

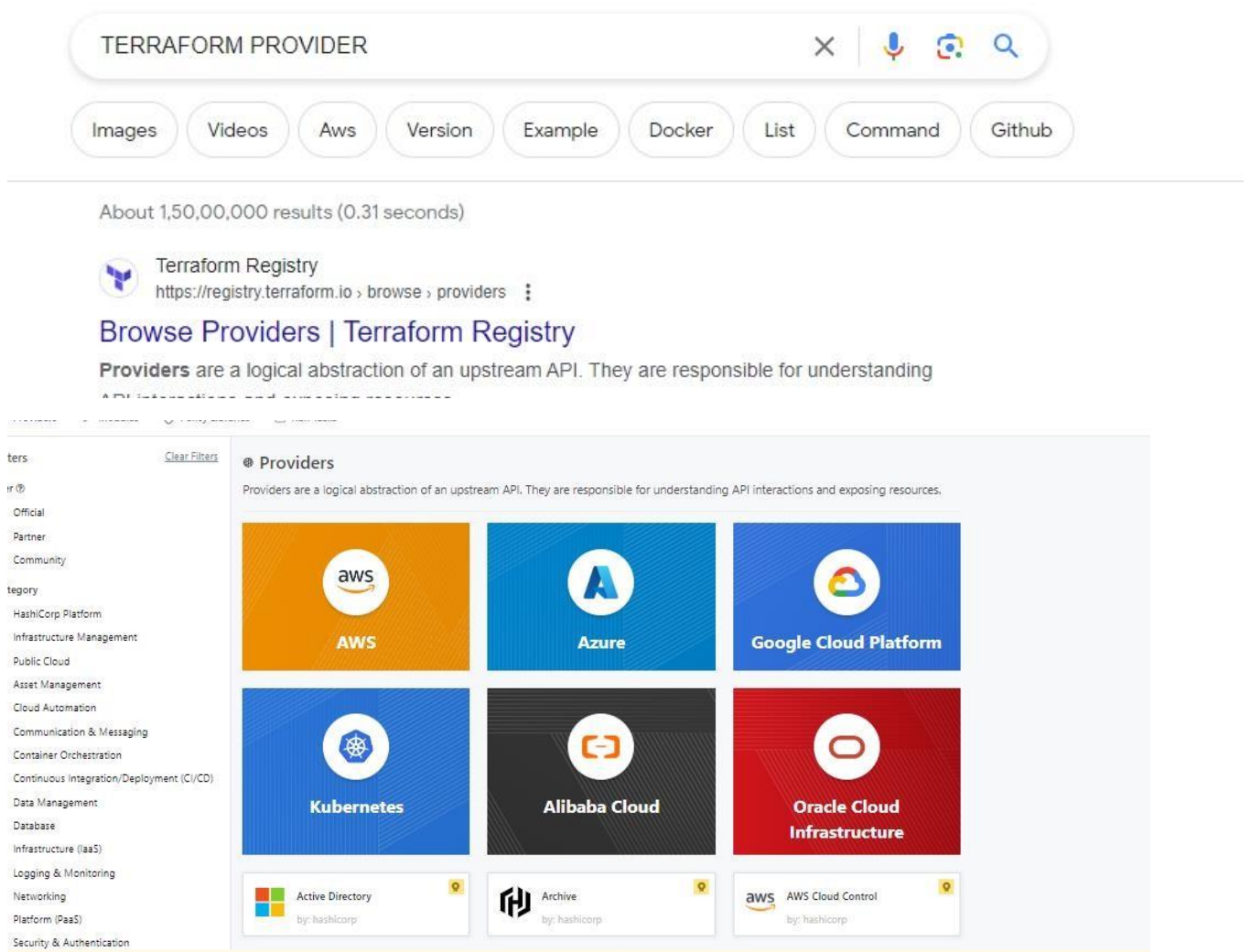
## TERRAFORM WITH AWS PROJECT

### AIM- SETTING UP INFRASTRUCTURE ON AWS USING TERRAFORM

#### ARCHITECTURE/TOPOLOGY-

1)CREATE AN I AM USER PROFILE AND GIVE PERMISSION OF EC2 ,S3 ,AND

2)NOW WRITE TERRAFORM SCRIPT SO FOR THAT GO TO GOOGLE AND SEARCH TERRAFORM PROVIDER



## SELECT AWS

The screenshot shows the Terraform Registry page for the `aws` provider. The breadcrumb navigation is `Providers / hashicorp / aws`. The current version is `5.41.0`, marked as the latest. A dropdown menu for versions is open, showing a list of versions from `5.38.0` to `5.41.0`. The `5.41.0` version is selected and has a checkmark. The page includes a description of the provider, a 'Public Cloud' badge, and a 'USE PROVIDER' button. On the right, there is a 'Provider Downloads' section showing statistics for the current version.

Downloads	Value
Downloads this week	25.7M
Downloads this month	55.3M
Downloads this year	249.8M
Downloads over all time	2.7B

## SELECT VERSION AND CLICK ON USE PROVIDER

This screenshot shows the same Terraform Registry page for the `aws` provider, but with the version dropdown menu open. The dropdown shows a list of versions, with `5.41.0` selected. The 'USE PROVIDER' button is visible. On the right, there is a code editor showing the Terraform configuration for the `aws` provider.

```
run terraform init.

Terraform 0.13+

terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.41.0"
    }
  }
}

provider "aws" {
  # Configuration options
}
```

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.41.0"
    }
  }
}

provider "aws" {
  # Configuration options
}
```

STEP 2- INSTALL TERRRAFORM FIRST for Ubuntu/linux

CMD-sudo apt-get update

Cmd 2- sudo apt-get install unzip

CMD wget

[https://releases.hashicorp.com/terraform/1.1.0/terraform\\_1.1.0\\_linux\\_amd64.zip](https://releases.hashicorp.com/terraform/1.1.0/terraform_1.1.0_linux_amd64.zip)

-Unzip the downloaded file:

Cmd -unzip terraform\_1.1.0\_linux\_amd64.zip

Move the Terraform binary to a directory in your PATH, such as /usr/local/bin/

sudo mv terraform /usr/local/bin/

to verify the terraform download cmd- terraform -v

NOW GO TO AWS

CMD- AWS CONFIGURE

ENTER YOUR I M USER CR3EDENTIAL NOT YOUR REAL ONE

STEP 3- GIVE CMD-

TERRAFORM INIT

CREATE VIM MAIN.TF

IN THESE OUR PROJECT START WITH CREATING VPC INSIDE IT

Create an instance of Ubuntu and download terraform and start the process



## STEP2- CREATE AN IAM USER NOW FOR OUR PROJECT

### 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.

**Are you providing console access to a person?**

User type

☐ Specify a user in Identity Center - Recommended  
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

☒ I want to create an IAM user  
We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

Console password

☐ Autogenerated password  
You can view the password after you create the user.

☒ Custom password  
Enter a custom password for the user.

We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

☒ I want to create an IAM user  
We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

Console password

☐ Autogenerated password  
You can view the password after you create the user.

☒ Custom password  
Enter a custom password for the user.

- Must be at least 8 characters long
- Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & \* ( ) \_ + - (hyphen) = [ ] { } | ' "

☒ 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](#) [to](#)

## GIVE ALL PERMISSION

### 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.

### Permissions policies (1/1204)

Choose one or more policies to attach to your new user.

Filter by Type: All types 266 matches

Policy name	Type	Attached entities
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	0
<input type="checkbox"/> AdministratorAccess-Amazon	AWS managed	0

### Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

#### User details

User name BICKY	Console password type Custom password	Require password reset No
--------------------	--	------------------------------

#### Permissions summary

Name	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy

#### Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

EC2 VPC S3 AWS Auto Scaling Simple Queue Service Simple Notification Service Key Management Service CloudTrail Amazon EventBridge

User created successfully

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

IAM > Users > Create user

Step 1

Specify user details

### Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This

## CREATE AN CREDITIAL I.E ACCESS KEY

Access key

### Access key best practices & alternatives [Info](#)

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

#### Use case

☒ **Command Line Interface (CLI)**

You plan to use this access key to enable the AWS CLI to access your AWS account.

☐ **Local code**

You plan to use this access key to enable application code in a local development environment to access your AWS account.

☐ **Application running on an AWS compute service**

You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

☐ **Third-party service**

You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

☐ **Application running outside AWS**

You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

&

### Set description tag - *optional* [Info](#)

The description for this access key will be attached to this user as a tag and shown alongside the access key.

#### Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

NINJA HATOR

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ . : / = + - @

Cancel

Previous

Create access key

The screenshot shows the AWS IAM console interface. At the top, a green banner states "Access key created" with a warning: "This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time." Below this, the "Retrieve access keys" page is displayed. It includes a sidebar with navigation steps: Step 1 (Access key best practices & alternatives), Step 2 - optional (Set description tag), and Step 3 (Retrieve access keys). The main content area shows the "Access key" details, including the Access key ID (AKIA5FTZBAIDQI4EVOBD) and the Secret access key (masked with asterisks and a "Show" link). A note at the bottom states: "If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive."

Access key	Secret access key
AKIA5FTZBAIDQI4EVOBD	***** <a href="#">Show</a>

sNOW CONNECT TO INSTANCE, GO TO AWS CONFIGURE AND AND INSTALL TERRAFORM



```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-25-47:/home/ubuntu# sudo apt-get install unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  zip
The following NEW packages will be installed:
  unzip
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 175 kB of archives.
After this operation, 386 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 unzip amd64 6.0-26ubuntu3.2 [175 kB]
Fetched 175 kB in 0s (7095 kB/s)
Selecting previously unselected package unzip.
(Reading database ... 65273 files and directories currently installed.)
Preparing to unpack .../unzip_6.0-26ubuntu3.2_amd64.deb ...
Unpacking unzip (6.0-26ubuntu3.2) ...
Setting up unzip (6.0-26ubuntu3.2) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes... [=====
```

```
root@ip-172-31-25-47:/home/ubuntu# wget https://releases.hashicorp.com/terraform/1.1.0/terraform_1.1.0_linux_amd64.zip
--2024-03-17 07:10:58-- https://releases.hashicorp.com/terraform/1.1.0/terraform_1.1.0_linux_amd64.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 108.138.85.53, 108.138.85.65, 108.138.85.30, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)|108.138.85.53|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 18683106 (18M) [application/zip]
Saving to: 'terraform_1.1.0_linux_amd64.zip'
```

```
terraform_1.1.0_linux_amd64.zip      100%[=====
2024-03-17 07:10:58 (146 MB/s) - 'terraform_1.1.0_linux_amd64.zip' saved [18683106/18683106]
```

```
root@ip-172-31-25-47:/home/ubuntu#
```

```
root@ip-172-31-25-47:/home/ubuntu# unzip terraform_1.1.0_linux_amd64.zip
Archive:  terraform_1.1.0_linux_amd64.zip
  inflating: terraform
root@ip-172-31-25-47:/home/ubuntu#
```



```
root@ip-172-31-25-47:/home/ubuntu# sudo mv terraform /usr/local/bin/  
root@ip-172-31-25-47:/home/ubuntu# terraform --v  
Usage: terraform [global options] <subcommand> [args]
```

The available commands for execution are listed below.  
The primary workflow commands are given first, followed by  
less common or more advanced commands.

Main commands:

init	Prepare your working directory for other commands
validate	Check whether the configuration is valid
plan	Show changes required by the current configuration
apply	Create or update infrastructure
destroy	Destroy previously-created infrastructure

All other commands:

console	Try Terraform expressions at an interactive command prompt
fmt	Reformat your configuration in the standard style
force-unlock	Release a stuck lock on the current workspace
get	Install or upgrade remote Terraform modules
graph	Generate a Graphviz graph of the steps in an operation

Now connect to aws

```
root@ip-172-31-25-47:/home/ubuntu# aws configure  
Command 'aws' not found, but can be installed with:  
snap install aws-cli # version 1.15.58, or  
apt install awscli # version 1.22.34-1  
See 'snap info aws-cli' for additional versions.  
root@ip-172-31-25-47:/home/ubuntu#
```

So install aws cli first



## ChatGPT

To install the AWS Command Line Interface (CLI) on Ubuntu, you can use the `awscli` package available in the official Ubuntu repositories. Here's how you can do it:

### 1. Update package index:

```
bash
```

[Copy code](#)

```
sudo apt update
```

### 2. Install AWS CLI:

```
bash
```

[Copy code](#)

```
sudo apt install awscli
```



```

root@ip-172-31-25-47:/home/ubuntu# sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-25-47:/home/ubuntu# sudo apt install awscli
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bzip2 docutils-common fontconfig fontconfig-config fonts-droid-fallback fonts-noto-mono fonts-urw-base35 ghostscript groff gsfonts hicolor-icon-theme
  imagemagick-6.q16 libaom3 libavahi-client3 libavahi-common-data libavahi-common3 libcairo2 libcups2 libdatrie1 libdav1d5 libde265-0 libffi8 libfontconfig1
  libfontconfig1-dev libgomp1 libgraphite2-3 libgs9 libgs9-common libharfbuzz0b libheif1 libice6 libidn2 libijs-0.35 libil2cpp-0 libjpeg-turbo8 libjpeg8
  libjxr-tools libjxr0 liblcms2-2 liblqr-1-0 libltdl7 libmagickcore-6.q16-6 libmagickcore-6.q16-6-extra libmagickwand-6.q16-6 libopenjp2-7 libpango-1.0-0
  libpangocairo-1.0-0 libpangoft2-1.0-0 libpaper-utils libpaper1 libpixman-1-0 librsvg2-bin librsvg2-common libsm6 libthai-data libwebpmux3 libwmf-lite-0.2-7
  libx265-199 libxaw7 libxcb-render0 libxcb-shm0 libxmu6 libxpm4 libxrender1 libxt6 mailcap mime-support netbase python3-dateutil python3-docutils python3-jmespath
  python3-olefile python3-pil python3-pygments python3-roman python3-rsa python3-s3transfer
Suggested packages:
  bzip2-doc fonts-noto fonts-freefont-otf | fonts-freefont-ttf fonts-texgyre ghostscript-x imagemagick-doc autotrace cups-bsd | lpr | lp | hplip |
  hp2xx html2ps libwmf-bin mplayer povray radiance sane-utils texlive-base-bin transfig ufw batch xdg-utils cups-common libfftw3-bin libfftw3-dev libfontconfig1-doc
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nlscript fonts-ns-simply
  fonts-urw-base35-extras

```

No VM guests are running outdated hypervisor (qemu) binaries on this host.

```
root@ip-172-31-25-47:/home/ubuntu#
```

```
root@ip-172-31-25-47:/home/ubuntu# aws configure
```

```
AWS Access Key ID [None]:
```

Enter your IAM user access key

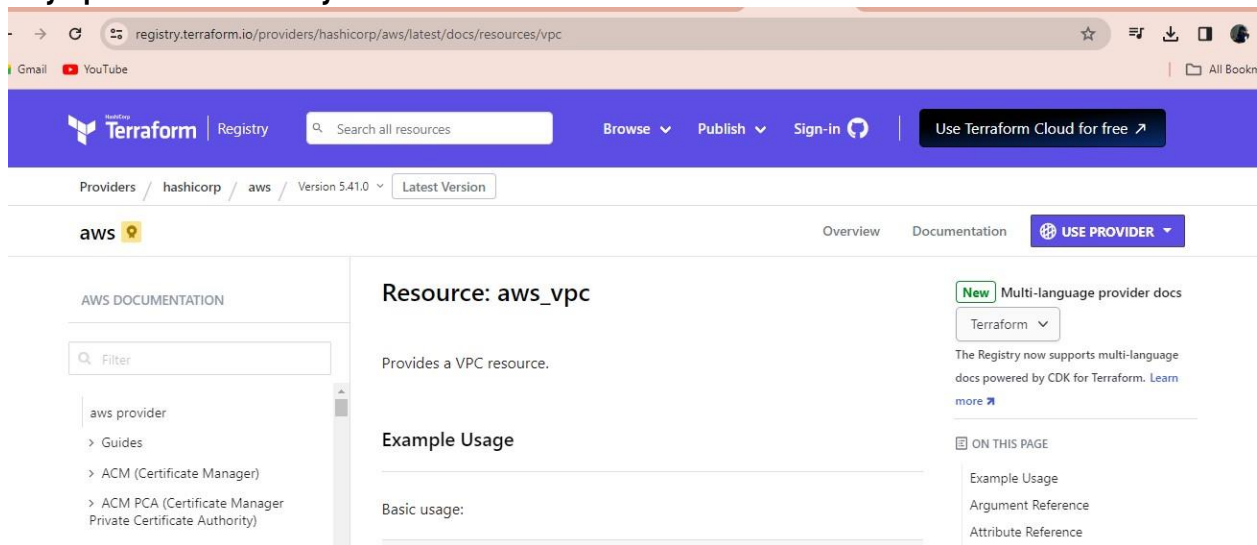
```

Default output format [None].
root@ip-172-31-25-47:/home/ubuntu# ls
terraform_1.1.0_linux_amd64.zip
root@ip-172-31-25-47:/home/ubuntu#

```

now create vpc

go to google and searchh terraform aws vpc and copyy the documentation  
or youp can create your own



```

resources "aws_vpc" "Bickyvpc" { cidr_block=
"10.0.0.0/16" }

```

these is terraform script for creating vpc now also create two subnet under these vpc

```

resource "aws_subnet" "subnet1" {
vpc_id= aws_vpc.Bickyvpc.id
cidr_block= "10.0.0.0/24" availability
zone= "us-east-1a"
map_public_ip_on_launch= true
}

```

#to get public ip use cmd- map\_public\_ip\_on\_launch= true

Now made another subnet same script copy and paste change name and cidr block

```
resource "aws_subnet" "subnet2" {  
  vpc_id= aws_vpc.Bickyvpc.id  
  cidr_block= "10.0.1.0/24" availability  
  zone= "us-east-1b"  
  map_public_ip_on_launch= true  
}
```

### Step-3

Create an internet gateway resource

```
"aws_internet_gateway" "igw" { vpc_id=  
aws_vpc.Bickyvpc.id }
```

### STEP-4

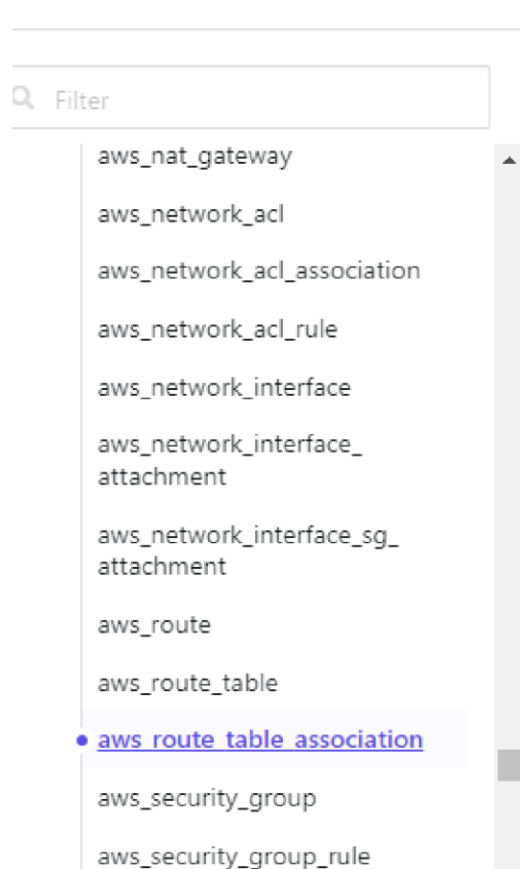
now create an route table and attach INTERNET GATEWAY TO IT

```
resource "aws_route_table" "route" {  
  vpc_id= aws_vpc.Bickyvpc.id route {  
    cidr_block= "0.0.0.0/0"  
    gateway_id = aws_internet_gateway.igw.id  
  } }
```

now its time to associate route to subnet so for

create resources and inside it put two thing the subnet id which you want to attach and where you want to attach i.e internet gateway part start

resource "aws\_route\_table\_association" "route1"



## Example Usage

```
resource "aws_route_table_association" "a" {  
  subnet_id      = aws_subnet.foo.id  
  route_table_id = aws_route_table.bar.id  
}
```

```
resource "aws_route_table_association" "b" {  
  gateway_id      = aws_internet_gateway.foo.id  
  route_table_id = aws_route_table.bar.id  
}
```

## Argument Reference

▲ NOTE

{ subnet\_id = aws\_subnet.subnet1.id

Route\_table\_id = aws\_route\_table.route.id }

resource "aws\_route\_table\_association" "route2"

{ subnet\_id = aws\_subnet.subnet2.id

Route\_table\_id = aws\_route\_table.route.id }

```
resource "aws_internet_gateway" "igw" { vpc_id=
aws_vpc.Bickyvpc.id }
```

Now complete terraform script is made let execute

Go to instance and create vim main.tf and enter these script

```
{ resources "aws_vpc" "Bickyvpc" { cidr_block=
"10.0.0.0/16" }
```

```
resource "aws_subnet" "subnet1" {
vpc_id= aws_vpc.Bickyvpc.id
cidr_block= "10.0.0.0/24" availability
zone= "us-east-1a"
map_public_ip_on_launch= true
}
```

```
resource "aws_subnet" "subnet2" {
vpc_id= aws_vpc.Bickyvpc.id
cidr_block= "10.0.1.0/24" availability
zone= "us-east-1b"
map_public_ip_on_launch= true
}
```



```
resource "aws_internet_gateway" "igw" { vpc_id=
aws_vpc.Bickyvpc.id }
```

```
resource "aws_route_table" "route" {
vpc_id= aws_vpc.Bickyvpc.id route {
  cidr_block= "0.0.0.0/0"
gateway_id = aws_internet_gateway.igw.id
} }
```

```
resource "aws_route_table_association" "route1"
{ subnet_id = aws_subnet.subnet1.id
Route_table_id = aws_route_table.route.id }
```

```
resource "aws_route_table_association" "route2" {
subnet_id = aws_subnet.subnet2.id
Route_table_id = aws_route_table.route.id } }
```

SO BAFTER RUNNING MY HANDMADE SCRIPT I GOT LOT OF SYNTAX AND  
UPPERCASE ERROR REMEMEVBRE IN TERRAFORM ALWAYS USED  
LOWERCASE RESOURCES NAME WHNEEVER YOUY GIVE

SO I USE CHATGPT TO CORR3ECT MY SUNTAX

## ChatGPT

It looks like you're trying to create a VPC (Virtual Private Cloud) in AWS using Terraform. However, there are a few syntax errors and incorrect resource names in your code. Here's a corrected version of your Terraform code:

```
hcl                                                                    Copy code

resource "aws_vpc" "bickyvpc" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "subnet1" {
  vpc_id            = aws_vpc.bickyvpc.id
  cidr_block        = "10.0.0.0/24"
  availability_zone = "us-east-1a"
}
```

message ChatGPT...

In this corrected version, I've made the following changes:

- Changed resource names to use lowercase (`bickyvpc`, `subnet1`, `subnet2`, etc.) as Terraform resource names should be lowercase.
- Corrected the `availability_zone` parameter in `aws_subnet` resources to use the correct syntax (`availability_zone = "us-east-1a"` instead of `availability zone= "us-east-1a"`).
- Corrected the `gateway_id` parameter in `aws_route_table` resource to use the correct resource reference (`gateway_id = aws_internet_gateway.igw.id`).
- Corrected the `Route_table_id` parameter in `aws_route_table_association` resources to use the correct resource reference (`route_table_id =`

/message ChatGPT...

CODE IS -

```
resource "aws_vpc" "bickyvpc" {
  cidr_block = "10.0.0.0/16"
```

```
}
```

```
resource "aws_subnet" "subnet1" {  
  vpc_id          = aws_vpc.bickyvpc.id  
  cidr_block      = "10.0.0.0/24"  
  availability_zone = "us-east-1a"  
  map_public_ip_on_launch = true  
}
```

```
resource "aws_subnet" "subnet2" {  
  vpc_id          = aws_vpc.bickyvpc.id  
  cidr_block      = "10.0.1.0/24"  availability_zone  
= "us-east-1b"  map_public_ip_on_launch = true  
}
```

```
resource "aws_internet_gateway" "igw" {  
  vpc_id = aws_vpc.bickyvpc.id  
}
```

```
resource "aws_route_table" "route" {  
  vpc_id = aws_vpc.bickyvpc.id  
  
  route {  
    cidr_block = "0.0.0.0/0"  
    gateway_id = aws_internet_gateway.igw.id  
  }  
}
```

```
}  
}
```

```
resource "aws_route_table_association" "route1" {  
  subnet_id    = aws_subnet.subnet1.id  route_table_id  
  = aws_route_table.route.id  
}
```

```
resource "aws_route_table_association" "route2" {  
  subnet_id    = aws_subnet.subnet2.id  route_table_id  
  = aws_route_table.route.id  
}
```

```
root@ip-172-31-25-47:/home/ubuntu# vim main.tf
root@ip-172-31-25-47:/home/ubuntu# terraform init
```

Initializing the backend...

Initializing provider plugins...

- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.41.0...
- Installed hashicorp/aws v5.41.0 (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-172-31-25-47:/home/ubuntu#
```

Run 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-172-31-25-47:/home/ubuntu# terraform plan
```

**Error:** Invalid provider configuration

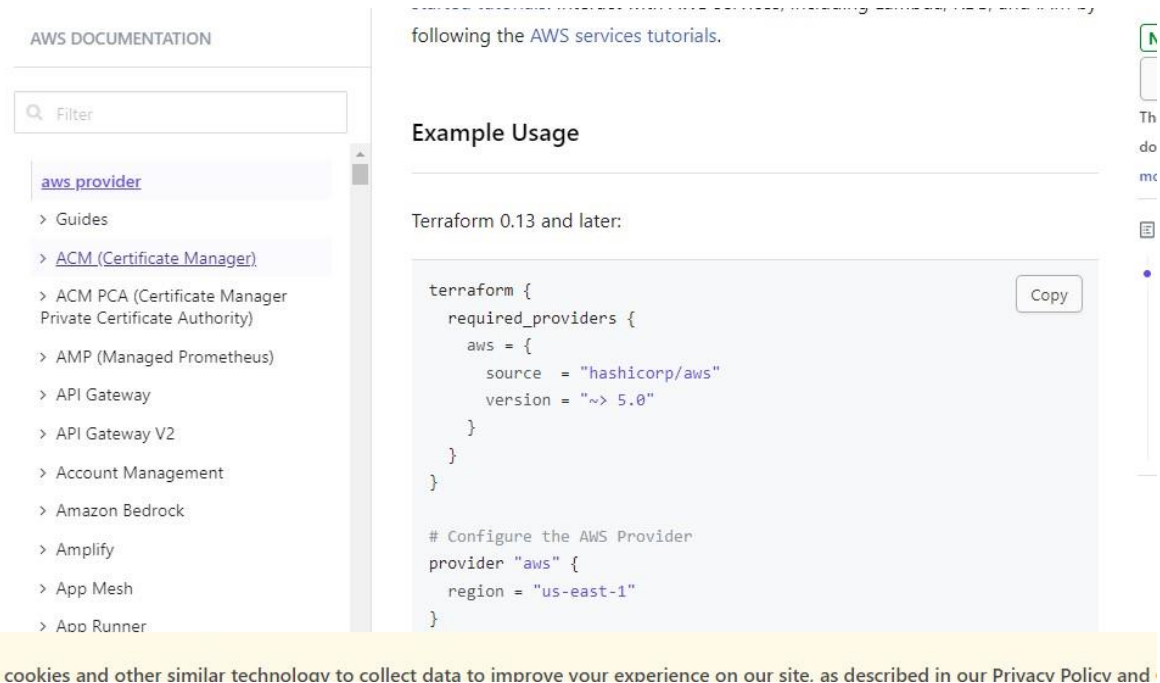
Provider "registry.terraform.io/hashicorp/aws" requires explicit configuration. Add a provider block to the root module and consult the provider documentation.

**Error:** Invalid AWS Region:

```
with provider["registry.terraform.io/hashicorp/aws"],
on <empty> line 0:
(source code not available)
```

```
root@ip-172-31-25-47:/home/ubuntu#
```

AFTER THESE I GOT SUCCESSFULLY TERRAFORM INIT WITHOUT ERROR AGAIN WE GETTING ERROR SOO LETS FIX IN OUR SCRIPT WE DIDN'T WRITTEN THE PROVIDER SCRIPT I.E



SO RE EDITING THE MAIN.TF AGAIN

```
terraform { required_providers {  
aws = {  
    source = "hashicorp/aws"  
    version = "~> 5.0"  
    }  
    }  
}
```

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

ADD THESE SCRIPT AT TOP



```

terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

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

resource "aws_vpc" "bickyvpc" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "subnet1" {
  vpc_id            = aws_vpc.bickyvpc.id
  cidr_block        = "10.0.0.0/24"
  availability_zone = "us-east-1a"
  map_public_ip_on_launch = true
}

resource "aws_subnet" "subnet2" {
  vpc_id            = aws_vpc.bickyvpc.id
  cidr_block        = "10.0.1.0/24"
  availability_zone = "us-east-1b"
}

"main.tf" 53L, 1135B

```

**Error:** Invalid AWS Region:

```

with provider["registry.terraform.io/hashicorp/aws"],
on <empty> line 0:
(source code not available)

```

```

root@ip-172-31-25-47:/home/ubuntu# vim main.tf
root@ip-172-31-25-47:/home/ubuntu# 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_internet_gateway.igw will be created
+ resource "aws_internet_gateway" "igw" {
+   arn          = (known after apply)
+   id           = (known after apply)
+   owner_id     = (known after apply)
+   tags_all     = (known after apply)
+   vpc_id       = (known after apply)
}

```

```

+ cidr_block              = "10.0.1.0/24"
+ enable_dns64            = false
+ enable_resource_name_dns_a_record_on_launch = false
+ enable_resource_name_dns_aaaa_record_on_launch = false
+ id                      = (known after apply)
+ ipv6_cidr_block_association_id = (known after apply)
+ ipv6_native             = false
+ map_public_ip_on_launch = true
+ owner_id                = (known after apply)
+ private_dns_hostname_type_on_launch = (known after apply)
+ tags_all                = (known after apply)
+ vpc_id                  = (known after apply)
}

```

aws\_vpc.bickyvpc will be created

```

resource "aws_vpc" "bickyvpc" {
+ arn                    = (known after apply)
+ cidr_block            = "10.0.0.0/16"
+ default_network_acl_id = (known after apply)
+ default_route_table_id = (known after apply)
+ default_security_group_id = (known after apply)
+ dhcp_options_id       = (known after apply)
+ enable_dns_hostnames   = (known after apply)
+ enable_dns_support     = true
+ enable_network_address_usage_metrics = (known after apply)
+ id                    = (known after apply)
+ instance_tenancy       = "default"
+ ipv6_association_id    = (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: yes

aws\_vpc.bickyvpc: Creating...

aws\_vpc.bickyvpc: Creation complete after 1s [id=vpc-02deef3f2cf5fc464]

aws\_subnet.subnet1: Creating...

aws\_internet\_gateway.igw: Creating...

aws\_subnet.subnet2: Creating...

aws\_internet\_gateway.igw: Creation complete after 1s [id=igw-04cf00b9ddec64e65]

aws\_route\_table.route: Creating...

aws\_route\_table.route: Creation complete after 0s [id=rtb-07fcb25366b5a5141]

aws\_subnet.subnet1: Still creating... [10s elapsed]

aws\_subnet.subnet2: Still creating... [10s elapsed]

aws\_subnet.subnet1: Creation complete after 11s [id=subnet-06a15732efe1fecab]

aws\_route\_table\_association.route1: Creating...

aws\_subnet.subnet2: Creation complete after 11s [id=subnet-0bf1de3c2bc1a8031]

aws\_route\_table\_association.route2: Creating...

aws\_route\_table\_association.route1: Creation complete after 0s [id=rtbassoc-08b940c55c4e65]

aws\_route\_table\_association.route2: Still creating... [10s elapsed]

aws\_route\_table\_association.route2: Still creating... [20s elapsed]

aws\_route\_table\_association.route2: Creation complete after 23s [id=rtbassoc-053fcc7399]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

root@ip-172-31-25-47:/home/ubuntu# ls

main.tf terraform.tfstate terraform 1.1.0 linux amd64.zip

```

    "dhcp_options_id": "dopt-0ba4c0aac11e2f1f9",
    "enable_dns_hostnames": false,
    "enable_dns_support": true,
    "enable_network_address_usage_metrics": false,
    "id": "vpc-02deef3f2cf5fc464",
    "instance_tenancy": "default",
    "ipv4_ipam_pool_id": null,
    "ipv4_netmask_length": null,
    "ipv6_association_id": "",
    "ipv6_cidr_block": "",
    "ipv6_cidr_block_network_border_group": "",
    "ipv6_ipam_pool_id": "",
    "ipv6_netmask_length": 0,
    "main_route_table_id": "rtb-04cfb7c624fad9e75",
    "owner_id": "905418179079",
    "tags": null,
    "tags_all": {}
  },
  "sensitive_attributes": [],
  "private": "eyJzY2h1bWVfdmVyc2lvbiI6IjEifQ=="
}
}
}
}
}
root@ip-172-31-25-47:/home/ubuntu# vim main.tf
root@ip-172-31-25-47:/home/ubuntu#

```

Hence we create automation creation of vpc using terraform

<input type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set	Mz
<input type="checkbox"/>	-	<a href="#">vpc-02deef3f2cf5fc464</a>	Available	10.0.0.0/16	-	<a href="#">dopt-0ba4c0aac11e2f1f9</a>	rtb-
<input type="checkbox"/>	-	<a href="#">vpc-04643f4dea97e29fa</a>	Available	172.31.0.0/16	-	<a href="#">dopt-0ba4c0aac11e2f1f9</a>	rtb-

#now create ec2 instance

```
resource "aws_instance" "webserver" {
```

```
ami           = "ami-08116b9957a259459"
```

```
instance_type = "t2.micro"
```

```
vpc_security_group_ids = aws_security_group.mysg.id
```

```
subnet_id      = aws_subnet.subnet1.id
```

```
}
```

#now for some additional task like load balancer application which we gona add later we will write a code script # batch! to depploy webserve to check traffic since we give port 80 http so simplify we can check these task also so lets start

#for these first create a file

Code- vim userdata.sh



Upload a file with your user data or enter it in the field.

 Choose file

☐ User data has already been base64 encoded

#then inside it enter a script but we need to encode the script first so for that used `base64encode(file("user_data.sh"))`

So your final code will be for creating instance is :-

```
resource "aws_instance" "webserver" {
```

```
ami          = "ami-08116b9957a259459"
```

```
instance_type = "t2.micro"
```

```
vpc_security_group_ids = aws_security_group.mysg.id
subnet_id              = aws_subnet.subnet1.id
user_data               = base64encode(file("user_data.sh"))
}
```

But before use cmd vim and create user\_data.sh and insert script

Batch script=

```
#!/bin/bash
```

```
apt update
```

```
apt install -y apache2
```

```
# Get the instance ID using the instance metadata
```

```
INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-data/instance-id)
```

```
# Install the AWS CLI
```

```
apt install -y awscli
```

```
# Download the images from S3 bucket
```

```
# aws s3 cp s3://myterraformprojectbucket2023/project.webp
/var/www/html/project.png --acl public-read
```

```
# Create a simple HTML file with the portfolio content and display the images
```

```
cat <<EOF > /var/www/html/index.html
```

```
<!DOCTYPE html>
```

```
<html>
<head>
  <title>My Portfolio</title>
  <style>
    /* Add animation and styling for the text */
    @keyframes colorChange {
      0% { color: red; }
      50% { color: green; }
      100% { color: blue; }
    }
    h1 {
      animation: colorChange 2s infinite;
    }
  </style>
</head>
<body>
  <h1>Terraform Project Server 1</h1>
  <h2>Instance ID: <span style="color:green">${INSTANCE_ID}</span></h2>
  <p>deep ka Project</p>
</body>
</html>
EOF
```

# Start Apache and enable it on boot

```
systemctl start apache2
```



systemctl enable apache2#so basically these is your script which you need to enter into user\_data.sh

Let go forward and create instance 2 in subnet 2 and enter same code with basis changes

```
resource "aws_instance" "webserver2" {  
  
    ami          = "ami-08116b9957a259459"  
    instance_type = "t2.micro"  
    vpc_security_group_ids = aws_security_group.mysg.id  
    subnet_id      = aws_subnet.subnet2.id  
    user_data      = base64encode(file("user_data.sh1"))  
}
```

#in these create user\_data.ssh1

So till now your vim msin.tf file will be

```
terraform {  
    required_providers {  
        aws = {  
            source = "hashicorp/aws"  
            version = "~> 5.0"  
        }  
    }  
}
```

```
resource "aws_vpc" "myvpc" {
```

```
    cidr_block = "10.0.0.0/16"
}
```

```
resource "aws_subnet" "subnet1" {
    vpc_id    = aws_vpc.myvpc.id
    cidr_block = "10.0.0.0/24"
```

```
    tags = {
        Name = "Main"
    }
}
```

```
resource "aws_subnet" "subnet2" {
    vpc_id    = aws_vpc.myvpc.id
    cidr_block = "10.0.1.0/24"
```

```
    tags = {
        Name = "Main"
    }
}
```

```
resource "aws_internet_gateway" "igw" {
    vpc_id = aws_vpc.myvpc.id
```

```
    tags = {
```

```
    Name = "main"
  }
}
```

```
resource "aws_route_table" "route1" {
  vpc_id = aws_vpc.myvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }
}
```

```
tags = {
  Name = "example"
}
}
```

```
resource "aws_route_table_association" "rt1" {
  subnet_id    = aws_subnet.subnet1.id
  route_table_id = aws_route_table.route1.id
}
```

```
resource "aws_route_table_association" "rt2" {
  subnet_id    = aws_subnet.subnet2.id
  route_table_id = aws_route_table.route1.id
}
```

```
}
```

```
resource "aws_security_group" "mysg" {  
  name      = "mysg"  
  description = "Allow TLS inbound traffic and all outbound traffic"  
  vpc_id    = aws_vpc.myvpc.id
```

```
  ingress {  
    description = "http"  
    from_port   = 80  
    to_port     = 80  
    protocol    = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
}
```

```
  ingress {  
    description = "ssh"  
    from_port   = 22  
    to_port     = 22  
    protocol    = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
}
```

```
  egress {  
    description = "all traffic"
```

```
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

```
resource "aws_s3_bucket" "example" {
  bucket = "your bucket name"
}
```

```
resource "aws_instance" "webserver" {
  ami           = "ami-08116b9957a259459"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.mysg.id]
  subnet_id       = aws_subnet.subnet1.id
  user_data       = base64encode(file("user_data.sh"))
}
```

```
resource "aws_instance" "webserver2" {
  ami           = "ami-08116b9957a259459"
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.mysg.id]
  subnet_id       = aws_subnet.subnet2.id
  user_data       = base64encode(file("user_data.sh1"))
}
```

```
}
```

Make sure that user\_data.sh and user\_data.sh1 is in same path as your terraform is there