# Day 60 - Terraform

Hello Learners, you guys are doing every task by creating an ec2 instance (mostly). Today let's automate this process. How to do it? Well Terraform is the solution.

What is Terraform?
Terraform is an infrastructure as code (IaC) tool that allows you to create, manage, and update infrastructure resources such as virtual machines, networks, and storage in a repeatable, scalable, and automated way.

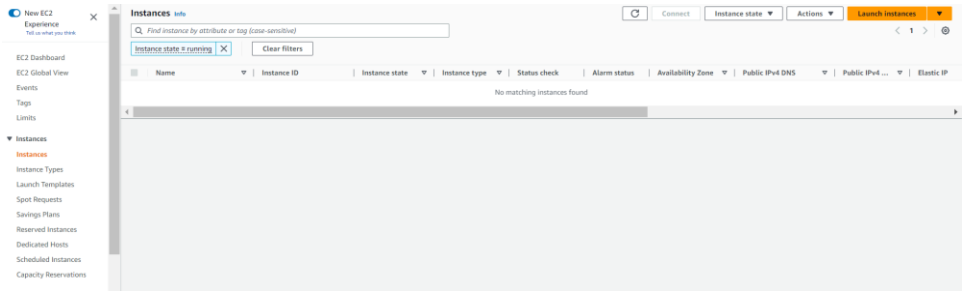
Task 1:
Install Terraform on your system Refer this below link for installation

Install Terraform on Linux

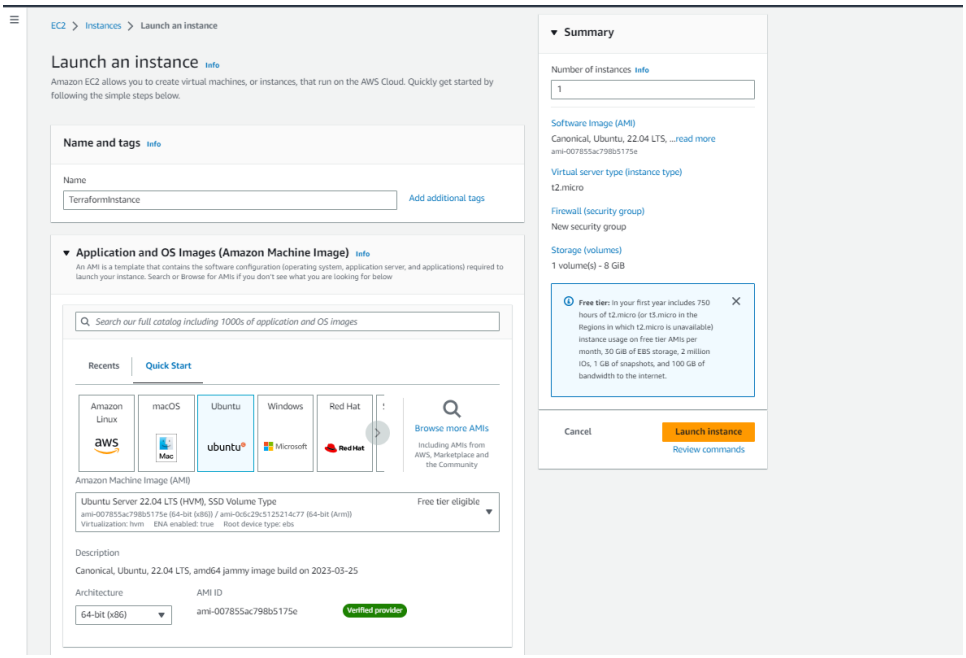
Terraform is an open source, so you can install terraform in any OS.

For this Demo, we are using Linux ubuntu.

First Create a Simple AWS EC2 Instance.

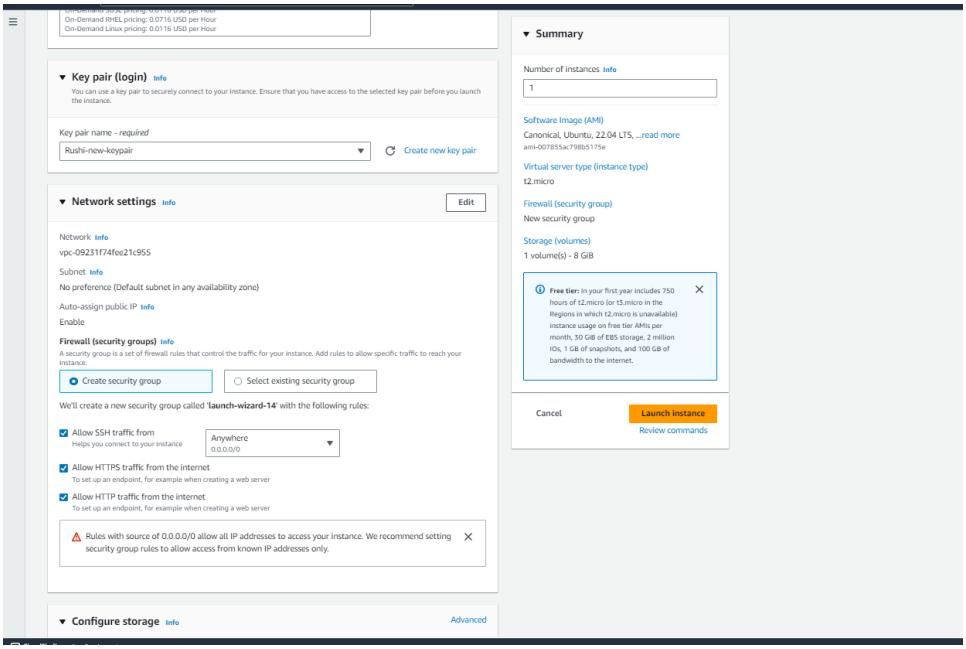



Select Ubuntu as an AMI (Amazon Machine Image)

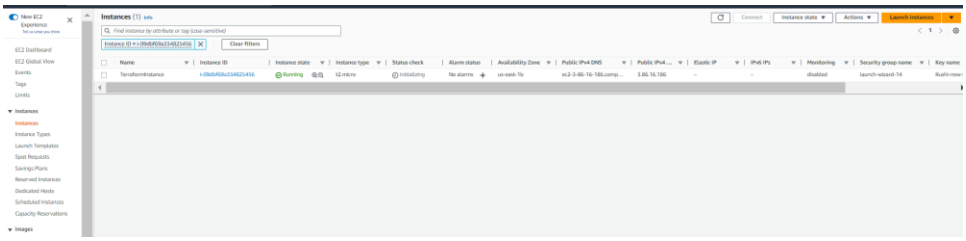



Create a new Key-Value pair.

After that, click on Launch Instance.

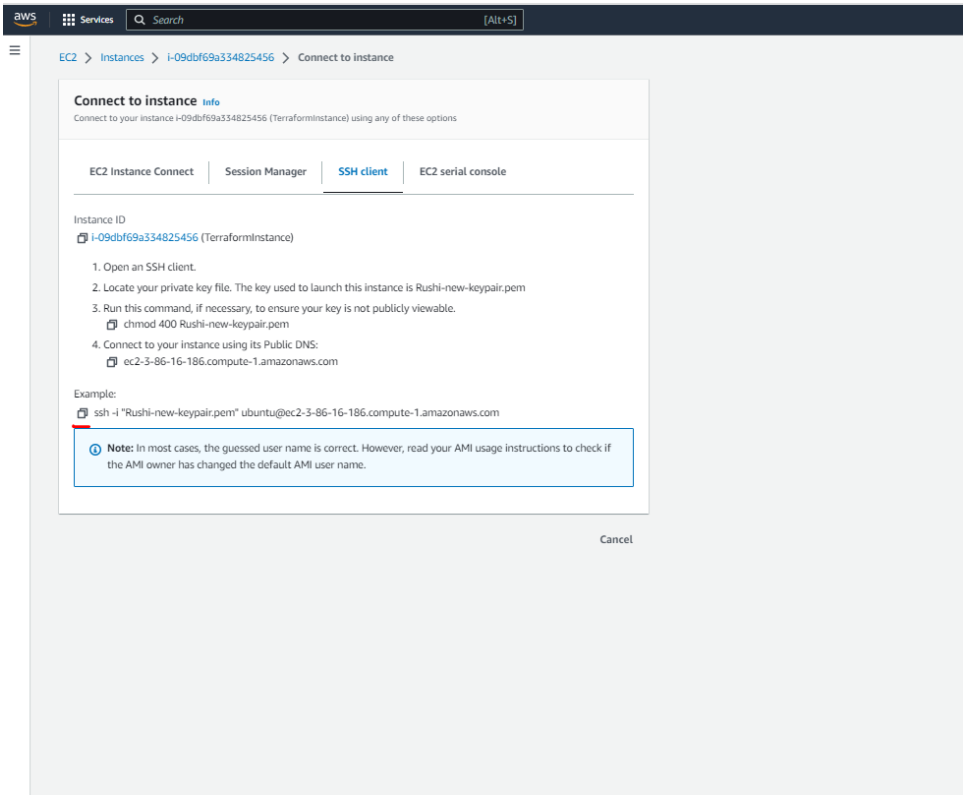



Your Instance are Running now



Do a SSH Connection, for securely communications between a local machine and a remote host

**Copy this ssh URL and paste on local.**




Open Command line and paste the link.

Make sure your downloaded key-value pair is also in the same location.

```
C:\Users\Rushikesh\Desktop>ssh -i "Rushi-new-keypair.pem" ubuntu@ec2-3-86-16-186.compute-1.amazonaws.com
The authenticity of host 'ec2-3-86-16-186.compute-1.amazonaws.com (64:ff9b::356:10ba)' can't be established.
ECDSA key fingerprint is SHA256:gQCmF9j8b2cDPkCHQKCOrfRwdCizx/dcdIhNnknANOE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-3-86-16-186.compute-1.amazonaws.com,64:ff9b::356:10ba' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-1031-aws x86_64)

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

  System information as of Thu Apr 13 06:00:10 UTC 2023

  System load:  0.26806640625     Processes:             110
  Usage of /:   20.2% of 7.57GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for eth0: 172.31.85.217
  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


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-85-217:~$
```

Go to /home/ubuntu path and create a new folder name as Terraform.

```
ubuntu@ip-172-31-85-217:~$ cd /
ubuntu@ip-172-31-85-217:/$ pwd
/
ubuntu@ip-172-31-85-217:/$ ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  lost+found  media  mnt  opt  proc  root  run  sbin  snap  srv  sys  tmp  usr  var
ubuntu@ip-172-31-85-217:/$ cd /home/ubuntu
ubuntu@ip-172-31-85-217:~$ ls
ubuntu@ip-172-31-85-217:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-85-217:~$ mkdir Terraform
ubuntu@ip-172-31-85-217:~$ c Terraform/
c: command not found
ubuntu@ip-172-31-85-217:~$ cd Terraform/
```

Use the commands below to install Terraform on Linux.

wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

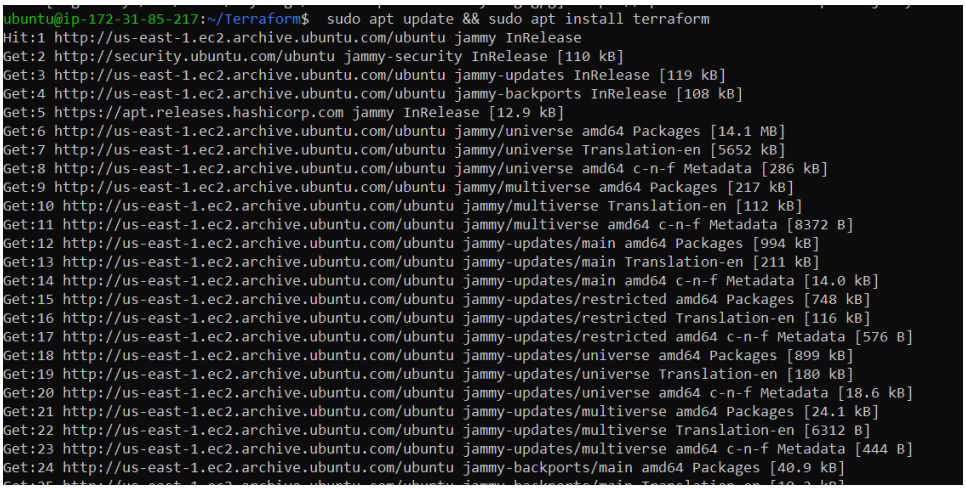
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update && sudo apt install terraform

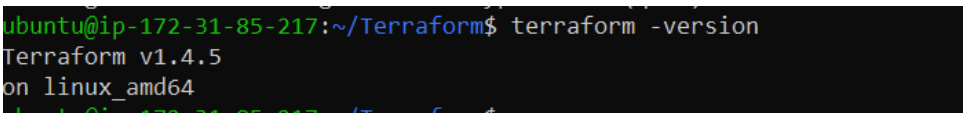
terraform -version

```
ubuntu@ip-172-31-85-217:~/Terraform$ wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
--2023-04-13 06:01:59--  https://apt.releases.hashicorp.com/gpg
Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 18.160.46.37, 18.160.46.62, 18.160.46.7, ...
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|18.160.46.37|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3980 (3.9K) [binary/octet-stream]
Saving to: 'STDOUT'

-                                   100%[===========================================================================================>]   3.89K  --.-KB/s    in 0s

2023-04-13 06:01:59 (208 MB/s) - written to stdout [3980/3980]
```

```
ubuntu@ip-172-31-85-217:~/Terraform$ 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
deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com jammy main
```




See, terraform installed with latest version.




**Task 2: Answer below questions**

- **Why we use terraform?**

  Terraform is a tool for building, changing, and versioning infrastructure in a safe, repeatable way. It enables teams to manage infrastructure as code, providing a single source of truth for the infrastructure and ensuring that it is always in the desired state. Terraform can be used to manage infrastructure across multiple cloud providers and on-premises infrastructure, and it supports a wide range of resource types.

- **What is Infrastructure as Code (IaC)?**

  Infrastructure as Code (IaC) is the managing and provisioning of infrastructure through code instead of through manual processes.

  With IaC, configuration files are created that contain your infrastructure specifications, which makes it easier to edit and distribute configurations. It also ensures that you provision the same environment every time. By codifying and documenting your configuration specifications, IaC aids configuration management and helps you to avoid undocumented, ad-hoc configuration changes.

  Deploying your infrastructure as code also means that you can divide your infrastructure into modular components that can then be combined in different ways through automation.

- **What is Resource?**

Resources are the most important element in the Terraform language. Each resource block describes one or more infrastructure objects, such as virtual networks, computer instances, or higher-level components such as DNS records.

resource syntax:

```
resource "resource_type" "resource_name"
{
config1 = value1
config2 = value2
 }
```

Here, resource_type = The type of the resource that we create/delete/modify resource_name = Give the resource a name for Terraform internal references. config(1-n) = The properties of the resource your manipulating.

In your main.tf, add this block next to your Provider definition.

an ec2 resource

```
 resource "aws_instance" "frontend" {
  ami= "ami-0ac019f4fcb7cb7e6"
  instance_type = "t2.micro"
}
```

- **What is Provider?**

Terraform relies on plugins called providers to interact with cloud providers, SaaS providers, and other APIs.

A provider is responsible for understanding API interactions and exposing resources. In Order to make a provider available on Terraform, we need to make a **terraform init.**

This command downloads any plugins we need for our providers.

- **What is State file in terraform? What's the importance of it?**

Terraform state is like a blueprint of the Real-world infrastructure with some unique ids and attributes.

This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures.

This state is stored by default in a local file named **"terraform.tfstate"**, but we recommend storing it in Terraform Cloud to version, encrypt, and securely share it with your team.

- **What is Desired and Current State?**

  In Terraform, the desired state is the state that you want your infrastructure to be in, as defined in your Terraform configuration files. The current state is the actual state of the infrastructure, as represented in the Terraform state file. When you run terraform apply, terraform compares the desired state with the current state and makes changes as needed to bring the infrastructure into the desired state.

Happy Learning:)