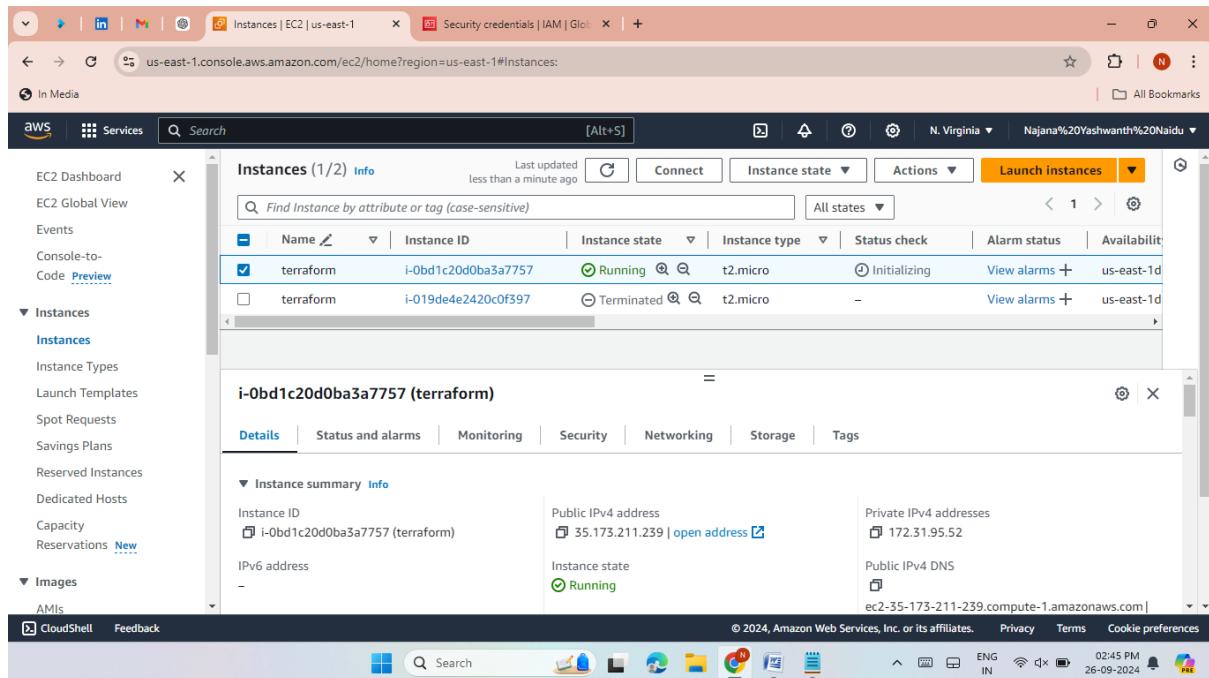


Project

DEPLOY THREE-TIRE ARCHITECTURE IN AWS USING TERRAFORM

- Terraform is an infrastructure-as-code software tool created by HashiCorp. Users define and provide data center infrastructure using a declarative configuration language known as HashiCorp Configuration Language (HCL).
- Create a EC2 instance then install terraform.



- Provide access keys and secret key.
- A software element known as a Terraform provider enables Terraform to communicate with a particular infrastructure platform. The resource kinds and data sources that Terraform can handle for that platform must be implemented by providers. Cloud platforms, data centres, network devices, databases, and other resources inside the target infrastructure or service can all be defined, configured, and managed by Terraform providers.

```
ec2-user@ip-172-31-95-52:~$ provider "aws" {
region = "us-east-1"
access_key = ""
secret_key = ""
}
~
```

The screenshot shows a Windows desktop environment with a terminal window open. The terminal window has a dark background and displays a single line of code: `provider "aws" {region = "us-east-1" access_key = "" secret_key = ""}`. Below the terminal window is a taskbar with various icons for applications like File Explorer, Google Chrome, and Microsoft Word. The system tray shows the date as 26-09-2024 and the time as 02:51 PM.

- Enter the terraform init command to initialized the terraform
- Terraform init command initializes a Terraform working directory by downloading and installing any required plugins and dependencies. It should be run before any other Terraform commands.

```
Verifying : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64
Verifying : git-2.40.1-1.amzn2.0.3.x86_64
Verifying : terraform-1.9.6-1.x86_64
Verifying : libperl-Error-0.17020-2.amzn2.noarch
Verifying : git-core-2.40.1-1.amzn2.0.3.x86_64
Verifying : git-core-doc-2.40.1-1.amzn2.0.3.noarch
Verifying : perl-Git-2.40.1-1.amzn2.0.3.noarch

Installed:
  terraform.x86_64 0:1.9.6-1

Dependency Installed:
  git.x86_64 0:2.40.1-1.amzn2.0.3          git-core.x86_64 0:2.40.1-1.amzn2.0.3
  perl-Error.noarch 1:0.17020-2.amzn2          perl-Git.noarch 0:2.40.1-1.amzn2.0.3

[ec2-user@ip-172-31-95-52 ~]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.68.0...
- Installed hashicorp/aws v5.68.0 (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!

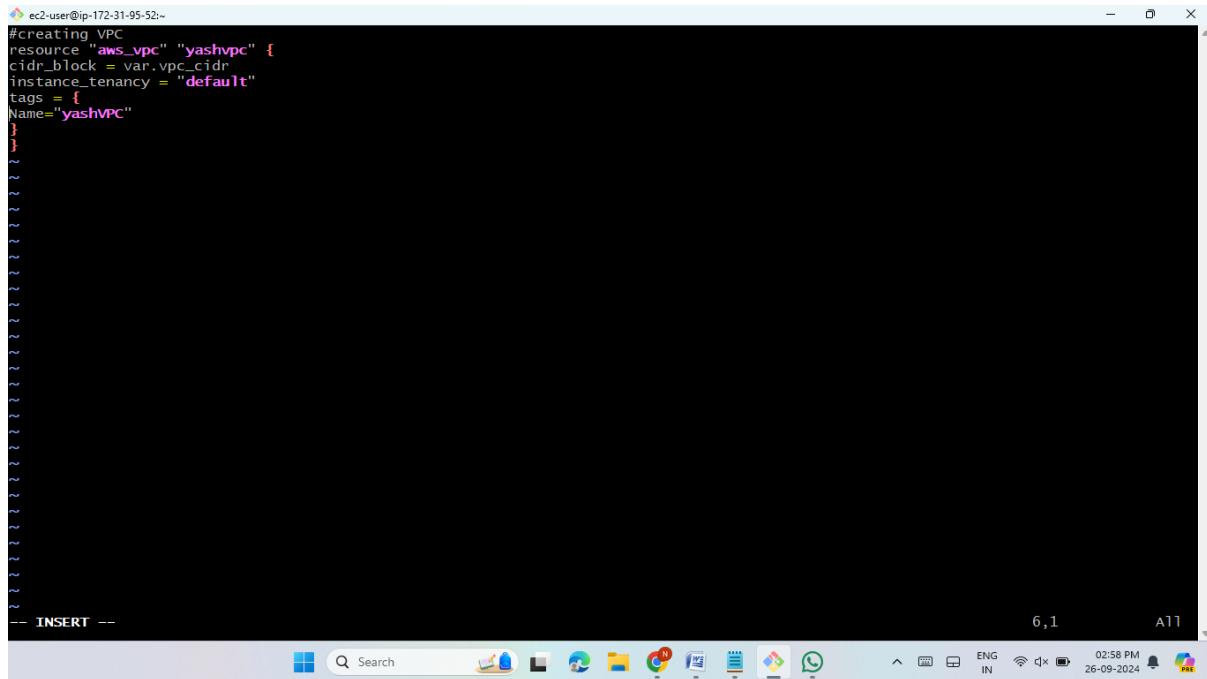
You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

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.
[ec2-user@ip-172-31-95-52 ~]$ ls
provider.tf
[ec2-user@ip-172-31-95-52 ~]$
```

The screenshot shows a Windows desktop environment with a terminal window displaying the output of the `terraform init` command. The terminal window shows the verification of provider packages, the installation of dependencies, and the creation of a lock file. The terminal window has a dark background and displays green text for success messages. Below the terminal window is a taskbar with various icons for applications like File Explorer, Google Chrome, and Microsoft Word. The system tray shows the date as 26-09-2024 and the time as 02:50 PM.

Create a file for the VPC

- Here Creating `vpc.tf` file
- `.tf` file in Terraform is a text file that contains the configuration code for Terraform to provision infrastructure. These files are written in **HashiCorp Configuration Language (HCL)**



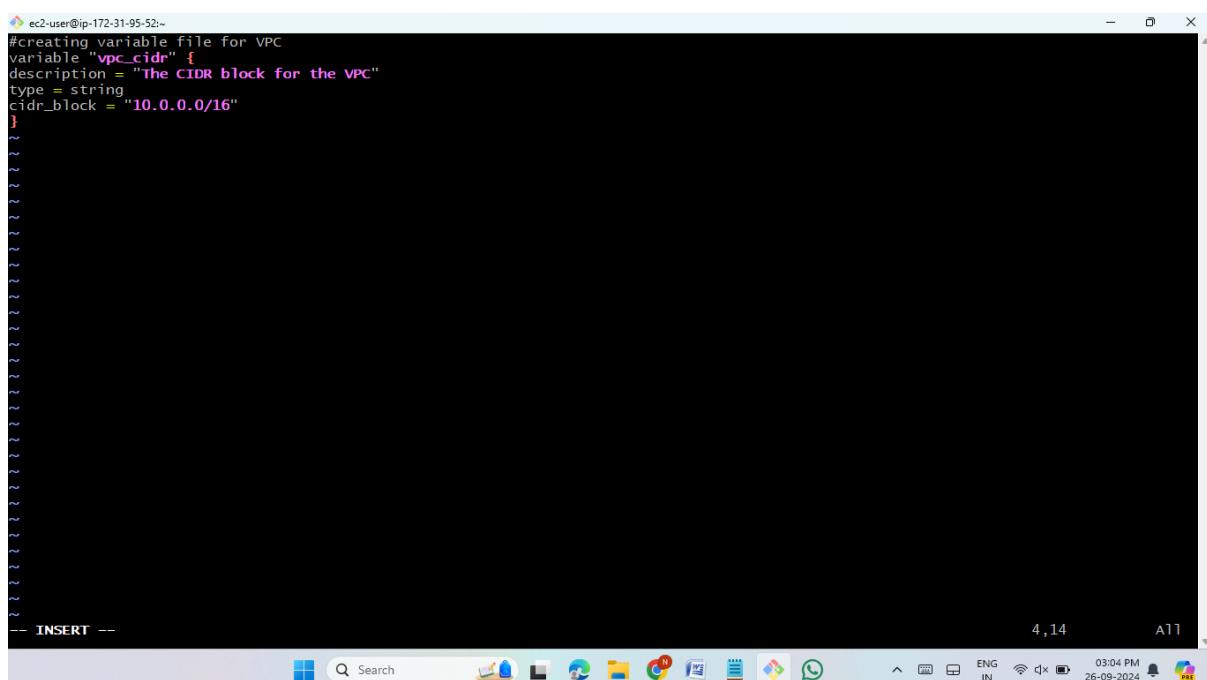
```
# creating VPC
resource "aws_vpc" "yashvpc" {
cidr_block = var.vpc_cidr
instance_tenancy = "default"
tags = {
Name="yashVPC"
}
}
```

-- INSERT --

6,1 All

Windows taskbar icons: Search, File Explorer, Edge, Google Chrome, File Manager, Paint 3D, Task View, WhatsApp, File History, File Explorer, File Manager, Edge, Google Chrome, File Manager, Paint 3D, Task View, WhatsApp, File History, ENG IN, 02:58 PM, 26-09-2024

- Here the creating variable file for the VPC.
- A **variable file** in Terraform is a file used to define and assign values to variables in your Terraform configuration.



```
#creating variable_file for VPC
variable "vpc_cidr" {
description = "The CIDR block for the VPC"
type = string
cidr_block = "10.0.0.0/16"
}
```

-- INSERT --

4,14 All

Windows taskbar icons: Search, File Explorer, Edge, Google Chrome, File Manager, Paint 3D, Task View, WhatsApp, File History, File Explorer, File Manager, Edge, Google Chrome, File Manager, Paint 3D, Task View, WhatsApp, File History, ENG IN, 03:04 PM, 26-09-2024

- Type the following command.
- **terraform fmt** (used to automatically format Terraform configuration files)
- **terraform validate** (used to check whether a Terraform configuration is syntactically valid and internally consistent. It verifies that the configuration files are structured correctly and that all resource definitions and configurations comply with Terraform's internal logic, but it doesn't actually interact with infrastructure or execute any changes.)
- **terraform plan** (used to create an execution plan, showing what actions Terraform will take to reach the desired state of your infrastructure based on your configuration files. It does not make any changes to the infrastructure but rather provides a preview of what Terraform will do when you apply the changes.)
- **terraform apply** (used to apply the changes required to reach the desired state of the configuration. It executes the planned actions, such as creating, modifying, or destroying infrastructure resources, based on the configuration defined in .tf files and the current state of the infrastructure.)
- We can see the vpc actions.

```

ec2-user@ip-172-31-95-52:~$ terraform plan
symbols:
+ create

Terraform will perform the following actions:

# aws_vpc.yashvpc will be created
+ resource "aws_vpc" "yashvpc" {
    + 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)"
    + ipv6_cidr_block = "(known after apply)"
    + ipv6_cidr_block_network_border_group = "(known after apply)"
    + main_route_table_id = "(known after apply)"
    + owner_id = "(known after apply)"
    + tags {
        + "Name" = "yashVPC"
    }
    + tags_all {
        + "Name" = "yashVPC"
    }
}

Plan: 1 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.

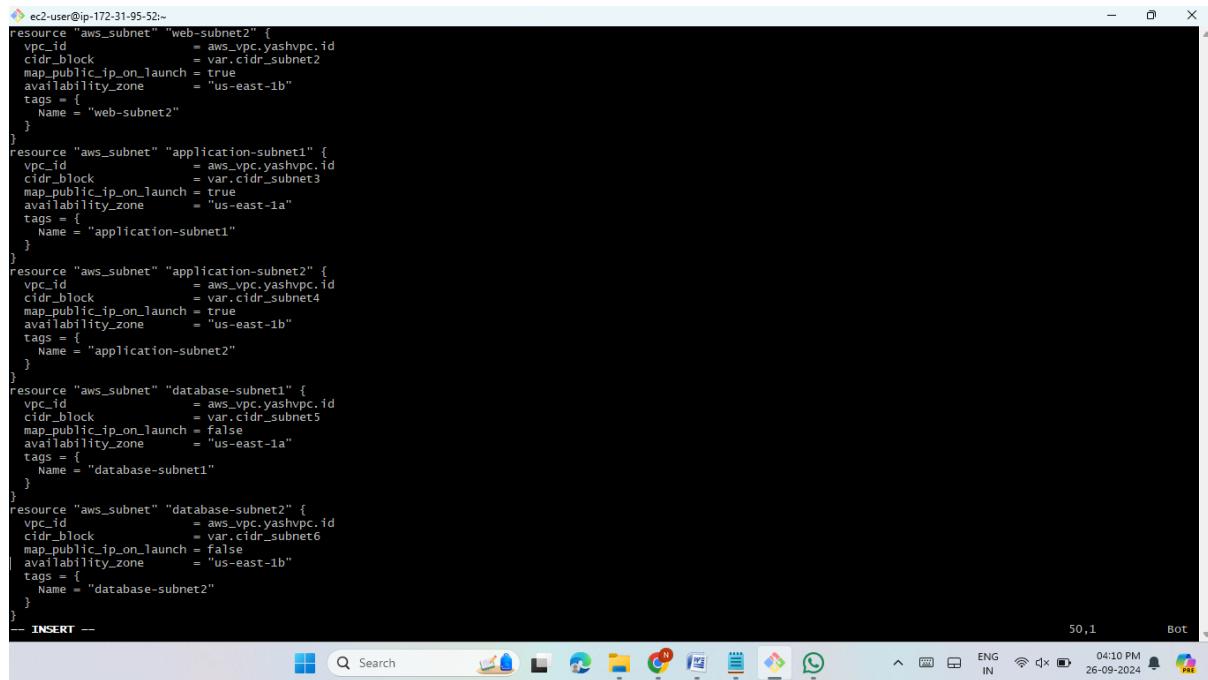
[ec2-user@ip-172-31-95-52 ~]$ 

```



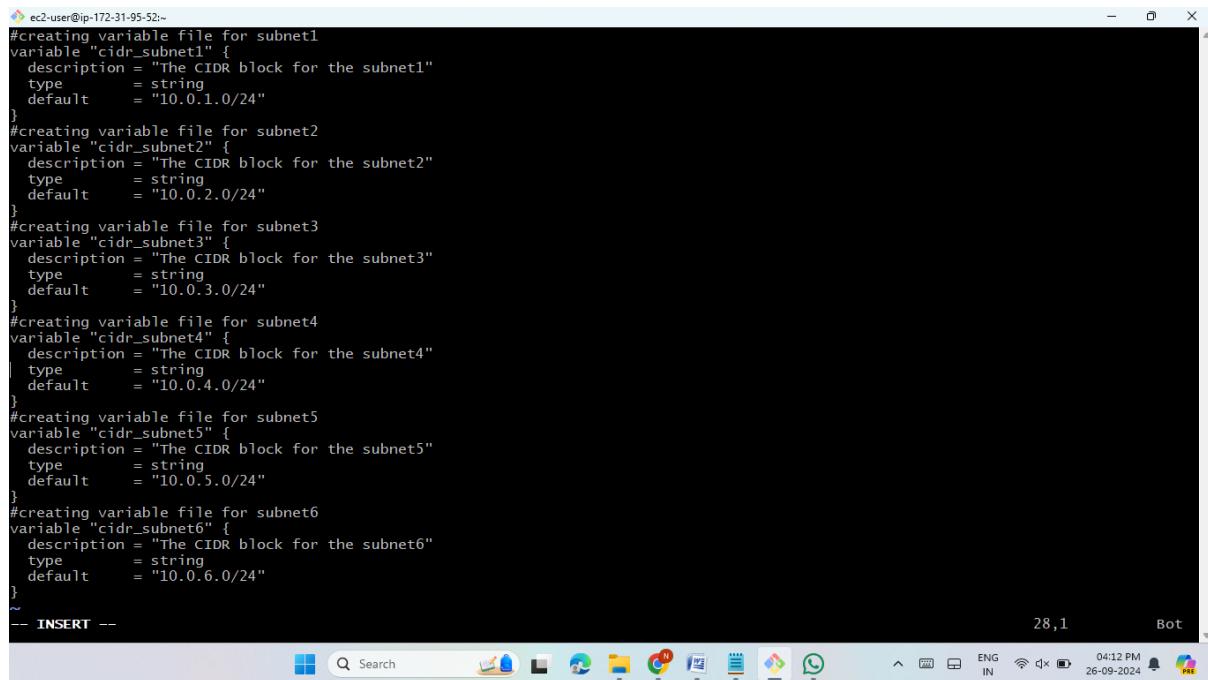
Create a file for the subnet

- In this project, I will create total 6 subnets for the front-end tier and back mixture of 4public & 2private subnet.
- Here Creating subnet.tf file



```
ec2-user@ip-172-31-95-52:~$ 
resource "aws_subnet" "web-subnet2" {
  vpc_id          = aws_vpc.yashvpc.id
  cidr_block      = var.cidr_subnet2
  map_public_ip_on_launch = true
  availability_zone = "us-east-1b"
  tags = {
    Name = "web-subnet2"
  }
}
resource "aws_subnet" "application-subnet1" {
  vpc_id          = aws_vpc.yashvpc.id
  cidr_block      = var.cidr_subnet3
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = {
    Name = "application-subnet1"
  }
}
resource "aws_subnet" "application-subnet2" {
  vpc_id          = aws_vpc.yashvpc.id
  cidr_block      = var.cidr_subnet4
  map_public_ip_on_launch = true
  availability_zone = "us-east-1b"
  tags = {
    Name = "application-subnet2"
  }
}
resource "aws_subnet" "database-subnet1" {
  vpc_id          = aws_vpc.yashvpc.id
  cidr_block      = var.cidr_subnet5
  map_public_ip_on_launch = false
  availability_zone = "us-east-1a"
  tags = {
    Name = "database-subnet1"
  }
}
resource "aws_subnet" "database-subnet2" {
  vpc_id          = aws_vpc.yashvpc.id
  cidr_block      = var.cidr_subnet6
  map_public_ip_on_launch = false
  availability_zone = "us-east-1b"
  tags = {
    Name = "database-subnet2"
  }
}
-- INSERT --
```

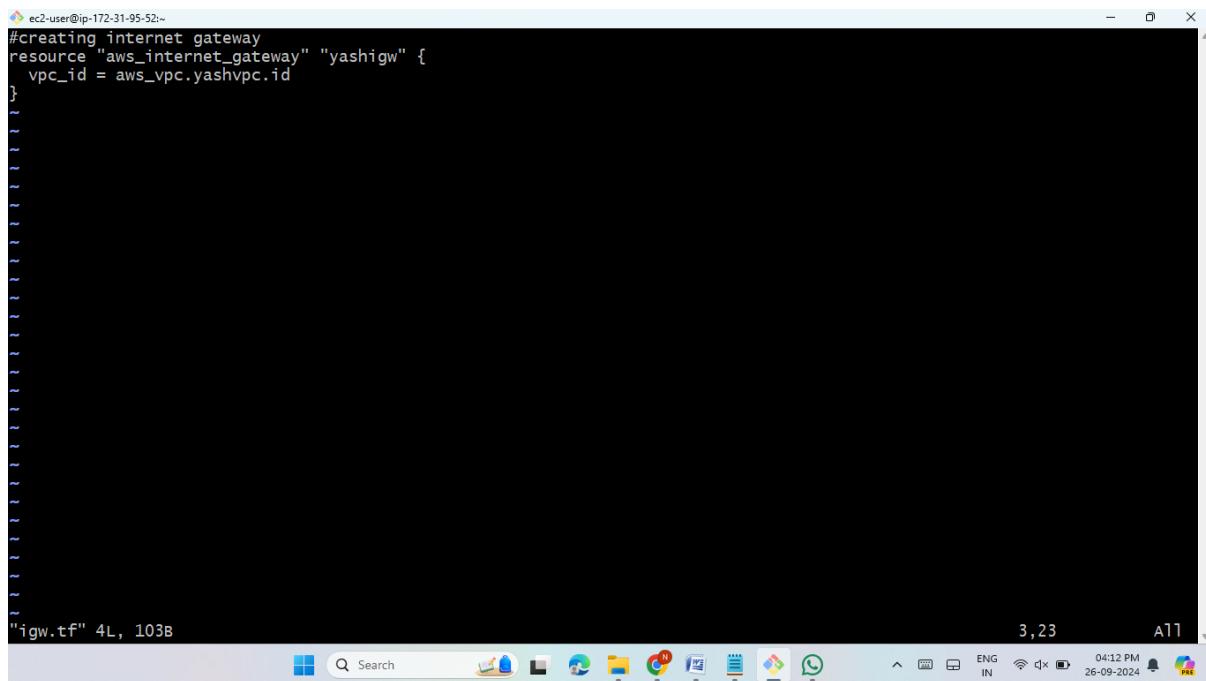
- Here Creating variable.tf file for 6 subnets



```
ec2-user@ip-172-31-95-52:~$ 
#creating variable file for subnet1
variable "cidr_subnet1" {
  description = "The CIDR block for the subnet1"
  type        = string
  default     = "10.0.1.0/24"
}
#creating variable file for subnet2
variable "cidr_subnet2" {
  description = "The CIDR block for the subnet2"
  type        = string
  default     = "10.0.2.0/24"
}
#creating variable file for subnet3
variable "cidr_subnet3" {
  description = "The CIDR block for the subnet3"
  type        = string
  default     = "10.0.3.0/24"
}
#creating variable file for subnet4
variable "cidr_subnet4" {
  description = "The CIDR block for the subnet4"
  type        = string
  default     = "10.0.4.0/24"
}
#creating variable file for subnet5
variable "cidr_subnet5" {
  description = "The CIDR block for the subnet5"
  type        = string
  default     = "10.0.5.0/24"
}
#creating variable file for subnet6
variable "cidr_subnet6" {
  description = "The CIDR block for the subnet6"
  type        = string
  default     = "10.0.6.0/24"
}
-- INSERT --
```

Create a file for Internet gateway

- Here Creating igw.tf file.

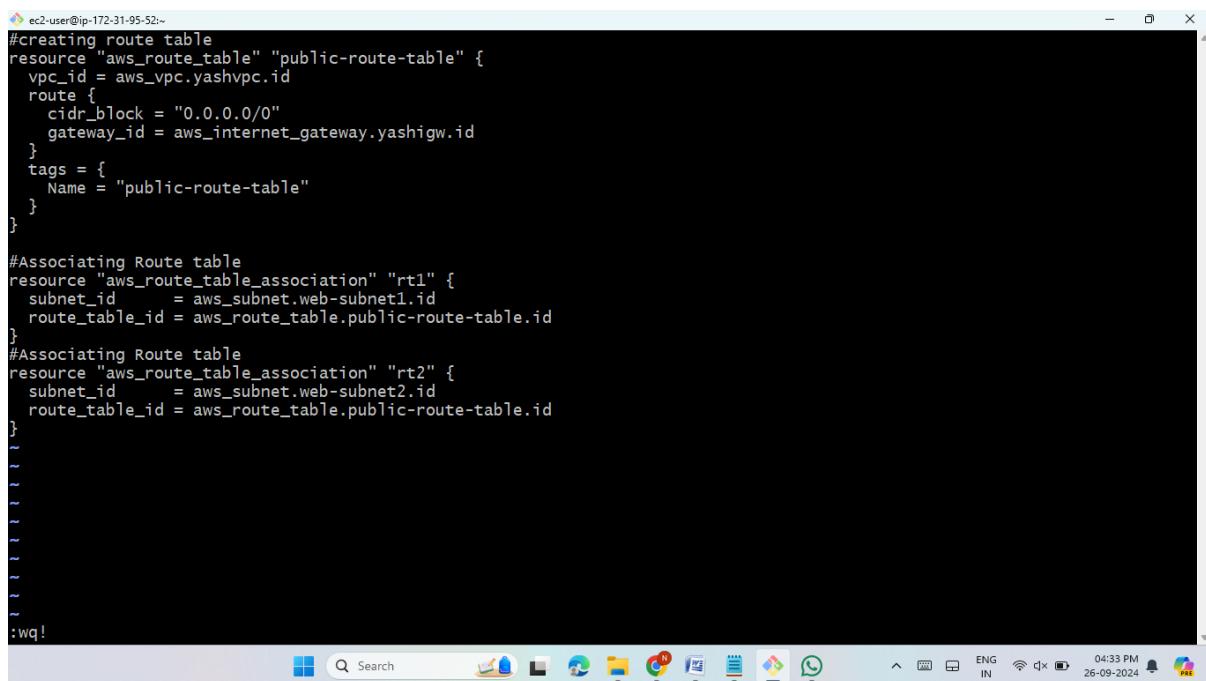


```
ec2-user@ip-172-31-95-52:~$ creating internet gateway
resource "aws_internet_gateway" "yashigw" {
  vpc_id = aws_vpc.yashvpc.id
}

"igw.tf" 4L, 103B
```

Create a file for the Route table

- Here Creating public- route-table.tf file.



```
ec2-user@ip-172-31-95-52:~$ creating route table
resource "aws_route_table" "public-route-table" {
  vpc_id = aws_vpc.yashvpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.yashigw.id
  }
  tags = [
    Name = "public-route-table"
  ]
}

#Associating Route table
resource "aws_route_table_association" "rt1" {
  subnet_id      = aws_subnet.web-subnet1.id
  route_table_id = aws_route_table.public-route-table.id
}
#Associating Route table
resource "aws_route_table_association" "rt2" {
  subnet_id      = aws_subnet.web-subnet2.id
  route_table_id = aws_route_table.public-route-table.id
}

:wq!
```

Create a file for security group

- Here Creating securitygroup1.tf file for instance1 and instance2 to launch the websites.

```
ec2-user@ip-172-31-95-52:~$ #creating security group
resource "aws_security_group" "securitygroup1" {
  vpc_id = aws_vpc.yashvpc.id

  #inbound Rules
  ingress {
    from_port   = "80"
    to_port     = "80"
    protocol   = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 443
    to_port     = 443
    protocol   = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 22
    to_port     = 22
    protocol   = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol   = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "securitygroup1"
  }
}
-- INSERT --
```

The screenshot shows a terminal window on a Windows operating system. The command entered is to create an AWS Security Group named 'securitygroup1'. It specifies the VPC ID and defines three inbound rules: one for port 80 (HTTP), one for port 443 (HTTPS), and one for port 22 (SSH). An egress rule is also defined to allow all outgoing traffic. The terminal window includes a status bar at the bottom showing network connectivity, battery level, and the date/time (26-09-2024, 04:39 PM).

Create a file for security group for the database

- Create database-securitygroup.tf file for the RDS Database.

```
ec2-user@ip-172-31-95-52:~$ resource "aws_security_group" "database-securitygroup" {
  vpc_id = aws_vpc.yashvpc.id
  ingress {
    from_port   = 3306
    to_port     = 3306
    protocol   = "tcp"
    security_groups = [aws_security_group.securitygroup1.id]
  }
  ingress {
    from_port   = 22
    to_port     = 22
    protocol   = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port   = 32768
    to_port     = 65535
    protocol   = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = " database-securitygroup"
  }
}
-- INSERT --
```

The screenshot shows a terminal window on a Windows operating system. The command entered is to create an AWS Security Group named 'database-securitygroup'. It specifies the VPC ID and defines two inbound rules: one for port 3306 (MySQL) and one for port 22 (SSH). An egress rule is defined to allow all outgoing traffic. The terminal window includes a status bar at the bottom showing network connectivity, battery level, and the date/time (26-09-2024, 05:47 PM).

Create a file for the EC2 instances

- Here Creating instance1.tf file for hosting website.

```
ec2-user@ip-172-31-05-52:~$ Creating 1st EC2 instance
resource "aws_instance" "instance1" {
  ami           = "ami-0e54eba7c51c234f6"
  instance_type = "t2.micro"
  key_name      = "yash"
  subnet_id     = aws_subnet.web-subnet1.id
  vpc_security_group_ids = [aws_security_group.securitygroup1.id]
  associate_public_ip_address = true
  user_data     = file("data.sh")

  tags = {
    Name = "instance1"
  }
}
:q!
```

- Here Creating instance2.tf file for hosting website.

```
ec2-user@ip-172-31-05-52:~$ Creating 2nd EC2 instance
resource "aws_instance" "instance2" {
  ami           = "ami-0e54eba7c51c234f6"
  instance_type = "t2.micro"
  key_name      = "yash"
  subnet_id     = aws_subnet.web-subnet2.id
  vpc_security_group_ids = [aws_security_group.securitygroup1.id]
  associate_public_ip_address = true
  user_data     = file("data1.sh")

  tags = {
    Name = "instance2"
  }
}
:q!
```

Create a file for application load balancer

- Here creating application load balancer and target group file.

```
# ec2-user@ip-172-31-95-52:~  
# Creating an External Application Load Balancer (ALB)  
resource "aws_lb" "external-alb" {  
    name          = "external-LB"  
    internal      = false  
    load_balancer_type = "application"  
    security_groups = [aws_security_group.securitygroup1.id]  
    subnets        = [aws_subnet.web-subnet1.id, aws_subnet.web-subnet2.id]  
  
    tags = {  
        Name = "external-alb"  
    }  
}  
  
# Creating a Target Group for the ALB  
resource "aws_lb_target_group" "target-elb" {  
    name          = "ALB-TG"  
    port          = 80  
    protocol     = "HTTP"  
    vpc_id        = aws_vpc.yashvpc.id  
  
    tags = {  
        Name = "target-elb"  
    }  
}  
  
# Attaching the EC2 instance to the Target Group (Instance 1)  
resource "aws_lb_target_group_attachment" "attachment1" {  
    target_group_arn = aws_lb_target_group.target-elb.arn  
    target_id       = aws_instance.instance1.id  
    port            = 80  
-- INSERT --  
5,37 Top
```

```
] }  
  
# Attaching the EC2 instance to the Target Group (Instance 2)  
resource "aws_lb_target_group_attachment" "attachment2" {  
    target_group_arn = aws_lb_target_group.target-elb.arn  
    target_id       = aws_instance.instance2.id  
    port            = 80  
  
    depends_on = [  
        aws_instance.instance2  
    ]  
}  
  
# Creating a Listener for the ALB  
resource "aws_lb_listener" "target-elb" {  
    load_balancer_arn = aws_lb.external-alb.arn  
    port             = 80  
    protocol         = "HTTP"  
  
    default_action {  
        type      = "forward"  
        target_group_arn = aws_lb_target_group.target-elb.arn  
    }  
  
    tags = {  
        Name = "target-listener"  
    }  
}  
-- INSERT --  
63,1 Bot
```

Create a file for Auto Scaling Group

- Here Creating auto scaling group.

```
ec2-user@ip-172-31-95-52:~$ resource "aws_launch_template" "web-launch-template" {  
    name_prefix = "web-launch-template"  
    image_id = "ami-0e54eba/c51c234f6"  
    instance_type = "t2.micro"  
    key_name = "yash"  
    vpc_security_group_ids = [aws_security_group.securitygroup1.id]  
    tag_specifications {  
        resource_type = "instance"  
        tags = {  
            Name = "webLaunchInstance"  
        }  
    }  
}  
resource "aws_autoscaling_group" "web-asg" {  
    desired_capacity = 3  
    max_size = 6  
    min_size = 1  
    launch_template {  
        id = aws_launch_template.web-launch-template.id  
        version = "$Latest"  
    }  
    vpc_zone_identifier = [aws_subnet.web-subnet1.id, aws_subnet.web-subnet2.id]  
    health_check_type = "EC2"  
    health_check_grace_period = 300  
    tag {  
        key = "Name"  
        value = "Webservernstance"  
        propagate_at_launch = true  
    }  
}  
resource "aws_autoscaling_attachment" "asg-attachments" {  
    autoscaling_group_name = aws_autoscaling_group.web-asg.name  
    lb_target_group_arn = aws_lb_target_group.target-elb.arn  
}  
-- INSERT --
```

Create a file for the RDS instance

- Here Creating RDS file with using MYSQL Database.

```
# Creating a DB Subnet Group  
resource "aws_db_subnet_group" "mydb" {  
    name = "main"  
    subnet_ids = [aws_subnet.database-subnet1.id, aws_subnet.database-subnet2.id]  
    tags = {  
        Name = "my DB subnet group"  
    }  
}  
  
# Creating an RDS Instance  
resource "aws_db_instance" "default" {  
    allocated_storage = 10  
    db_subnet_group_name = aws_db_subnet_group.mydb.id  
    engine = "mysql"  
    engine_version = "8.0.39"  
    instance_class = "db.t3.micro"  
    multi_az = true  
    username = "admin"  
    password = "Najana999"  
    skip_final_snapshot = true  
    vpc_security_group_ids = [aws_security_group.database-securitygroup.id]  
    tags = {  
        Name = "mydb"  
    }  
}  
-- INSERT --
```

Create a file for output

- Creating output.tf file for host websites using DNS.

Create a file for user data

- Here Creating data.sh file for ecomm.git website.

- Here Creating data1.sh file for Mario.git website.

- Here I am using plan command to show the execution plan.
 - We can see all the resources with the single command.

```
ec2-user@ip-172-31-95-52:~  
data1.sh      igw.tf    instance2.tf    output.tf      rds.tf      securitygroup1.tf    subnet.tf      vpc.tf  
[ec2-user@ip-172-31-95-52 ~]$ 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_autoscaling_attachment.assg_attachments will be created  
+ resource "aws_autoscaling_attachment" "assg_attachments" {  
    + autoscaling_group_name = (known after apply)  
    + id                     = (known after apply)  
    + lb_target_group_arn   = (known after apply)  
}  
  
# aws_autoscaling_group.web_asg will be created  
+ resource "aws_autoscaling_group" "web_asg" {  
    + arn           = (known after apply)  
    + availability_zones = (known after apply)  
    + default_cooldown = (known after apply)  
    + desired_capacity = 3  
    + force_delete   = false  
    + force_delete_warm_pool = false  
    + health_check_grace_period = 300  
    + health_check_type = "ELB"  
    + id             = (known after apply)  
    + ignore_failed_scaling_activities = false  
    + load_balancers = (known after apply)  
    + max_size       = 6  
    + metrics_granularity = "1Minute"  
    + min_size       = 1  
    + name           = (known after apply)  
    + name_prefix    = (known after apply)  
    + predicted_capacity = (known after apply)  
    + protect_from_scale_in = false  
    + service_linked_role_arn = (known after apply)  
    + target_group_arns = (known after apply)  
    + vpc_zone_identifier = (known after apply)  
    + wait_for_capacity_timeout = "10m"  
    + warm_pool_size = (known after apply)  
  
    + launch_template {  
        + id     = (known after apply)  
        + name   = (known after apply)  
        + version = "$Latest"  
    }  
  
+ mixed_instances_policy (known after apply)
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ mixed_instances_policy (known after apply)
+ tag {
  + key   = "Name"
  + propagate_at_launch = true
  + value  = "Webservernstace"
}
+ traffic_source (known after apply)

# aws_db_instance.default will be created
resource "aws_db_instance" "default" {
  address           = (known after apply)
  allocated_storage = 10
  apply_immediately = false
  arn               = (known after apply)
  auto_minor_version_upgrade = true
  availability_zone = (known after apply)
  backup_retention_period = (known after apply)
  backup_target      = (known after apply)
  build_time        = (known after apply)
  ca_certificate_identifier = (known after apply)
  character_set_name = (known after apply)
  copy_tags_to_snapshot = true
  db_subnet_group_name = (known after apply)
  dedicated_log_volume = false
  enable_automated_backups = false
  domain_fqdn       = (known after apply)
  endpoint          = (known after apply)
  engine            = "mysql"
  engine_features_lifecycle_support = (known after apply)
  engine_version    = "8.0.20"
  engine_version_actual = (known after apply)
  engine_version_requested = (known after apply)
  id                = (known after apply)
  identifier         = (known after apply)
  identifier_prefix = (known after apply)
  instance_class     = "db.t2.micro"
  iops              = (known after apply)
  kms_key_id        = (known after apply)
  latest_restoreable_time = (known after apply)
  license_model      = (known after apply)
  listener_endpoint = (known after apply)
  maintenance_window = (known after apply)
  master_user_name   = (known after apply)
  master_user_password = (known after apply)
  master_user_secret_kms_key_id = (known after apply)
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ engine_features_lifecycle_support = "mysql"
+ engine_version                  = "8.0.20"
+ engine_version_actual          = (known after apply)
+ engine_version_requested       = (known after apply)
+ id                            = (known after apply)
+ identifier                     = (known after apply)
+ identifier_prefix              = (known after apply)
+ instance_class                 = "db.t2.micro"
+ iops                          = (known after apply)
+ kms_key_id                     = (known after apply)
+ latest_restoreable_time       = (known after apply)
+ license_model                  = (known after apply)
+ listener_endpoint              = (known after apply)
+ maintenance_window             = (known after apply)
+ master_user_secret_kms_key_id = (known after apply)
+ monitoring_arn                 = (known after apply)
+ multi_az                       = true
+ network_character_set_name     = (known after apply)
+ network_type                   = (known after apply)
+ option_group_name              = (known after apply)
+ parameter_group_name           = (known after apply)
+ performance_insights_enabled = false
+ performance_insights_kms_key_id = (known after apply)
+ performance_insights_retention_period = (known after apply)
+ port                           = (known after apply)
+ publicly_accessible           = false
+ provisioned_iops_mode          = (known after apply)
+ replicas                      = (known after apply)
+ resource_id                    = (known after apply)
+ skip_final_snapshot            = (known after apply)
+ snapshot_identifier             = (known after apply)
+ status                         = (known after apply)
+ storage_throughput             = (known after apply)
+ storage_type                   = (known after apply)
+ tags {
  + "Name" = "mydb"
}
+ tags_all {
  + "Name" = "mydb"
}
+ timezone                      = (known after apply)
+ username                       = "username"
+ vpc_security_group_ids          = (known after apply)
```

```
ec2-user@ip-172-31-95-52:~
```

```
# aws_db_subnet_group.default will be created
resource "aws_db_subnet_group" "default" {
  arn           = (known after apply)
  description   = "Managed by Terraform"
  name          = "main"
  name_prefix   = (known after apply)
  subnet_ids    = (known after apply)
  supported_network_types = (known after apply)
  tags {
    + "Name" = "my DB subnet group"
  }
  tags_all {
    + "Name" = "my DB subnet group"
  }
  vpc_id        = (known after apply)
}

# aws_instance.instance1 will be created
resource "aws_instance" "instance1" {
  ami           = "ami-0e54eba7c51c34f6"
  associate_public_ip_address = true
  availability_zone = (known after apply)
  cpu_core_count   = (known after apply)
  cpu_threads_per_core = (known after apply)
  disable_api_stop = (known after apply)
  disable_api_termination = (known after apply)
  ebs_optimized    = (known after apply)
  get_password_data = false
  host_id         = (known after apply)
  host_resource_group_arn = (known after apply)
  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         = "key"
  monitoring        = (known after apply)
  outpost_arn      = (known after apply)
  password_data    = (known after apply)
  placement_group   = (known after apply)
  placement_partition_number = (known after apply)
  primary_network_interface_id = (known after apply)
  private_ip        = (known after apply)
  public_ip         = (known after apply)
  public_dns        = (known after apply)
  secondary_private_ips = (known after apply)
```

```
ec2-user@ip-172-31-95-52:~
```

```
  ipv6_addresses_count          = (known after apply)
  ipv6_addresses                = (known after apply)
  key_name                      = "yash"
  monitoring                   = (known after apply)
  owner_id                      = (known after apply)
  password_data                = (known after apply)
  placement_group               = (known after apply)
  placement_partition_number    = (known after apply)
  primary_network_interface_id = (known after apply)
  private_dns                   = (known after apply)
  private_ip                    = (known after apply)
  public_dns                     = (known after apply)
  public_ip                      = (known after apply)
  security_group_ids            = (known after apply)
  security_groups               = (known after apply)
  source_dest_check             = true
  subnet_id                     = (known after apply)
  subnet_id                     = (known after apply)
  tags                          = {
```

```
    + "Name" = "instance1"
  } + tags_all = {
```

```
    + "Name" = "instance1"
  }
```

```
  tenancy                       = (known after apply)
  user_data                      = (known after apply)
  user_data_base64               = (known after apply)
  user_data_replace_on_change    = false
  vpc_security_group_ids         = (known after apply)
```

```
  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)
  private_dns_name_options (known after apply)
  root_block_device (known after apply)
}
```

```
ec2-user@ip-172-31-95-52:~
```

```
# aws_instance.instance2 will be created
resource "aws_instance" "instance2" {
  + ami                           = "ami-0e54eba7c51c234f6"
  + arn                           = (known after apply)
  + associate_public_ip_address   = true
  + availability_zone              = (known after apply)
  + ec2_cfn_update                = (known after apply)
  + cpus_per_hours_per_core        = (known after apply)
  + disable_api_stop               = (known after apply)
  + disable_ami_termination        = (known after apply)
  + optimized                     = (known after apply)
  + get_password_data              = false
  + host_id                       = (known after apply)
  + hibernation_source_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                  = t2.micro
  + ipv4_addresses_count           = (known after apply)
  + ipv6_addresses                 = (known after apply)
  key_name                        = "yash"
  monitoring                      = (known after apply)
  outpost_arn                     = (known after apply)
  password_data                   = (known after apply)
  placement_group_id              = (known after apply)
  placement_partition_number       = (known after apply)
  primary_network_interface_id    = (known after apply)
  private_dns                      = (known after apply)
  private_ip                       = (known after apply)
  public_dns                        = (known after apply)
  public_ip                         = (known after apply)
  security_group_ids               = (known after apply)
  security_groups                  = (known after apply)
  source_dest_check                = true
  subnet_id                        = (known after apply)
  subnet_id                        = (known after apply)
  tags                           = {
```

```
    + "Name" = "instance2"
  } + tags_all = {
```

```
    + "Name" = "instance2"
  }
```

```
  tenancy                       = (known after apply)
  user_data                      = (known after apply)
  user_data_base64               = (known after apply)
  user_data_replace_on_change    = false
  vpc_security_group_ids         = (known after apply)
```

```
  capacity_reservation_specification (known after apply)
}
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ 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)
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
```

```
# aws_internet_gateway.yashigw will be created
resource "aws_internet_gateway" "yashigw" {
  + arn                           = (known after apply)
  + id                            = (known after apply)
  + owner_id                      = (known after apply)
  + tags_all                      = (known after apply)
  + vpc_id                         = (known after apply)
}
```

```
# aws_launch_template.web-launch-template will be created
resource "aws_launch_template" "web-launch-template" {
  + arn                           = (known after apply)
  + default_version               = (known after apply)
  + id                            = (known after apply)
  + image_id                      = "ami-0e54eba7c51c234f6"
  + instance_type                 = "t2.micro"
  + key_name                      = "yash"
  + latest_version                = (known after apply)
  + network_interfaces            = (known after apply)
  + name_prefix                   = "web-launch-template"
  + tags_all                      = (known after apply)
  + vpc_security_group_ids        = (known after apply)
```

```
+ metadata_options (known after apply)
+ tag_specifications [
  + resource_type = "instance"
  + tags [
    + "Name" = "weblanuchinstance"
  ]
]
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ latest_version = "(known after apply)"
+ name = "(known after apply)"
+ name_prefix = "web-lanuch-template"
+ tags_all = "(known after apply)"
+ vpc_security_group_ids = "(known after apply)"

+ metadata_options (known after apply)
+ tag_specifications [
+   resource_type = "instance"
+   tags = [
+     "Name" = "webLaunchInstance"
+   ]
+ ]
}

# aws_lb.external-alb will be created
resource "aws_lb" "external-alb" {
  arn
+ arn_suffix
+ cross_az_load_balancer
+ defragment_ip_header
+ desync_mitigation_mode
+ dns_name
+ drop_invalid_header_fields
+ enable_ip_deception_protection
+ enable_ip_header_protection
+ enable_html_error_page
+ enable_tls_version_and_cipher_suite_headers
+ enable_xff
+ enable_xff_client_port
+ enforce_security_group_inbound_rules_on_private_link_traffic
+ id
+ idle_timeout
+ internal
+ ip_address_type
+ load_balancer_type
+ name
+ name_prefix
+ preserve_host_header
+ security_groups
+ subnets
+ tags
+ "Name" = "external-alb"
}
tags_all["Name" = "external-alb"]
+ vpc_id
+ xff_header_processing_mode
+ zone_id

+ subnet_mapping (known after apply)
```

```
ec2-user@ip-172-31-95-52:~ # aws_lb_listener_target_elb will be created
+ resource "aws_lb_listener" "target_elb" {
+   arn = (known after apply)
+   cert_arn = (known after apply)
+   load_balancer_arn = (known after apply)
+   port = 80
+   protocol = "HTTP"
+   ssl_policy = (known after apply)
+   tags = [
+     {
+       "Name" = "target-listener"
+     }
+   ]
+   tags_all = [
+     {
+       "Name" = "target-listener"
+     }
+   ]
+
+   default_action {
+     type = "forward"
+     target_group_arn = (known after apply)
+   }
+
+   mutual_authentication (known after apply)
}
+
# aws_lb_target_group_target_elb will be created
+ resource "aws_lb_target_group" "target_elb" {
+   arn = (known after apply)
+   arn_suffix = (known after apply)
+   connection_termination = (known after apply)
+   deregistration_delay = "300"
+   idle_timeout = (known after apply)
+   ip_address_type = (known after apply)
+   lambda_multi_value_headers_enabled = false
+   load_balancer_arn = (known after apply)
+   load_balancing_algorithm_type = (known after apply)
+   load_balancing_anomaly_mitigation = (known after apply)
+   load_balancing_cross_zone_enabled = (known after apply)
+   name = "target-elb"
+   name_prefix = (known after apply)
+   port = 80
+   preserve_client_ip = (known after apply)
+   protocol = "HTTP"
+   protocol_version = (known after apply)
+   proxy_protocol_v2 = false
+   target_start = 0
+   tags = [
+     {
+       "Name" = "target-elb"
+     }
+   ]
+   tags_all = [
+     {
+       "Name" = "target-elb"
+     }
+   ]
+   target_type = "instance"
}
```

```
ec2-user@ip-172-31-95-52:~$
```

```
+ tags = { "Name" = "target-elb" }
+ }
+ tags_all = {
+   "Name" = "target-elb"
+ }
+ target_type = "instance"
+ vpc_id = "(known after apply)"

+ health_check (known after apply)
+ stickiness (known after apply)
+ target_failover (known after apply)
+ target_group_health (known after apply)
+ target_health_state (known after apply)
}

# aws_lb_target_group_attachment.attachment1 will be created
resource "aws_lb_target_group_attachment" "attachment1" {
+ id = "(known after apply)"
+ port = 80
+ target_group_arn = "(known after apply)"
+ target_id = "(known after apply)"
}

# aws_lb_target_group_attachment.attachment2 will be created
resource "aws_lb_target_group_attachment" "attachment2" {
+ id = "(known after apply)"
+ port = 80
+ target_group_arn = "(known after apply)"
+ target_id = "(known after apply)"
}

# aws_route_table.public-route-table will be created
resource "aws_route_table" "public-route-table" {
+ arn = "(known after apply)"
+ id = "(known after apply)"
+ owner_id = "(known after apply)"
+ propagating_vgws = "(known after apply)"
+ route = [
+   {
+     cidr_block = "0.0.0.0/0"
+     gateway_id = "(known after apply)"
+     # (11 unchanged attributes hidden)
+   },
+ ],
+ tags = {
+   "Name" = "public-route-table"
+ }
+ tags_all = {
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ tags = {
+   "Name" = "public-route-table"
+ }
+ tags_all = [
+   "Name" = "public-route-table"
+ ]
+ vpc_id = (known after apply)
}

# aws_route_table_association.rt1 will be created
resource "aws_route_table_association" "rt1" {
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = (known after apply)
}

# aws_route_table_association.rt2 will be created
resource "aws_route_table_association" "rt2" {
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = (known after apply)
}

# aws_security_group.database-securitygroup will be created
resource "aws_security_group" "database-securitygroup" {
+ arn = (known after apply)
+ description = "Managed by Terraform"
+ egress = [
+   {
+     cidr_blocks = [
+       "0.0.0.0/0"
+     ]
+     from_port = 32768
+     ipv4.cidr_blocks = []
+     prefix_list_ids = []
+     protocol = "tcp"
+     security_groups = []
+     self = false
+     to_port = 65535
+   },
+   {
+     ingress = [
+       {
+         cidr_blocks = [
+           "0.0.0.0/0"
+         ]
+         from_port = 22
+         ipv4.cidr_blocks = []
+         prefix_list_ids = []
+         protocol = "tcp"
+       }
+     ]
+   }
+ ]
+ id = (known after apply)
+ tags = [
+   "Name" = "database-securitygroup"
+ ]
+ tags_all = [
+   "Name" = "database-securitygroup"
+ ]
+ vpc_id = (known after apply)
}

# aws_security_group.securitygroup1 will be created
resource "aws_security_group" "securitygroup1" {
+ arn = (known after apply)
+ description = "Managed by Terraform"
+ egress = [
+   {
+     cidr_blocks = [
+       "0.0.0.0/0"
+     ]
+     from_port = 0
+     ipv4.cidr_blocks = []
+     prefix_list_ids = []
+     protocol = "1"
+     security_groups = []
+     self = false
+     to_port = 0
+   },
+   {
+     ingress = [
+       {
+         cidr_blocks = [
+           "0.0.0.0/0"
+         ]
+         from_port = 443
+         ipv4.cidr_blocks = []
+         prefix_list_ids = []
+         protocol = "tcp"
+         security_groups = []
+         self = false
+         to_port = 443
+       },
+       {
+         cidr_blocks = [
+           "0.0.0.0/0"
+         ]
+         from_port = 80
+         ipv4.cidr_blocks = []
+         prefix_list_ids = []
+         protocol = "tcp"
+         security_groups = []
+         self = false
+         to_port = 80
+       }
+     ],
+     id = (known after apply)
+   }
+ ]
+ tags_all = [
+   "Name" = "securitygroup1"
+ ]
+ vpc_id = (known after apply)
}
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ prefix_list_ids = []
+ protocol = "tcp"
+ security_groups = []
+ self = false
+ to_port = 22
# (1 unchanged attribute hidden)
},
{
+ cidr_blocks = []
+ from_port = 3206
+ ipv4.cidr_blocks = []
+ prefix_list_ids = []
+ protocol = "tcp"
+ security_groups = (known after apply)
+ self = false
+ to_port = 3206
# (1 unchanged attribute hidden)
},
{
+ name = "database-securitygroup"
+ tags_all = [
+   "Name" = "database-securitygroup"
+ ]
+ vpc_id = (known after apply)
}

# aws_security_group.securitygroup1 will be created
resource "aws_security_group" "securitygroup1" {
+ arn = (known after apply)
+ description = "Managed by Terraform"
+ egress = [
+   {
+     cidr_blocks = [
+       "0.0.0.0/0"
+     ]
+     from_port = 0
+     ipv4.cidr_blocks = []
+     prefix_list_ids = []
+     protocol = "1"
+     security_groups = []
+     self = false
+     to_port = 0
+   },
+   {
+     ingress = [
+       {
+         cidr_blocks = [
+           "0.0.0.0/0"
+         ]
+         from_port = 443
+         ipv4.cidr_blocks = []
+         prefix_list_ids = []
+         protocol = "tcp"
+         security_groups = []
+         self = false
+         to_port = 443
+       }
+     ],
+     id = (known after apply)
+   }
+ ]
+ tags_all = [
+   "Name" = "securitygroup1"
+ ]
+ vpc_id = (known after apply)
}
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ id = (known after apply)
+ ingress = [
+   {
+     cidr_blocks = [
+       "0.0.0.0/0"
+     ]
+     from_port = 22
+     ipv4.cidr_blocks = []
+     prefix_list_ids = []
+     protocol = "tcp"
+     security_groups = []
+     self = false
+     to_port = 22
# (1 unchanged attribute hidden)
},
{
+ cidr_blocks = [
+   "0.0.0.0/0"
+ ]
+ from_port = 443
+ ipv4.cidr_blocks = []
+ prefix_list_ids = []
+ protocol = "tcp"
+ security_groups = []
+ self = false
+ to_port = 443
# (1 unchanged attribute hidden)
},
{
+ cidr_blocks = [
+   "0.0.0.0/0"
+ ]
+ from_port = 80
+ ipv4.cidr_blocks = []
+ prefix_list_ids = []
+ protocol = "tcp"
+ security_groups = []
+ self = false
+ to_port = 80
# (1 unchanged attribute hidden)
},
{
+ name = (known after apply)
+ name_prefix = (known after apply)
+ owner_id = (known after apply)
+ revoke_rules_on_delete = false
+ tags = [
+   "Name" = "securitygroup1"
+ ]
+ tags_all = [
+   "Name" = "securitygroup1"
+ ]
+ vpc_id = (known after apply)
}
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ vpc_id = (Known after apply)
```

```
# aws_subnet.application-subnet1 will be created
```

```
+ resource "aws_subnet" "application-subnet1" {
```

```
+   arn = (Known after apply)
```

```
+   assign_ipv6_address_on_creation = false
```

```
+   availability_zone = "us-east-1a"
```

```
+   availability_zone_id = (Known after apply)
```

```
+   cidr_block = "10.0.3.0/24"
```

```
+   enable_dns64 = false
```

```
+   enable_resource_name_dns_a_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 = {
```

```
+     "Name" = "application-subnet1"
```

```
+   }
```

```
+   tags_all = {
```

```
+     "Name" = "application-subnet1"
```

```
+   }
```

```
+   vpc_id = (Known after apply)
```

```
}
```

```
# aws_subnet.application-subnet2 will be created
```

```
+ resource "aws_subnet" "application-subnet2" {
```

```
+   arn = (Known after apply)
```

```
+   assign_ipv6_address_on_creation = false
```

```
+   availability_zone = "us-east-1b"
```

```
+   availability_zone_id = (Known after apply)
```

```
+   cidr_block = "10.0.4.0/24"
```

```
+   enable_dns64 = false
```

```
+   enable_resource_name_dns_a_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 = {
```

```
+     "Name" = "application-subnet2"
```

```
+   }
```

```
+   tags_all = {
```

```
+     "Name" = "application-subnet2"
```

```
+   }
```

```
+   vpc_id = (Known after apply)
```

```
}
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ vpc_id = (Known after apply)
```

```
# aws_subnet.database-subnet1 will be created
```

```
+ resource "aws_subnet" "database-subnet1" {
```

```
+   arn = (Known after apply)
```

```
+   assign_ipv6_address_on_creation = false
```

```
+   availability_zone = "us-east-1a"
```

```
+   availability_zone_id = (Known after apply)
```

```
+   cidr_block = "10.0.5.0/24"
```

```
+   enable_dns64 = false
```

```
+   enable_resource_name_dns_a_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 = {
```

```
+     "Name" = "database-subnet1"
```

```
+   }
```

```
+   tags_all = {
```

```
+     "Name" = "database-subnet1"
```

```
+   }
```

```
+   vpc_id = (Known after apply)
```

```
}
```

```
# aws_subnet.database-subnet2 will be created
```

```
+ resource "aws_subnet" "database-subnet2" {
```

```
+   arn = (Known after apply)
```

```
+   assign_ipv6_address_on_creation = false
```

```
+   availability_zone = "us-east-1b"
```

```
+   availability_zone_id = (Known after apply)
```

```
+   cidr_block = "10.0.6.0/24"
```

```
+   enable_dns64 = false
```

```
+   enable_resource_name_dns_a_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 = {
```

```
+     "Name" = "database-subnet2"
```

```
+   }
```

```
+   tags_all = {
```

```
+     "Name" = "database-subnet2"
```

```
+   }
```

```
+   vpc_id = (Known after apply)
```

```
}
```

```
ec2-user@ip-172-31-95-52:~
```

```
+ vpc_id = (Known after apply)
```

```
# aws_subnet.web-subnet1 will be created
```

```
+ resource "aws_subnet" "web-subnet1" {
```

```
+   arn = (Known after apply)
```

```
+   assign_ipv6_address_on_creation = false
```

```
+   availability_zone = "us-east-1a"
```

```
+   availability_zone_id = (Known after apply)
```

```
+   cidr_block = "10.0.1.0/24"
```

```
+   enable_dns64 = false
```

```
+   enable_resource_name_dns_a_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 = {
```

```
+     "Name" = "web-subnet1"
```

```
+   }
```

```
+   tags_all = {
```

```
+     "Name" = "web-subnet1"
```

```
+   }
```

```
+   vpc_id = (Known after apply)
```

```
}
```

```
# aws_subnet.web-subnet2 will be created
```

```
+ resource "aws_subnet" "web-subnet2" {
```

```
+   arn = (Known after apply)
```

```
+   assign_ipv6_address_on_creation = false
```

```
+   availability_zone = "us-east-1b"
```

```
+   availability_zone_id = (Known after apply)
```

```
+   cidr_block = "10.0.2.0/24"
```

```
+   enable_dns64 = false
```

```
+   enable_resource_name_dns_a_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 = {
```

```
+     "Name" = "web-subnet2"
```

```
+   }
```

```
+   tags_all = {
```

```
+     "Name" = "web-subnet2"
```

```
+   }
```

```
+   vpc_id = (Known after apply)
```

```
}
```

```

ec2-user@ip-172-31-95-52:~ 
enable_resource_name_dns_a_record_on_launch = false
enable_resource_name_dns_ae_record_on_launch = false
id = (known after apply)
ipv6_cidr_block_association_id = (known after apply)
map_public_ip_on_launch = true
owner_id = (known after apply)
private_dns_hostname_type_on_launch = (known after apply)
tags = {
  "Name" = "web-subnet2"
}
tags_all = {
  "Name" = "web-subnet2"
}
vpc_id = (known after apply)

# aws_vpc.yashvpc will be created
resource "aws_vpc" "yashvpc" {
  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)
  enable_dhcp = true
  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"
  ipv4_association_id = (known after apply)
  ipv4_cidr_block = (known after apply)
  ipv6_cidr_block_network_border_group = (known after apply)
  main_route_table_id = (known after apply)
  owner_id = (known after apply)
  tags = {
    "Name" = "yashVPC"
  }
  tags_all = {
    "Name" = "yashVPC"
  }
}

Plan: 25 to add, 0 to change, 0 to destroy.

Changes to outputs:
+ lb_dns_name = (known after apply)

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-172-31-95-52 ~]$ 

```

```

ec2-user@ip-172-31-95-52:~ 
} 
+ vpc_id = (known after apply)

# aws_vpc.yashvpc will be created
resource "aws_vpc" "yashvpc" {
  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"
  ipv4_association_id = (known after apply)
  ipv4_cidr_block = (known after apply)
  ipv6_cidr_block_network_border_group = (known after apply)
  main_route_table_id = (known after apply)
  owner_id = (known after apply)
  tags = {
    "Name" = "yashVPC"
  }
  tags_all = {
    "Name" = "yashVPC"
  }
}

Plan: 25 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ lb_dns_name = (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

```

```

ec2-user@ip-172-31-95-52:~ 
aws_db_instance.default: still creating... [7m10s elapsed]
aws_db_instance.default: still creating... [7m20s elapsed]
aws_db_instance.default: still creating... [7m30s elapsed]
aws_db_instance.default: still creating... [7m40s elapsed]
aws_db_instance.default: still creating... [7m50s elapsed]
aws_db_instance.default: still creating... [8m0s elapsed]
aws_db_instance.default: still creating... [8m10s elapsed]
aws_db_instance.default: still creating... [8m20s elapsed]
aws_db_instance.default: still creating... [8m30s elapsed]
aws_db_instance.default: still creating... [8m40s elapsed]
aws_db_instance.default: still creating... [8m50s elapsed]
aws_db_instance.default: still creating... [9m0s elapsed]
aws_db_instance.default: still creating... [9m10s elapsed]
aws_db_instance.default: still creating... [9m20s elapsed]
aws_db_instance.default: still creating... [9m30s elapsed]
aws_db_instance.default: still creating... [9m40s elapsed]
aws_db_instance.default: still creating... [9m50s elapsed]
aws_db_instance.default: still creating... [10m0s elapsed]
aws_db_instance.default: still creating... [10m10s elapsed]
aws_db_instance.default: still creating... [10m20s elapsed]
aws_db_instance.default: still creating... [10m30s elapsed]
aws_db_instance.default: still creating... [10m40s elapsed]
aws_db_instance.default: still creating... [10m50s elapsed]
aws_db_instance.default: still creating... [11m0s elapsed]
aws_db_instance.default: still creating... [11m10s elapsed]
aws_db_instance.default: still creating... [11m20s elapsed]
aws_db_instance.default: still creating... [11m30s elapsed]
aws_db_instance.default: still creating... [11m40s elapsed]
aws_db_instance.default: still creating... [11m50s elapsed]
aws_db_instance.default: still creating... [12m0s elapsed]
aws_db_instance.default: Creation complete after 12m10s [id=db-s7HPU4XLHCNJRNMNLUGC7RQSXQ]

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

Outputs:
lb_dns_name = "external-LB-1476713536.us-east-1.elb.amazonaws.com"
[ec2-user@ip-172-31-95-52 ~]$ 

```

Verify the resources

- Terraform will create all resources.
- Here the VPC was created.

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
yashVPC	vpc-02ceea516d413a5c8	Available	10.0.0.0/16	-
-	vpc-05f2f9c35f844e371	Available	172.31.0.0/16	-

- Here Four public and two private subnets was created.

Name	Subnet ID	State	VPC	IPv4
database-subnet1	subnet-043b397b61bb946c4	Available	vpc-02ceea516d413a5c8 yash...	10.0.
-	subnet-0418e4d5d513ce75	Available	vpc-05f2f9c35f844e371	172.3.
application-subnet1	subnet-0b4c1f3ca6731d769	Available	vpc-02ceea516d413a5c8 yash...	10.0.
-	subnet-0b9cdb5a0e5ae871a	Available	vpc-05f2f9c35f844e371	172.3.
application-subnet2	subnet-0dd1bd25dc7f5338f	Available	vpc-02ceea516d413a5c8 yash...	10.0.
-	subnet-03ebaa5e1f54cd90c	Available	vpc-05f2f9c35f844e371	172.3.
database-subnet2	subnet-07bb7ac342ad46091	Available	vpc-02ceea516d413a5c8 yash...	10.0.
web-subnet1	subnet-0fd12e68f3a46183d	Available	vpc-02ceea516d413a5c8 yash...	10.0.
-	subnet-02e04a21c22395dfa	Available	vpc-05f2f9c35f844e371	172.3.
web-subnet2	subnet-040b4beceacd795a5	Available	vpc-02ceea516d413a5c8 yash...	10.0.
-	subnet-08d27d7fbae8eae6c	Available	vpc-05f2f9c35f844e371	172.3.

- Here Internet gateway was created.

Internet gateways (1/2) Info

Name	Internet gateway ID	State	VPC ID
-	igw-00213540771abe2c3	Attached	vpc-05f2f9c35f844e371
<input checked="" type="checkbox"/>	igw-059f24cc01b38b07c	Attached	vpc-02ceea516d413a5c8 yashVPC

igw-059f24cc01b38b07c

Details | Tags

Details

Internet gateway ID igw-059f24cc01b38b07c	State Attached	VPC ID vpc-02ceea516d413a5c8 yashVPC	Owner 637423323663
--	-------------------	---	-----------------------

- Here the route table was created.

Route tables (1/3) Info

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC
<input checked="" type="checkbox"/> public-route-table	rtb-0763feda7588cedf7	2 subnets	-	No	vp
-	rtb-097c70e1ff620f552	-	-	Yes	vp
-	rtb-004452a24f929a179	-	-	Yes	vp

rtb-0763feda7588cedf7 / public-route-table

Details | **Routes** | Subnet associations | Edge associations | Route propagation | Tags

Routes (2)

Filter routes

- Here we can see the two security groups were created.

The screenshot shows the AWS VPC Security Groups console. The left sidebar navigation includes Network & Security, Load Balancing, Auto Scaling, and Settings. The main content area displays a table titled "Security Groups (2/6) Info". The table has columns for Name, Security group ID, Security group name, and VPC ID. Two rows are listed: "database-securitygroup" (sg-0bb09e8ce6a9222f2) and "securitygroup1" (sg-01d8b8e09726e5fbf), both associated with the VPC vpc-02cea516d413a5c8. A message at the bottom states "Security Groups: sg-0bb09e8ce6a9222f2, sg-01d8b8e09726e5fbf".

- Here the Load Balancer and Target Group was created.

The screenshot shows the AWS Load Balancers console. The left sidebar navigation includes Network & Security, Load Balancing, Auto Scaling, and Settings. The main content area displays a table titled "Load balancers (1/1)". It shows one entry for "external-LB" with a status of "Active" and a VPC ID of vpc-02cea516d413a5c8. Below this, a detailed view of the "external-LB" load balancer is shown, with tabs for Details, Listeners and rules, Network mapping, Resource map - new, Security, Monitoring, and Integrations. The "Details" tab is selected, showing information such as Load balancer type (Application), Status (Active), VPC (vpc-02cea516d413a5c8), and Load balancer IP address type (IPv4).

The screenshot shows the AWS EC2 Target groups page. The left sidebar navigation includes: Lifecycle Manager, Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups, Trust Stores), and Auto Scaling (Auto Scaling Groups, Settings). The main content area displays 'Target groups (1/1) Info' with a table showing one entry: ALB-TG. The table columns are Name, ARN, Port, Protocol, and Target type. The ARN is arn:aws:elasticloadbalancing:us-east-1:637423323663:targetgroup/ALB-TG/b83c0a38f1ad73a4, Port is 80, Protocol is HTTP, and Target type is Instance. Below this, a detailed view for 'Target group: ALB-TG' is shown with tabs for Details, Targets, Monitoring, Health checks, Attributes, and Tags. The Details tab shows the ARN again and lists Target type, Protocol : Port, Protocol version, and VPC.

➤ Here the Auto Scaling Group was created.

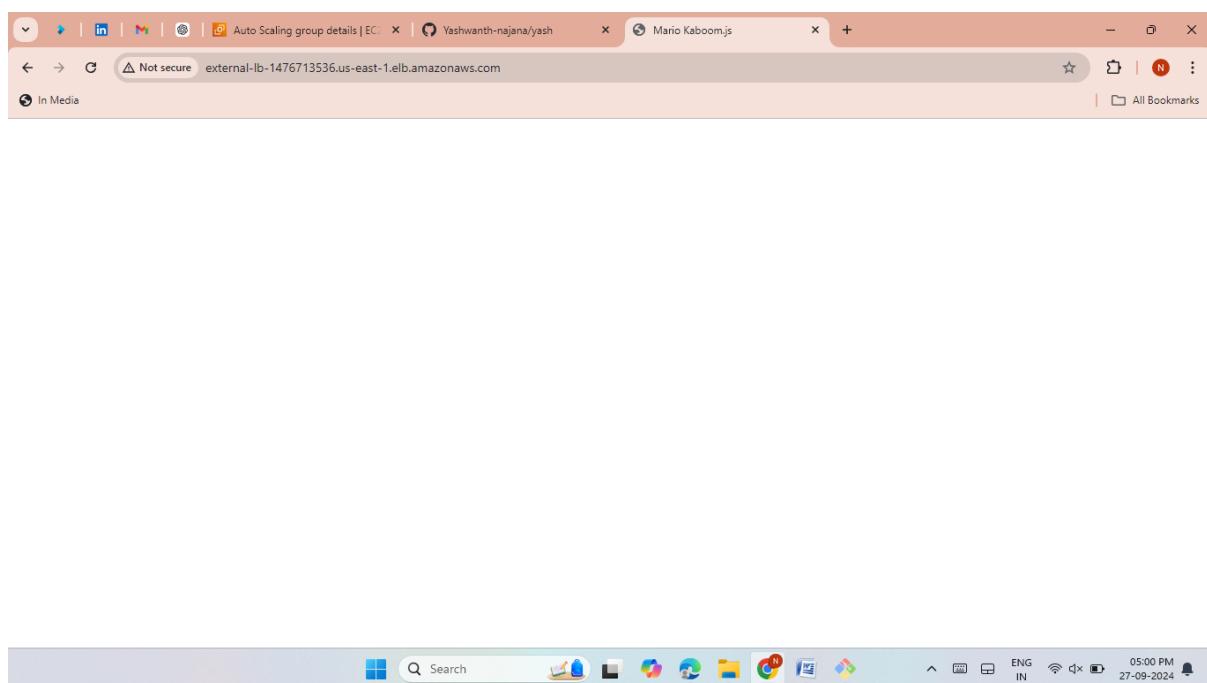
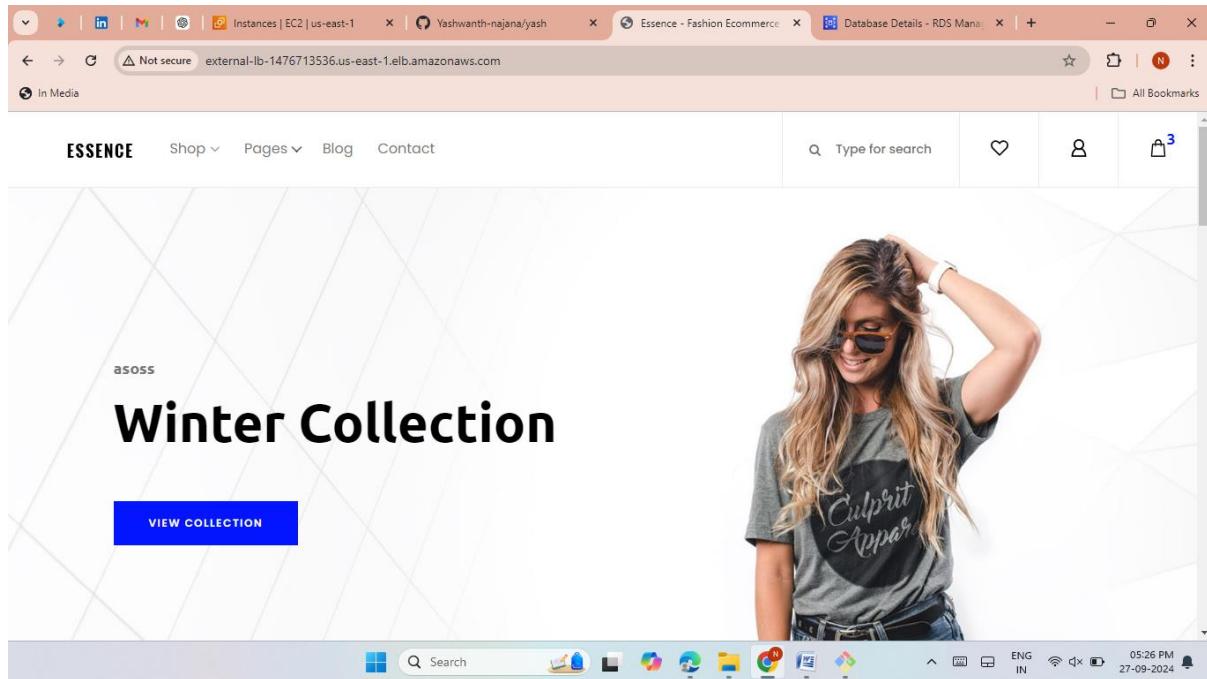
The screenshot shows the AWS EC2 Auto Scaling groups page. The left sidebar navigation includes: Lifecycle Manager, Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups, Trust Stores), and Auto Scaling (Auto Scaling Groups, Settings). The main content area displays 'Auto Scaling groups (1/1) Info' with a table showing one entry: terraform-. The table columns are Name, Launch template/configuration, Instances, Status, and Desired. The name is terraform-, the launch template/configuration is web-lanuuch-template20240927110454:, there are 3 instances, the status is -, and the desired capacity is 3. Below this, a detailed view for 'Auto Scaling group: terraform-20240927110503066700000009' is shown with tabs for Details, Activity, Automatic scaling, Instance management, Monitoring, and Instance refresh. The Automatic scaling tab is selected, showing the Group details section with fields: Auto Scaling group name (terrafarm-), Desired capacity (3), Desired capacity type (Units (number of instances)), and Amazon Resource Name (ARN) (arn:aws:autoscaling:us-east-1:637423323663:autoScalingGroup:637423323663:terrafarm-).

The screenshot shows the AWS Auto Scaling group details page. The left sidebar navigation includes Services, Lifecycle Manager, Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups, Trust Stores), and Auto Scaling (Auto Scaling Groups). The main content area displays the 'terraform-20240927110503066700000009' Auto Scaling group. The 'Group details' section shows the Auto Scaling group name, desired capacity (3), minimum capacity (1), maximum capacity (6), and Amazon Resource Name (ARN). The 'Launch template' section shows the ARN of the launch template used for this group.

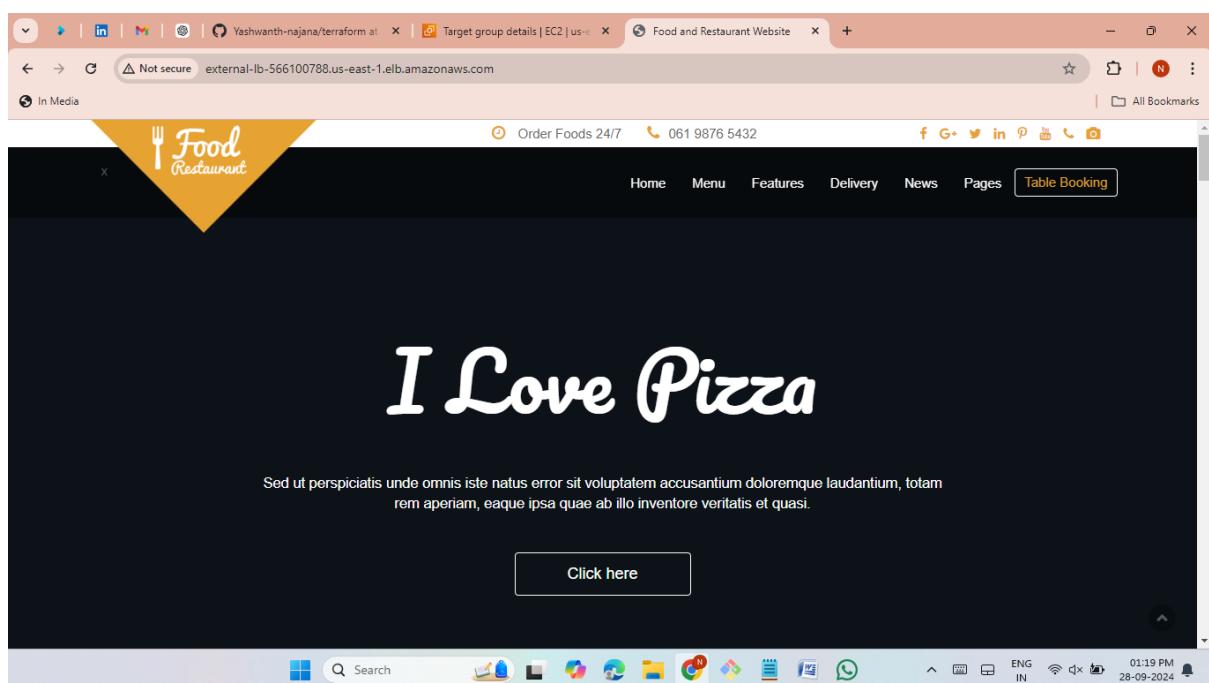
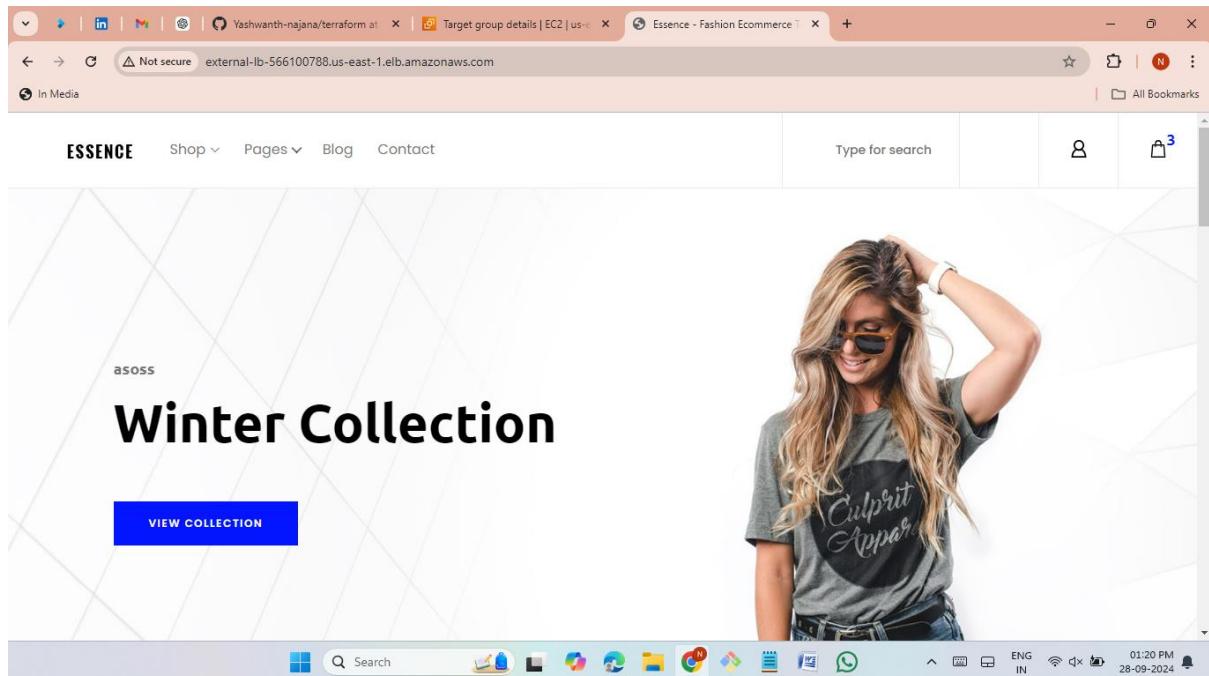
- Instance1 and Instance2 are created instances. Remaining instances are created by the Auto Scaling Group.

The screenshot shows the AWS Instances page. The left sidebar navigation includes EC2 Global View, Events, Console-to-Code (Preview), Instances (Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations), Images (AMIs, AMI Catalog), and CloudWatch Metrics. The main content area displays a table of 11 instances. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability zone. Instances listed include 'Webservermstae' (terminated, t2.micro), 'instance1' (running, t2.micro), 'terraform' (running, t2.micro), 'Webservermstae' (terminated, t2.micro), 'Webservermstae' (terminated, t2.micro), 'instance2' (running, t2.micro), 'Webservermstae' (running, t2.micro), and 'instance2' (terminated, t2.micro).

- Once the Resource creation finishes we can get the DNS of a load balancer and paste it into the browser you can see Load Balancer will send the request to two instances.
- Hosting and balancing the load with help of load balancer DNS.
- We can see the two websites in single DNS link.



- Here I am hosting ecomm and food websites also.



➤ Here the RDS Database was created.

The screenshot shows the AWS RDS Databases page. On the left, there's a sidebar with options like Dashboard, Databases (which is selected), Query Editor, etc. The main area shows a table of databases. A modal window at the top right provides information about Blue/Green Deployments. The database listed is:

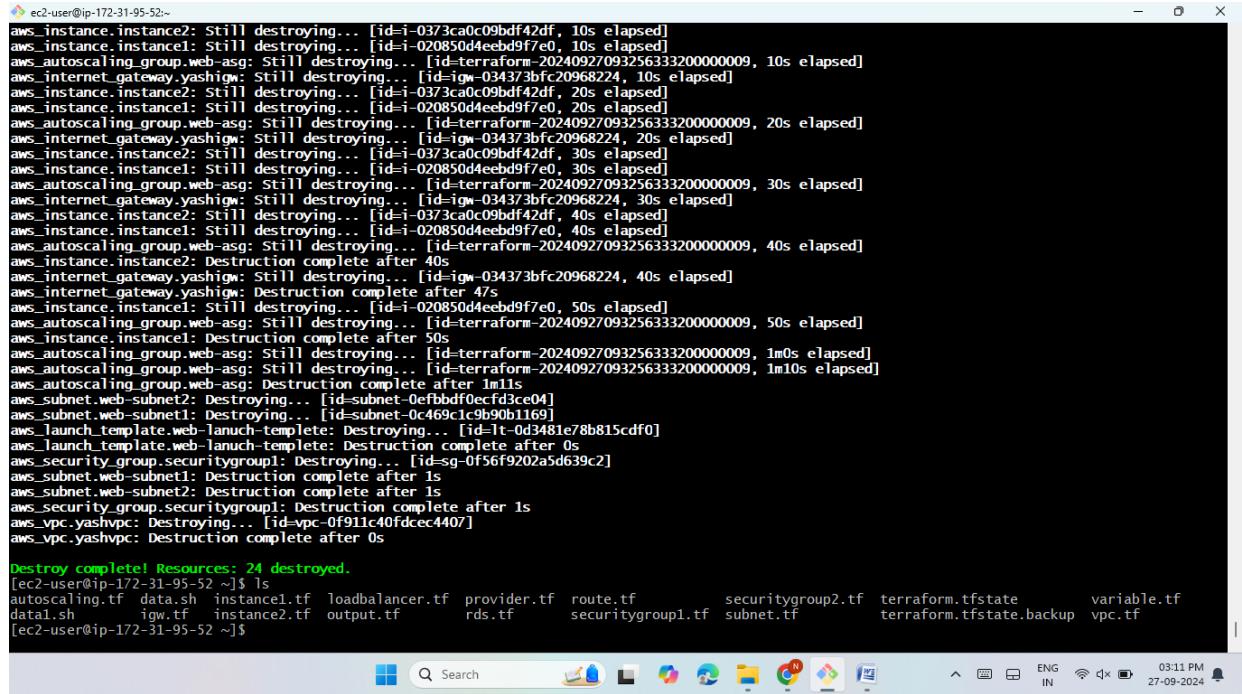
DB identifier	Status	Role	Engine	Region & AZ
terraform-20240927110456490700000006	Available	Instance	MySQL Community	us-east-1b

The screenshot shows the AWS RDS Database Details page for the database 'terraform-20240927110456490700000006'. The left sidebar is identical to the previous screenshot. The main area has a 'Summary' section with details:

DB identifier	Status	Role	Engine	Recommendations
terraform-20240927110456490700000006	Available	Instance	MySQL Community	
CPU	Class db.t3.micro	Current activity 0	Region & AZ us-east-1b	

Below the summary, there are tabs for Connectivity & security, Monitoring, Logs & events, Configuration, Zero-ETL integrations, and Maintenance. The Connectivity & security tab is currently active.

- Finally we can destroy all the resources with the single command **terraform destroy**.
- The terraform destroy command is used to completely remove all resources that were previously created and managed by Terraform in your infrastructure.



```

ec2-user@ip-172-31-95-52:~$ terraform destroy
aws_instance.instance2: Still destroying... [id=i-0373ca0c09bdf42df, 10s elapsed]
aws_instance.instance1: Still destroying... [id=i-020850d4eebd9f7e0, 10s elapsed]
aws_autoscaling_group.web-asg: Still destroying... [id=terraform-20240927093256333200000009, 10s elapsed]
aws_internet_gateway.yashigw: Still destroying... [id=igw-034373bfc20968224, 10s elapsed]
aws_instance.instance2: Still destroying... [id=i-0373ca0c09bdf42df, 20s elapsed]
aws_instance.instance1: Still destroying... [id=i-020850d4eebd9f7e0, 20s elapsed]
aws_internet_gateway.yashigw: Still destroying... [id=igw-034373bfc20968224, 20s elapsed]
aws_instance.instance2: Still destroying... [id=i-0373ca0c09bdf42df, 30s elapsed]
aws_instance.instance1: Still destroying... [id=i-020850d4eebd9f7e0, 30s elapsed]
aws_autoscaling_group.web-asg: Still destroying... [id=terraform-20240927093256333200000009, 30s elapsed]
aws_internet_gateway.yashigw: Still destroying... [id=igw-034373bfc20968224, 30s elapsed]
aws_instance.instance2: Still destroying... [id=i-0373ca0c09bdf42df, 40s elapsed]
aws_instance.instance1: Still destroying... [id=i-020850d4eebd9f7e0, 40s elapsed]
aws_autoscaling_group.web-asg: Still destroying... [id=terraform-20240927093256333200000009, 40s elapsed]
aws_instance.instance2: Destruction complete after 40s
aws_internet_gateway.yashigw: Still destroying... [id=igw-034373bfc20968224, 40s elapsed]
aws_internet_gateway.yashigw: Destruction complete after 47s
aws_instance.instance1: Still destroying... [id=i-020850d4eebd9f7e0, 50s elapsed]
aws_autoscaling_group.web-asg: Still destroying... [id=terraform-20240927093256333200000009, 50s elapsed]
aws_instance.instance1: Destruction complete after 50s
aws_autoscaling_group.web-asg: Still destroying... [id=terraform-20240927093256333200000009, 1m0s elapsed]
aws_autoscaling_group.web-asg: Still destroying... [id=terraform-20240927093256333200000009, 1m10s elapsed]
aws_autoscaling_group.web-asg: Destruction complete after 1m1s
aws_subnet.web-subnet2: Destroying... [id=subnet-0efbbdb0c0cf3ce04]
aws_subnet.web-subnet1: Destroying... [id=subnet-0c469c1c9b90b11691]
aws_launch_template.web-launch-template: Destroying... [id=lt-0d3481e78b815cdf0]
aws_launch_template.web-launch-template: Destruction complete after 0s
aws_security_group.securitygroup1: Destroying... [id=sg-0f56f9202a5d639c2]
aws_subnet.web-subnet1: Destruction complete after 1s
aws_subnet.web-subnet2: Destruction complete after 1s
aws_security_group.securitygroup1: Destruction complete after 1s
aws_vpc.yashvpc: Destroying... [id=vpc-0f911c40fdcec4407]
aws_vpc.yashvpc: Destruction complete after 0s

Destroy complete! Resources: 24 destroyed.

[ec2-user@ip-172-31-95-52 ~]$ ls
autoscaling.tf  data.sh  instance1.tf  loadbalancer.tf  provider.tf  route.tf      securitygroup2.tf  terraform.tfstate   variable.tf
data1.sh        igw.tf   instance2.tf  output.tf       rds.tf       securitygroup1.tf  subnet.tf      terraform.tfstate.backup  vpc.tf
[ec2-user@ip-172-31-95-52 ~]$ 
```