pair access to SSH and specific port of Jenkins

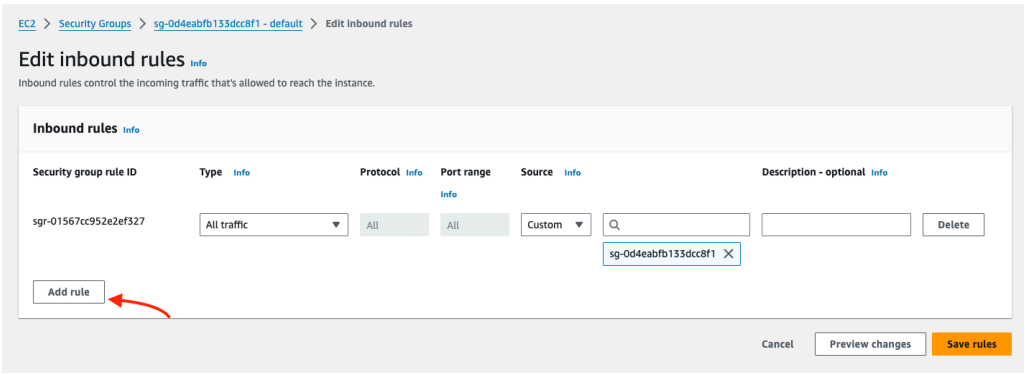
1. Go to EC2 and look for “Security Groups”.



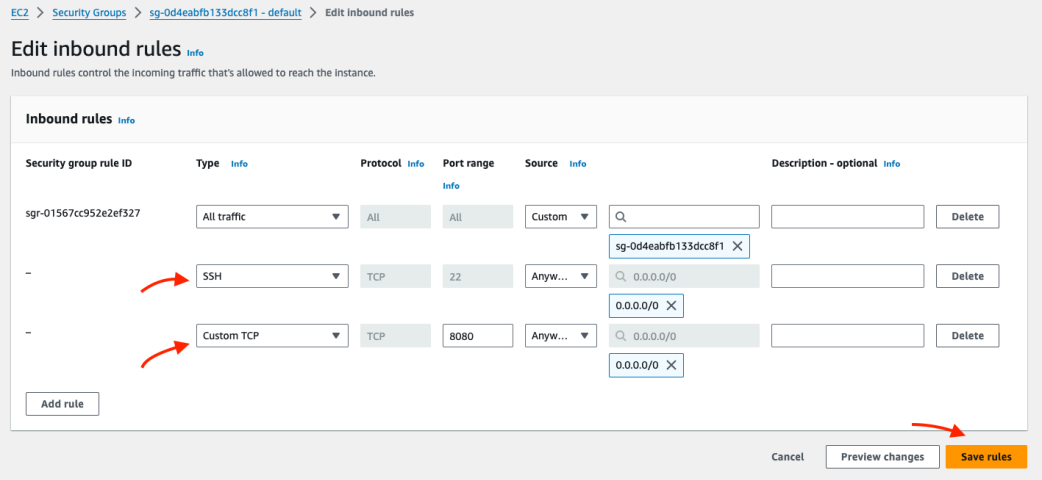
2. On the list, select and check the default security group. Select the drop-down “Actions > Edit inbound rules”.



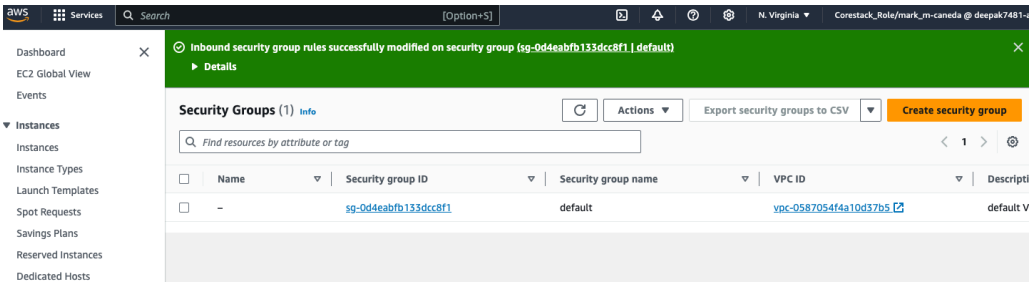
3. Inside the Edit inbound rules, click “Add rule”.



4. Add under type “SSH” with port range : 22 for ssh automation of Terraform and “Custom TCP” with port “8080” to access Jenkins default port. Source “Anywhere” with “0.0.0/0”. Click “Save rules” once done.



5. Now, Terraform automatically can access the specified port.



IV. Setup Terraform Configuration Files

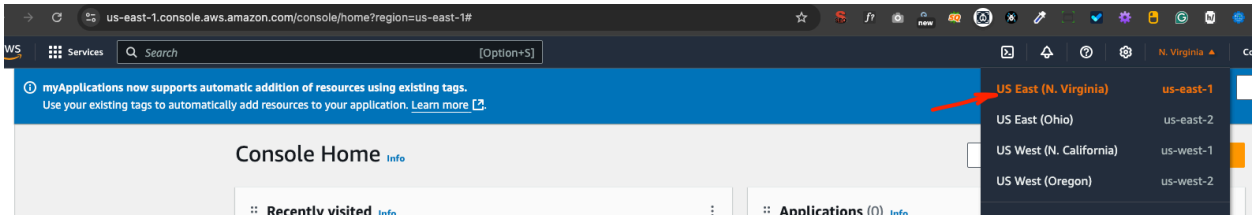
1. Create a directory for your project and navigate into it:

```
mkdir terraform-jenkins && cd terraform-jenkins
```

2. Create the following Terraform files: `main.tf`
3. This file contains the Terraform configuration to launch the EC2 instance.

```
provider "aws" {
  region     = "[REGION USED]"
  access_key = "[ACCESS KEY ID]"
  secret_key = "[SECRET ACCESS KEY]"
}

resource "aws_instance" "jenkins_server" {
  ami           = "[AMI TYPE]" # check for the AMI ubuntu 24 free tier as an example
  instance_type = "[INSTANCE TYPE]"
  key_name      = "[NAME OF THE SSH KEY PAIR]"

  tags = {
    Name = "Jenkins-Server"
  }

  provisioner "remote-exec" {
    inline = [
      # Update and install required packages
      "sudo apt update -y",
      "sudo apt upgrade -y",

      # Install Java 17 (required for Jenkins) and fontconfig
      "sudo apt install -y fontconfig openjdk-17-jdk",

      # Download and configure the stable Jenkins repository
      "curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee
/usr/share/keyrings/jenkins-keyring.asc > /dev/null",

      # Add the Jenkins repository to the package sources
      "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",

      # Update package list to include Jenkins and install Jenkins
      "sudo apt update -y",
      "sudo apt install -y jenkins",

      # Start and enable Jenkins service
      "sudo systemctl daemon-reload",
      "sudo systemctl start jenkins",
      "sudo systemctl enable jenkins"
    ]

    connection {
      type        = "ssh"
      user        = "ubuntu" #ubuntu default username as root
      private_key = file("[PATH OF THE .PEM FILE]")
      host        = aws_instance.jenkins_server.public_ip
      timeout     = "5m"  # Increase the timeout as needed. To prevent from timeout.
    }
  }
}

output "instance_ip" {
  value = aws_instance.jenkins_server.public_ip
}
```

- For the REGION USED: `us-east-1`

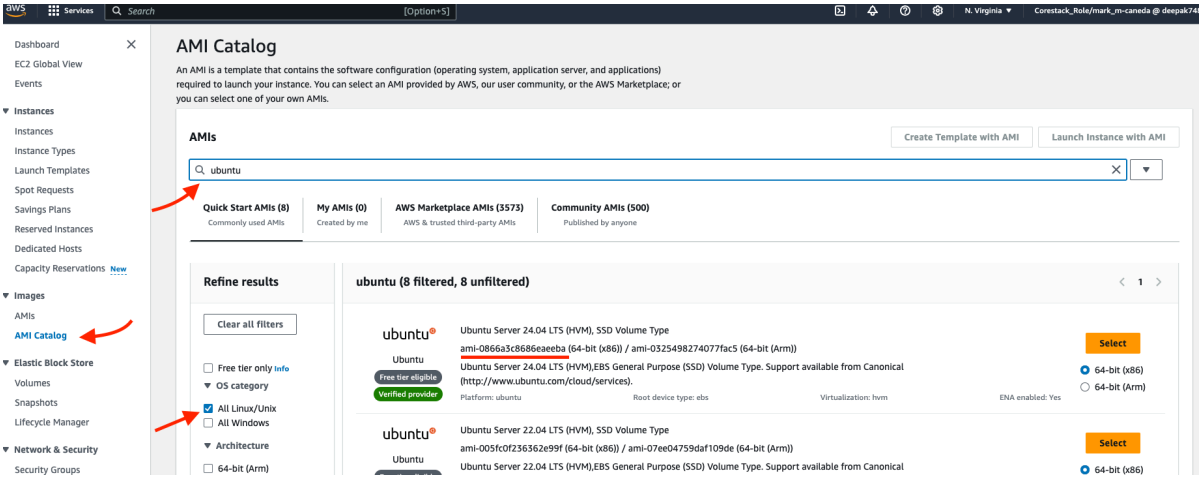


- For the ACCESS AND SECRET KEY, can be found from the created IAM Account and Access Keys (ref. Generate AWS Access and Secret Keys for Terraform - above steps).

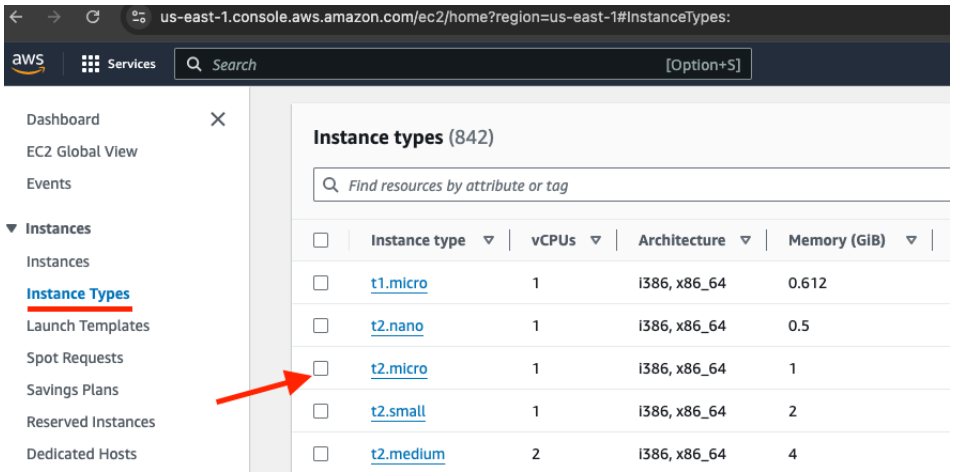
terraform_ami_accessKeys (1).csv

Access key ID	Secret access key
AKIAUAHP6XOUV6M5QE7M	t0exaRvDEo+IZVUcuIidn0S91YQUWwZJ/OYTDzo3

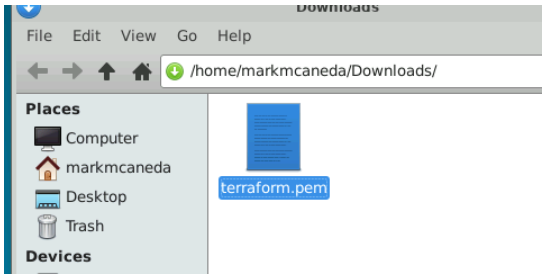
- For AMI Type, this can be found in EC2 Instances AMI Catalog. Search for AMI for "ubuntu" and under All Linux/Unix OS Category. Get the name of the AMI (example: `ami-0866a3c8686eaeeba`) for the 64-bit)



- For INSTANCE TYPE, this can be found from EC2 Instances Types. Copy the name of the Instance type: `t2.micro` as example free-tier of this project.



- For the NAME OF THE SSH KEY PAIR: `terraform` as for example. (ref. Generate Key Pairs of EC2 Instance from above step)



- For the PATH OF THE .PEM FILE, copy the file directory and the filename. Example:

`/home/markmcaneda/terraform-jenkins/terraform.pem`, for which it was located.

- These following lines will be executed inside the EC2 instance remotely after it was created.

```
    provisioner "remote-exec" {
      inline = [
  # Update and install required packages
        "sudo apt update -y",
        "sudo apt upgrade -y",

        # Install Java 17 (required for Jenkins) and fontconfig
        "sudo apt install -y fontconfig openjdk-17-jdk",

        # Download and configure the stable Jenkins repository
        "curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null",

        # Add the Jenkins repository to the package sources
        "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",

        # Update package list to include Jenkins and install Jenkins
        "sudo apt update -y",
        "sudo apt install -y jenkins",

        # Start and enable Jenkins service
        "sudo systemctl daemon-reload",
        "sudo systemctl start jenkins",
        "sudo systemctl enable jenkins"
      ]
```

V. Configure Ansible (for Post-Installation Task)

This will display the Java version, Python version, to verify the Installation and Jenkins for setup.

- Create an Ansible playbook named: **`jenkins-setup.yml`**
- Copy the code below and run after the Launch of the Terraform & EC2 Instance.

```
- hosts: all
  become: true
  tasks:
    - name: Install Java and Python
      apt:
        name:
          - openjdk-17-jdk
          - python3
        state: present

    - name: Add Jenkins repo and install Jenkins
      shell: |
        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
      args:
        warn: false

    - name: Install Jenkins
      apt:
        name: jenkins
        state: present

    - name: Start Jenkins
      service:
        name: jenkins
        state: started
```

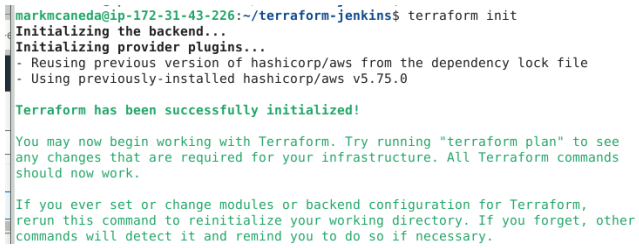
```
        enabled: true

    - name: Display Jenkins initialAdminPassword
      shell: cat /var/lib/jenkins/secrets/initialAdminPassword
      register: jenkins_password
      changed_when: false

    - name: Echo Jenkins initialAdminPassword
      debug:
        msg: "Jenkins initialAdminPassword: {{ jenkins_password.stdout }}"

    - name: Check Java version
      shell: java -version
      register: java_version
      ignore_errors: true  # Ignore errors if Java isn't installed
      changed_when: false

    - name: Display Java version
      debug:
        msg: "Java Version: {{ java_version.stderr_lines[0] if java_version.stderr else 'Java
not installed' }}"

    - name: Check Python version
      shell: python3 --version
      register: python_version
      ignore_errors: true  # Ignore errors if Python isn't installed
      changed_when: false

    - name: Display Python version
      debug:
        msg: "Python Version: {{ python_version.stdout if python_version.stdout else 'Python
not installed' }}"

    - name: Check Jenkins service status
      shell: systemctl is-active jenkins
      register: jenkins_status
      changed_when: false

    - name: Display Jenkins service status
      debug:
        msg: "Jenkins Service Status: {{ 'active' if jenkins_status.stdout == 'active' else
'inactive or not running' }}"

    - name: Display Jenkins access URL
      debug:
        msg: "Access Jenkins at: http://{{ ansible_host }}:8080"
```

VI. Initialize Terraform and Launch an EC2 Instance using Terraform

Run the following commands to initialize Terraform and apply the configuration.

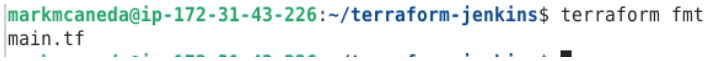
1. Initialize the project:

```
terraform init
```



2. Format the configuration

```
terraform fmt
```



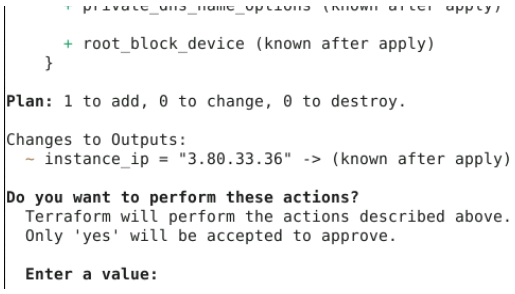
3. Validate the configuration

```
terraform validate
```

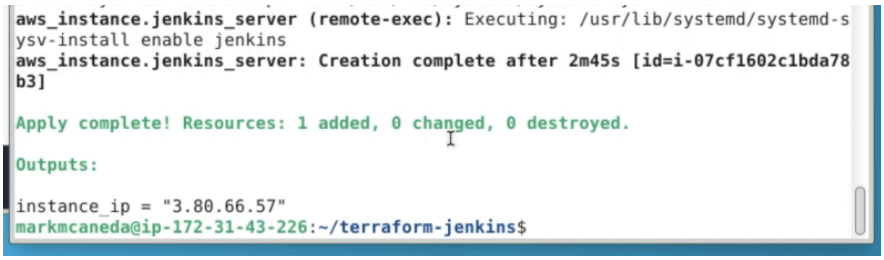
4. Apply the configuration

```
terraform apply
```

5. Confirm the changes by typing `yes` when prompted.



6. Once completed, the **instance_ip** will be displayed. It will used for later configuration.



VII. Verify EC2 Instance and Connect to the EC2 Instance via SSH.

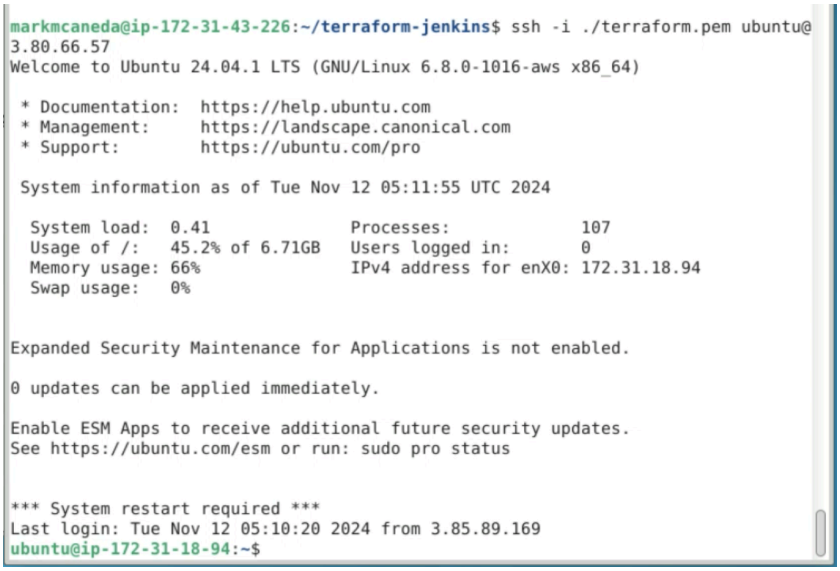
- After Terraform completes, it will output the public IP address of the instance. Use it to SSH into the instance. To connect:

```
ssh -i <key pair name>.pem ubuntu@<instance_ip>
```

- Make sure to adjust the .pem file permissions to chmod 400. Enter this command:

```
chmod 400 <key pair name>.pem
```

- Once done, run the ssh command above. When you see the Welcome screen message, you have successfully entered the Ubuntu instances of EC2.

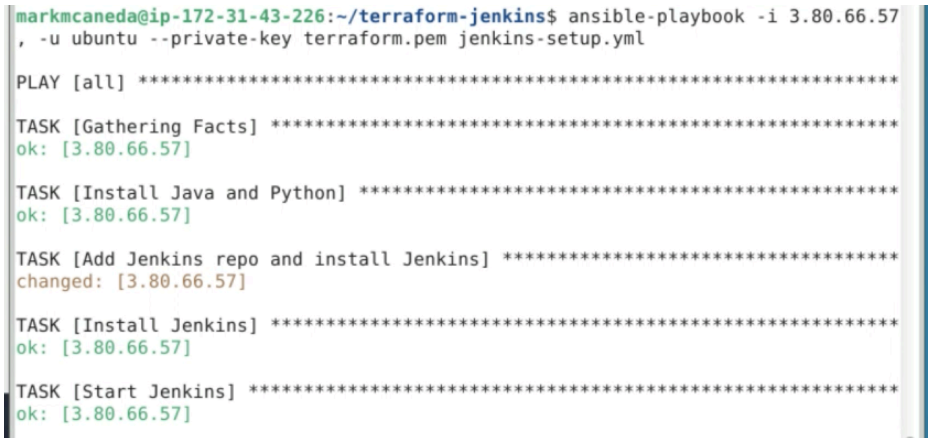


VIII. Run the Ansible playbook (for Java, Python, and Jenkins verification)

- After the above Terraform apply (ref. VI. Initialize Terraform and Launch an EC2 Instance using Terraform) and connected to SSH (ref. VII. Verify EC2 Instance and Connect to the EC2 Instance via SSH.), Exit to the SSH and make sure you're in the local machine.
- Run this playbook to your terminal.

```
ansible-playbook -i <instance_ip>, -u ubuntu --private-key <my-keypair-name>.pem
jenkins-setup.yml
```

- You may see the echo of the following command for Java version, Python version and Jenkins was running and active.



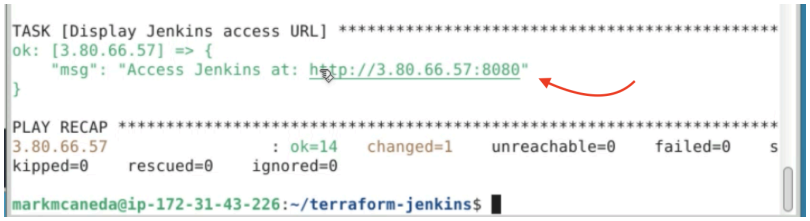
- It will show the Jenkin InitialAdminPassword for later access of Jenkins webpage.
- The Java that has been installed and its version.

- Display the Python version and the Jenkins remote URL

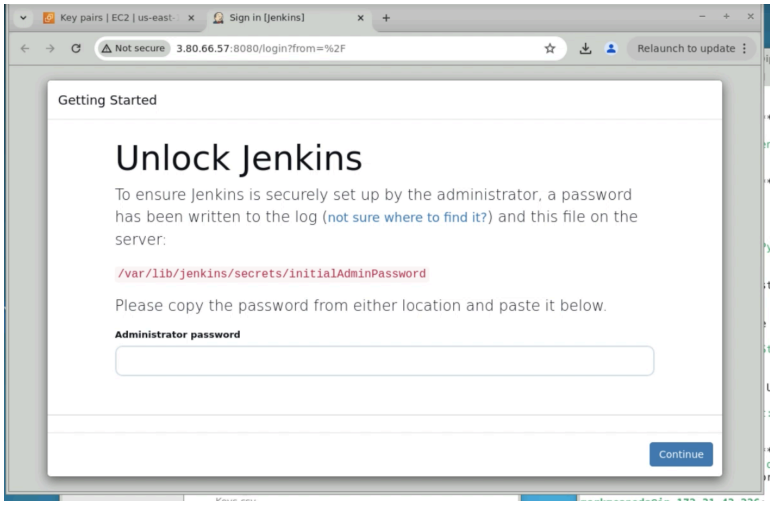
IX. Access and Launch the Jenkins

- Once confirmed and copy all the details above related to the Jenkins.
- You may now, access Jenkins: via

`http://<instance_ip>:8080`

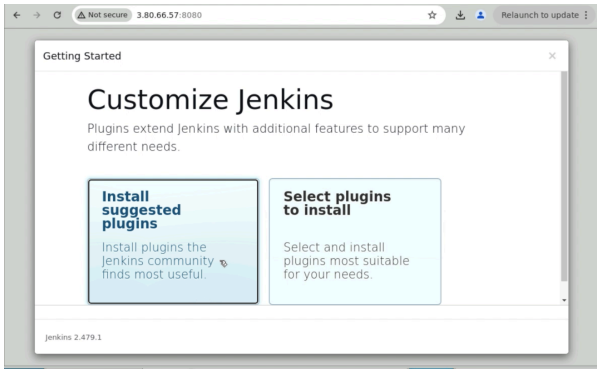


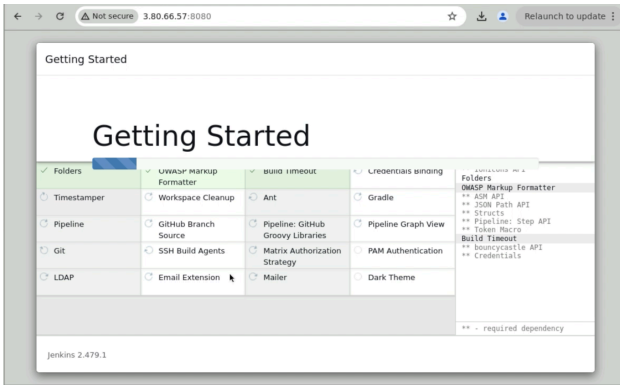
- Once visit the page, you may see this request.



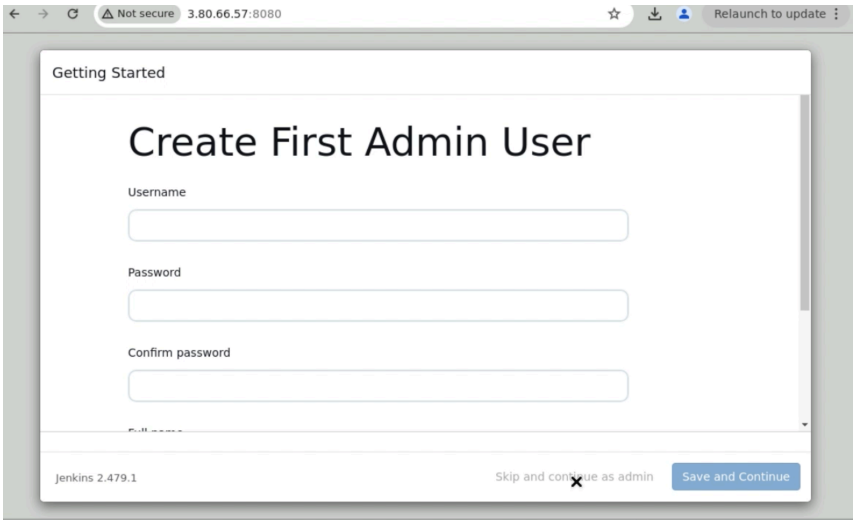
- Copy the Jenkin InitialAdminPassword. From the output ofthe Ansible command.

- Once done, just continue customize the Jenkins. Select Install suggested plugins and continue the process until it finished.

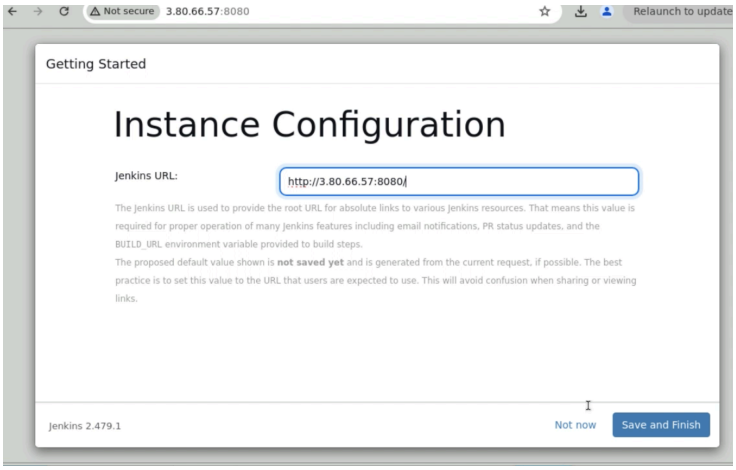


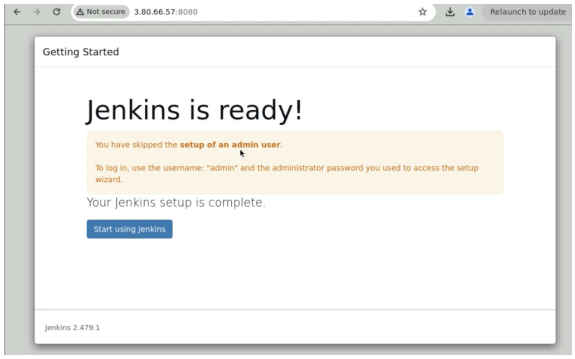


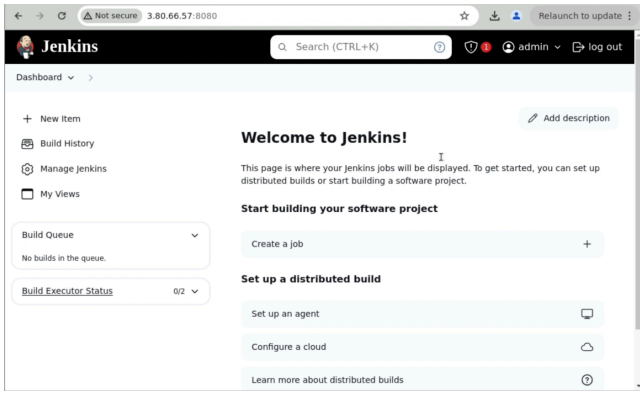
- You may skip the admin creation or fill out the fields. Save and Continue.



- Just continue the configuration. Until you land to the dashboard.







- Jenkins is now done.

Expected Deliverables:

- Launch an EC2 instance using Terraform (ref. IV. Setup Terraform Configuration Files, VI. Initialize Terraform and Launch an EC2 Instance using Terraform)
- Connect to the instance (ref. VII. Verify EC2 Instance and Connect to the EC2 Instance via SSH.).
- Install Jenkins, Java, and Python in the instance (ref. IV. Setup Terraform Configuration Files, VI. Initialize Terraform and Launch an EC2 Instance using Terraform, VIII. Run the Ansible playbook (for Java, Python, and Jenkins verification), and IX. Access and Launch the Jenkins)

- END -