

PROJECT-1

Deploy Three Tier Architecture In AWS Using Terraform

Task-1:- Create a provider file and initialize the terraform folder by using “terraform init” command.

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 4.6"
    }
  }
}
provider "aws" {
  profile = "default"
  region = "us-east-1c"
  access_key = "AKIA5PWVUBL44F2DTFM2"
  secret_key = "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"
```

```
[ec2-user@ip-172-31-82-108 Terraform]$ terraform init
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.67.0

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.
[ec2-user@ip-172-31-82-108 Terraform]$ terraform fmt
[ec2-user@ip-172-31-82-108 Terraform]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-82-108 Terraform]$ terraform plan
No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
[ec2-user@ip-172-31-82-108 Terraform]$ terraform apply
No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-82-108 Terraform]$ ll
total 8
```

Task-2:- Create a file for aws vpc (vpc.tf) and write a script for vpc in the file by using vi command.

- Apply Terraform fmt, terraform validate, terraform plan and terraform apply commands after writing the script.

```
ec2-user@ip-172-31-82-108:~/Terraform
```

```
resource "aws_vpc" "myvpc" {
  cidr_block = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

~
~
~
```

```
ec2-user@ip-172-31-82-108:~/Terraform
[ec2-user@ip-172-31-82-108 Terraform]$ cat provider
cat: provider: No such file or directory.
[ec2-user@ip-172-31-82-108 Terraform]$ vi provider.tf
[ec2-user@ip-172-31-82-108 Terraform]$ cat provider.tf
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 4.6"
    }
  }
}

provider "aws" {
  profile = "default"
  region = "us-east-1c"
  access_key = "AKIASPMVUBL4FZDTPW2"
  secret_key = "M3OXCE7wdd4UW00F503n26lq8FwCffFznglx"
}
[ec2-user@ip-172-31-82-108 Terraform]$ terraform init
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.67.0

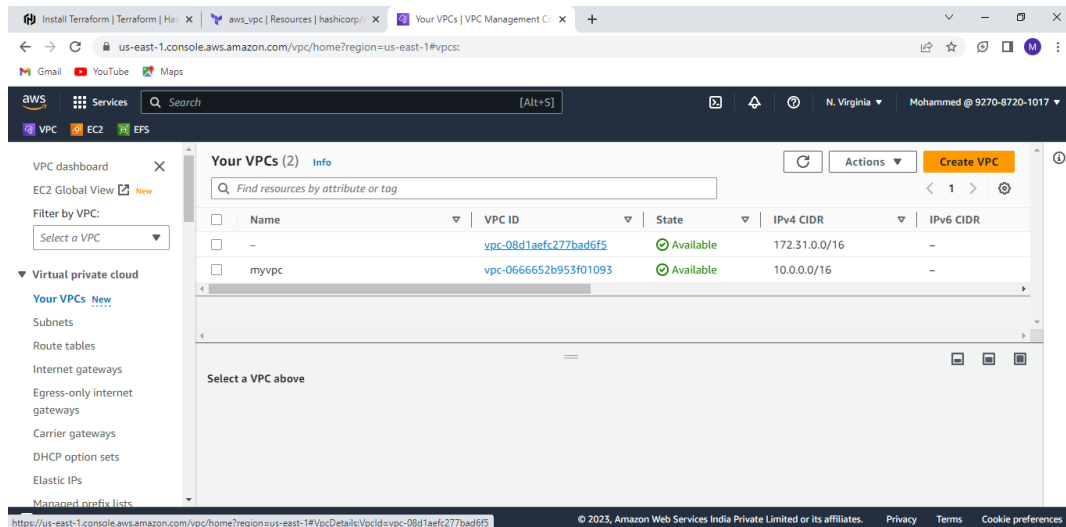
Terraform has been successfully initialized!

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,
run this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-82-108 Terraform]$ terraform cat
[ec2-user@ip-172-31-82-108 Terraform]$ terraform validate
Success! The configuration is valid.
[ec2-user@ip-172-31-82-108 Terraform]$ terraform plan
No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
[ec2-user@ip-172-31-82-108 Terraform]$ terraform apply
No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
```



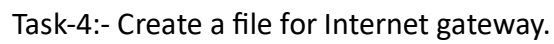
Task-3:- Create a file AWS subnet resource.

```
ec2-user@ip-172-31-82-108:~/Terraform
#creating 1st web subnet
resource "aws_subnet" "pub-sub-1" {
  vpc_id         = "aws_vpc.mmyvpc.id"
  cidr_block     = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = {
    Name = "pub-sub-1"
  }
}

#creating 2nd web subnet
resource "aws_subnet" "pub-sub-2" {
  vpc_id         = "aws_vpc.mmyvpc.id"
  cidr_block     = "10.0.2.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1b"
  tags = {
    Name = "pub-sub-2"
  }
}

#creating 3rd web subnet
resource "aws_subnet" "pvt-sub-1" {
  vpc_id         = "aws_vpc.mmyvpc.id"
  cidr_block     = "10.0.3.0/24"
  map_public_ip_on_launch = false
  availability_zone = "us-east-1a"
  tags = {
    Name = "pvt-sub-1"
  }
}

#creating 4th web subnet
resource "aws_subnet" "pvt-sub-2" {
  vpc_id         = "aws_vpc.mmyvpc.id"
  cidr_block     = "10.0.4.0/24"
  map_public_ip_on_launch = false
  availability_zone = "us-east-1b"
  tags = {
    Name = "pvt-sub-2"
  }
}
```

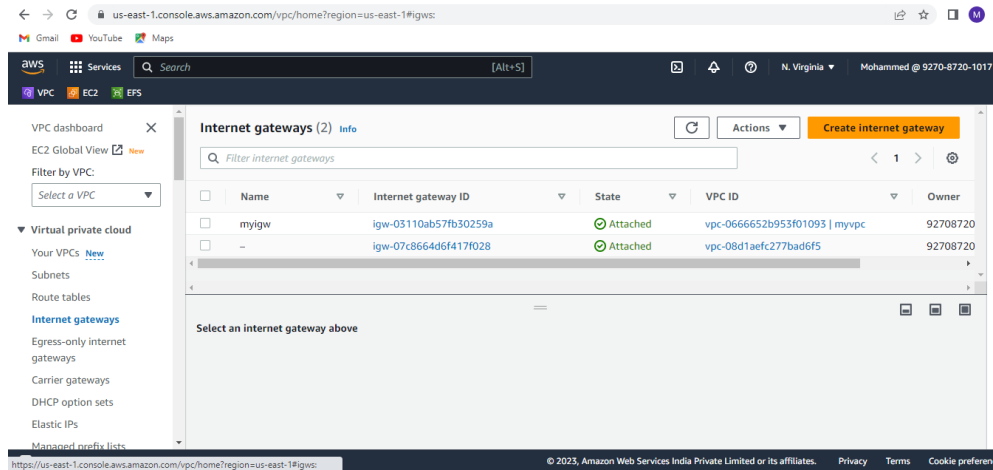


```
resource "aws_internet_gateway" "myigw" {
  vpc_id = aws_vpc.mmyvpc.id
  tags = {
    Name = "myigw"
  }
}
```

```

"internetgw.tf" 6L, 107B

```



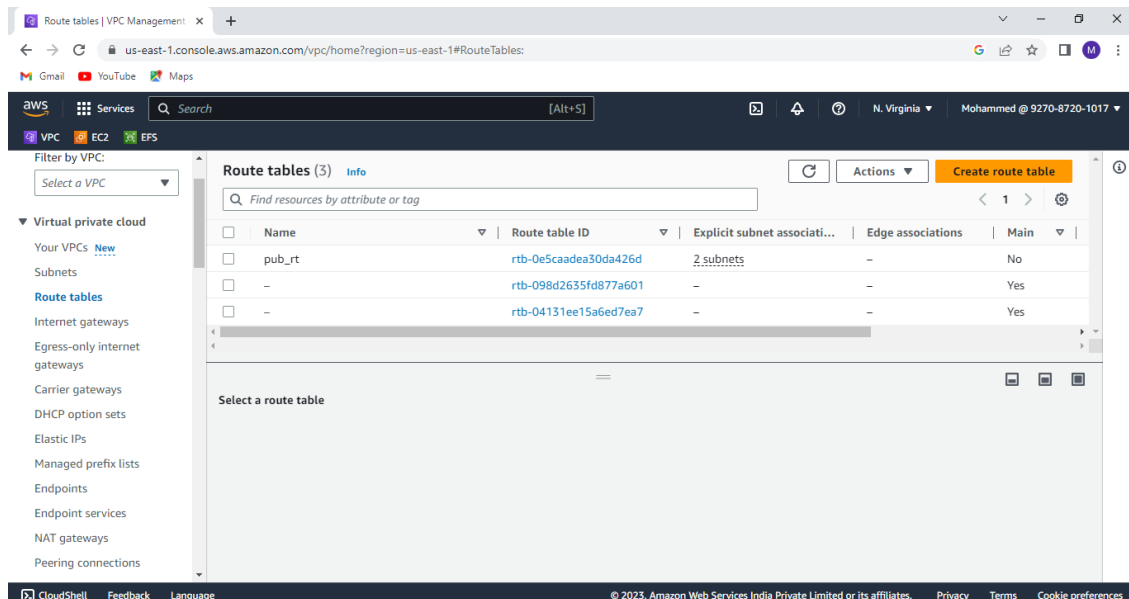
Task-5:- Create a file for Routetable.

```
ec2-user@ip-172-31-82-108:~/Terraform
resource "aws_route_table" "pub_rt" {
  vpc_id = aws_vpc.mmyvpc.id
  tags = {
    Name = "pub_rt"
  }
}

resource "aws_route" "route" {
  route_table_id = aws_route_table.pub_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id = aws_internet_gateway.myigw.id
}

#Associating route table
resource "aws_route_table_association" "route1" {
  subnet_id = aws_subnet.pub-sub-1.id
  route_table_id = aws_route_table.pub_rt.id
}

resource "aws_route_table_association" "route2" {
  subnet_id = aws_subnet.pub-sub-2.id
  route_table_id = aws_route_table.pub_rt.id
}
```



Task-6:- Create a file for AWS EC2 Instance Resource.

The screenshot shows the AWS Management Console interface for the 'us-east-1' region. The main content area displays the 'Instances (3)' page. The left sidebar contains the navigation menu, and the top bar shows the user's profile and account information.

Instances Table:

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	project1	i-051e71d66793a83ea	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c
<input type="checkbox"/>	ec2_1	i-046b84cf16cc4fb0c	Running	t2.micro	Initializing	No alarms	us-east-1a
<input type="checkbox"/>	ec2_2	i-0e2a9a3da84b29540	Running	t2.micro	Initializing	No alarms	us-east-1b

Left Sidebar Navigation:

- EC2 Dashboard
- EC2 Global View
- Events
- Limits
- Instances**
 - Instances
 - Instance Types
 - Launch Templates
 - Spot Requests
 - Savings Plans
 - Reserved Instances
 - Dedicated Hosts
 - Scheduled Instances
 - Capacity Reservations

Top Bar:

- Instances | EC2 Management Console
- terraform load balancer target
- us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:v=3;case=tags:true%5C.client:false;\$regex=tags:false%5C.client:false
- Gmail YouTube Maps
- Services Search [Alt+S]
- N. Virginia
- Mohammed @ 9270-8720-1017

Task-7:- Create a file for Security Group for the Frontend Tier.

```

ec2-user@ip-172-31-82-108:~/Terraform
resource "aws_security_group" "webosg" {
  vpc_id = aws_vpc.myyvpc.id

  #inbound rules
  ingress {
    from_port = 80
    to_port   = 80
    protocol  = "tcp"
    cidr_block = ["0.0.0.0/0"]
  }

  #HTTPS access from anywhere
  ingress {
    from_port = 443
    to_port   = 443
    protocol  = "tcp"
    cidr_block = ["0.0.0.0/0"]
  }

  #ssh access from anywhere
  ingress {
    from_port = 22
    to_port   = 22
    protocol  = "tcp"
    cidr_block = ["0.0.0.0/0"]
  }

  #outbound rules
  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_block = ["0.0.0.0/0"]
  }

  tags = {
    Name = "webosg"
  }
}

-- INSERT --

```

```

ec2-user@ip-172-31-82-108:~/Terraform
+ prefix_list_ids = []
+ protocol        = "tcp"
+ security_groups = []
+ self            = false
+ to_port         = 443
+ },
+ {
+   + cidr_blocks = [
+     + "0.0.0.0/0",
+   ]
+   + description = ""
+   + from_port   = 80
+   + ipv6_cidr_blocks = []
+   + prefix_list_ids = []
+   + protocol      = "tcp"
+   + security_groups = []
+   + self          = false
+   + to_port       = 80
+ },
+ name              = (known after apply)
+ name_prefix       = (known after apply)
+ owner_id          = (known after apply)
+ revoke_rules_on_delete = false
+ tags              = {
+   + "Name" = "webosg"
+ }
+ tags_all          = {
+   + "Name" = "webosg"
+ }
+ vpc_id            = "vpc-0666652b953f01093"
}

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_security_group.webosg: Creating...
aws_security_group.webosg: Creation complete after 2s [id=sg-08c5afc949a2795fb]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-82-108 Terraform]$

```

VPC Management Console | aws_security_group | Resources | +

us-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#SecurityGroups:

Services | Search | [Alt+S] | N. Virginia | Mohammed @ 9270-8720-1017

VPC | EC2 | EFS

Peering connections

Security

Network ACLs

Security groups

DNS firewall

Rule groups

Domain lists

Network Firewall

Firewalls

Firewall policies

Network Firewall rule groups

TLS inspection configurations New

Network Firewall resource groups New

Security Groups (5) Info

Filter security groups

<input type="checkbox"/>	Name	Security group ID	Security group name	VPC ID	Description
<input type="checkbox"/>	-	sg-0fbb80da9171522f4	launch-wizard-2	vpc-08d1aefc277bad6f5	launch-wizard-2 create...
<input type="checkbox"/>	-	sg-08e387e560703a2a4	launch-wizard-1	vpc-08d1aefc277bad6f5	launch-wizard-1 create...
<input type="checkbox"/>	-	sg-09ed31b8df72b8554	default	vpc-08d1aefc277bad6f5	default VPC security gr...
<input type="checkbox"/>	webosg	sg-08c5afc949a2795fb	terraform-202306071...	vpc-0666652b953f01093	Managed by Terraform

CloudShell | Feedback | Language

© 2023, Amazon Web Services India Private Limited or its affiliates. | Privacy | Terms | Cookie preferences

Task-8:- Create a file for Security Group for the Database Tier.

```
ec2-user@ip-172-31-82-108:~/Terraform
resource "aws_security_group" "databasesg" {
  vpc_id = aws_vpc.mmyvpc.id

  ingress {
    description = "TLS from vpc"
    from_port   = 3306
    to_port     = 3306
    protocol    = "tcp"
    security_groups = [aws_security_group.websg.id]
  }

  egress {
    from_port   = 32768
    to_port     = 65535
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "database_sg"
  }
}
```

```
ec2-user@ip-172-31-82-108:~/Terraform
Terraform will perform the following actions:

# aws_security_group.databasesg is tainted, so must be replaced
./ resource "aws_security_group" "databasesg" {
  ~ arn           = "arn:aws:ec2:us-east-1:927087201017:security-group/sg-073db4e2b0f73ae06" -> (known after apply)
  ~ egress        = [
    + {
      + cidr_blocks = [
        + "0.0.0.0/0",
      ]
      + description = ""
      + from_port   = 0
      + ipv6_cidr_blocks = []
      + prefix_list_ids = []
      + protocol    = "-1"
      + security_groups = []
      + self         = false
      + to_port      = 0
    },
  ]
  ~ id           = "sg-073db4e2b0f73ae06" -> (known after apply)
  ~ name          = "terraform-202306071139208527000000001" -> (known after apply)
  ~ name_prefix   = "terraform-" -> (known after apply)
  ~ owner_id      = "927087201017" -> (known after apply)
  ~ tags          = {
    "Name" = "database_sg"
  }
  # (5 unchanged attributes hidden)
}

Plan: 1 to add, 0 to change, 1 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_security_group.databasesg: Destroying... [id=sg-073db4e2b0f73ae06]
aws_security_group.databasesg: Destruction complete after 0s
aws_security_group.databasesg: Creating...
aws_security_group.databasesg: Creation complete after 2s [id=sg-04a7847ca876cb253]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
[ec2-user@ip-172-31-82-108 Terraform]$
```


Load balancers | EC2 Management console

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:

Services Search [Alt+S] N. Virginia Mohammed @ 9270-8720-1017

VPC EC2 EFS

Load balancers (1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Actions Create load balancer

Find resources by attribute or tag

<input type="checkbox"/>	Name	DNS name	State	VPC ID	Availability Zones
<input type="checkbox"/>	external-alb	external-alb-610570701.u...	Active	vpc-0666652b953f01093	2 Availability Zones

0 load balancers selected

Select a load balancer above.

Target groups | EC2 Management console

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:

Services Search [Alt+S] N. Virginia Mohammed @ 9270-8720-1017

VPC EC2 EFS

Target groups (1)

Actions Create target group

Find resources by attribute or tag

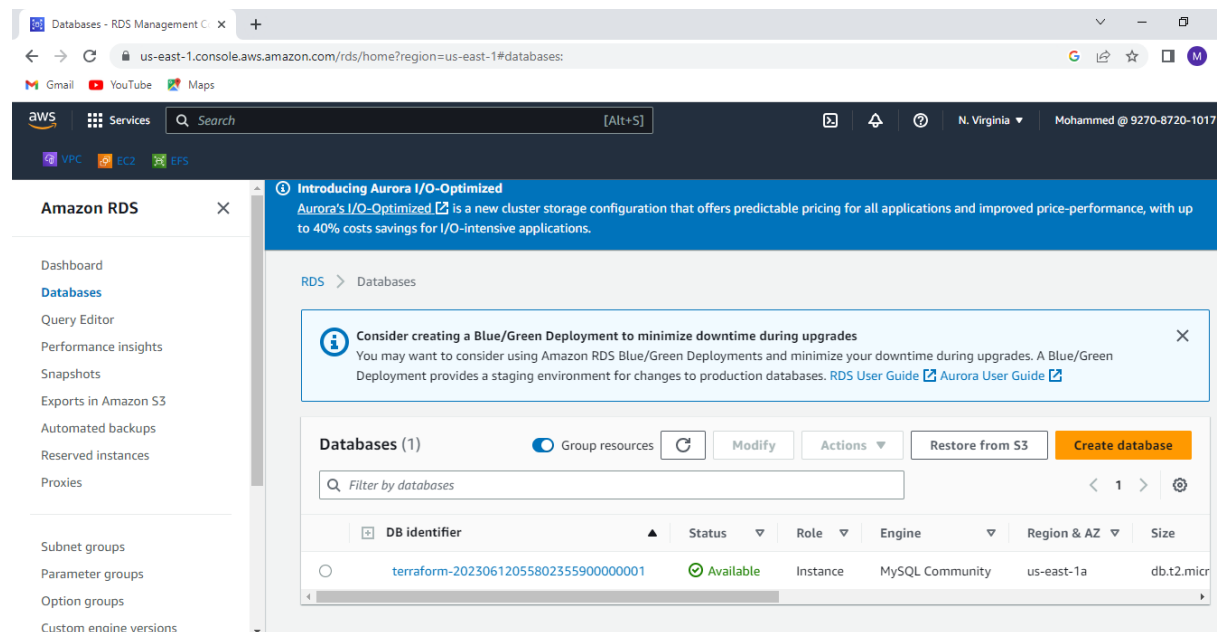
<input type="checkbox"/>	Name	ARN	Port	Protocol	Target type
<input type="checkbox"/>	targets	arn:aws:elasticloadbalanci...	80	HTTP	Instance

0 target groups selected

Select a target group above.

Task-10:- Create a file for RDS Instance.

```
ec2-user@ip-172-31-82-108:~/Terraform
#creating RDS instance
resource "aws_db_subnet_group" "rds1" {
  name       = "main"
  subnet_ids = [aws_subnet.pvt-sub-1.id, aws_subnet.pvt-sub-2.id]
  tags = {
    Name = " DB subnet group"
  }
}
resource "aws_db_instance" "db-instance" {
  allocated_storage    = 10
  db_subnet_group_name = aws_db_subnet_group.rds1.id
  engine               = "mysql"
  engine_version       = "5.7"
  instance_class       = "db.t2.micro"
  multi_az             = true
  username             = "username"
  password             = "password"
  skip_final_snapshot  = true
  vpc_security_group_ids = [aws_security_group.databasesg.id]
  tags = {
    Name = "mydb"
  }
}
```



Task-11:- Create a file for Outputs.

```
ec2-user@ip-172-31-82-108:~/Terraform
#getting the DNS of load balancer
output "lb_dns_name" {
  description = "The DNS name of the load balancer"
  value       = aws_lb.external-alb.dns_name
}

"outputs.tf" 61, 157B
```

```

[ec2-user@ip-172-31-82-108 ~]$ terraform apply
Changes to Outputs:
  + lb_dns_name = "external-alb-610570701.us-east-1.elb.amazonaws.com"

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

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.
[ec2-user@ip-172-31-82-108 ~]$ terraform apply
aws_vpc.mmvpc: Refreshing state... [id=vpc-0666652b953f01093]
aws_route_table.pub.rt: Refreshing state... [id=rtb-0e5caade30da426d]
aws_subnet.pvt-sub-2: Refreshing state... [id=subnet-0f6260cddf663b05]
aws_lb_listener.frontend: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:targetgroup/targets/80c71830147e0f40]
aws_subnet.pub-sub-2: Refreshing state... [id=subnet-012fdd1e0d70744]
aws_subnet.pub-sub-1: Refreshing state... [id=subnet-0752938f9f080825c]
aws_internet_gateway.mwigw: Refreshing state... [id=igw-03110ab57fb30259a]
aws_security_group.websg: Refreshing state... [id=sg-08c5f4cf949a2795fb]
aws_subnet.pvt-sub-1: Refreshing state... [id=subnet-0822d0bc3380d0d0]
aws_instance.ec2_1[0]: Refreshing state... [id=i-030e14ccdb5b44f5]
aws_route_table.association.route1: Refreshing state... [id=rtbassoc-02630bfcf448a64c0]
aws_route.route: Refreshing state... [id=rtb-0e5caade30da426d1808289494]
aws_security_group.database: Refreshing state... [id=sg-0478a7ca836c2953]
aws_lb_external-alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:loadbalancer/app/external-alb/96c60c5e952a2b]
aws_route_table.association.route2: Refreshing state... [id=rtbassoc-0fb0437962c15b32f]
aws_db_subnet_group.rds1: Refreshing state... [id=main]
aws_instance.ec2_0: Refreshing state... [id=i-06a902c209efcf3fc]
aws_db_instance.db-instance: Refreshing state... [id=terraform-20230612055802355900000001]
aws_lb_listener.frontend: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:listener/app/external-alb/96c60c5e952a2b/1d73b985fd403987]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:targetgroup/targets/80c71830147e0f40-20230612050834981900000004]
aws_lb_target_group_attachment.attachment2: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:targetgroup/targets/80c71830147e0f40-20230612050824960700000003]

Changes to Outputs:
  + lb_dns_name = "external-alb-610570701.us-east-1.elb.amazonaws.com"

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

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

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

Outputs:
lb_dns_name = "external-alb-610570701.us-east-1.elb.amazonaws.com"
[ec2-user@ip-172-31-82-108 ~]$ terraform apply

```

Task-12:- Create a file for user data with .sh extension.

```
ec2-user@ip-172-31-82-108: ~/Terraform
#!/bin/bash
sudo yum update -y
sudo yum install httpd -y
sudo yum install git -y
sudo systemctl start httpd
sudo systemctl enable httpd
echo "Hello World From $(hostname -f)" > /var/www/html/index.html
```

Task-13:- Verify the resource.

Terraform created below resources

- Vpc
- Public&private subnets
- Route tables
- Internet Gateway
- EC2 instances
- RDS instance
- Application Load Balancer
- Security Groups for web & RDS instance

TERRAFORM SCRIPT AUTOMATION WITH JENKIN

Step-1:- Launch a EC2 instance by giving jenkins port number 8080 in security group.

Step-2:- Connect the EC2 Instance.

Install Jenkins in EC2 instance by executing following commads.

- `sudo yum update -y`
- `sudo wget -O /etc/yum.repos.d/jenkins.repo \`
`https://pkg.jenkins.io/redhat-stable/jenkins.repo`
- `sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key`
- `sudo yum upgrade`
- `sudo amazon-linux-extras install java-openjdk11 -y`
- `sudo yum install jenkins -y`
- `sudo systemctl enable Jenkins`
- `sudo systemctl start Jenkins`

Step-3:- Connect to `http://<your_server_public_DNS>:8080` from your browser. You will be able to access Jenkins through its management interface

Step-4:- Enter the password found in `/var/lib/jenkins/secrets/initialAdminPassword`.

Use the following command to display this password:

```
$ sudo cat /var/lib/Jenkins/secrets/InitialAdminPassword
```

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Continue

Step-5:- The Jenkins installation script directs you to the Customize Jenkins page. Click Install suggested plugins.

- Once the installation is complete, the Create First Admin User will open. Enter your information, and then select Save and Continue.

Getting Started

Create First Admin User

Username:

admin

Password:

.....

Confirm password:

.....

Full name:

E-mail address:

Jenkins 2.263.1

[Skip and continue as admin](#)

Save and Continue

Step-6:- Once the Jenkins profile setup is completed create a new job by clicking on new item. Select free style project.


Enter an item name


Hello World


1


Required field


2


**Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any tool used for something other than software build.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple builds, etc.

**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, separate namespace, so you can have multiple things of the same name as long as they are in different namespaces.

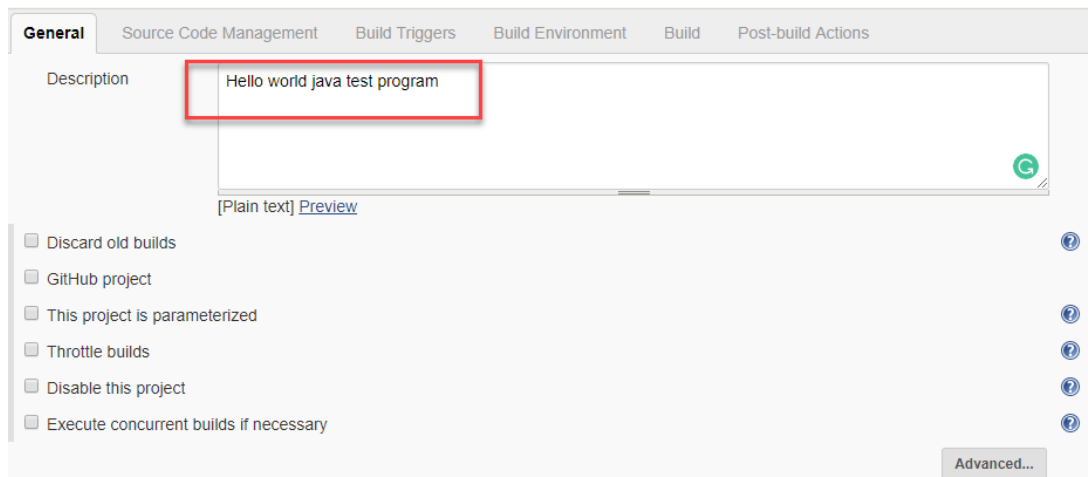
**GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

3

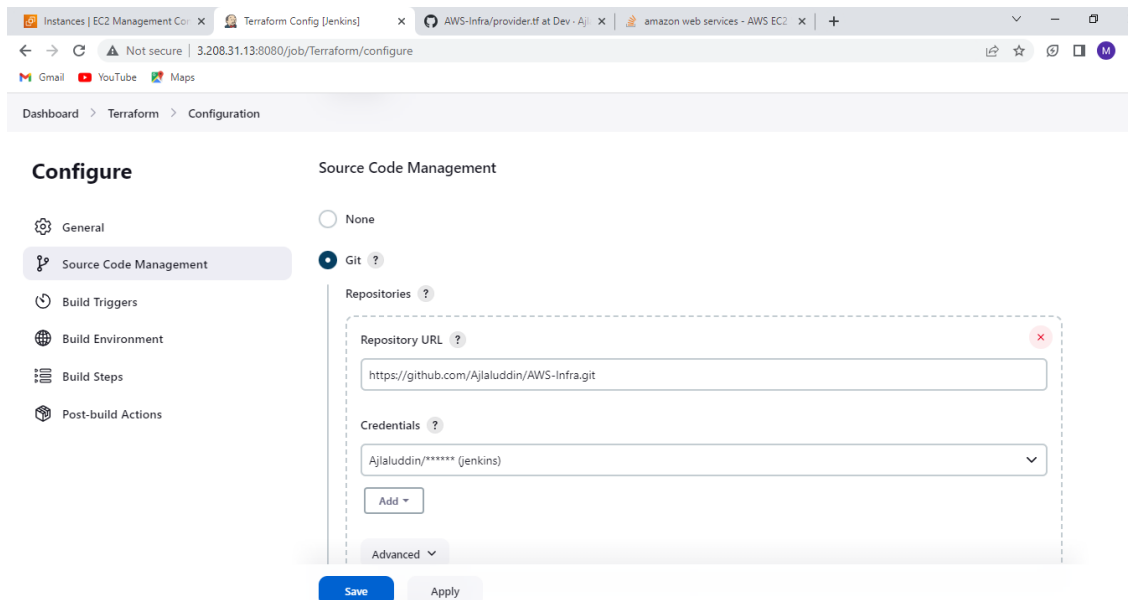
OK

Step-7:- Enter the details of the project you want to test.



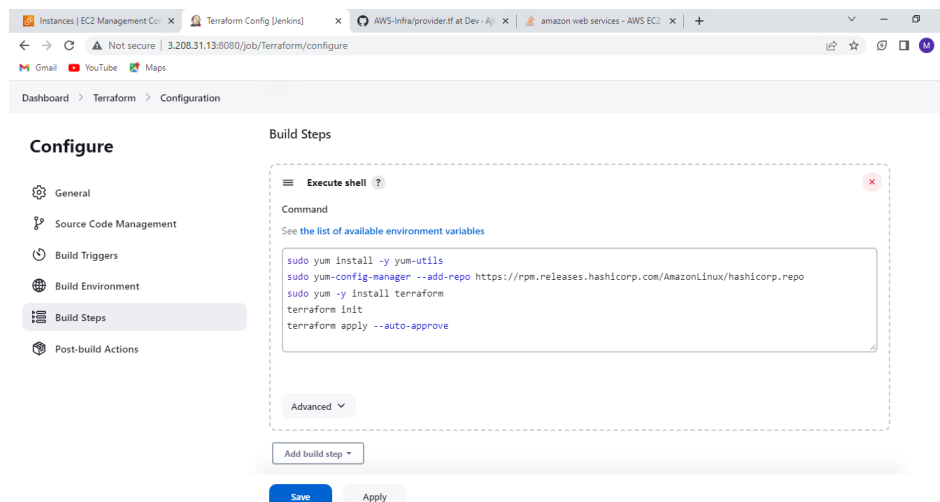
The screenshot shows the Jenkins configuration page for a new job, with the 'General' tab selected. The 'Description' field is highlighted with a red box and contains the text 'Hello world java test program'. Below the description field, there are several checkboxes: 'Discard old builds', 'GitHub project', 'This project is parameterized', 'Throttle builds', 'Disable this project', and 'Execute concurrent builds if necessary'. An 'Advanced...' button is located at the bottom right of the configuration area.

Step-8:- Under Source Code Management, Enter your repository URL.



The screenshot shows the Jenkins configuration page for a new job, with the 'Source Code Management' tab selected. The 'Repository URL' field is highlighted with a red box and contains the text 'https://github.com/AjJaluddin/AWS-Infra.git'. The 'Credentials' dropdown menu is set to 'AjJaluddin/***** (jenkins)'. There are 'Add', 'Save', and 'Apply' buttons at the bottom of the configuration area.

Step-9:- Under the build section select “Add build step” and click on “Execute Shell” and add the commands which you want to execute during the build process.



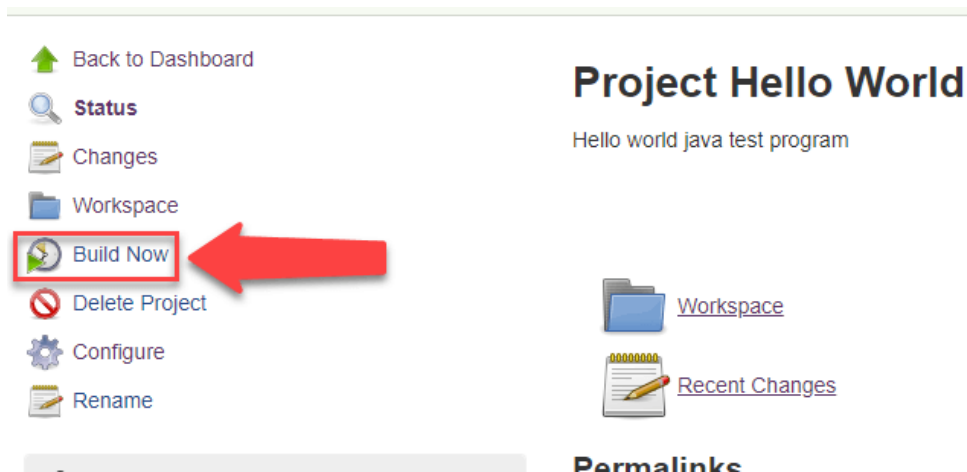
The screenshot shows the Jenkins configuration page for a new job, with the 'Build Steps' tab selected. The 'Execute shell' step is highlighted with a red box. The 'Command' field contains the following text: 'sudo yum install -y yum-utils', 'sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo', 'sudo yum -y install terraform', 'terraform init', and 'terraform apply --auto-approve'. There are 'Add build step', 'Save', and 'Apply' buttons at the bottom of the configuration area.

Step-10:- Click apply and save the project.

Step-11:- Build source code.

Now, in the main screen, Click the **Build Now** button on the left-hand side to build the source code.

After clicking on **Build now**, you can see the status of the build you run under **Build History**.



Step-12:- Click on **console output** to see the status of the build you run.

