

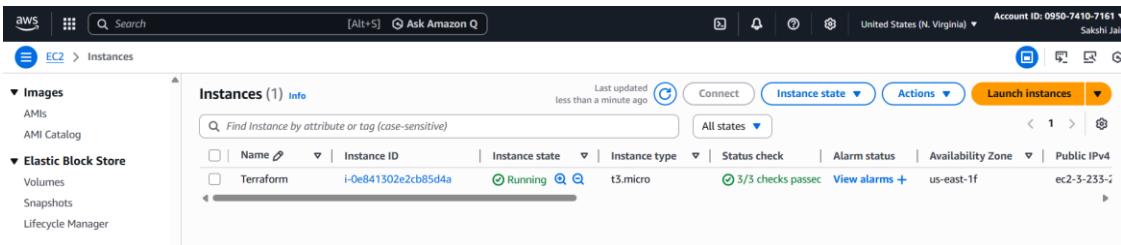
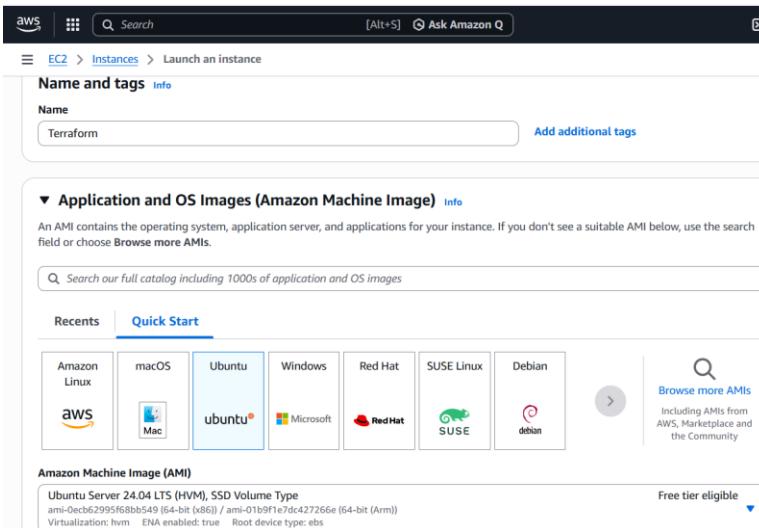
Terraform

○ Terraform IAC, Information & Installation on AWS

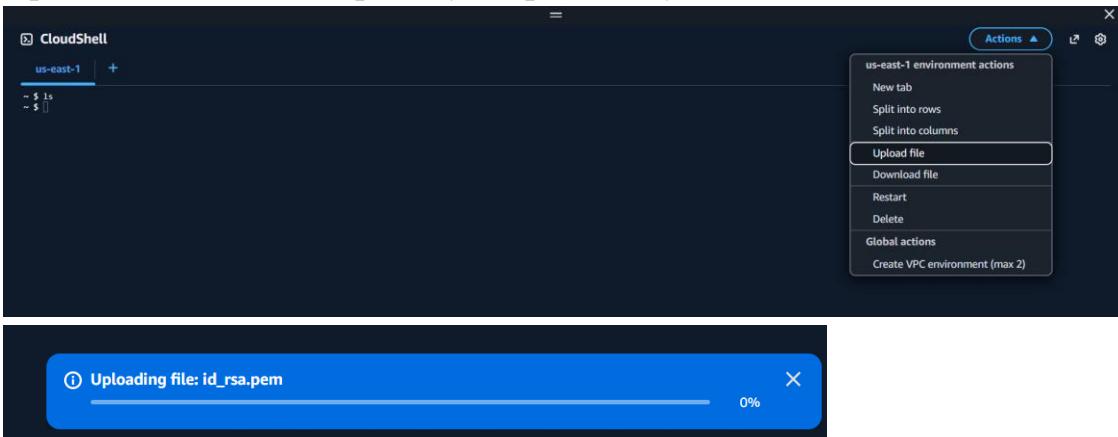
Terraform is an Infrastructure as Code (IaC) tool created by HashiCorp.

It allows you to **create, update, and delete cloud infrastructure using code**, instead of doing everything manually in the cloud console.

1. Create EC2 Instance



2. Open Cloud shell and upload your public key



Copy Public IP of your Instance to get access on cloud shell

The screenshot shows the AWS EC2 Instances page. A single instance named "Terraform" is listed, which is a t3.micro instance with a public IPv4 address of 44.197.230.41. The "Public IPv4" address is highlighted with a tooltip: "Public IPv4 address copied".

Take access of your ec2

```
ubuntu@44.197.230.41: Permission denied (publickey).
~ $ chmod 400 id_rsa.pem
~ $ ssh -i id_rsa.pem ubuntu@44.197.230.41
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

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

System information as of Tue Jan 13 06:25:33 UTC 2026

System load: 0.08      Temperature:      -273.1 °C
Usage of /: 25.8% of 6.71GB Processes:          113
Memory usage: 23%      Users logged in:    0
Swap usage:  0%      IPv4 address for ens5: 172.30.5.91

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-172-30-5-91:~$
```

3. Now Got to official site of terraform and copy Linux/ubuntu command to install Terraform

The screenshot shows the HashiCorp Terraform website. On the left, there's a sidebar with 'Operating Systems' including macOS, Windows, Linux (selected), FreeBSD, OpenBSD, and Solaris. The main content area has sections for '386' and 'AMD64' with 'Download' buttons. Below this is a 'Linux' section for 'Package manager' under 'Ubuntu/Debian'. It contains a command-line interface showing the execution of:

```
wget -O - https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg  
echo "deb [arch=$(dpkg --print-architecture)] signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg" | sudo tee /etc/apt/sources.list.d/hashicorp.list  
sudo apt update & sudo apt install terraform
```

Below this is a CloudShell terminal window titled 'us-east-1' showing the execution of the same apt-get command. The terminal output includes:

```
Get:52 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [527 kB]  
Get:53 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [208 B]  
Get:54 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [27.0 kB]  
Get:55 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [608 B]  
Get:56 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]  
Get:57 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [304 B]  
Fetched 39.5 MB in 7s (5951 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
32 packages upgraded, Run 'apt list --upgradable' to see them.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following NEW packages will be installed:  
  terraform  
0 upgraded, 1 newly installed, 0 to remove and 82 not upgraded.  
Need to get 30.6 MB of archives.  
After this operation, 101 MB of additional disk space will be used.  
Get:1 https://apt.releases.hashicorp.com/noble/main amd64 terraform amd64 1.14.3-1 [30.6 MB]  
Fetched 30.6 MB in 0s (124 MB/s)  
Selecting previously unselected package terraform.  
(Reading database ... 71735 files and directories currently installed.)  
Preparing to unpack .../terraform_1.14.3-1_amd64.deb ...  
Unpacking terraform (1.14.3-1) ...  
Setting up terraform (1.14.3-1) ...  
Scanning processes...  
Scanning linux images...  
Running kernel seems to be up-to-date.  
No services need to be restarted.  
No containers need to be restarted.  
No user sessions are running outdated binaries.  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
root@ip-172-30-5-91:~#
```

○ write configuration file, terraform life-cycle.

Required Visual Studio

Use this Registry for reference

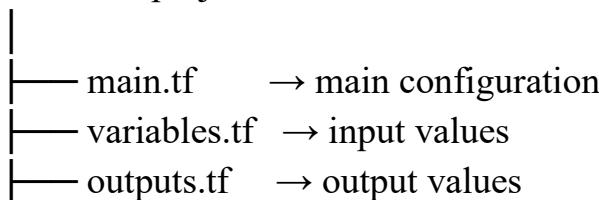
[Terraform AWS Registry](#)

Install Terraform Extension in VS code

The screenshot shows the Microsoft Visual Studio Code Marketplace. In the center, the 'HashiCorp Terraform' extension is displayed with a star rating of 4.8 and 5,875 reviews. It has an 'Auto Update' option checked. Below it, the 'Terraform Extension for Visual Studio Code' is described as adding editing features for Terraform, Terraform Stacks and Terraform Search files. To the right, a 'Marketplace' sidebar shows details for the HashiCorp Terraform extension, including its identifier, version (2.37.6), publication date (9 years ago), and last release (1 month ago). There are also 'Categories' and 'Quick Start' links.

Basic Terraform File Structure

terraformer-project/



main.tf (Core File)

This file contains:

1. Provider
2. Resources

Example: main.tf

```
provider "aws" {  
    region = "ap-south-1"  
}
```

👉 Meaning:

- Use AWS
- Region: Mumbai

Add a Resource (EC2 Example)

The screenshot shows the AWS CloudFront interface for launching a new EC2 instance. The top navigation bar includes links for Home, Services, and Support, along with a search bar and a user icon.

The main content area is titled "Launch an Instance". It displays a grid of available AMIs:

- Amazon Linux
- macOS
- Ubuntu
- Windows
- Red Hat
- SUSE Linux
- Debian
- ubuntu** (selected)
- Microsoft
- Red Hat
- SUSE
- debian

Below the AMI grid, there is a section for "Amazon Machine Image (AMI)":

- Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
- ami-0ecb62995f68bb549 (64-bit (x86)) / ami-01b9f1e7d427266e (64-bit (Arm))
- Virtualization: hvm
- ENI enabled: true
- Root device type: ebs
- Free tier eligible

Below this, there are fields for "Description", "Architecture", "AMI ID", "Publish Date", "Username", and a "Verified provider" badge.

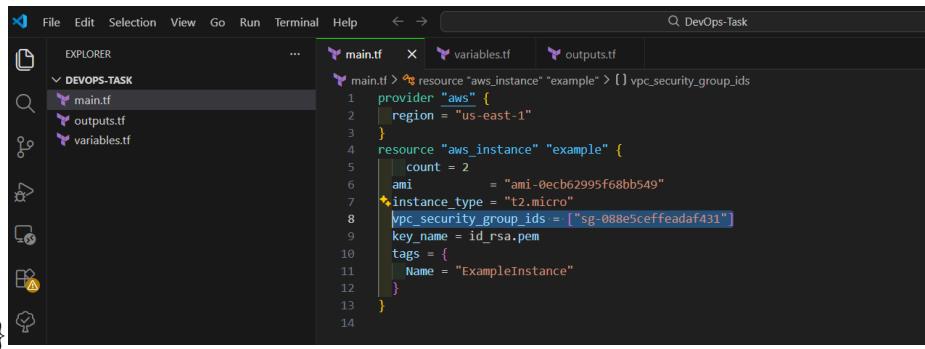
The bottom section is titled "Network settings" and includes tabs for "Network" (selected), "Info", "Subnet", and "Auto-assign public IP". It also features a "Firewall (security groups)" section where the "Select existing security group" option is selected, and a "Common security groups" dropdown containing "mysg sg-08e5ceffeadaf431".

you have to use this ID's in code for Automatically creation

```
resource "aws_instance" "ExampleInstance" {
    ami          = "ami-0ecb62995f68bb549"
    instance_type = "t3.micro"

    subnet_id      = "subnet-0ff35381a830914ad"
    vpc_security_group_ids = ["sg-088e5ceffeadaf431"]
```

```
tags = {
    Name = "ExampleInstance"
}
```



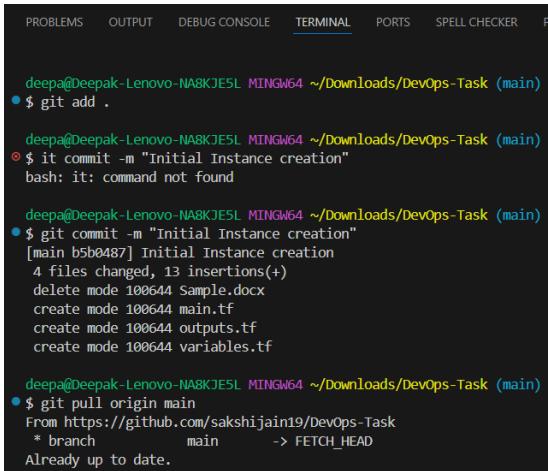
```
provider "aws" {
    region = "us-east-1"
}

resource "aws_instance" "example" {
    count = 2
    ami          = "ami-0ecb62995f68bb549"
    instance_type = "t2.micro"
    vpc_security_group_ids = ["sg-088e5ceffeadaf431"]
    key_name = id_rsa.pem
    tags = [
        { Name = "ExampleInstance" }
    ]
}
```

👉 Meaning:

- resource → what to create
- aws_instance → EC2
- my_ec2 → logical name (Terraform uses it)
- ami, instance_type → EC2 settings

Push this files to your Main GitHub Repository



```
deepa@Deepak-Lenovo-NA8KJESL MINGW64 ~/Downloads/DevOps-Task (main)
$ git add .

deepa@Deepak-Lenovo-NA8KJESL MINGW64 ~/Downloads/DevOps-Task (main)
$ git commit -m "Initial Instance creation"
bash: it: command not found

deepa@Deepak-Lenovo-NA8KJESL MINGW64 ~/Downloads/DevOps-Task (main)
$ git commit -m "Initial Instance creation"
[main b5b487] Initial Instance creation
 4 files changed, 13 insertions(+)
 delete mode 100644 Sample.docx
 create mode 100644 main.tf
 create mode 100644 outputs.tf
 create mode 100644 variables.tf

deepa@Deepak-Lenovo-NA8KJESL MINGW64 ~/Downloads/DevOps-Task (main)
$ git pull origin main
From https://github.com/sakshijain19/DevOps-Task
 * branch            main       -> FETCH_HEAD
Already up to date.
```

```
deepa@Deepak-Lenovo-NA8KJE5L MINGW64 ~/Downloads/DevOps-Task (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 529 bytes | 264.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sakshijain19/DevOps-Task.git
  9edf679..b5b0487  main -> main
```

You Need to communicate to AWS for that required one Instance which help you to communicate AWS for that I will create one IAM Role and attach it to this Instance

The screenshot shows three panels of the AWS Management Console:

- Instances (1/1) Info:** Shows a single EC2 instance named "Terraform" with ID i-080b359695161deac, which is currently stopped.
- Roles (1/15) Info:** Shows a single IAM role named "TerraformRole" attached to the "ec2" service.
- Instances (1/1) Info (Details View):** Shows the same instance details. A context menu is open over the instance row, with the "Modify IAM role" option highlighted.
- Modify IAM role Info:** A modal dialog box where the "TerraformRole" is selected from a dropdown menu under the "IAM role" section. There are "Create new IAM role" and "Update IAM role" buttons at the bottom.

Copy Public IP of Instance and through SSH take remote access of Instance

aws | ⚡ | Search

CloudShell

us-east-1 +

```

~ $ ls
id_rsa.pem
~ $ ssh -i id_rsa.pem ubuntu@100.27.234.69
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1018-aws x86_64)

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

System information as of Thu Jan 15 05:26:47 UTC 2026

System load: 0.0 Temperature: -273.1 C
Usage of /: 36.6% of 6.71GB Processes: 117
Memory usage: 22% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 172.30.5.91

* Ubuntu Pro delivers the most comprehensive open source security and
compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

43 updates can be applied immediately.
1 of these updates is a standard security update.
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: Wed Jan 14 13:38:44 2026 from 3.84.9.207
ubuntu@ip-172-30-5-91:~$ █

```

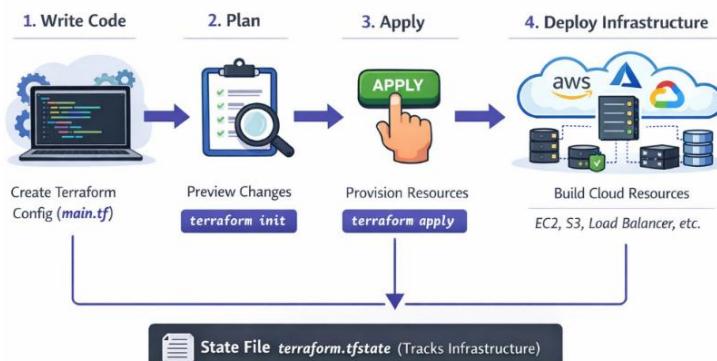
Clone your Repository in your Instance

```

ubuntu@ip-172-30-5-91:~$ sudo -i
root@ip-172-30-5-91:~# git clone https://github.com/sakshijain19/DevOps-Task.git
Cloning into 'DevOps-Task'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 0), reused 4 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (7/7), 11.83 KiB | 3.94 MiB/s, done.
root@ip-172-30-5-91:~# ls
DevOps-Task snap
root@ip-172-30-5-91:~# cd DevOps-Task/
root@ip-172-30-5-91:~/DevOps-Task# ls
main.tf outputs.tf variables.tf
root@ip-172-30-5-91:~/DevOps-Task# █

```

Terraform Workflow/ Lifecycle



What is terraform init? (Very Simple)

terraform init is the **first command** you run in any Terraform project.

Why Do We Need terraform init?

Terraform cannot work until it knows:

- Which **cloud provider** you are using
- Where to store the **state file**
- Which **plugins** are required

terrafrom init does all this setup automatically

```
root@ip-172-30-5-91:~/DevOps-Task# terraform --version
Terraform v1.14.3
on linux_amd64
root@ip-172-30-5-91:~/DevOps-Task# terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.28.0...
- Installed hashicorp/aws v6.28.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
root@ip-172-30-5-91:~/DevOps-Task# ls -a
. .git .terraform .terraform.lock.hcl main.tf outputs.tf variables.tf
root@ip-172-30-5-91:~/DevOps-Task#
```

You will get all details about provider

```
root@ip-172-30-5-91:~/DevOps-Task# ls -a
. .git .terraform .terraform.lock.hcl main.tf outputs.tf variables.tf
root@ip-172-30-5-91:~/DevOps-Task# cd .terraform/
root@ip-172-30-5-91:~/DevOps-Task/.terraform# ls
providers
root@ip-172-30-5-91:~/DevOps-Task/.terraform# cd providers/
root@ip-172-30-5-91:~/DevOps-Task/.terraform/providers# ls
registry.terraform.io
root@ip-172-30-5-91:~/DevOps-Task/.terraform/providers# cd registry.terraform.io/
root@ip-172-30-5-91:~/DevOps-Task/.terraform/providers/registry.terraform.io# ls
hashicorp
root@ip-172-30-5-91:~/DevOps-Task/.terraform/providers/registry.terraform.io# cd hashicorp/
root@ip-172-30-5-91:~/DevOps-Task/.terraform/providers/registry.terraform.io/hashicorp# ls
aws
root@ip-172-30-5-91:~/DevOps-Task/.terraform/providers/registry.terraform.io/hashicorp# cd aws/
root@ip-172-30-5-91:~/DevOps-Task/.terraform/providers/registry.terraform.io/hashicorp/aws# ls
6.28.0
root@ip-172-30-5-91:~/DevOps-Task/.terraform/providers/registry.terraform.io/hashicorp/aws#
```

What is .terraform.lock.hcl?

.terraform.lock.hcl is a **lock file** that tells Terraform **which exact provider versions to use**.

Why is .terraform.lock.hcl Important?

Without this file:

- Different team members may use **different provider versions**

- Terraform behavior may change
- Builds may fail unexpectedly

What is terraform plan?

terraform plan shows you **what Terraform is going to do** before it actually makes any changes.

Why Do We Use terraform plan?

It helps you:

- Avoid mistakes
- Understand changes clearly
- Review before applying
- Detect configuration drift

```
us-east-1 | +
root@ip-172-30-5-91:~/DevOps-Task# terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.example[0] will be created
+ aws_instance "aws_instance" "example" [
  + ami                               = "ami-0ecb62995f68bb549"
  + arn                             = (known after apply)
  + associate_public_ip_address      = (known after apply)
  + availability_zone                = (known after apply)
  + display_name                     = (known after apply)
  + disable_api_termination          = (known after apply)
  + ebs_optimized                    = (known after apply)
  + enable_primary_ipv6              = (known after apply)
  + force_destroy                   = false
  + get_password_data               = (known after apply)
  + host_id                         = (known after apply)
  + iam_instance_profile             = (known after apply)
  + id                               = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle               = (known after apply)
  + instance_state                  = (known after apply)
  + instance_type                   = t2.micro
  + ipv6_address_count              = (known after apply)
  + ipv6_addresses                  = (known after apply)
  + key_name                        = (known after apply)
  + monitoring                      = (known after apply)
  + outpost_arn                     = (known after apply)
  + password_data                  = (known after apply)
  + placement_group                 = (known after apply)
  + placement_group_id              = (known after apply)
]
```

Here your IAM Role is work as credential

```
vpc_security_group_ids           = [
  "sg-08e5cfeffeadaf431",
]
+ capacity_reservation_specification (known after apply)
+ cpu_options (known after apply)
+ ebs_block_device (known after apply)
+ enclave_options (known after apply)
+ ephemeral_block_device (known after apply)
+ instance_market_options (known after apply)
+ maintenance_options (known after apply)
+ metadata_options (known after apply)
+ network_interface (known after apply)
+ primary_network_interface (known after apply)
+ private_dns_name_options (known after apply)
+ root_block_device (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@ip-172-30-5-91:~/DevOps-Task#
```

terraform apply —

terraform apply is the command that **actually creates or updates your infrastructure.**

1. Reads your .tf files
2. Checks the **state** file
3. Compares with **real cloud resources**
4. Shows a plan (what will change)
5. Asks for confirmation (yes)
6. **Creates / updates / deletes** resources

👉 This is when **real AWS resources are created** (EC2, S3, etc.).

```
Note: You must use the -out option to save this plan, so Terraform can guarantee to take exactly these actions if you run `terraform apply'.
ubuntu@ip-172-30-5-91:~/DevOps-Task$ terraform apply
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
Terraform will perform the following actions:

# aws_instance.ExampleInstance will be created
resource "aws_instance" "ExampleInstance" {
  ami                               = "ami-0ccb62995f68bb549"
  ami_id                           = (known after apply)
  associate_public_ip_address      = (known after apply)
  availability_zone                = (known after apply)
  disable_api_stop                 = (known after apply)
  dynamic_ip_termination          = (known after apply)
  ebs_optimized                    = (known after apply)
  enable_primary_ipv6              = (known after apply)
  force_destroy                    = false
  get_password_data               = false
  host                                = (known after apply)
  host_resource_group_arn           = (known after apply)
  iam_instance_profile              = (known after apply)
  id                                 = (known after apply)
  instance_initiated_shutdown_behavior = (known after apply)
  instance.lifecycle                = (known after apply)
  instance.state                   = (known after apply)
  instance_type                     = "t3.micro"
  ipv6_address_count               = (known after apply)
  ipv6_addresses                   = (known after apply)
  key_name                          = (known after apply)
  monitoring                        = (known after apply)
  outpost_arn                       = (known after apply)
  password_data                     = (known after apply)
}

+ ebs_block_device (known after apply)
+ enclave_options (known after apply)
+ ephemeral_block_device (known after apply)
+ instance_market_options (known after apply)
+ maintenance_options (known after apply)
+ metadata_options (known after apply)
+ network_interface (known after apply)
+ primary_network_interface (known after apply)
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}

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

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

Enter a value: yes

aws_instance.ExampleInstance: Creating...
aws_instance.ExampleInstance: Still creating... [00m10s elapsed]
aws_instance.ExampleInstance: Creation complete after 12s [id=i-094cb88d75600ff52c]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
ubuntu@ip-172-30-5-91:~/DevOps-Task$
```

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with 'EC2' selected. The main area has a title 'Instances (2) Info'. It shows two instances: 'ExampleInstance' (ID: i-094cb88d75600f52c) and 'Terraform' (ID: i-0806359695161dec). Both instances are listed as 'Running' with 't3.micro' instance type. There are filters for 'Name', 'Instance ID', 'Instance state', 'Instance type', 'Status check', 'Alarm status', and 'Availability Zone'. A 'Launch instances' button is at the top right. Below the table, there are buttons for 'View alarms' and 'View logs'.

What is terraform.tfstate? (Simple Explanation)

terraform.tfstate is a **state file** that Terraform uses to **remember what infrastructure it has created**.

```
ubuntu@ip-172-30-5-91:~/DevOps-Task$ ls -a
.  ..  .git  .terraform  .terraform.lock.hcl  main.tf  outputs.tf  terraform.tfstate  terraform.tfstate.backup  variables.tf
```

Why Terraform Needs terraform.tfstate

Terraform must know:

- What resources already exist
- Their IDs (EC2 ID, S3 bucket name, IPs)
- Current configuration

terraform destroy —

terraform destroy is the command used to **delete everything created by Terraform**.

1. Reads the **state file**
2. Finds all resources it created
3. Shows what will be deleted
4. Asks for confirmation (yes)
5. **Deletes all resources**

👉 Used to avoid **unnecessary cloud billing**

```

ubuntu@ip-172-30-5-91:~/DevOps-Task$ terraform destroy
aws_instance.ExampleInstance: Refreshing state... [id=i-094cb88d75600f52c]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_instance.ExampleInstance will be destroyed
- resource "aws_instance" "ExampleInstance" {
    - ami                                = "ami-0ecb62995f68bb549" -> null
    - arn                                = "arn:aws:ec2:us-east-1:095074107161:instance/i-094cb88d75600f52c" -> null
    - associate_public_ip_address          = true -> null
    - availability_zone                   = "us-east-1F" -> null
    - disable_api_stop                    = false -> null
    - disable_api_termination             = false -> null
    - ebs_optimized                      = false -> null
    - force_destroy                       = false -> null
    - get_password_data                  = false -> null
    - hibernation                         = false -> null
    - id                                 = "i-094cb88d75600f52c" -> null
    - instance_initiated_shutdown_behavior = "stop" -> null
    - instance_state                      = "running" -> null
    - instance_type                       = "t3.micro" -> null
    - ipv6_address_count                 = 0 -> null
    - ipv6_addresses                     = [] -> null
    - monitoring                          = false -> null
    - placement_partition_number          = 0 -> null
    - primary_network_interface_id       = "eni-0b6cc9167a8a19614" -> null
    - private_dns                         = "ip-172-30-5-44.ec2.internal" -> null
    - private_ip                           = "172.30.5.44" -> null
    - public_dns                          = "ec2-13-220-190-124.compute-1.amazonaws.com" -> null
    - public_ip                            = "13.220.190.124" -> null
}

```

```

us-east-1 | +
          - hostname_type           = "ip-name" -> null
        }

      - root_block_device {
        - delete_on_termination = true -> null
        - device_name          = "/dev/sda1" -> null
        - encrypted             = false -> null
        - iops                  = 3000 -> null
        - tags                  = {} -> null
        - tags_all              = {} -> null
        - throughput             = 125 -> null
        - volume_id              = "vol-06354e00b5497c45a" -> null
        - volume_size             = 8 -> null
        - volume_type             = "gp3" -> null
        # (1 unchanged attribute hidden)
      }
    }

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

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_instance.ExampleInstance: Destroying... [id=i-094cb88d75600f52c]
aws_instance.ExampleInstance: Still destroying... [id=i-094cb88d75600f52c, 00m10s elapsed]
aws_instance.ExampleInstance: Still destroying... [id=i-094cb88d75600f52c, 00m20s elapsed]
aws_instance.ExampleInstance: Still destroying... [id=i-094cb88d75600f52c, 00m30s elapsed]
aws_instance.ExampleInstance: Still destroying... [id=i-094cb88d75600f52c, 00m40s elapsed]
aws_instance.ExampleInstance: Destruction complete after 50s

Destroy complete! Resources: 1 destroyed.

```

The screenshot shows the AWS EC2 Instances page. The left sidebar has 'EC2' selected under 'Instances'. The main area displays a table titled 'Instances (2) Info' with the following data:

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	ExampleInstance	i-094cb88d75600f52c	Terminated	t3.micro	-	View alarms +	us-east-1F
<input type="checkbox"/>	Terraform	i-080b359695161deac	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1F

○ Terraform block, set backend, variables and output block

Create S3 Bucket

The screenshot shows the AWS S3 'Create bucket' wizard. In the 'General configuration' section, the AWS Region is set to 'US East (N. Virginia) us-east-1'. The 'Bucket type' is set to 'General purpose' (selected). The 'Bucket name' is 'sample001-test'. In the 'Object Ownership' section, 'ACLs disabled (recommended)' is selected. Under 'Bucket Versioning', 'Enable' is selected. The final step shows a green success message: 'Successfully created bucket "sample001-test"'. The bucket is listed in the 'General purpose buckets' table.

Name	AWS Region	Creation date
sample001-test	US East (N. Virginia) us-east-1	January 16, 2026, 22:15:17 (UTC+05:30)

Edit role of your terraform Instance and add s3 permission to it and reattach to Instance

The screenshot shows the AWS IAM Roles page. A role named "TerraformRole" is selected. The "Permissions" tab is active, displaying one managed policy attached: "AmazonEC2FullAccess". The ARN of the role is listed as "arn:aws:iam::095074107161:role/TerraformRole".

The screenshot shows the "Add permissions" step for the TerraformRole. It lists the "AmazonS3FullAccess" policy from the "Other permissions policies" section. The "Description" for this policy is "Provides full access to all buckets via t...".

The screenshot shows the AWS EC2 Instances page. An instance named "Terraform" (ID: i-080b359695161deac) is running. A context menu is open over the instance, with the "Modify IAM role" option highlighted.

The screenshot shows the "Modify IAM role" dialog for the Terraform instance. The "Instance ID" is listed as "+044fb3acd23bde9f3 (Terraform)". The "IAM role" dropdown is set to "TerraformRole". There is a "Create new IAM role" button and a "Cancel" button.

This Backend blog will store tfstate file in the s3

```

EXPLORER                         ...   main.tf    variables.tf    outputs.tf
DEVOPS-TASK
  main.tf
  outputs.tf
  variables.tf

main.tf > ...
1
2  terraform {
3    backend "s3" {
4      bucket = "sample001-test"
5      key    = "terraform.tfstate"
6      region = "us-east-1"
7    }
8  }
9  provider "aws" {
10   region = "us-east-1"
11 }
12
13 resource "aws_instance" "ExampleInstance" {
14   ami           = var.ami_id
15   instance_type = var.instance_type
16
17   subnet_id     = "subnet-0ff35381a830914ad"
18   vpc_security_group_ids = ["sg-0abc1234def567890"] # <-- FIX HERE
19
20   tags = {
21     Name = "ExampleInstance"
22   }
23 }

```

```

EXPLORER                         ...   main.tf    variables.tf    outputs.tf
DEVOPS-TASK
  main.tf
  outputs.tf
  variables.tf

variables.tf > variable "ami_id" > abc default
variable "ami_id" {
  default = "ami-0ecb62995f68bb549"
}
variable "instance_type" {
  default = "t3.micro"
}

```

```

EXPLORER                         ...   main.tf    variables.tf    outputs.tf
DEVOPS-TASK
  main.tf
  outputs.tf
  variables.tf

outputs.tf > output "id"
output "public_ip" {
  value = aws_instance.ExampleInstance.public_ip
}

output "id" {
  value = aws_instance.ExampleInstance.id
}

```

```

deepa@Deepak-Lenovo-NA8KJESL MINGW64 ~/Downloads/DevOps-Task (main)
$ git add .
warning: in the working copy of 'main.tf', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'outputs.tf', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'variables.tf', LF will be replaced by CRLF the next time Git touches it

deepa@Deepak-Lenovo-NA8KJESL MINGW64 ~/Downloads/DevOps-Task (main)
$ git commit -m "s3 backend added"
[main 8a88da2] s3 backend added
 3 files changed, 31 insertions(+), 6 deletions(-)

deepa@Deepak-Lenovo-NA8KJESL MINGW64 ~/Downloads/DevOps-Task (main)
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 751 bytes | 250.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sakshijain19/DevOps-Task.git
  306046c..8a88da2  main -> main

```

```

ubuntu@ip-172-30-5-19:~/DevOps-Task$ git pull origin
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 8 (delta 1), pack-reused 0 (from 0)
Unpacking objects: 100% (8/8), 1.04 KiB | 530.00 KiB/s, done.
From https://github.com/sakshijain19/DevOps-Task
  306046c..be1c54c  main -> origin/main
Updating 306046c..be1c54c
Fast-forward
 main.tf | 23 ++++++-----+
 outputs.tf |  8 ++++++
 variables.tf |  6 ++++++
 3 files changed, 31 insertions(+), 6 deletions(-)

```

Then run terraform init, plan, apply

```
ubuntu@ip-172-30-5-19:~/DevOps-Task$ terraform init
Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.28.0

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.
ubuntu@ip-172-30-5-19:~/DevOps-Task$ terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.my_ec2 will be created
+ resource "aws_instance" "my_ec2" {
  + ami = "ami-0ecb62995368bb549"
  + arn = "(known after apply)"
  + associate_public_ip_address = "(known after apply)"
  + availability_zone = "(known after apply)"
  + disable_api_stop = "(known after apply)"
  + disable_api_termination = "(known after apply)"
  + ebs_optimized = "(known after apply)"
  + enable_primary_ipv6 = "(known after apply)"
```

```
ubuntu@ip-172-30-5-19:~/DevOps-Task$ terraform apply
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.my_ec2 will be created
+ resource "aws_instance" "my_ec2" {
  + ami = "ami-0ecb62995368bb549"
  + arn = "(known after apply)"
  + associate_public_ip_address = "(known after apply)"
  + availability_zone = "(known after apply)"
  + disable_api_stop = "(known after apply)"
  + disable_api_termination = "(known after apply)"
  + ebs_optimized = "(known after apply)"
  + enable_primary_ipv6 = "(known after apply)"
  + force_destroy = "false"
  + get_password_data = "false"
  + host_id = "(known after apply)"
  + host_resource_group_arn = "(known after apply)"
  + iam_instance_profile = "(known after apply)"
  + id = "(known after apply)"
  + instance_initiated_shutdown_behavior = "(known after apply)"
  + instance.lifecycle = "(known after apply)"
  + instance_state = "(known after apply)"
  + instance_type = "t3.micro"
  + ipv4_address_count = "(known after apply)"
  + ipv6_addresses = "(known after apply)"
  + key_name = "(known after apply)"
  + monitoring = "(known after apply)"
  + outpost_arn = "(known after apply)"
  + password_data = "(known after apply)"
  + placement_group = "(known after apply)"
  + placement_group_id = "(known after apply)"
```

The screenshot shows the AWS S3 console. The left sidebar shows 'Amazon S3' with 'Buckets' expanded, showing 'sample001-test'. The main area shows the 'sample001-test' bucket details. Under 'Objects', there is one item: 'terraform.tfstate' (tfstate). The file was last modified on January 16, 2026, at 23:15:54 (UTC+05:30), has a size of 181.0 B, and is stored in the Standard storage class.

The screenshot shows the AWS EC2 Instances page. The left sidebar shows 'EC2' with 'Instances' expanded. The main area shows 'Instances (2)'. There are two instances listed: 'Terraform' (with ID i-044f858acd23bde9f3) and 'ExampleInstance' (with ID i-02dc0de05e392c606). Both instances are in the 'Running' state, have a t3.micro instance type, and are located in the us-east-1f Availability Zone. Their Public IP addresses are ec2-100-48-51-28.com... and ec2-44-223-46-126.co... respectively, with public access enabled.

○ Local & data block, modules, and VPC create

What are locals?

locals are named values you define **once** and reuse many times in your Terraform code.

```
main.tf M X variables.tf outputs.tf

locals {
  Instance_name = "ExampleInstance"
  region        = "us-east-1"
}

terraform {
  backend "s3" {
    bucket = "sample001-test"
    key    = "terraform.tfstate"
    region = local.region
  }
  provider "aws" {
    region = local.region
  }
}

resource "aws_instance" "ExampleInstance" {
  ami           = var.ami_id
  instance_type = var.instance_type

  subnet_id      = "subnet-0ff35381a8309014ad"
  vpc_security_group_ids = ["sg-088ae5ceffeadaf431"] # <-- FIX HERE

  tags = {
    Name = local.Instance_name
  }
}
```

What is a data block?

A **data block** is used to **read existing infrastructure** instead of creating new resources.

When to use data blocks?

- Use an existing VPC
 - Use an existing AMI
 - Use an existing Security Group

```
main.tf
provider "aws" {
  region = local.region
}

data "aws_security_group" "mysg" {
  filter {
    name   = "group-name"
    values = ["mysg"]
  }

  filter {
    name   = "vpc-id"
    values = ["vpc-0b5dc84df5e7cb8a9"]
  }
}

resource "aws_instance" "ExampleInstance" {
  ami           = var.ami_id
  instance_type = var.instance_type

  subnet_id          = "subnet-0ff35381a830914ad"
  vpc_security_group_ids = [data.aws_security_group.mysg.id]
}
```

Using data block

The screenshot shows two separate AWS service pages. The top part is the 'Instances' page under the EC2 service, showing a list of running instances. The bottom part is the 'sample001-test' bucket page under the S3 service, showing a single object named 'terraform.tfstate'.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
Terraform	i-044f83acd23bde9f5	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1f	ec2-100-27-252-108.co...	100.27
ExampleInstance	i-02ca32997db6990e	Running	t3.micro	Initializing	View alarms +	us-east-1f	ec2-44-220-243-34.co...	44.220

Amazon S3 > Buckets > sample001-test

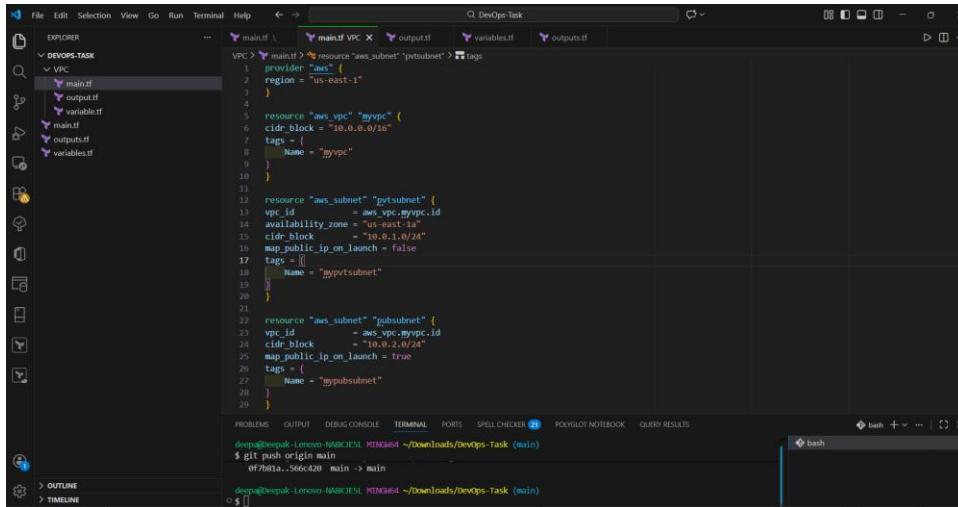
Name	Type	Last modified	Size	Storage class
terraform.tfstate	tfstate	January 17, 2026, 12:46:26 (UTC+05:30)	6.4 KB	Standard

VPC Creation

```
provider "aws" {
region = "us-east-1"
}
resource "aws_vpc" "myvpc" {
cidr_block = "10.0.0.0/16"
tags = {
  Name = "myvpc"
}
}
resource "aws_subnet" "pvtsubnet" {
vpc_id      = aws_vpc.myvpc.id
availability_zone = "us-east-1a"
cidr_block   = "10.0.1.0/24"
map_public_ip_on_launch = false
tags = {
  Name = "mypvtsubnet"
}
}
resource "aws_subnet" "pubsubnet" {
vpc_id      = aws_vpc.myvpc.id
cidr_block   = "10.0.2.0/24"
```

```
map_public_ip_on_launch = true
tags = {
    Name = "mypubsubnet"
}
}
}
resource "aws_internet_gateway" "myigw" {
vpc_id = aws_vpc.myvpc.id
tags = {
    Name = "myigw"
}
}
}
resource "aws_default_route_table" "myroute" {
default_route_table_id = aws_vpc.myvpc.default_route_table_id
route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.myigw.id
}
tags = {
    Name = "myroute"
}
}
}
resource "aws_route_table_association" "routeassoc" {
subnet_id    = aws_subnet.pubsubnet.id
route_table_id = aws_default_route_table.myroute.id
}
```

Push it into main GitHub Repository



```
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "myvpc" {
  cidr_block = "10.0.0.0/16"
  tags = [
    { Name = "myvpc" }
  ]
}

resource "aws_subnet" "privsubnet" {
  vpc_id          = aws_vpc.myvpc.id
  availability_zone = "us-east-1a"
  cidr_block      = "10.0.1.0/24"
  map_public_ip_on_launch = false
  tags = [
    { Name = "myprivsubnet" }
  ]
}

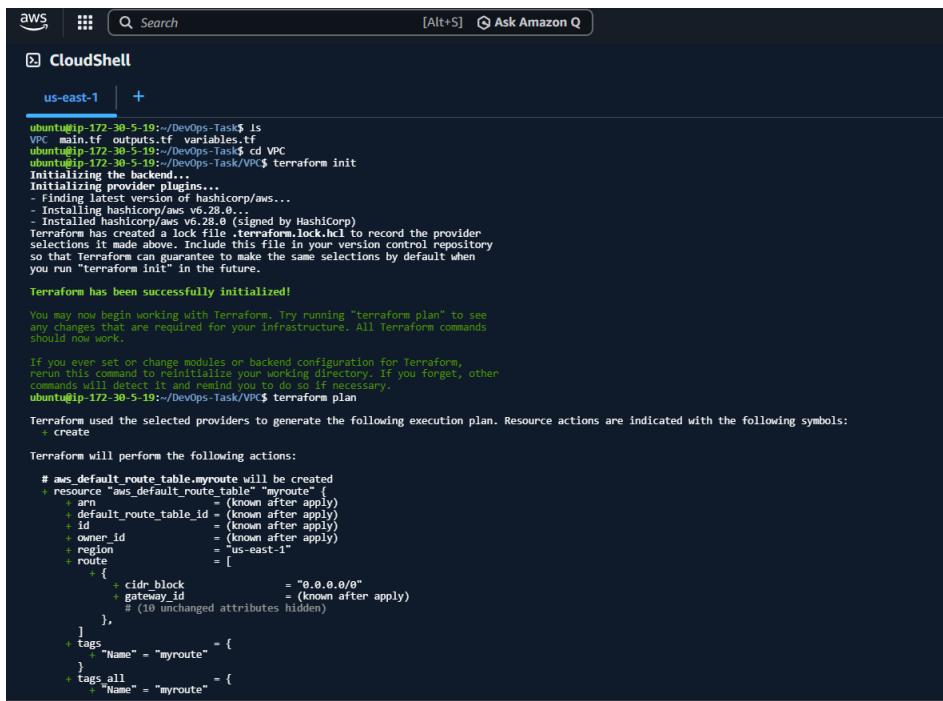
resource "aws_subnet" "pubsubnet" {
  vpc_id          = aws_vpc.myvpc.id
  cidr_block      = "10.0.2.0/24"
  map_public_ip_on_launch = true
  tags = [
    { Name = "mypubsubnet" }
  ]
}

provider "aws" {
  region = "us-east-1"
}

resource "aws_default_route_table" "myroute" {
  arn           = (known after apply)
  default_route_table_id = (known after apply)
  id            = (known after apply)
  owner_id      = (known after apply)
  region         = "us-east-1"
  route {
    cidr_block      = "0.0.0.0/0"
    gateway_id     = (known after apply)
    # (10 unchanged attributes hidden)
  },
  tags {
    Name = "myroute"
  }
  tags_all {
    Name = "myroute"
  }
}
```

Then run **git pull origin main**

Then run **terraform init, plan, apply**



```
ubuntu@ip-172-30-5-19:~/DevOps-Task$ ls
VPC  main.tf  outputs.tf  variables.tf
ubuntu@ip-172-30-5-19:~/DevOps-Task$ cd VPC
ubuntu@ip-172-30-5-19:~/DevOps-Task/VPC$ terraform init
Initializing the backend...
Initializing provider plugins...
[...]
Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.

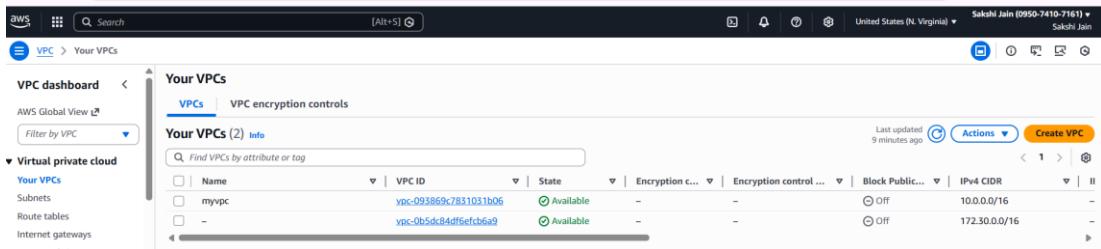
Terraform has been successfully initialized!

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.
ubuntu@ip-172-30-5-19:~/DevOps-Task/VPC$ terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_default_route_table.myroute will be created
+ resource "aws_default_route_table" "myroute" {
  arn           = (known after apply)
  default_route_table_id = (known after apply)
  id            = (known after apply)
  owner_id      = (known after apply)
  region         = "us-east-1"
  route {
    cidr_block      = "0.0.0.0/0"
    gateway_id     = (known after apply)
    # (10 unchanged attributes hidden)
  },
  tags {
    Name = "myroute"
  }
  tags_all {
    Name = "myroute"
  }
}
```



Name	VPC ID	State	Encryption controls	Encryption control ...	Block Public...	IPv4 CIDR
myvpc	vpc-093869c7831031b06	Available	-	-	Off	10.0.0.0/16
-	vpc-0b5dc84df6efcb6a9	Available	-	-	Off	172.30.0.0/16

Screenshot of the AWS VPC Subnets dashboard. The left sidebar shows 'Virtual private cloud' with 'Subnets' selected. The main table lists 8 subnets:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 C
mypubsubnet	subnet-07fb562a7fbfb6ca3f	Available	vpc-093869c7831031b06 myv...	Off	10.0.2.0/24	-
-	subnet-0ff55381a830914ad	Available	vpc-0b5dc84df6efcb6a9	Off	172.30.5.0/24	-
-	subnet-03c6720f7681e511b	Available	vpc-0b5dc84df6efcb6a9	Off	172.30.0.0/24	-
-	subnet-0b07a058ae5a91478	Available	vpc-0b5dc84df6efcb6a9	Off	172.30.2.0/24	-
-	subnet-01dc8a13f2b4a0bf0	Available	vpc-0b5dc84df6efcb6a9	Off	172.30.1.0/24	-
myvtsubnet	subnet-0ed0573bb5b1c4c16	Available	vpc-093869c7831031b06 myv...	Off	10.0.1.0/24	-
-	subnet-0e2c85afc36149c1a	Available	vpc-0b5dc84df6efcb6a9	Off	172.30.4.0/24	-
-	subnet-00031c7a91d0b08f6	Available	vpc-0b5dc84df6efcb6a9	Off	172.30.3.0/24	-

Screenshot of the AWS VPC Route tables dashboard. The left sidebar shows 'Virtual private cloud' with 'Route tables' selected. The main table lists 2 route tables:

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC	Owner ID
-	rtb-00e4113176962c05f	-	-	Yes	vpc-0b5dc84df6efcb6a9	095074107161
myroute	rtb-0567e657b988cb7d	subnet-07b562a7fbfb6ca...	-	Yes	vpc-093869c7831031b06 myv...	095074107161

Screenshot of the AWS VPC Internet gateways dashboard. The left sidebar shows 'Virtual private cloud' with 'Your VPCs' selected. The main table lists 2 internet gateways:

Name	Internet gateway ID	State	VPC ID	Owner
-	igw-06f00e001160b3752	Attached	vpc-0b5dc84df6efcb6a9	095074107161
myigw	igw-0fb24e7296c12bdff5	Attached	vpc-093869c7831031b06 myv...	095074107161

EC2 Instance Creation-inside-VPC

Find code Here

EC2-inside-VPC

Commands to Run on Cloud shell

Git pull origin main
 terraform init
 terraform plan
 terraform apply

Screenshot of the AWS EC2 Instances and Security Groups dashboards. The left sidebar shows 'Instances' selected.

Instances (3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs
Terraform	i-044f83acd23bde9f3	Running	t3.micro	2/3 checks passed	View alarms +	us-east-1f	ec2-100-27-252-108.co...	100.27.252.108	-	-
clouteam-ins...	i-0a4ee9fbbeadbfb285	Running	t3.micro	2/3 checks passed	View alarms +	us-east-1f	-	3.237.31.119	-	-
clouteam-ins...	i-0b42a581a798856d9	Running	t3.micro	2/3 checks passed	View alarms +	us-east-1f	-	98.81.28.211	-	-

Security Groups (5) Info

Name	Security group ID	Security group name	VPC ID	Description	Owner	Inbs
-	sg-0be20f27f8a6fc49a	launch-wizard-1	vpc-0b5dc84df6efcb6a9	launch-wizard-1 created 2025-12-28T1...	095074107161	1 Pe
clouteam-mysg	sg-02462c02182f5416f	clouteam-mysg	vpc-058f1778e356da81	Allow httpd service	095074107161	2 Pe

The screenshot shows the AWS VPC dashboard with the following details:

- VPCs** tab selected.
- Your VPCs (2)** section.
- Name**: cloudteam_module, **VPC ID**: vpc-058f177be836daa81, **State**: Available, **Encryption controls**: -, **Encryption control type**: -, **Block Public Access**: Off, **IPv4 CIDR**: 10.0.0.0/16, **IPv6 CIDR**: -, **DHCP option set**: dopt-01ba473.
- Name**: -, **VPC ID**: vpc-0b5dc84df6efcb5a9, **State**: Available, **Encryption controls**: -, **Encryption control type**: -, **Block Public Access**: Off, **IPv4 CIDR**: 172.30.0.0/16, **IPv6 CIDR**: -, **DHCP option set**: dopt-01ba473.

Subnets (8) Info										
	Name	Subnet ID	State	VPC	Block Public... Off	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID	Last updated 2 minutes ago	Actions
<input type="checkbox"/>	clouddemo_pub_subnet	subnet-0e611021a9d6372a7	Available	vpc-05bf177be8356da81 clou...	<input type="radio"/> Off	10.0.1.0/24	-	-		Actions
<input type="checkbox"/>	-	subnet-0ff53581a50914ad	Available	vpc-0b5dc4df6efcb6a9	<input type="radio"/> Off	172.0.5.0/24	-	-		Actions
<input type="checkbox"/>	-	subnet-03c672077681e511b	Available	vpc-0b5dc4df6efcb6a9	<input type="radio"/> Off	172.30.0/24	-	-		Actions
<input type="checkbox"/>	-	subnet-0007058a9391478	Available	vpc-0b5dc4df6efcb6a9	<input type="radio"/> Off	172.30.2.0/24	-	-		Actions
<input type="checkbox"/>	-	subnet-01d8a1f2b4a0hf0	Available	vpc-0b5dc4df6efcb6a9	<input type="radio"/> Off	172.30.1.0/24	-	-		Actions
<input type="checkbox"/>	-	subnet-0e2c85afc361491ca	Available	vpc-0b5dc4df6efcb6a9	<input type="radio"/> Off	172.30.4.0/24	-	-		Actions
<input type="checkbox"/>	-	subnet-00031c7a91j008fb	Available	vpc-0b5dc4df6efcb6a9	<input type="radio"/> Off	172.30.3.0/24	-	-		Actions
<input type="checkbox"/>	clouddemo_pvt_subnet	subnet-0e3b40d8014ae461	Available	vpc-05bf177be8356da81 clou...	<input type="radio"/> Off	10.0.0.0/24	-	-		Actions

Route tables (2) Info								Last updated	Actions	Create route table
								less than a minute ago		
								< 1 >		
	Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC	Owner ID			
Filter by VPC	rtb-004e4113176b62c05f	-	-	-	Yes	vpc-0b5dc84d6fe6cb6a9	095074107161			
Virtual private cloud	clouddteam_myRT	rtb-0034b5ef3fb0e6051	-	-	Yes	vpc-05f177fe836da811clou...	095074107161			

Internet gateways (2) <small>Info</small>						
	Name	Internet gateway ID	State	VPC ID	Owner	Actions
<input type="checkbox"/>	clousteam_igw	igw-02625187a5fa7aa45	Attached	vpc-05f8f1778e836daa811clousteam...	095074107161	Edit Delete Details
<input type="checkbox"/>	-	igw-06fd0ed01160b3752	Attached	vpc-0b5d484ffefcf6a9	095074107161	Edit Delete Details