Project Description

Use Terraform to provision infrastructure

**Description:**

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.

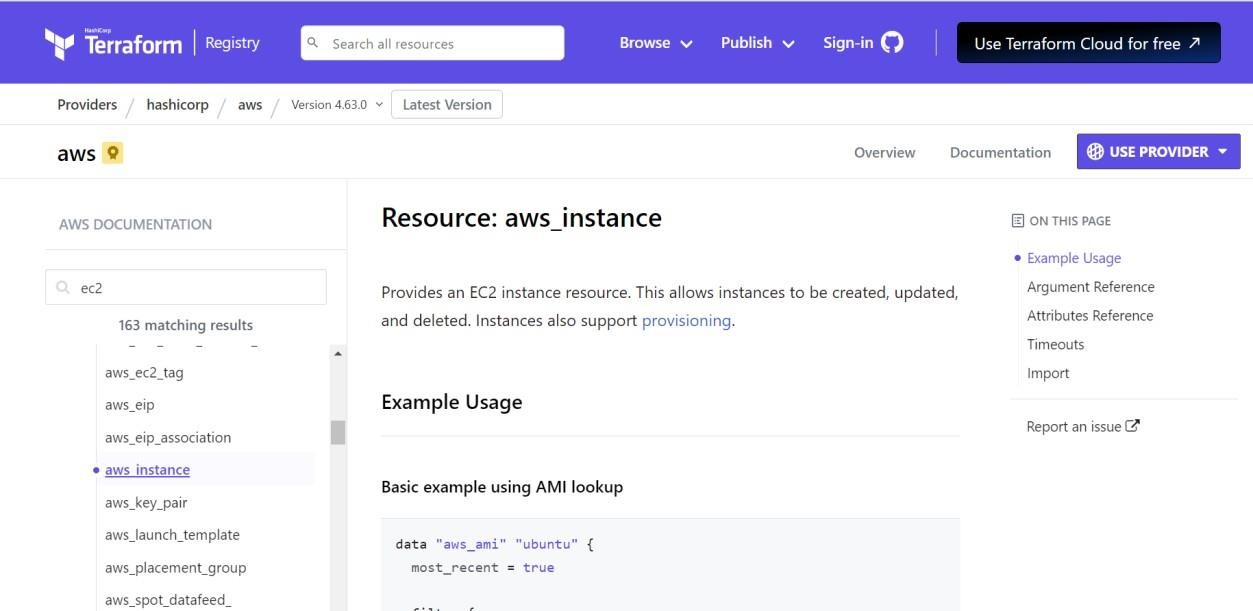
**Tools required:** Terraform, AWS account with security credentials, Keypair

**Expected Deliverables:**

* + Launch an EC2 instance using Terraform
  + Connect to the instance
  + Install Jenkins, Java, and Python in the instance
* Tip: launch ec2 instance from aws and use aws provider.

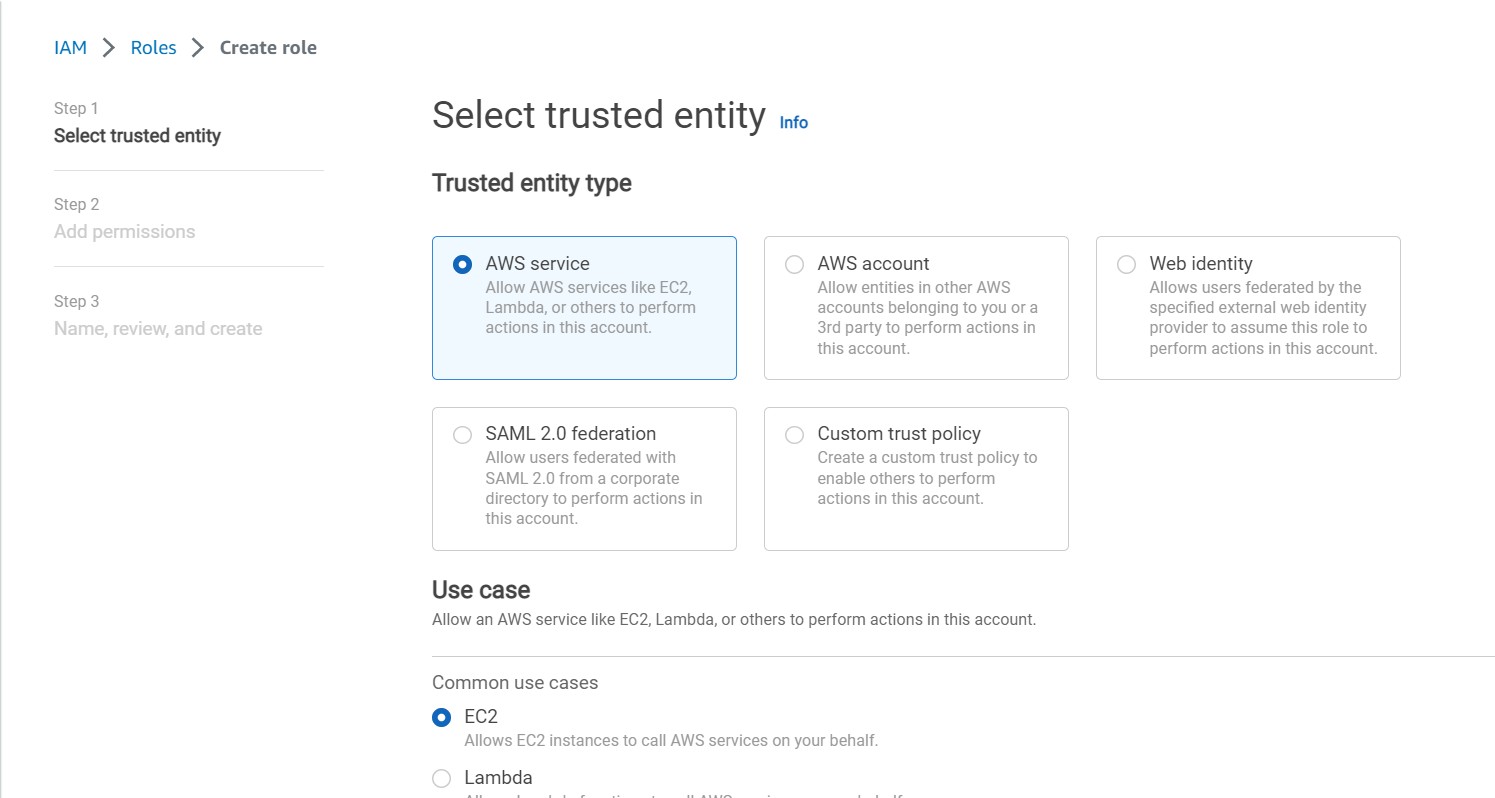
**Implementation:**

# Step 1: Install Terraform and set up AWS credentials

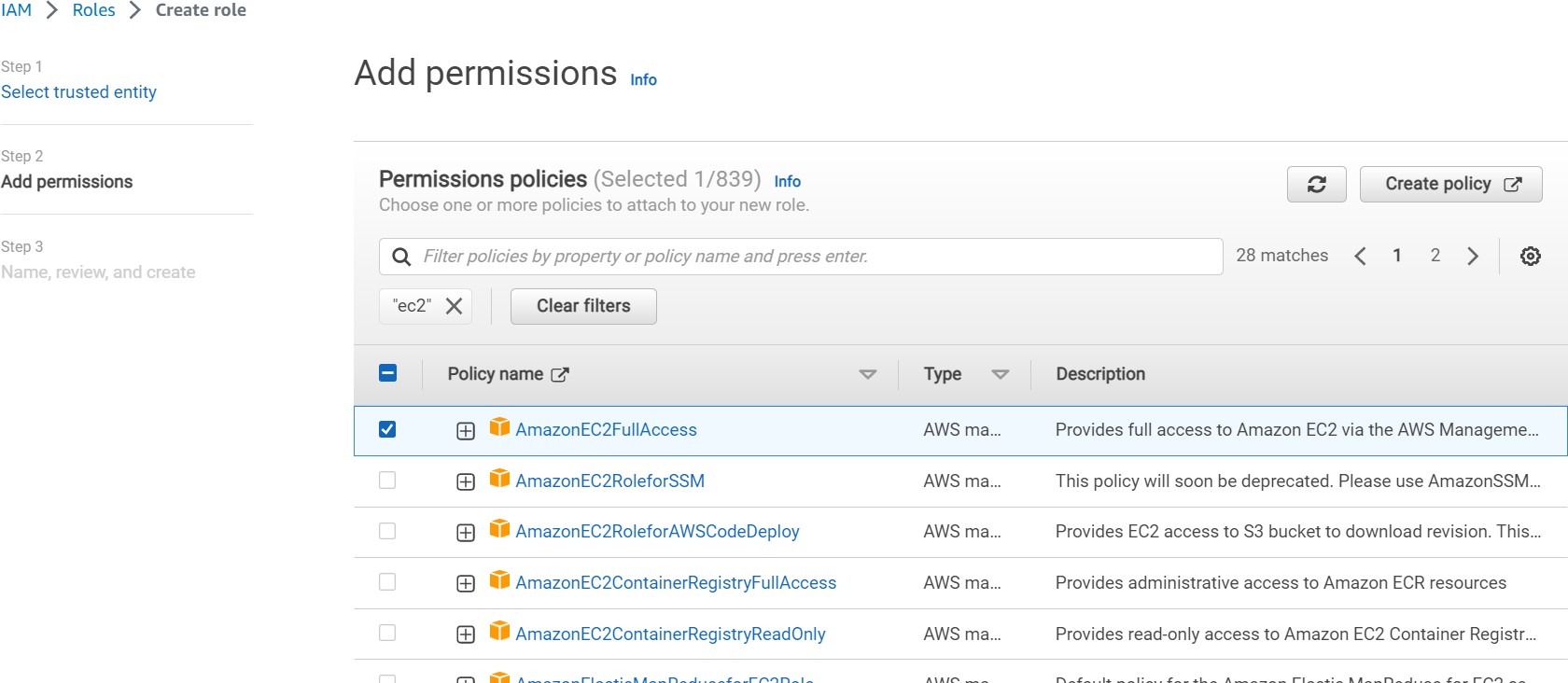


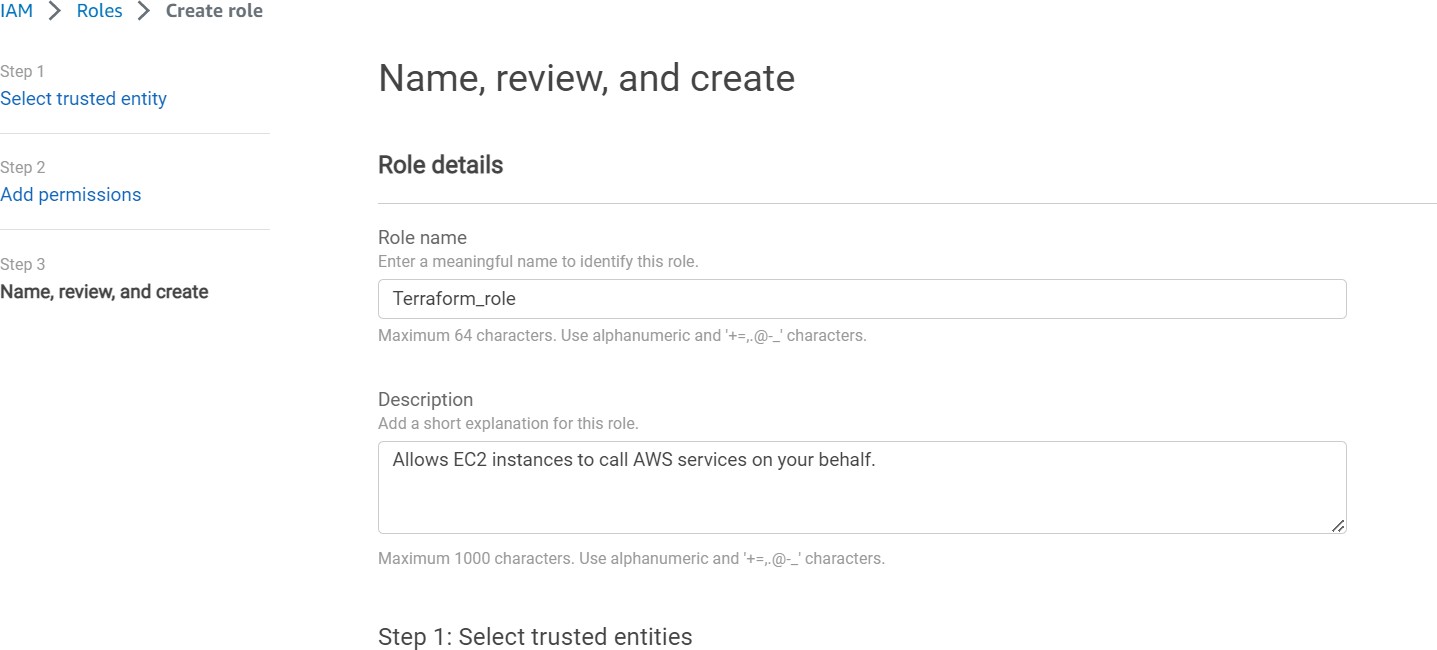
Here are the steps to create an IAM role and generate access keys:

1. Log in to the AWS Management Console and navigate to the IAM service.
2. Click on "Roles" in the left-hand navigation menu and then click the "Create Role" button.
3. Choose the "AWS service" as the trusted entity and then select "EC2" as the service that will use this role.

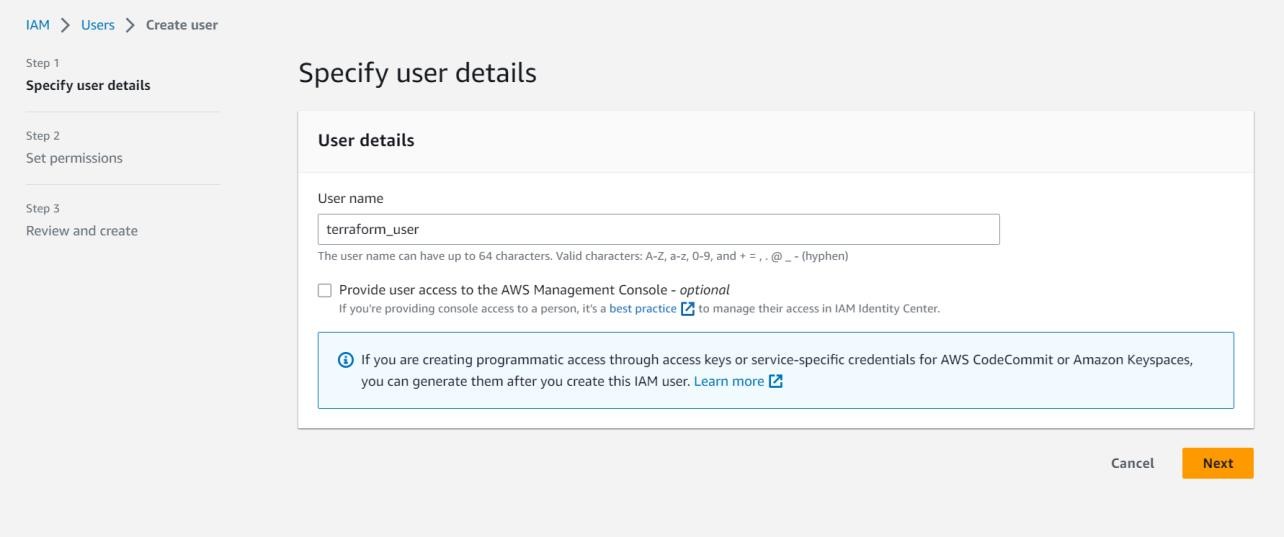


1. Attach the appropriate policies to the role based on the access level required for your infrastructure automation. For this project, you can attach the "AmazonEC2FullAccess" policy.

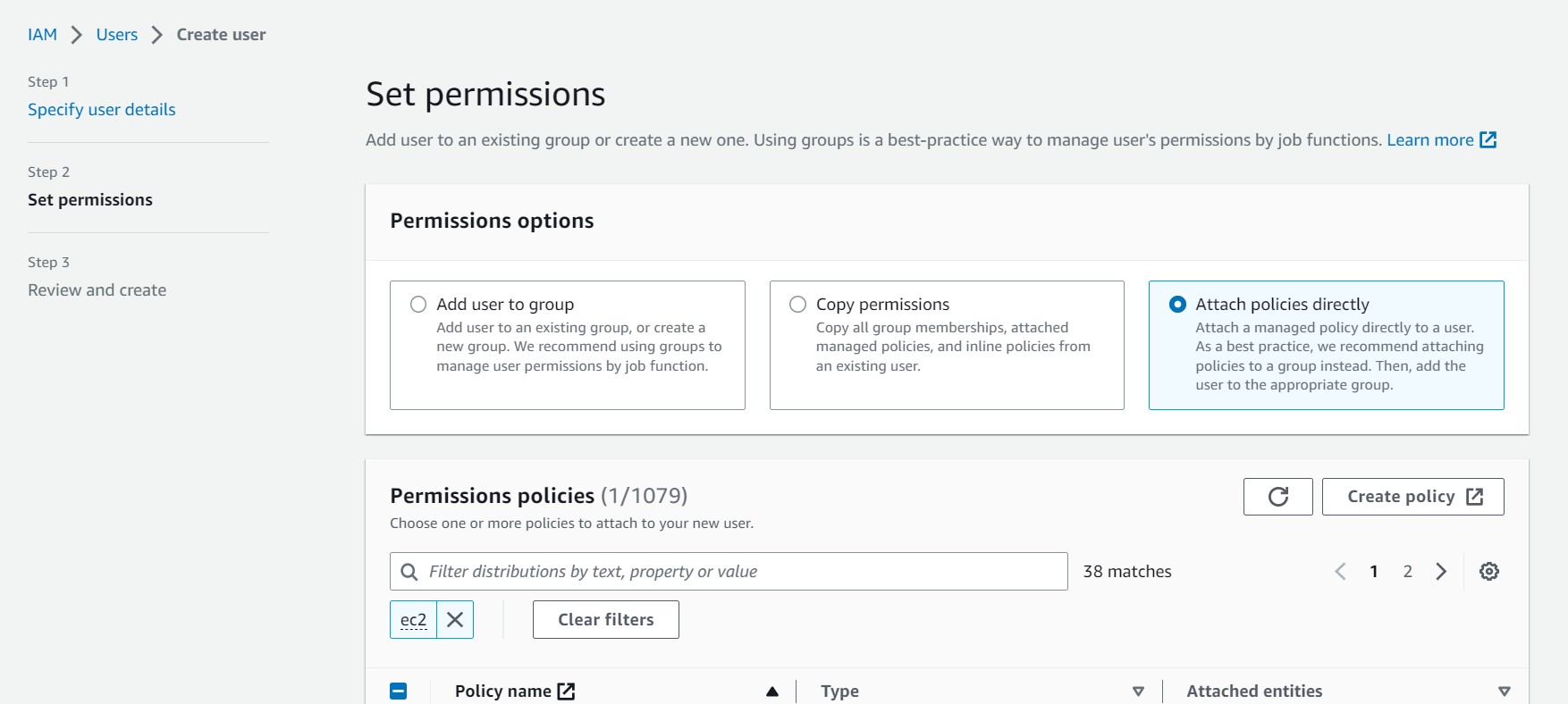


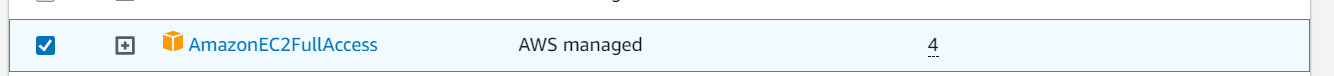


1. Review the settings and create the role.
2. Navigate to the "Users" section of the IAM console and click on "Add User".

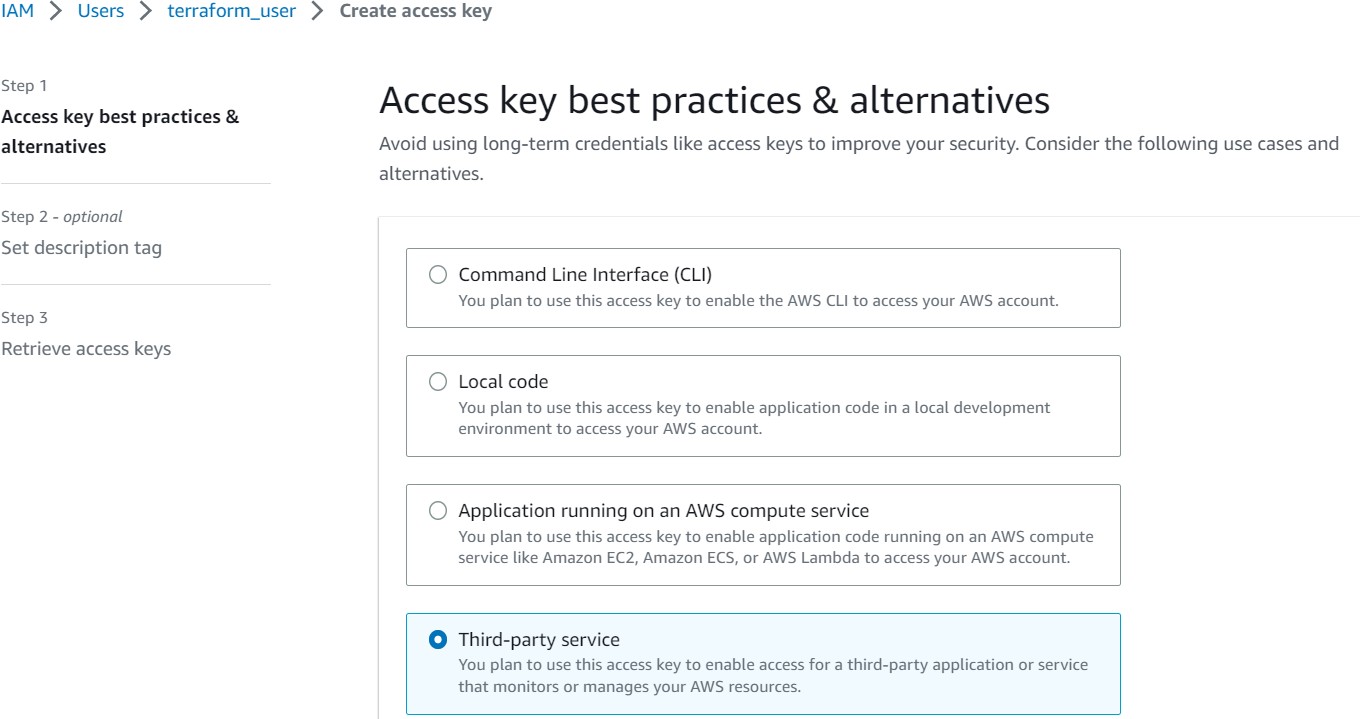
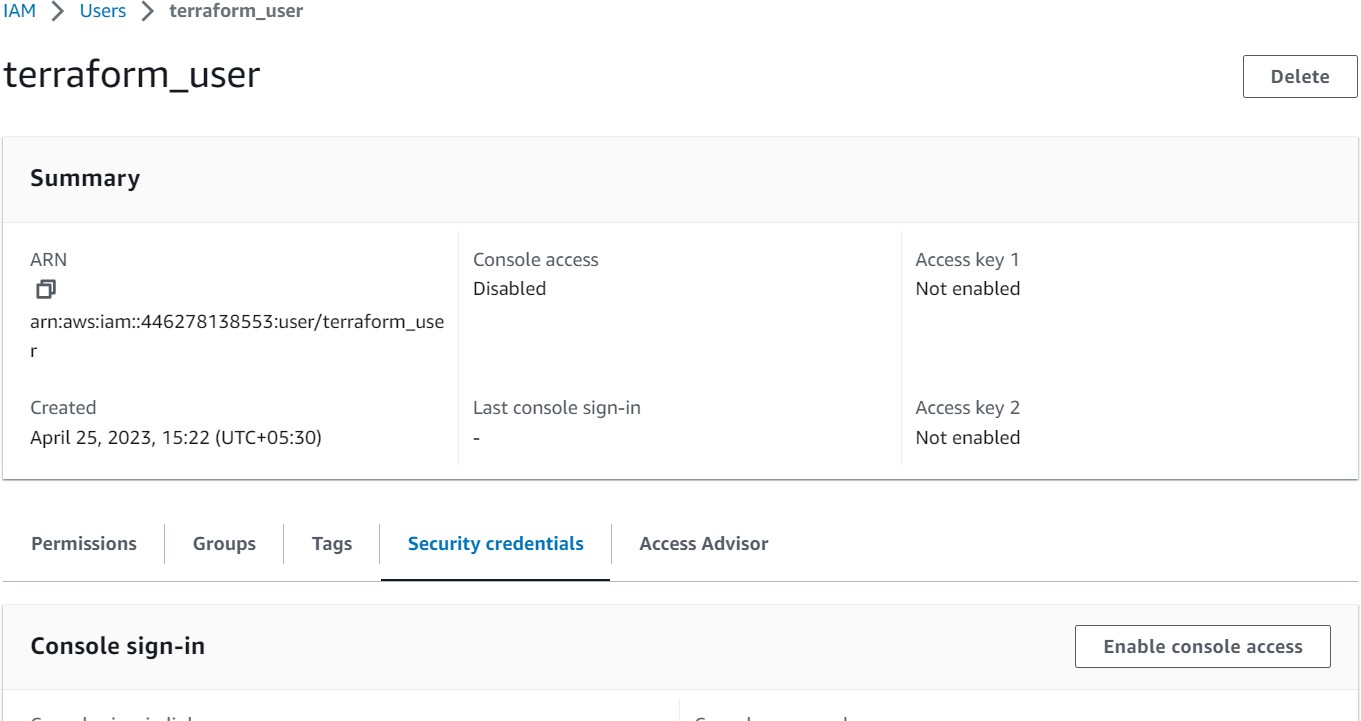


1. Give the user a name and select "Programmatic access" as the access type.
2. Attach the IAM role you created in step 4 to the user.

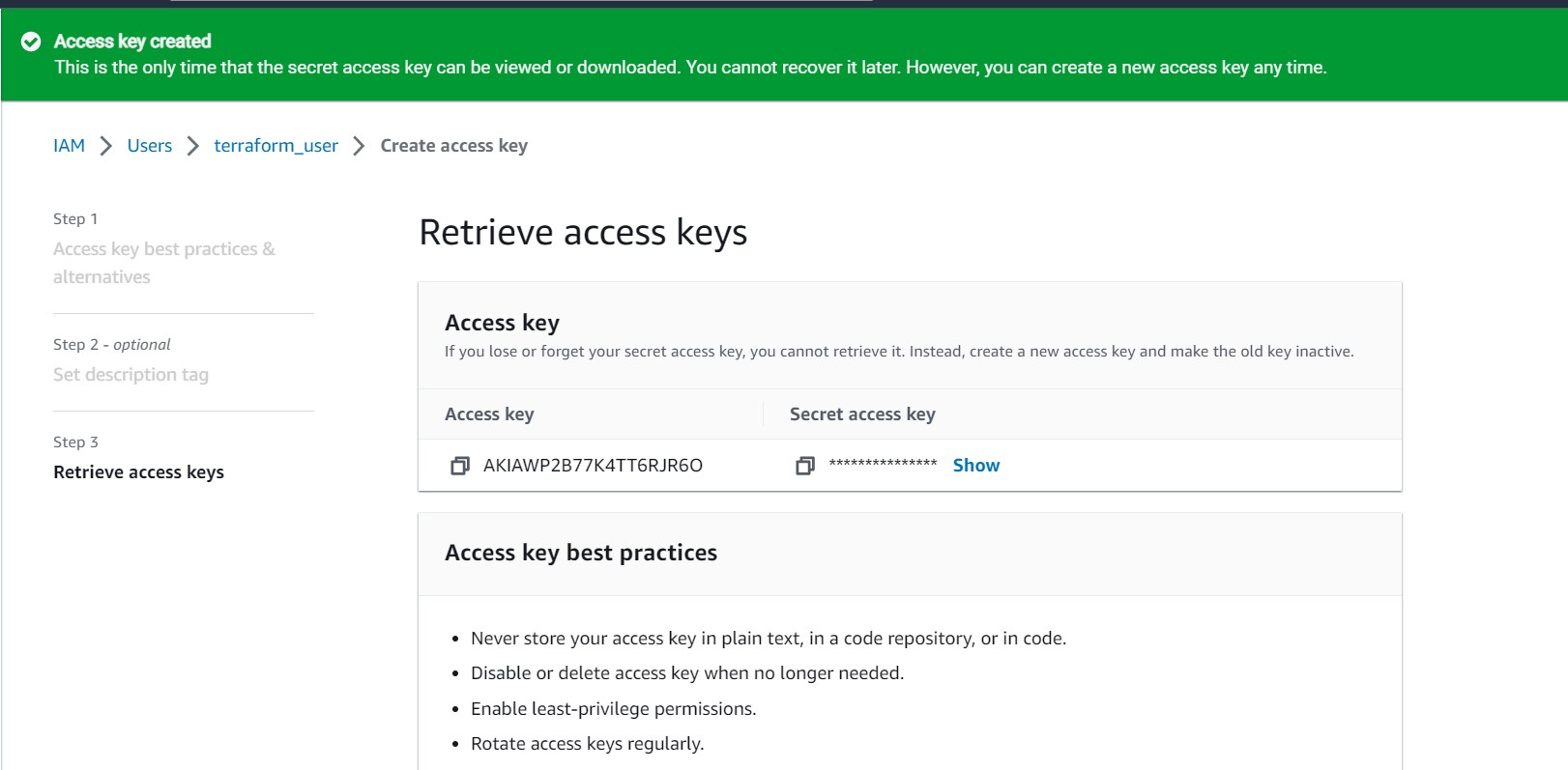




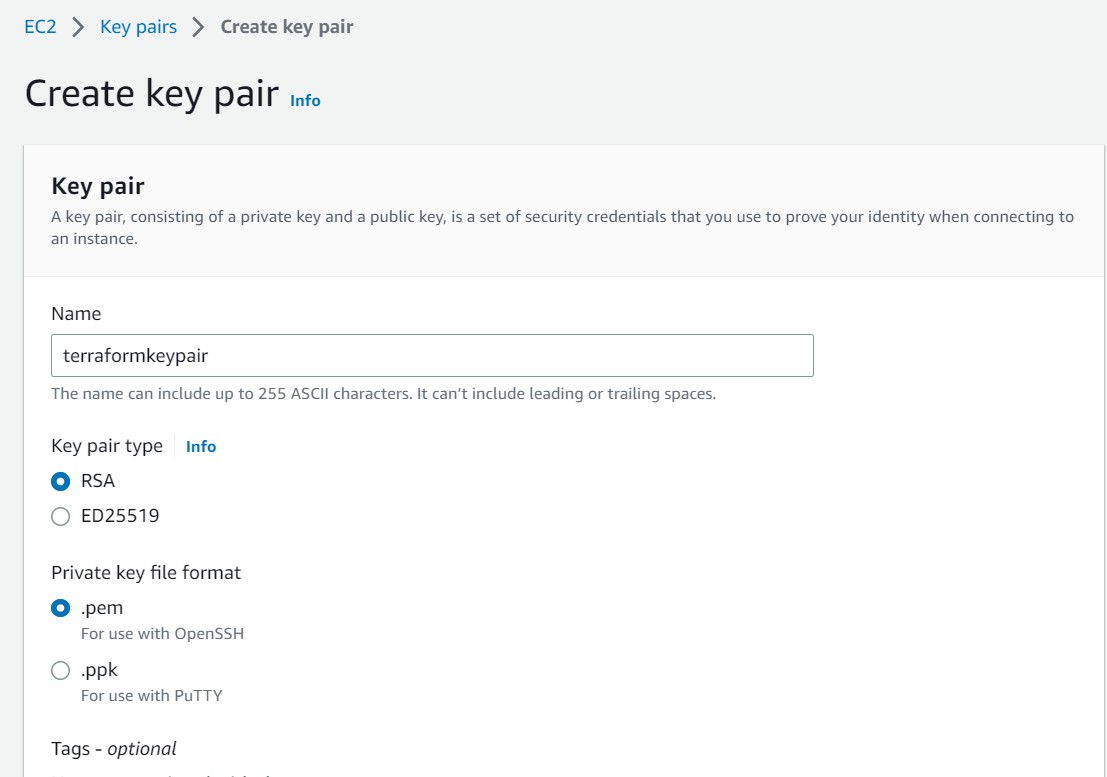
1. Review the settings and create the user.



1. After the user is created, you will be presented with an access key and secret key. Make sure to copy these keys to a secure location as they will not be displayed again.



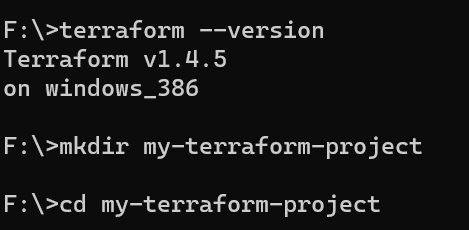
Once you have the access key and secret key, you can configure them in your Terraform provider configuration. Create an AWS key pair to access the EC2 instance later.



* Make sure you have the AWS CLI (Command Line Interface) installed on your local machine. If not, you can install it by following the instructions here: <https://aws.amazon.com/cli/>

# Step 2: Write Terraform script to launch EC2 instance

1. Create a new directory for your Terraform project on your local machine.



1. Inside the project directory, create a new file called "main.tf" (or any other name with ".tf" extension) which will contain your Terraform configuration.
2. Open "main.tf" in a text editor and write the following code:

In command prompt, type **notepad main.tf**

You can do this by adding the following code to your Terraform configuration file:

## provider "aws" {

**access\_key = "XXXXXXXXXXXXXXXXX"**

## secret\_key = "XXXXXXXXX" region = "us-east-1"

**}**

## resource "aws\_instance" "project-1-instance" { ami = "ami-007855ac798b5175e" instance\_type = "t2.micro"

**key\_name = "terraformkeypair"**

## tags = {

**Name = "Terraform-automated-instance"**

## }

**}**



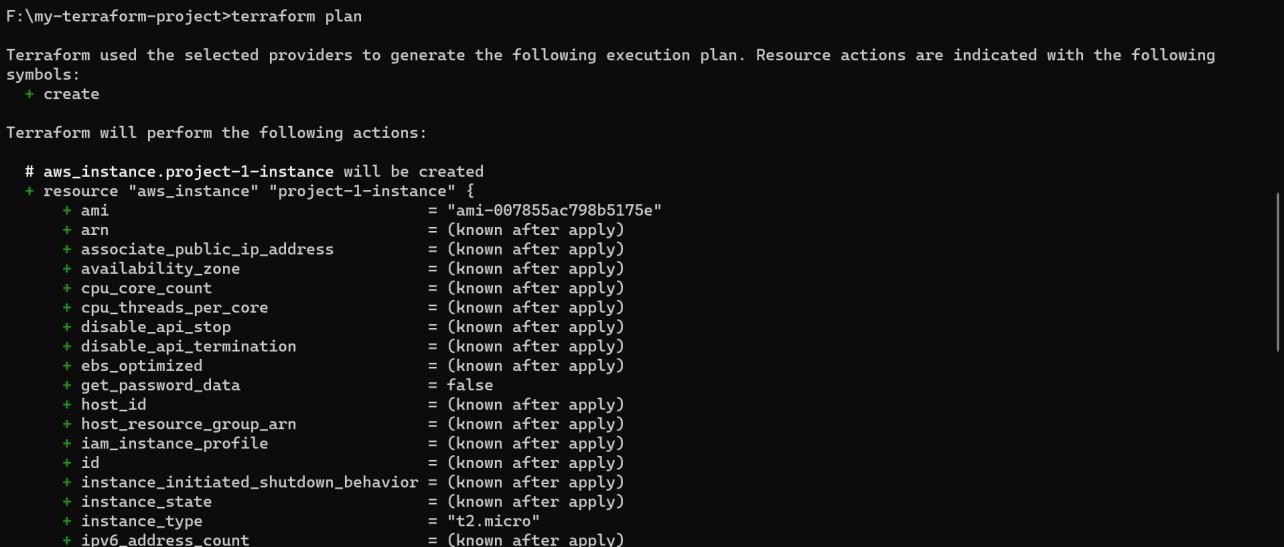
Step 3: Run Terraform script to launch EC2 instance

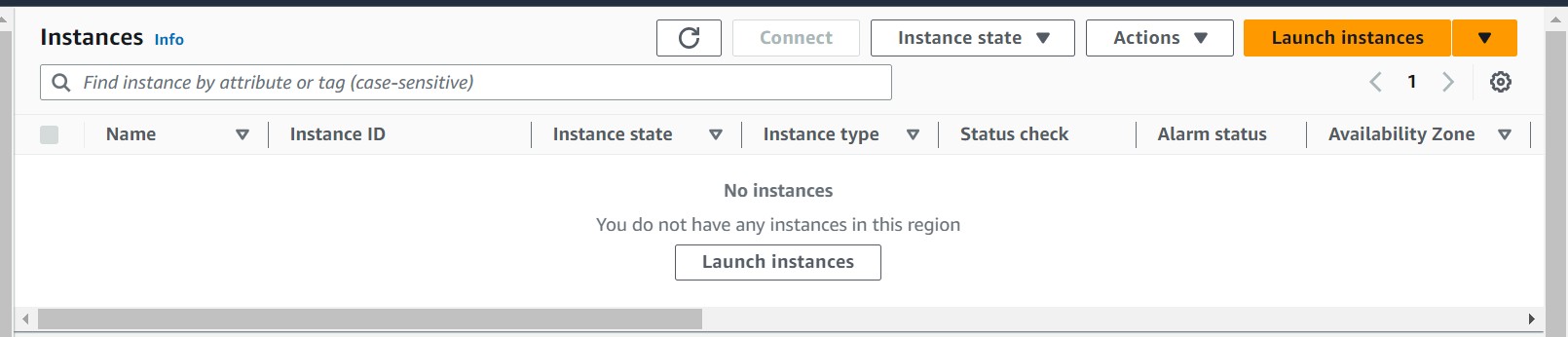
1. Open a terminal and navigate to the directory where you created your Terraform project.
2. Initialize the Terraform working directory by running the following command:

## terraform init

1. Validate your Terraform configuration by running the following command:

## terraform validate or terraform plan

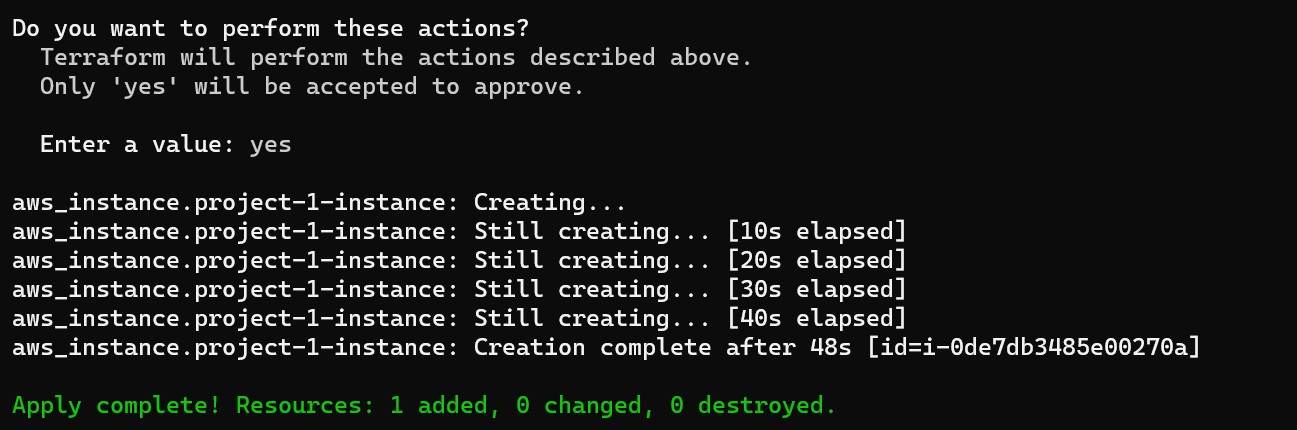


**Initially no instances are there in AWS**

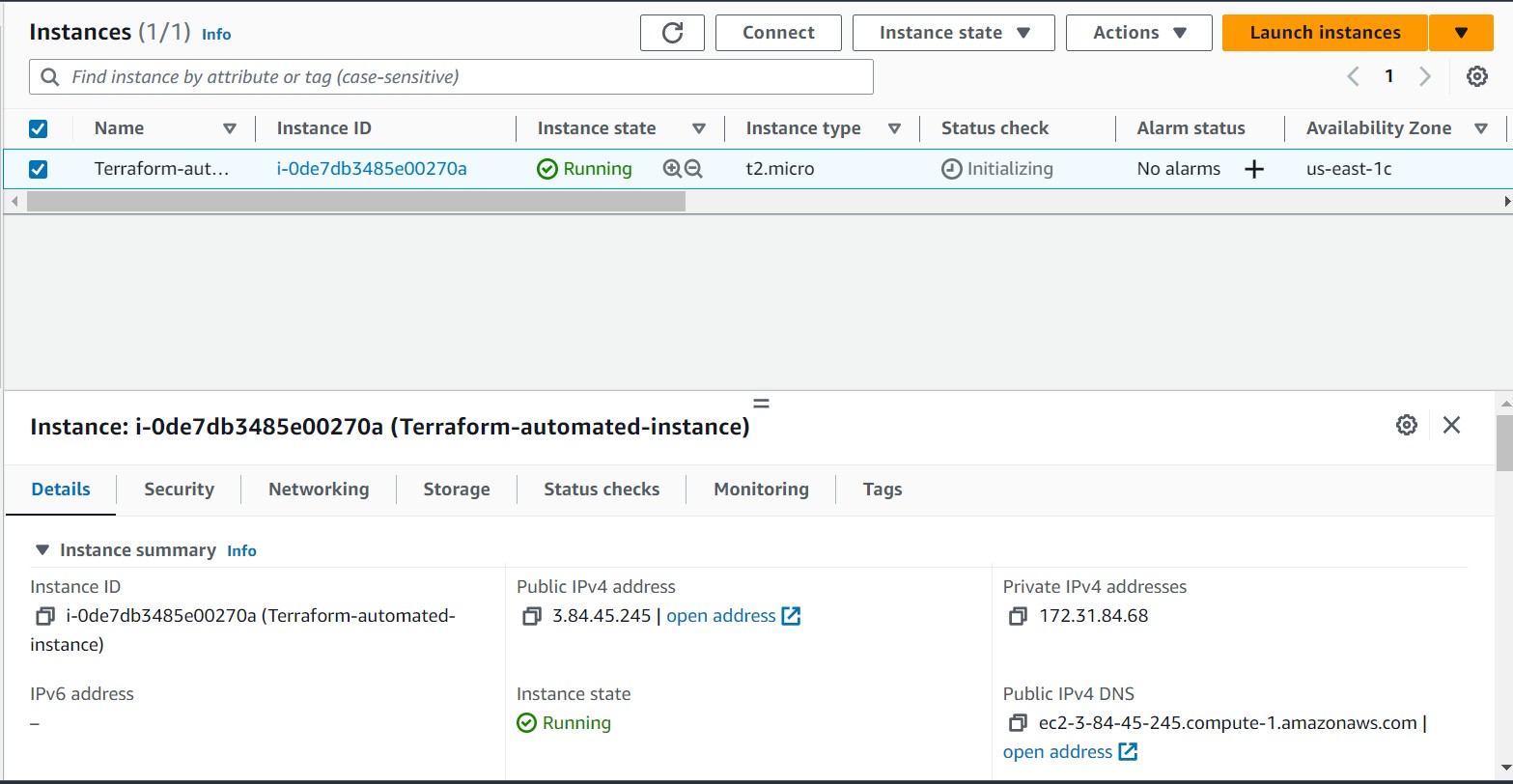
1. If there are no errors, you can proceed to apply the Terraform configuration by running the following command:

## terraform apply

1. Terraform will prompt you to confirm the changes. Type "yes" and press Enter to confirm and launch the EC2 instance.



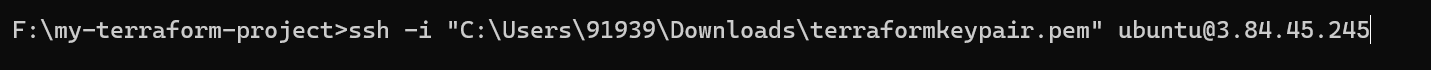
Now observe the AWS console(refresh)



# Step 4: Connect to EC2 instance using SSH

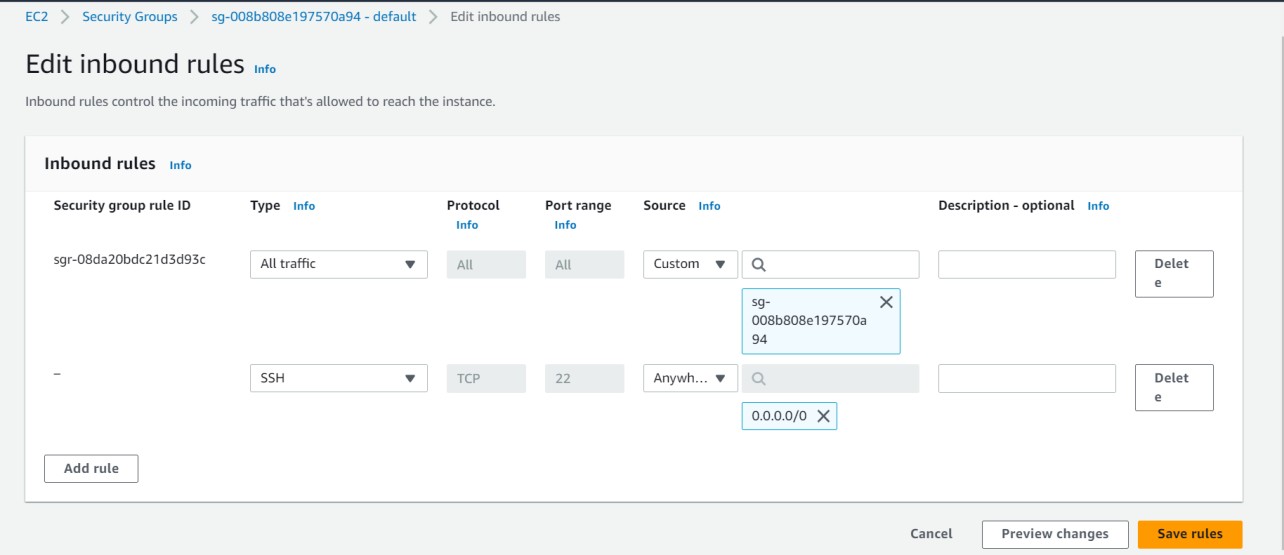
1. Once the EC2 instance is launched, Terraform will output the public IP address of the instance. Make a note of it.
2. Open a terminal on your local machine and navigate to the directory where you saved your AWS key pair.
3. Set the permissions of the key pair file to be readable only by the owner by running the following command:

## chmod 400 YOUR\_KEY\_PAIR\_NAME.pem

1. Connect to the EC2 instance using SSH and the key pair by running the following command:

## ssh -i YOUR\_KEY\_PAIR\_NAME.pem ubuntu@PUBLIC\_IP\_ADDRESS

Note: Replace "YOUR\_KEY\_PAIR\_NAME.pem" or keypair path with the name of your key [edit inbound rule to allow SSh ttaffic on port 22]

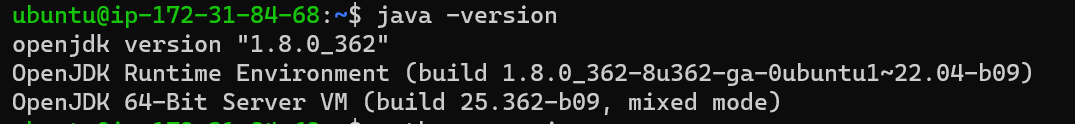
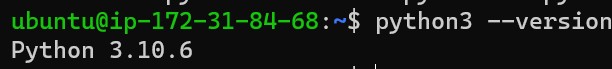


# Step 5: Install Jenkins, Java, and Python on EC2 instance

1. Once connected to the EC2 instance, you can install Jenkins, Java, and Python using the appropriate package manager for the operating system of your EC2 instance. Here are the commands for some common Linux distributions:

## sudo apt-get update

**sudo apt install openjdk-8-jdk python3 python3-pip**



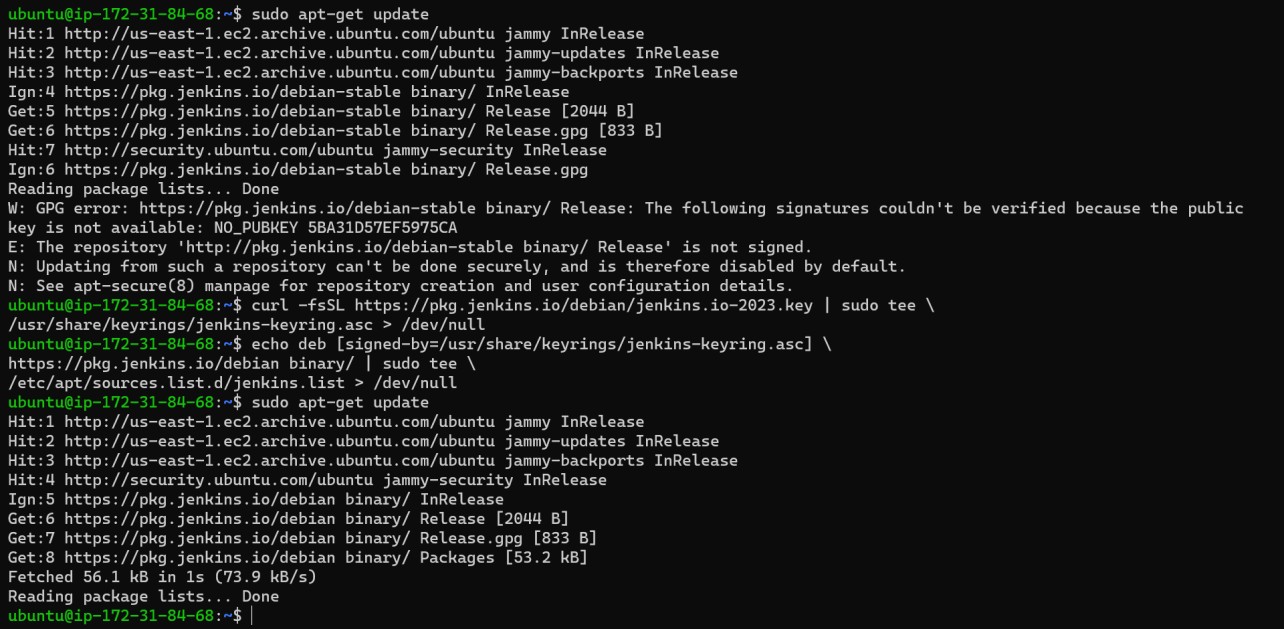
**To install Jenkins, java must be installed**

**curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \**

## /usr/share/keyrings/jenkins-keyring.asc > /dev/null

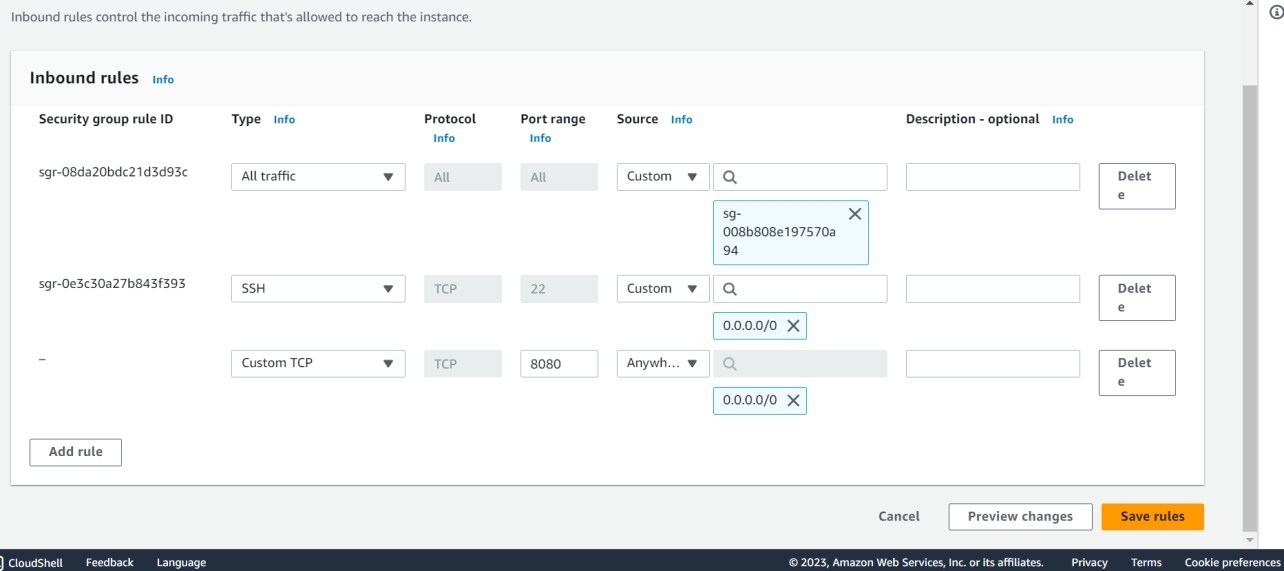
**echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \ https://pkg.jenkins.io/debian binary/ | sudo tee \**

## /etc/apt/sources.list.d/jenkins.list > /dev/null

**sudo apt-get update**

## sudo apt-get install jenkins

Source: Jenkins installation <https://pkg.jenkins.io/debian/>and choose Ubuntu

And also include port 8080 in inbound rules for the instance

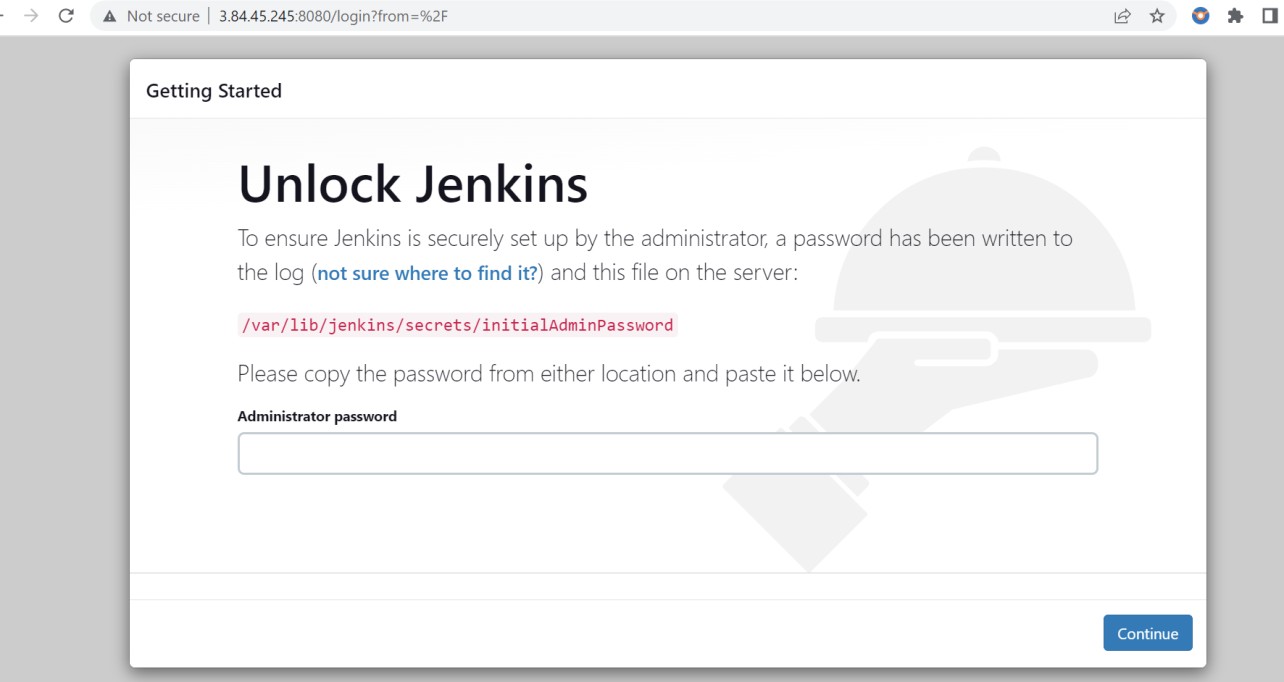
Note: You may need to modify the package names or package manager commands based on the specific Linux distribution you are using.

1. Start the Jenkins service by running the following command:

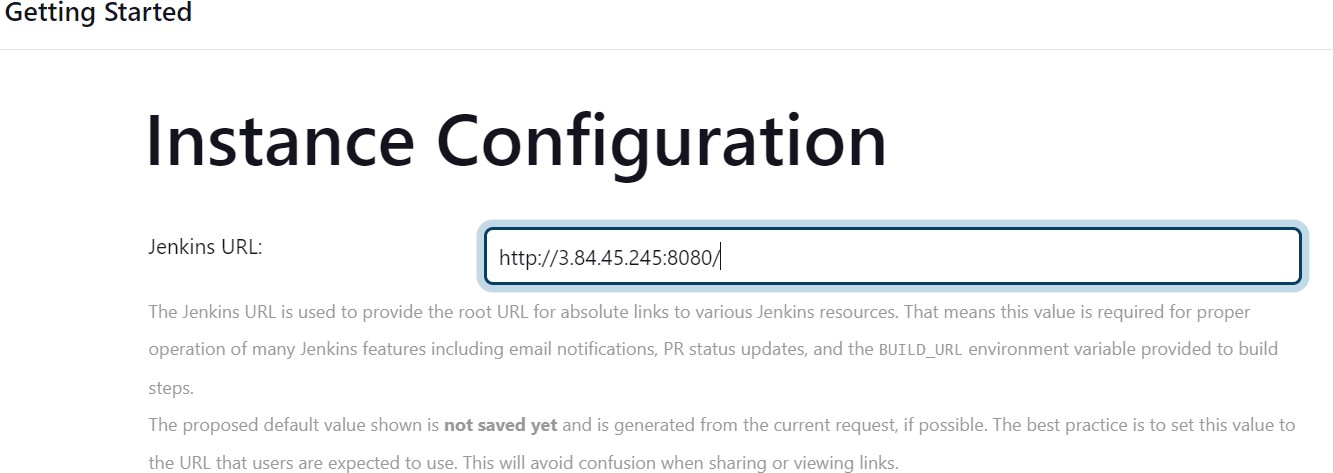
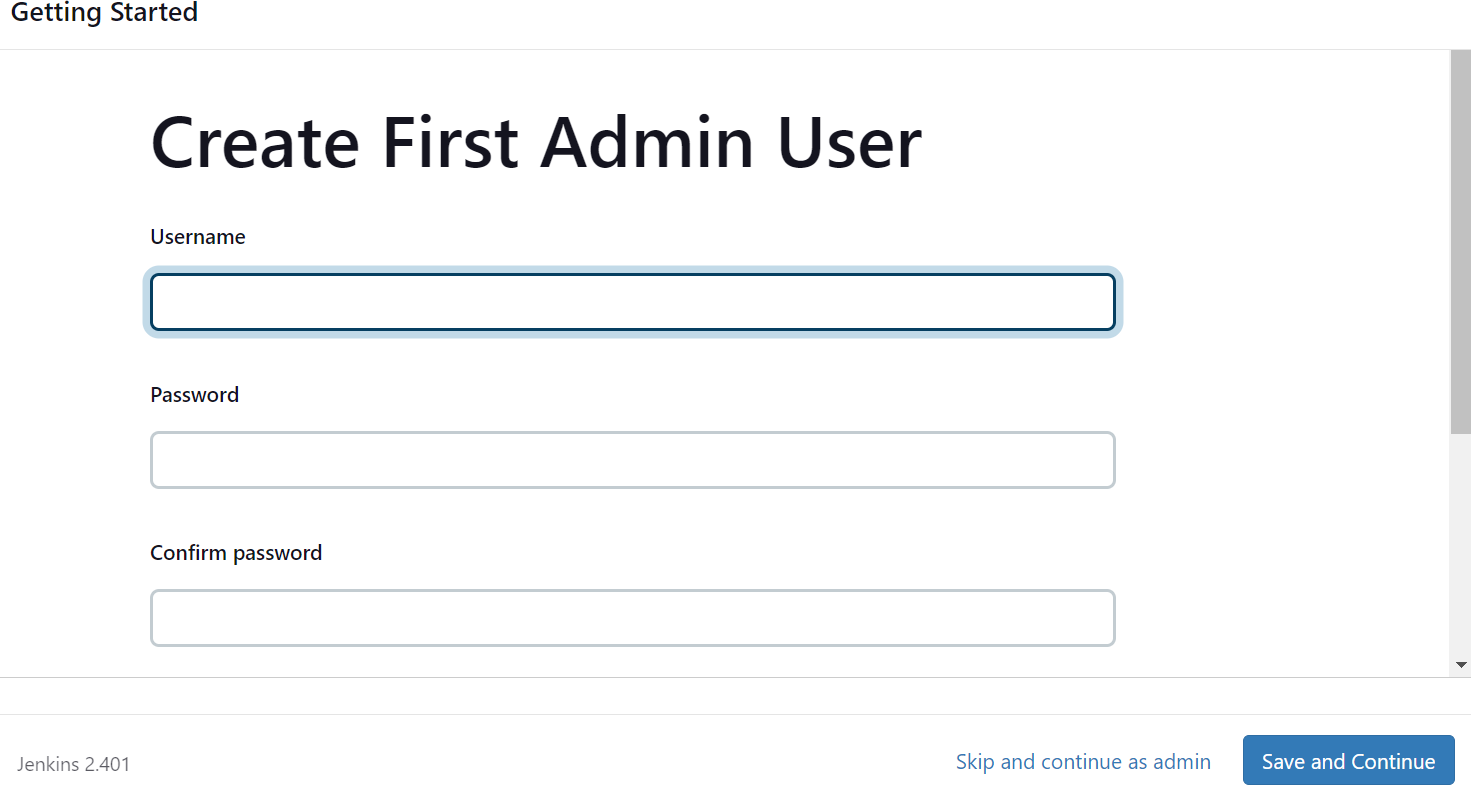
## sudo systemctl start jenkins

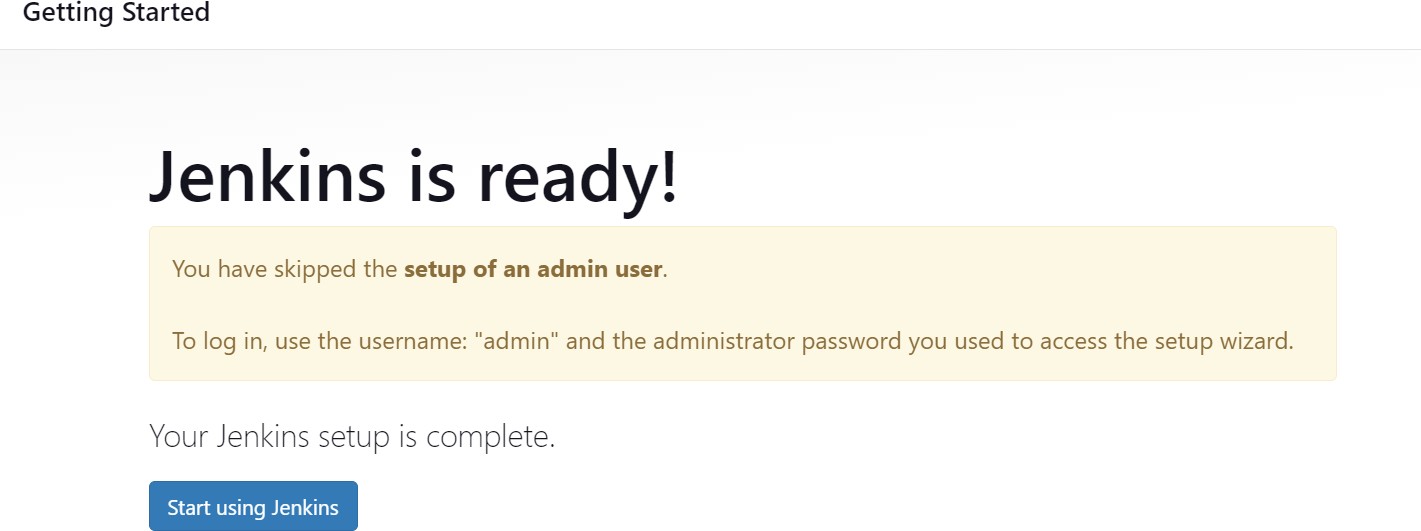
1. Open a web browser and access the Jenkins web interface by navigating to the public IP address of your EC2 instance followed by port 8080 (e.g., http://PUBLIC\_IP\_ADDRESS:8080).
2. Retrieve the Jenkins unlock key by running the following command on the EC2 instance:

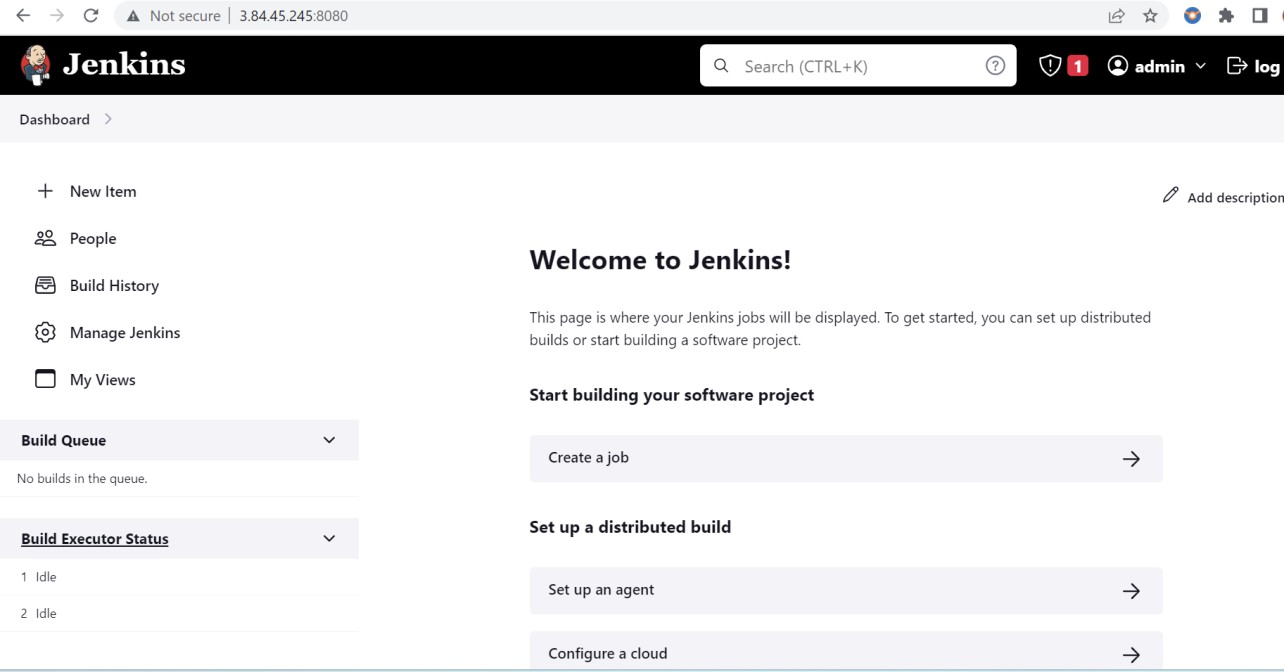
## sudo cat /var/lib/jenkins/secrets/initialAdminPassword



1. Copy the unlock key and paste it in the Jenkins web interface to unlock Jenkins and set up the admin user.
2. Follow the on-screen instructions to complete the Jenkins setup wizard and install any required plugins.







That's it! You now have an EC2 instance running with Jenkins, Java, and Python installed, ready for automation tasks. You can further configure Jenkins and install additional plugins as needed for your specific project requirements.