

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
```



```
markmcaneda@ip-172-31-43-226: ~
File Edit View Search Terminal Help
markmcaneda@ip-172-31-43-226:~$ ansible --version
ansible [core 2.12.10]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['~/home/markmcaneda/.ansible/plugins/modules',
  '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = ~/home/markmcaneda/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Sep 11 2024, 15:47:36) [GCC 11.4.0]
  jinja version = 3.0.3
  libyaml = True
markmcaneda@ip-172-31-43-226:~$
```

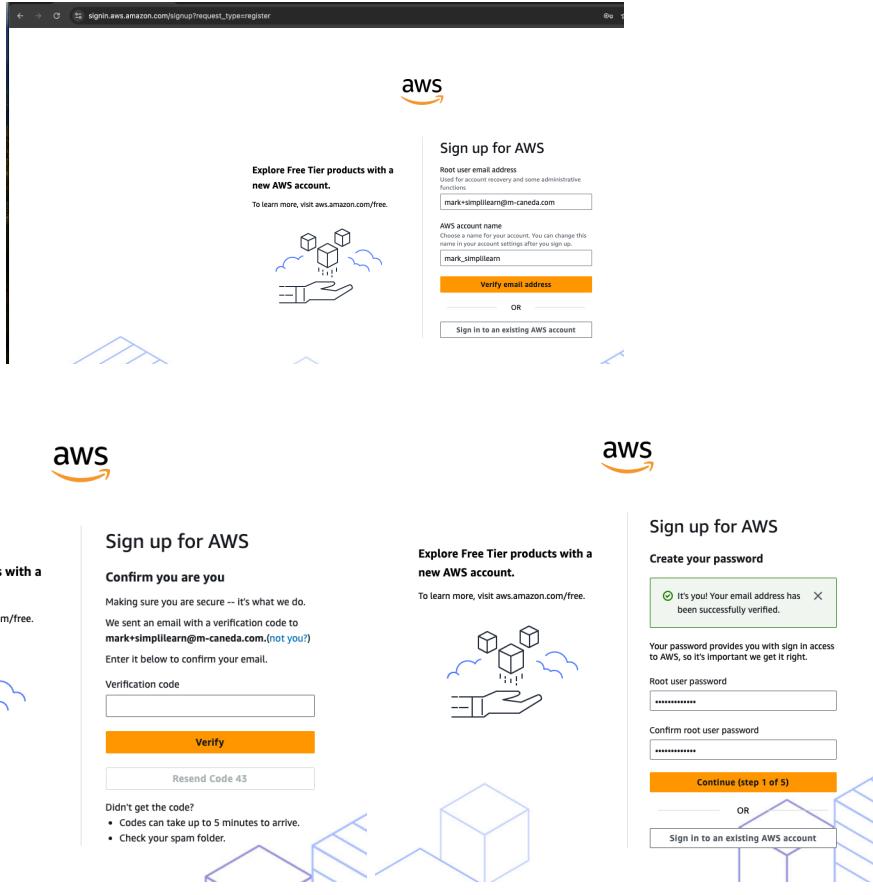
III. AWS Account and Keypair Setup (AWS Account)

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."



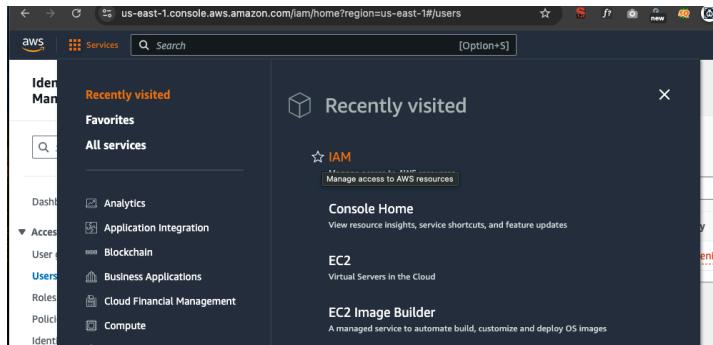
2. **Sign Up:** Enter your email address, create a password, and set an account name. Click "Verify email address" to continue and complete the registration steps.



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

A screenshot of the AWS IAM Users page. The left sidebar shows "Identity and Access Management (IAM)" with a "Users" section selected. The main area displays a table titled "Users (1) Info". The table has one row with the following data:

User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access keys
corestack-639f2	/	0	Access denied	-	1584 days	-	Access keys

The "Create user" button is located at the top right of the table area.

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.

IAM > Users > Create user

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Step 4
Retrieve password

Specify user details

User details

User name: The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and * = , . @ _ - (hyphen)

Provide user access to the AWS Management Console - optional If you're providing console access to a person, it's a best practice [to manage their access in IAM Identity Center](#).

Console password:

- Autogenerated password You can view the password after you create the user.
- Custom password Enter a custom password for the user.

Show password

Users must create a new password at next sign-in - Recommended Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel **Next**

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.

IAM > Users > Create user

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Step 4
Retrieve password

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

- Add user to group Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- Copy permissions Copy all group memberships, attached managed policies, and inline policies from an existing user.
- Attach policies directly Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

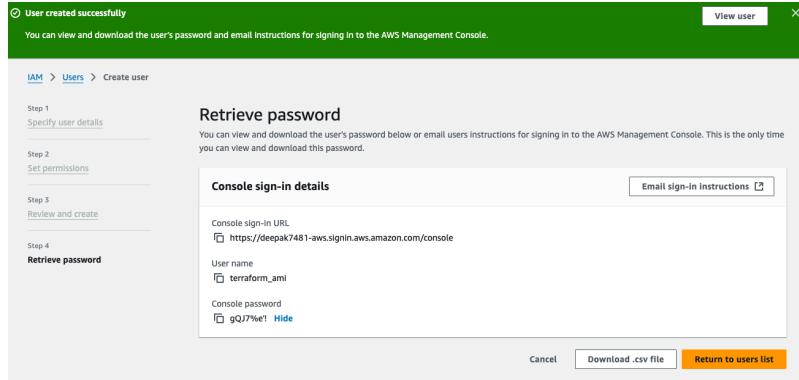
User groups (2/5)

Group name	Users	Attached policies	Created
admin	0	AdministratorAccess	2021-12-11 (2 years ago)
MygroupForLab	0	AdministratorAccess	2020-11-19 (3 years ago)
ProjectCompanyAdmin	0	-	2023-03-26 (1 year ago)
SysAdmin	0	AmazonEC2FullAccess	2021-07-13 (3 years ago)
vjgroup	0	AmazonEC2FullAccess and ...	2020-07-21 (4 years ago)

Set permissions boundary - optional

Cancel **Previous** **Next**

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.



- 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.

The screenshot shows the AWS IAM 'Users' page with three entries:

User name	Path	Group	Last activity	MFA	Password age	Console
corestack-639f2	/	0	Access denied	-	1584 days	-
terraform	/	1	-	-	1 minute	-
terraform_ami	/	0	-	-	-	-

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

The screenshot shows the AWS IAM 'User Details' page for 'terraform_ami'. The 'Security credentials' tab is selected. In the 'Access keys' section, there is a 'Create access key' button highlighted with a red arrow.

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

The screenshot shows the 'Create access key' wizard, Step 3: Use case. The 'Application running on an AWS compute service' option is selected and highlighted with a red arrow.

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

Screenshot 1: Set description tag - optional

The 'Description tag value' field contains the text 'AWS Terraform AMI'. Below it, a note states: 'The description for this access key will be attached to this user as a tag and shown alongside the access key.' A note at the bottom specifies: 'Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ : / = - @'.

Screenshot 2: Retrieve access keys

The 'Access key' table shows one entry:

Access key	Secret access key
<input checked="" type="checkbox"/> AKIAUUAHP6XOUV6M5QE7M	<input type="checkbox"/> ***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

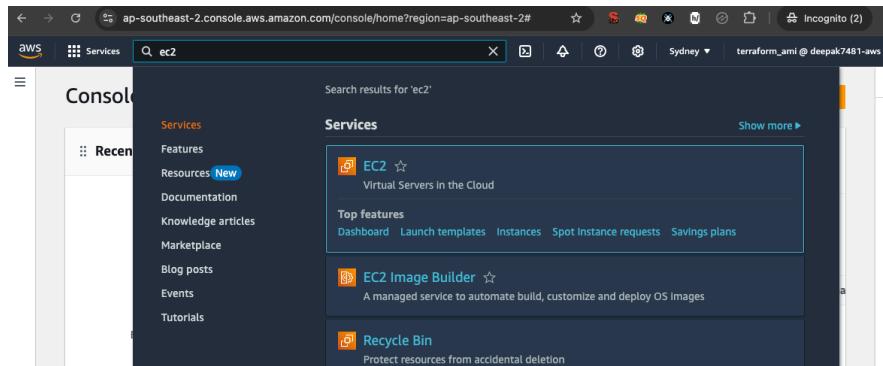
Buttons at the bottom: 'Download .csv file' and 'Done'.

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

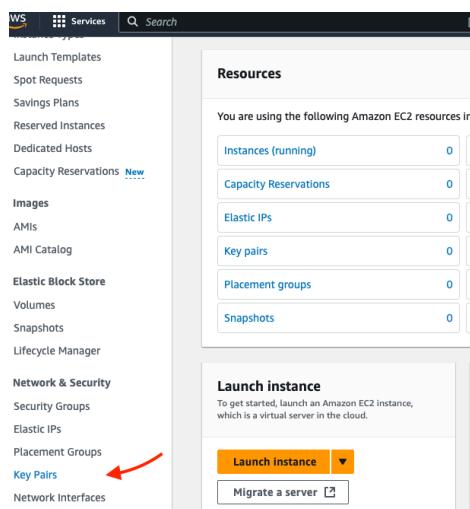
Access key ID	Secret access key
AKIAUUAHP6XOUV6M5QE7M	t0exaRvDEo+IZVuclldn0S91YQUWwZJ/OYTDzo3

Generate Key Pairs of EC2 Instance

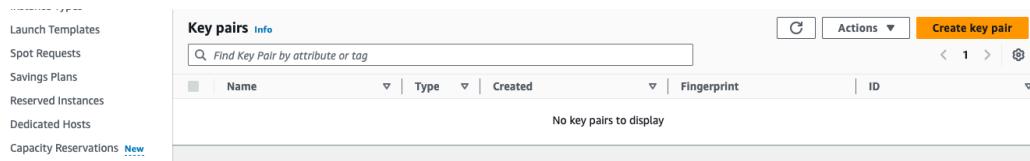
- Once done above, log in to the AWS, after logging in to the AWS Console, Type “EC2” in the search bar to land in the EC2 Dashboard.



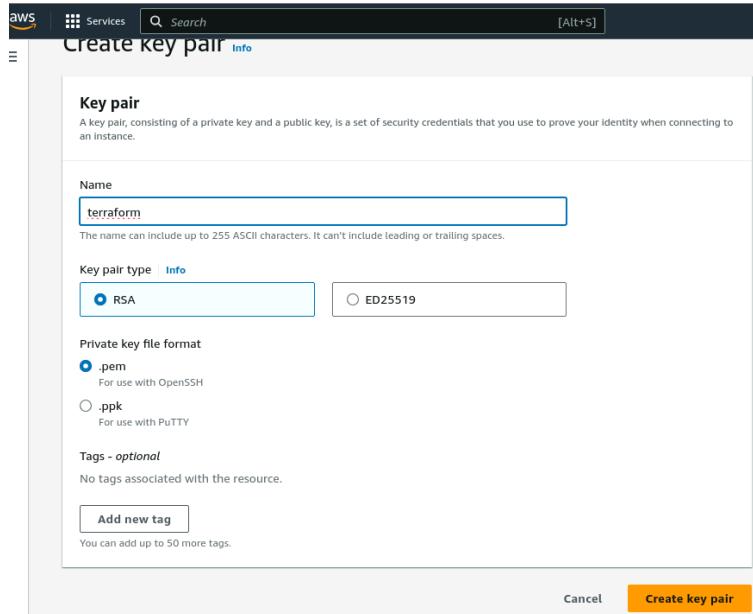
- Search for “Key Pairs”, select and go to the page. This is another required keys that Terraform need for accessing EC2 Instance.



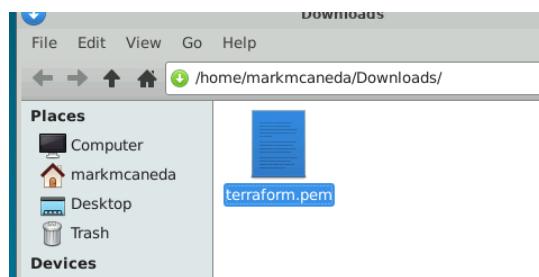
- Click “Create key pair”.



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.

Key pairs (1/1) Info					
Find Key Pair by attribute or tag Actions ▾ Create key pair					
Name	Type	Created	Fingerprint	ID	▼
terraform	rsa	2024/11/12 04:19 GMT+0	0e:01:da:a6:5a:83:b9:21:18:5c:7d:3d:73:1a:92:f...	key-00ae9c392...	▼

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

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

The screenshot shows the AWS EC2 console with the "Security Groups" section selected. A single security group, "sg-0d4eabfb133dcc8f1", is listed under the "default" VPC. The list includes columns for Name, Security group ID, Security group name, and VPC ID. The "Security Groups" link in the left sidebar is highlighted in blue, indicating it is the active section.

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

The screenshot shows the same EC2 Security Groups page as before, but with a red arrow pointing to the "Edit inbound rules" option in the context menu that appears when right-clicking the selected security group. The "Actions" dropdown menu is open, and the "Edit inbound rules" item is clearly visible.

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

The screenshot shows the "Edit inbound rules" configuration page for the selected security group. At the bottom left, there is a prominent "Add rule" button, which is highlighted with a red arrow. The page includes fields for Security group rule ID (sg-01567cc952e2ef327), Type (All traffic), Protocol (All), Port range (All), Source (Custom), and Description (optional). Buttons for "Cancel", "Preview changes", and "Save rules" are at the bottom right.

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.

Edit inbound rules [Info](#)
Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules [Info](#)

Security group rule ID	Type Info	Protocol Info	Port range	Source Info	Description - optional Info
sgr-01567cc952e2ef327	All traffic	All	All	Custom	<input type="text" value="sg-0d4eabfb133dcc8f1"/> Delete
-	SSH	TCP	22	Anyw...	<input type="text" value="0.0.0.0/0"/> Delete
-	Custom TCP	TCP	8080	Anyw...	<input type="text" value="0.0.0.0/0"/> Delete

[Add rule](#) [Cancel](#) [Preview changes](#) **Save rules**

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

Inbound security group rules successfully modified on security group (sg-0d4eabfb133dcc8f1 | default)

Details

Security Groups (1) [Info](#)

Name	Security group ID	Security group name	VPC ID	Description
sg-0d4eabfb133dcc8f1	sg-0d4eabfb133dcc8f1	default	vpc-0587054f4a10d37b5	default V

IV. Setup Terraform Configuration Files

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

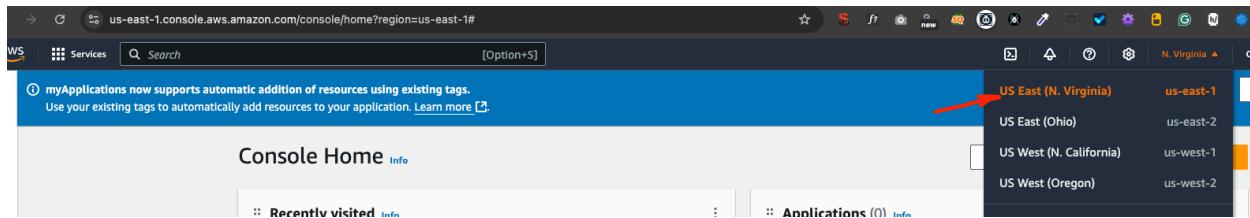
```
mkdir terraform-jenkins && cd terraform-jenkins
```

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)

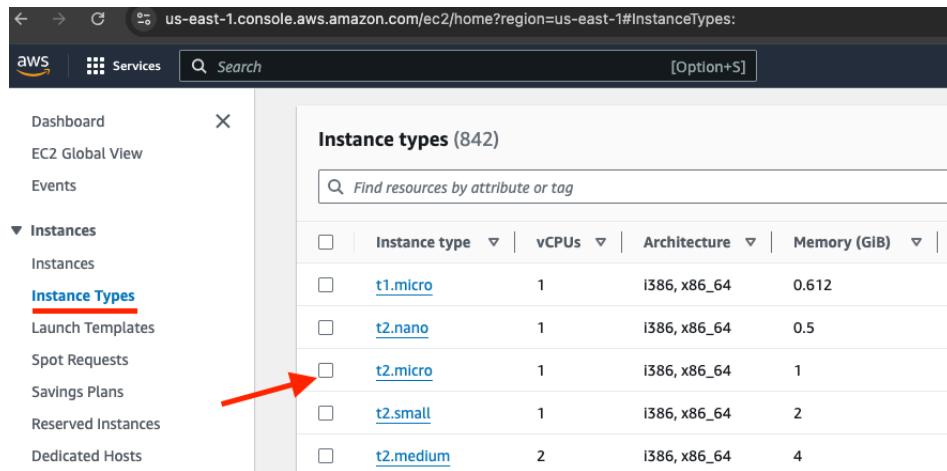
The screenshot shows the AWS AMI Catalog search results for "ubuntu". The search bar at the top has "ubuntu" typed into it. Below the search bar, there are several filters:

- Quick Start AMIs (8)
- My AMIs (0)
- AWS Marketplace AMIs (3573)
- Community AMIs (500)

 On the left, there is a sidebar with navigation links like Dashboard, EC2 Global View, Events, Instances, Images, and Elastic Block Store. The "AMI Catalog" link is highlighted with a red arrow. Below the search bar, there is a "Refine results" section with a "Clear all filters" button and a checked checkbox for "All Linux/Unix". Other available filters include "Free tier only", "OS category", "Architecture", and "64-bit (Arm)". The main results table shows two entries for "ubuntu":

Platform	Description	Select
Ubuntu	Ubuntu Server 24.04 LTS (HVM), SSD Volume Type ami-0866a3c8686eaeeba (64-bit (x86)) / ami-0325498274077fac5 (64-bit (Arm)) Ubuntu Server 24.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services). Platform: ubuntu Root device type: ebs Virtualization: hvm ENA enabled: Yes	<input checked="" type="button" value="Select"/> <input type="radio" value="64-bit (x86)"/> 64-bit (x86) <input type="radio" value="64-bit (Arm)"/> 64-bit (Arm)
Ubuntu	Ubuntu Server 22.04 LTS (HVM), SSD Volume Type ami-005fc0f236362d99f (64-bit (x86)) / ami-07ee04759daf109de (64-bit (Arm)) Ubuntu Server 22.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical	<input type="button" value="Select"/> <input checked="" type="radio" value="64-bit (x86)"/> 64-bit (x86)

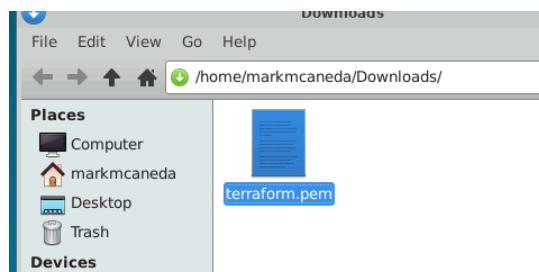
- For INSTANCE TYPE, this can be found from EC2 Instances Types. Copy the name of the Instance type: `t2.micro` as example free-tier of this project.



The screenshot shows the AWS EC2 Instance Types page. On the left, there's a sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types (highlighted), Launch Templates, Spot Requests, Savings Plans, Reserved Instances, and Dedicated Hosts. The main area is titled "Instance types (842)". It has a search bar and a table with columns: Instance type, vCPUs, Architecture, and Memory (GiB). The table lists several instance types, with `t2.micro` highlighted and a red arrow pointing to its row. The table data is as follows:

	Instance type	vCPUs	Architecture	Memory (GiB)
<input type="checkbox"/>	t1.micro	1	i386, x86_64	0.612
<input type="checkbox"/>	t2.nano	1	i386, x86_64	0.5
<input type="checkbox"/>	t2.micro	1	i386, x86_64	1
<input type="checkbox"/>	t2.small	1	i386, x86_64	2
<input type="checkbox"/>	t2.medium	2	i386, x86_64	4

- 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.

```

16   provisioner "remote-exec" {
17     inline = [
18       # Update and install required packages
19       "sudo apt update -y",
20       "sudo apt upgrade -y",
21
22       # Install Java 17 (required for Jenkins) and fontconfig
23       "sudo apt install -y fontconfig openjdk-17-jdk",
24
25       # Download and configure the stable Jenkins repository
26       "curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null",
27
28       # Add the Jenkins repository to the package sources
29       "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",
30
31       # Update package list to include Jenkins and install Jenkins
32       "sudo apt update -y",
33       "sudo apt install -y jenkins",
34
35       # Start and enable Jenkins service
36       "sudo systemctl daemon-reload",
37       "sudo systemctl start jenkins",
38       "sudo systemctl enable jenkins"
39     ]
}

```

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
```

```
markmcaneda@ip-172-31-43-226:~/terraform-jenkins$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.75.0

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.
```

2. Format the configuration

```
terraform fmt
```

```
markmcaneda@ip-172-31-43-226:~/terraform-jenkins$ terraform fmt
main.tf
```

3. Validate the configuration

```
terraform validate
```

```
markmcaneda@ip-172-31-43-226:~/terraform-jenkins$ terraform validate
Success! The configuration is valid.
```

4. Apply the configuration

```
terraform apply
```

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

```
+ PRIVATE_AMI_NAME_OPTIONS (known after apply)
+ root_block_device (known after apply)
}

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
~ instance_ip = "3.80.33.36" -> (known after apply)

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

Enter a value:
```

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

```
aws_instance.jenkins_server (remote-exec): Executing: /usr/lib/systemd/systemd-s
ysv-install enable jenkins
aws_instance.jenkins_server: Creation complete after 2m45s [id=i-07cf1602c1bda78
b3]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

instance_ip = "3.80.66.57"
markmcaneda@ip-172-31-43-226:~/terraform-jenkins$
```

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.

```
markmcaneda@ip-172-31-43-226:~/terraform-jenkins$ ssh -i ./terraform.pem ubuntu@3.80.66.57
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

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

System information as of Tue Nov 12 05:11:55 UTC 2024

System load:  0.41           Processes:          107
Usage of /:   45.2% of 6.71GB  Users logged in:    0
Memory usage: 66%
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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

*** System restart required ***
Last login: Tue Nov 12 05:10:20 2024 from 3.85.89.169
ubuntu@ip-172-31-18-94:~$
```

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.

```
markmcaneda@ip-172-31-43-226:~/terraform-jenkins$ ansible-playbook -i 3.80.66.57  
, -u ubuntu --private-key terraform.pem jenkins-setup.yml  
  
PLAY [all] *****  
  
TASK [Gathering Facts] *****  
ok: [3.80.66.57]  
  
TASK [Install Java and Python] *****  
ok: [3.80.66.57]  
  
TASK [Add Jenkins repo and install Jenkins] *****  
changed: [3.80.66.57]  
  
TASK [Install Jenkins] *****  
ok: [3.80.66.57]  
  
TASK [Start Jenkins] *****  
ok: [3.80.66.57]
```

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

```
TASK [Install Jenkins] *****  
ok: [3.80.66.57]  
  
TASK [Display Jenkins initialAdminPassword] *****  
ok: [3.80.66.57]  
  
TASK [Display Jenkins initialAdminPass] (80x33) *****  
ok: [3.80.66.57]  
  
TASK [Echo Jenkins initialAdminPassword] *****  
ok: [3.80.66.57] => {  
    "msg": "Jenkins initialAdminPassword: d5f93631a875402c895d39f0d9774393"  
}  
  
TASK [Check Java version] *****  
ok: [3.80.66.57]  
  
TASK [Display Java version] *****  
ok: [3.80.66.57] => {  
    "msg": "Java Version: openjdk version \"17.0.13\" 2024-10-15"  
}
```



- Display the Python version and the Jenkins remote URL

```

TASK [Check Python version] ****
ok: [3.80.66.57]

TASK [Display Python version] ****
ok: [3.80.66.57] => {
    "msg": "Python Version: Python 3.12.3"
}

TASK [Check Jenkins service status] ****
ok: [3.80.66.57]

TASK [Display Jenkins service status] ****
ok: [3.80.66.57] => {
    "msg": "Jenkins Service Status: active"
}

TASK [Display Jenkins access URL] ****
ok: [3.80.66.57] => {
    "msg": "Access Jenkins at: http://3.80.66.57:8080"
}

PLAY RECAP ****
3.80.66.57 : ok=14    changed=1      unreachable=0    failed=0    s
skipped=0   rescued=0   ignored=0

markmcaneda@ip-172-31-43-226:~/terraform-jenkins$ █

```

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

```

TASK [Display Jenkins access URL] ****
ok: [3.80.66.57] => {
    "msg": "Access Jenkins at: http://3.80.66.57:8080"
}

PLAY RECAP ****
3.80.66.57 : ok=14    changed=1      unreachable=0    failed=0    s
skipped=0   rescued=0   ignored=0

markmcaneda@ip-172-31-43-226:~/terraform-jenkins$ █

```

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



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

```

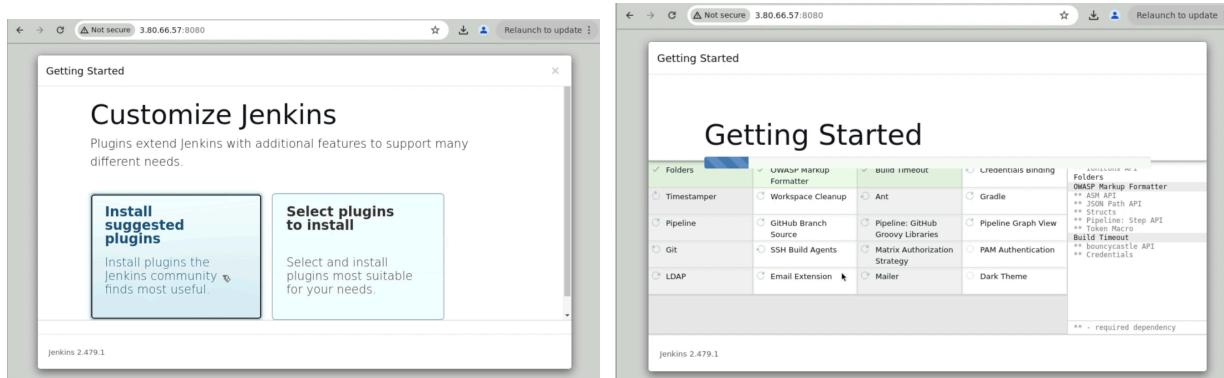
TASK [Echo Jenkins initialAdminPassword] *****
ok: [3.80.66.57] => {
    "msg": "Jenkins initialAdminPassword: d5f93631a875402c895d39f0d9774393"
}

TASK [Check Java version] *****
ok: [3.80.66.57]

```

A red arrow points to the line containing the Jenkins initialAdminPassword value: "Jenkins initialAdminPassword: d5f93631a875402c895d39f0d9774393".

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

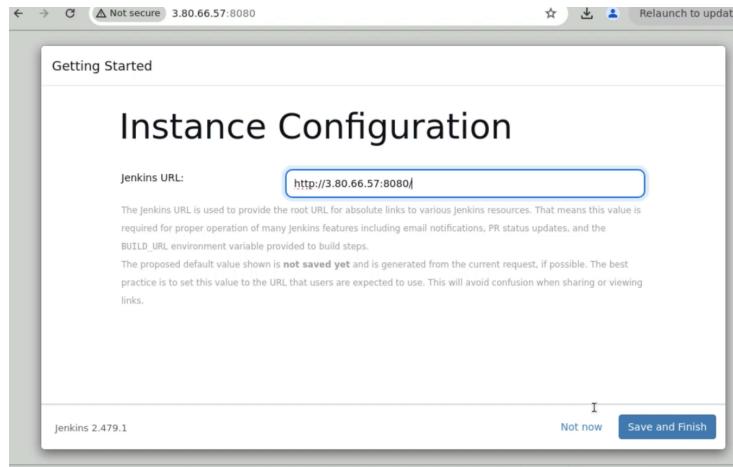


- You may skip the admin creation or fill out the fields. Save and Continue.

Username	<input type="text"/>
Password	<input type="password"/>
Confirm password	<input type="password"/>
<input type="button" value="Skip and continue as admin"/> <input type="button" value="Save and Continue"/>	

Jenkins 2.479.1

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



The screenshot shows the Jenkins 'Dashboard' after configuration. The left sidebar displays a 'Getting Started' message stating 'Jenkins is ready!' and noting that the setup of an admin user was skipped. The main dashboard features a 'Welcome to Jenkins!' message, a 'Build History' section, and a 'Start building your software project' section with options like 'Create a job', 'Set up a distributed build', 'Set up an agent', and 'Configure a cloud'.

- 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 -