Terraform

Introduction

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, or optionally JSON.

Programming language: Go

Developer: HashiCorp

Initial release: 28 July 2014; 9 years ago

License: Business Source License v1.1(source-available)

Operating system: Linux, FreeBSD, macOS, OpenBSD, Solaris, and Microsoft Windows

Repository: github.com/hashicorp/terraform

Stable release: 1.5.5 / 9 August 2023; 8 months ago

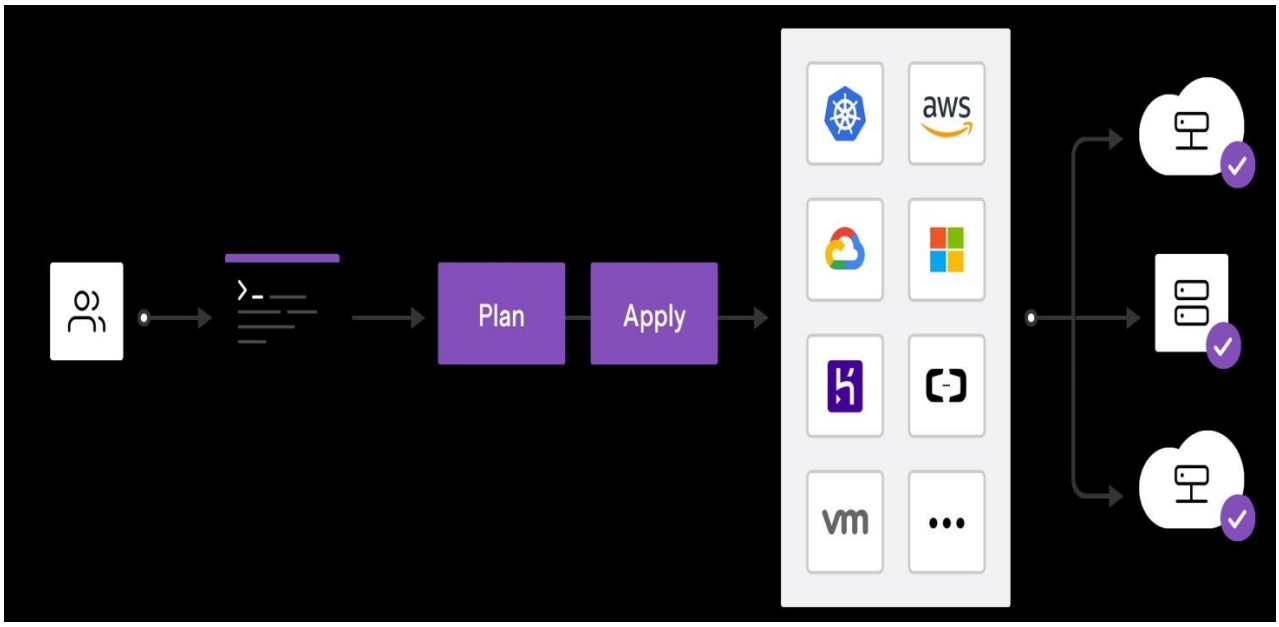
Terraform providers:

HashiCorp Terraform now supports more than 3,000 integrations with more than 250 partners, including

Amazon Web Services (AWS), Microsoft Azure, Google Cloud, Confluent, Datadog, MongoDB, Palo Alto Networks, ServiceNow, and Zscaler.

Terraform work flow:

1. **Terraform init**
2. **Terraform fmt**
3. **Terraform validate**
4. **Terraform plan**
5. **Terraform apply**



What are the use cases in Terraform?

✓ 6 ways to use Terraform:

1. Multi-Cloud Management
2. Self-Service Infrastructure
3. Improve Speed and Ease of Use
4. Enable High Collaboration
5. Reduce Errors
6. Create Consistency

What is an example of IaC (Infrastructure-as-code)?

Examples of infrastructure-as-code tools include AWS CloudFormation, Chef, Google Cloud Deployment Manager, Hashicorp Terraform, Microsoft Azure Resource Manager, Puppet, Red Hat Ansible and SaltStack. Some tools rely on a domain-specific language, while others use a standard template format, such as YAML and JSON.

What is an example of declarative configuration language?

Common declarative languages include those of database query languages (e.g., SQL, XQuery) regular expressions, logic programming (e.g. Prolog, Datalog, answer set programming), functional programming, configuration management, and algebraic modeling systems.

Step1: create a file for variables

- ✓ **Create var.tf file and add the below code to it:**

```
# Defining CIDR Block for VPC

variable "vpc_cidr" {
  default = "10.0.0.0/16"
}


# Defining CIDR Block for 1st subnet

variable "subnet1_cidr" {
  default = "10.0.1.0/24"
}


# Defining CIDR Block for 2nd subnet

variable "subnet2_cidr" {
  default = "10.0.2.0/24"
}


# Defining CIDR Block for 3rd subnet

variable "subnet3_cidr" {
  default = "10.0.3.0/24"
}


# Defining CIDR Block for 4th subnet
```

```
variable "subnet4_cidr" {
  default = "10.0.4.0/24"
}
```

Defining CIDR Block for 5th subnet

```
variable "subnet5_cidr" {
  default = "10.0.5.0/24"
}
```

Defining CIDR Block for 6th subnet

```
variable "subnet6_cidr" {
  default = "10.0.6.0/24"
}
```

Step 2: Create a file for the Provider

✓ Create provider.tf file and add the below code to it:

```
provider "aws" {

region = "us-east-1"

access_key = "AKIA5FTZEZ3GKDZUQI4U"

secret_key = "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"

}
```

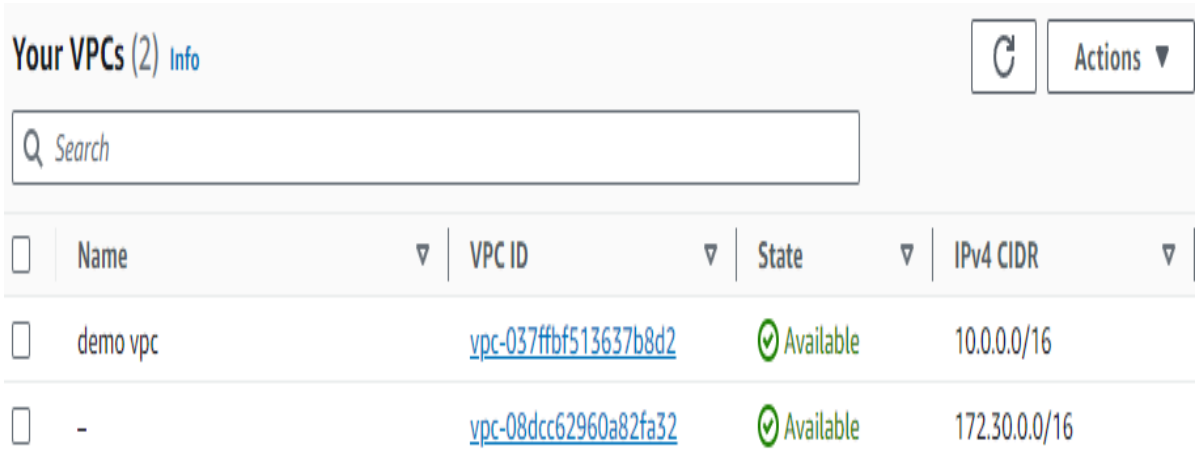
Step 3: Create a file for the VPC

✓ Create vpc.tf file and add the below code to it:

Creating VPC

```
resource "aws_vpc" "demovpc" {

cidr_block = var.vpc_cidr

instance_tenancy = "default"

tags = {

    Name = "demo vpc"
  }
 }
```

```
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_vpc.demovpc: Creating...
aws_vpc.demovpc: Creation complete after 1s [id=vpc-037ffbf513637b8d2]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-3-38 ~]$
```



Step4: Create a file for the subnet

- ✓ Create subnet.tf file and add the below code to it:
- ✓ Creating 1st-subnet(web subnet)
- ✓ Creating 2nd-subnet(web subnet)
- ✓ Creating 1st-application subnet
- ✓ Creating 2nd-application subnet
- ✓ Creating database private subnet
- ✓ Creating database private subnet

```
# Create web subnet1

resource "aws_subnet" "public_subnet-1” {

vpc_id  = aws_vpc.demovpc.id

cidr_block = var.subnet1_cidr

map_public_ip_on_launch = true

availability_zone = "us-east-1a"

 tags = {

    Name = "web subnet 1"

    }
  }
# Create web subnet2

resource "aws_subnet" "public_subnet-2" {
```

```
  vpc_id = aws_vpc.demovpc.id
 cidr_block = var.subnet2_cidr
 map_public_ip_on_launch = true
 availability_zone = "us-east-1b"
  tags = {
     Name = "web subnet 2"
      }
    }
# Create application subnet1
 resource "aws_subnet" "application_subnet-1" {
 vpc_id = aws_vpc.demovpc.id
 cidr_block = var.subnet3_cidr
 map_public_ip_on_launch = false
 availability_zone = "us-east-1a"
 tags = {
   Name = "application subnet 1"
     }
    }
# Create application subnet2
  resource "aws_subnet" "application_subnet-2" {
  vpc_id = aws_vpc.demovpc.id
  cidr_block = var.subnet4_cidr
```

```
    map_public_ip_on_launch = false
    availability_zone = "us-east-1b"
    tags = {
        Name = "application subnet 2"
         }
        }
# Create Database private subnet
     resource "aws_subnet" "database_subnet-1" {
     vpc_id = aws_vpc.demovpc.id
    cidr_block = var.subnet5_cidr
   availability_zone = "us-east-1a"
   tags = {
     Name = "database subnet1"
        }
       }
# create Database private subnet
      resource "aws_subnet" "database_subnet-2" {
      vpc_id = aws_vpc.demovpc.id
      cidr_block = var.subnet6_cidr
      availability_zone = "us-east-1b"
       tags = {
          Name = "database subnet1"
          }
```

}

```
aws_subnet.database_subnet-1: Creating...
aws_subnet.public_subnet-2: Creating...
aws_subnet.database_subnet-2: Creating...
aws_subnet.public_subnet-1: Creating...
aws_subnet.application_subnet-1: Creating...
aws_subnet.application_subnet-2: Creating...
aws_subnet.database_subnet-1: Creation complete after 0s [id=subnet-0fcd0ce35d4304c56]
aws_subnet.application_subnet-1: Creation complete after 0s [id=subnet-03c62158e56fd4979]
aws_subnet.database_subnet-2: Creation complete after 0s [id=subnet-0759940b60f266c12]
aws_subnet.application_subnet-2: Creation complete after 0s [id=subnet-0897264f338355e0c]
aws_subnet.public_subnet-1: Still creating... [10s elapsed]
aws_subnet.public_subnet-2: Still creating... [10s elapsed]
aws_subnet.public_subnet-2: Creation complete after 10s [id=subnet-03f85b9f8e45773ec]
aws_subnet.public_subnet-1: Creation complete after 11s [id=subnet-0ff27f3ac2d5e0284]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-3-38 ~]$
```

Subnets (12) Info

Actions ▼ | Create subnet

Find resources by attribute or tag

1

Name	Subnet ID	State	VPC	IPv4
database subnet1	subnet-0fcd0ce35d4304c56	Available	vpc-037ffbf513637b8d2 | dem...	10.0
–	subnet-0883c066f13c7d8d6	Available	vpc-08dcc62960a82fa32	172.
database subnet1	subnet-0759940b60f266c12	Available	vpc-037ffbf513637b8d2 | dem...	10.0
–	subnet-0c29c1fbcd41446ca	Available	vpc-08dcc62960a82fa32	172.
–	subnet-0942dfb406fa63fc9	Available	vpc-08dcc62960a82fa32	172.
application subnet 1	subnet-03c62158e56fd4979	Available	vpc-037ffbf513637b8d2 | dem...	10.0
application subnet 2	subnet-0897264f338355e0c	Available	vpc-037ffbf513637b8d2 | dem...	10.0
–	subnet-0ae2bca8eecabc170	Available	vpc-08dcc62960a82fa32	172.
web subnet 2	subnet-03f85b9f8e45773ec	Available	vpc-037ffbf513637b8d2 | dem...	10.0
web subnet 1	subnet-0ff27f3ac2d5e0284	Available	vpc-037ffbf513637b8d2 | dem...	10.0
–	subnet-066f698f3e92a7fc8	Available	vpc-08dcc62960a82fa32	172.

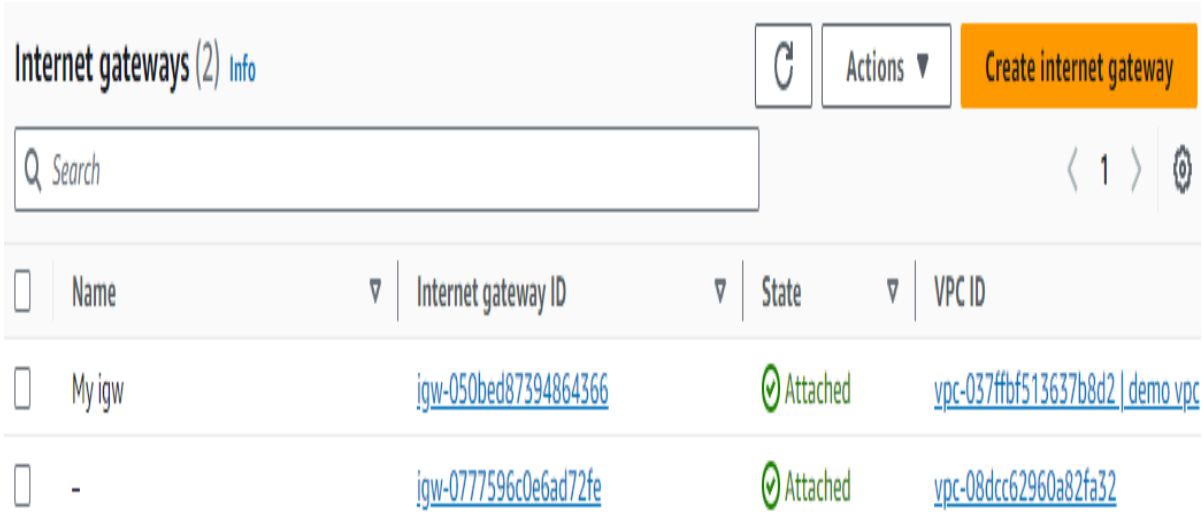
Step5: Create a file for the internet gateway

✓ **Create a file igw.tf and add the below code to it:**

```
# Creating internet gateway

resource "aws_internet_gateway" "demogateway" {
```

```
vpc_id = aws_vpc.demovpc.id

tags = {

   Name = "My igw"
   }
 }
```

```
aws_internet_gateway.demogateway: Destroying... [id=igw-0193617bf5da3c985]
aws_internet_gateway.demogateway: Destruction complete after 0s
aws_internet_gateway.demogateway: Creating...
aws_internet_gateway.demogateway: Creation complete after 1s [id=igw-050bed87394864366]

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



Step6: Create route_table

- ✓ **Create a file public.tf file and add the below code to it:**
- ✓ **# Creating route table**
- ✓ **# Associated route table(rt1)**
- ✓ **# Associated route table(rt2)**

 # Creating route table

```
resource "aws_route_table" "route" {
vpc_id = aws_vpc.demovpc.id
route {
cidr_block = "0.0.0.0/0"
gateway_id = aws_internet_gateway.demogateway.id
    }
tags = {
   Name = "route to internet"
    }
  }
```

 # Associating route table

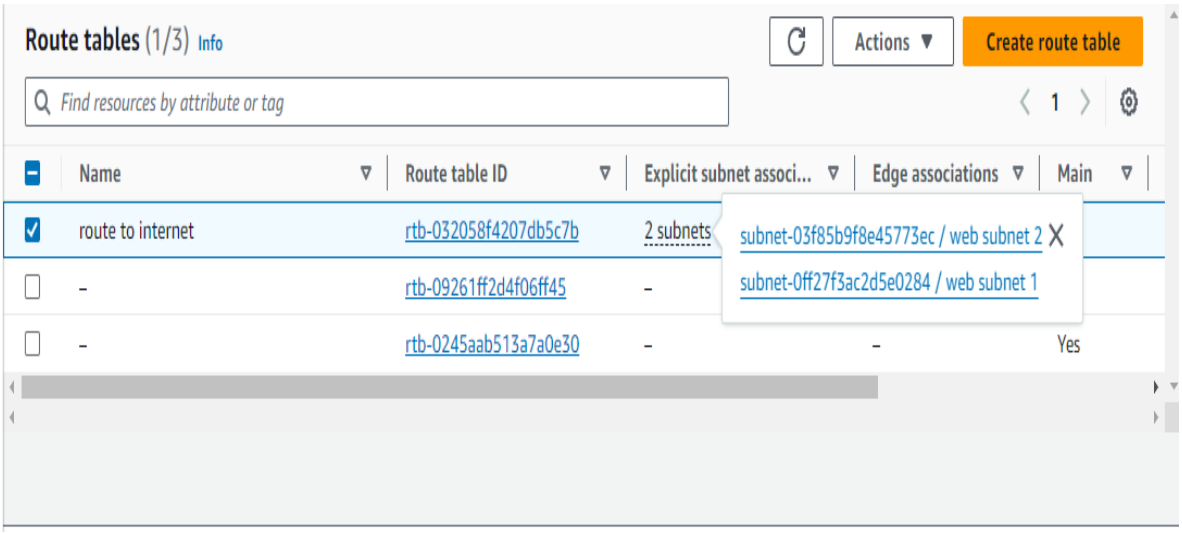
```
resource "aws_route_table_association" "rt1" {
subnet_id = aws_subnet.public_subnet-1.id
route_table_id = aws_route_table.route.id
  }
```

 # Associating route table

```
resource "aws_route_table_association" "rt2" {
```

```
subnet_id = aws_subnet.public_subnet-2.id

route_table_id = aws_route_table.route.id

  }
```

```
  Enter a value: yes

aws_route_table.route: Creating...
aws_route_table.route: Creation complete after 1s [id=rtb-032058f4207db5c7b]
aws_route_table_association.rt1: Creating...
aws_route_table_association.rt2: Creating...
aws_route_table_association.rt2: Still creating... [10s elapsed]
aws_route_table_association.rt1: Still creating... [10s elapsed]
aws_route_table_association.rt1: Creation complete after 13s [id=rtbassoc-08d16abff71e0f3ca]
aws_route_table_association.rt2: Creation complete after 13s [id=rtbassoc-0be990f37c012c338]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-3-38 ~]$
```



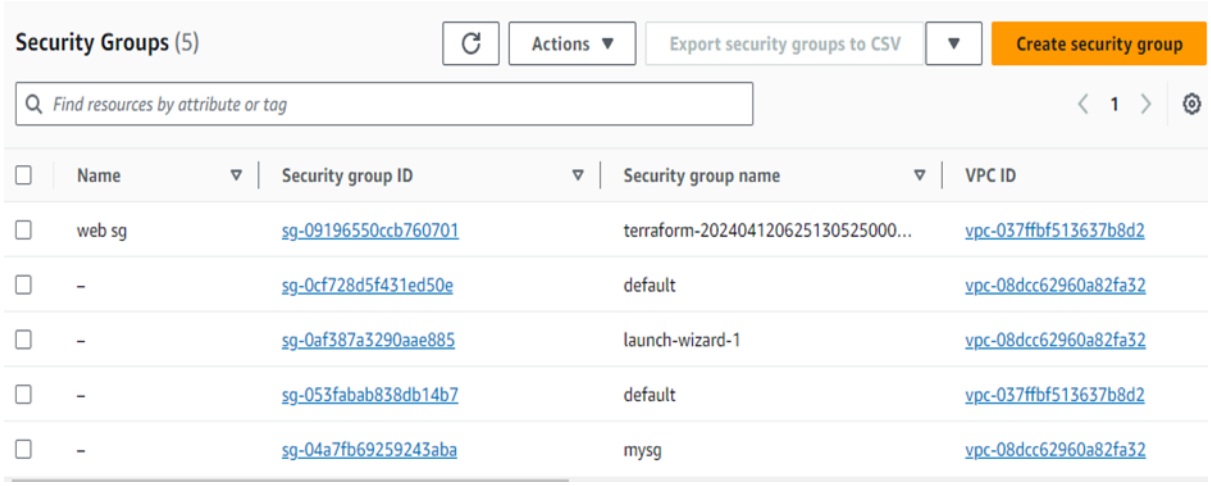
Step7: Creating a file for the security group

- ✓ Create web_sg.tf file and add the below code to it:
- ✓ # Creating security group
- ✓ # Inbound rules
- ✓ # HTTP access from anywhere
- ✓ # HTTPS access from anywhere
- ✓ # SSH access from anywhere
- ✓ # Outbound rules

```
# Creating security group

    resource "aws_security_group" "demosg" {

    vpc_id = aws_vpc.demovpc.id

 # Inbound rules

 # HTTP access from anywhere

   ingress {

   from_port   = 80

   to_port     = 80

   protocol    = "tcp"

   cidr_blocks = ["0.0.0.0/0"]

       }

# HTTPS access from anywhere

     ingress {

     from_port = 443

     to_port = 443

     protocol = "tcp"
```

```
    cidr_blocks = ["0.0.0.0/0"]
      }
# SSH access from anywhere
    ingress {
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
      }
# Outbound rules
# Internet access to anywhere
    egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
     }
     tags = {
       Name = "web sg"
     }
    }
```

```
aws_security_group.demosg: Creating...
aws_security_group.demosg: Creation complete after 2s [id=sg-09196550ccb760701]
aws_instance.demoinstance[0]: Creating...
aws_instance.demoinstance1[0]: Creating...
aws_instance.demoinstance[0]: Still creating... [10s elapsed]
aws_instance.demoinstance1[0]: Still creating... [10s elapsed]
aws_instance.demoinstance[0]: Still creating... [20s elapsed]
aws_instance.demoinstance1[0]: Still creating... [20s elapsed]
aws_instance.demoinstance[0]: Creation complete after 21s [id=i-036f4de085f8a4bce]
aws_instance.demoinstance1[0]: Still creating... [30s elapsed]
aws_instance.demoinstance1[0]: Creation complete after 31s [id=i-009badbe2971b576b]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-3-38 ~]$
```



Step8: Create a file for User Data

✓ **Create data.sh file and add the below code to it:**

```
#!/bin/bash
yum update -y
yum install -y httpd.x86_64
systemctl start httpd.service
systemctl enable httpd.service
```

```
  echo "hello world from $(hostname -f)" >
/var/www/html/index.html
```

Step9: Creating a file for EC2 instance

- ✓ Creating ec2.tf file and add the below code to it:

```
# creating instance
```

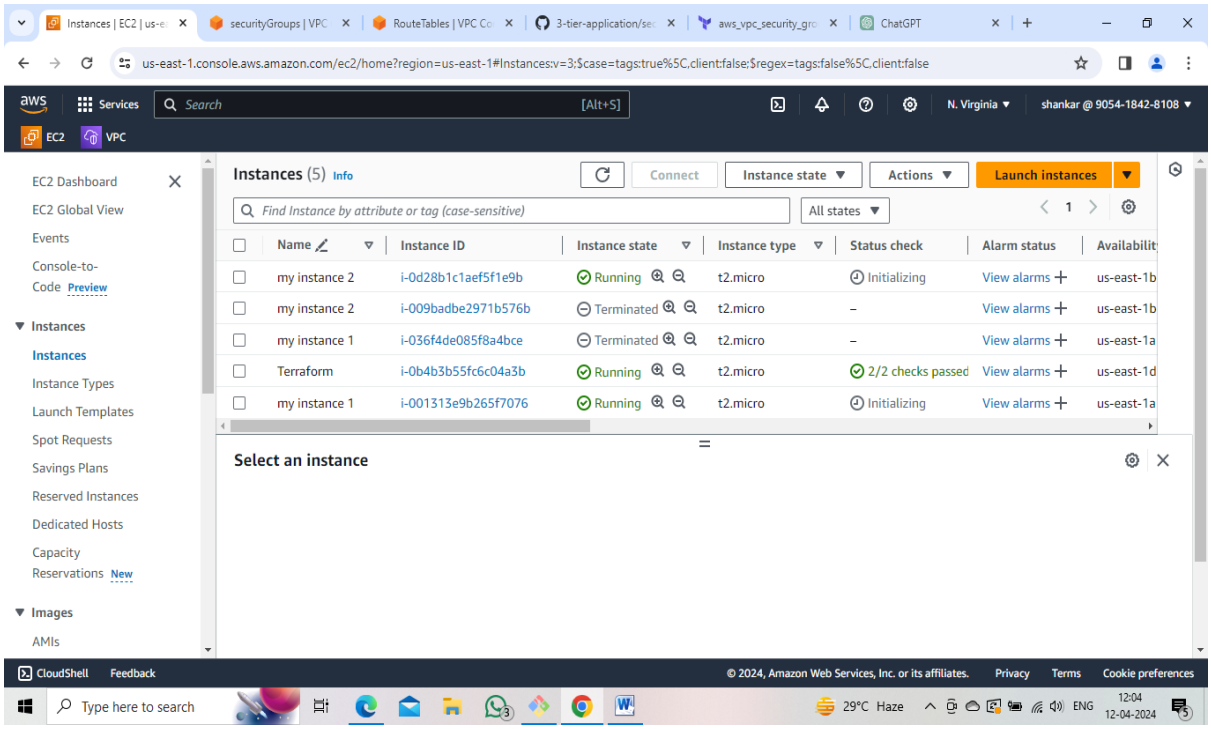
- ✓ Creating 1st EC2 instance in public subnet
- ✓ Creating 2nd EC2 instance in public subnet

```
# creating 1st instance in public subnet

resource "aws_instance" "demo-instance" {

ami = "ami-051f8a213df8bc089"

instance_type = "t2.micro"

count = 1

key_name= "tests-key"

security_groups= [aws_security_group.demosg.id]

subnet_id= aws_subnet.public_subnet-1.id

associate_public_ip_address = true
```

```
    user_data = file ("data.sh")
    tags = {
       Name = "instance1"
  }
}
   resource "aws_key_pair" "tests" {
   key_name   = "tests-key"
   public_key = "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXd
MsQz1x2cEikKDEY0aIj41qgxMCP/iteneqXSIFZBp5vizPvaoIR3Um9xK
7PGoW8giupGn+EPuxIA4cDM4vzOqOkiMPhz5XK0whEjkVzTo4+S0p
uvDZuwIsdiW9mxhJc7tgBNL0cYlWSYVkz4G/fslNfRPW5mYAM49f4f
htxPb5ok4Q2Lg9dPKVHO/Bgeu5woMc7RY0p1ej6D4CKFE6lymSDJp
W0YHX/wqE9+cfEauh7xZcG0q9t2ta6F6fmX0agvpFyZo8aFbXeUBr7o
sSCJNgvavWbM/06niWrOvYX2xwWdhXmXSrbX8ZbabVohBK41
email@example.com"
}
#creating 2nd ec2 instance in public subnet
   resource "aws_instance" "demo-instance1" {
   ami= "ami-051f8a213df8bc089"
   instance_type= "t2.micro"
   count = 1
   key_name= "tests-key"
   security_groups= [aws_security_group.demosg.id]
```

```
  subnet_id= aws_subnet.public_subnet-2.id

  associate_public_ip_address = true

  user_data= file ("data.sh")

  tags = {

    Name = "instance2"

 }

}
```

```
aws_instance.demoinstance1[0]: Destroying... [id=i-009badbe2971b576b]
aws_instance.demoinstance[0]: Destroying... [id=i-036f4de085f8a4bce]
aws_instance.demoinstance1[0]: Still destroying... [id=i-009badbe2971b576b, 10s elapsed]
aws_instance.demoinstance[0]: Still destroying... [id=i-036f4de085f8a4bce, 10s elapsed]
aws_instance.demoinstance1[0]: Still destroying... [id=i-009badbe2971b576b, 20s elapsed]
aws_instance.demoinstance[0]: Still destroying... [id=i-036f4de085f8a4bce, 20s elapsed]
aws_instance.demoinstance1[0]: Still destroying... [id=i-009badbe2971b576b, 30s elapsed]
aws_instance.demoinstance[0]: Still destroying... [id=i-036f4de085f8a4bce, 30s elapsed]
aws_instance.demoinstance1[0]: Still destroying... [id=i-009badbe2971b576b, 40s elapsed]
aws_instance.demoinstance[0]: Still destroying... [id=i-036f4de085f8a4bce, 40s elapsed]
aws_instance.demoinstance1[0]: Destruction complete after 40s
aws_instance.demoinstance1[0]: Creating...
aws_instance.demoinstance[0]: Destruction complete after 41s
aws_instance.demoinstance[0]: Creating...
aws_instance.demoinstance1[0]: Still creating... [10s elapsed]
aws_instance.demoinstance[0]: Still creating... [10s elapsed]
aws_instance.demoinstance1[0]: Still creating... [20s elapsed]
aws_instance.demoinstance[0]: Still creating... [20s elapsed]
aws_instance.demoinstance1[0]: Still creating... [30s elapsed]
aws_instance.demoinstance[0]: Still creating... [30s elapsed]
aws_instance.demoinstance[0]: Creation complete after 31s [id=i-001313e9b265f7076]
aws_instance.demoinstance1[0]: Creation complete after 31s [id=i-0d28b1c1aef5f1e9b]

Apply complete! Resources: 2 added, 0 changed, 2 destroyed.
[ec2-user@ip-172-30-3-38 ~]$
```



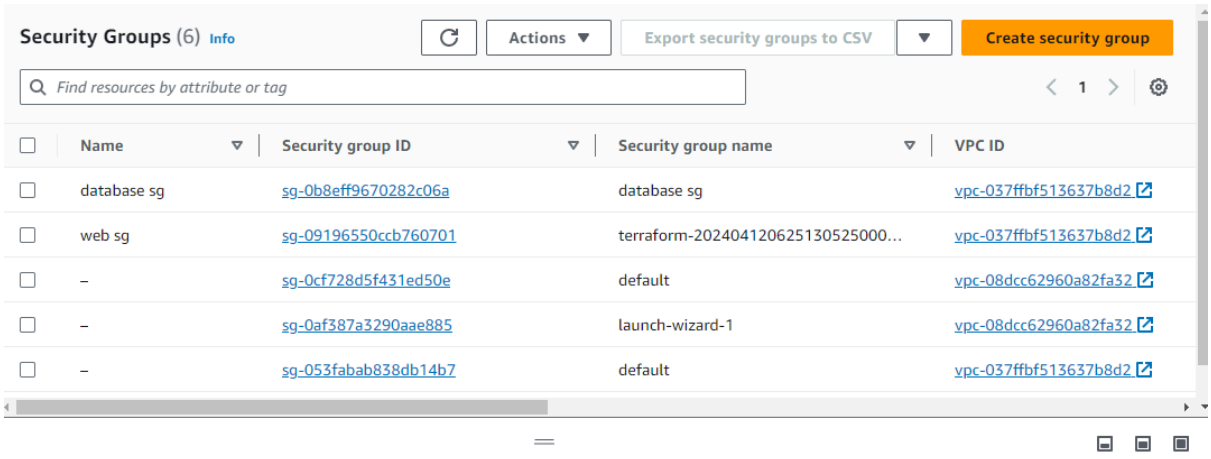
Step10: Create a file for security group for the database tier

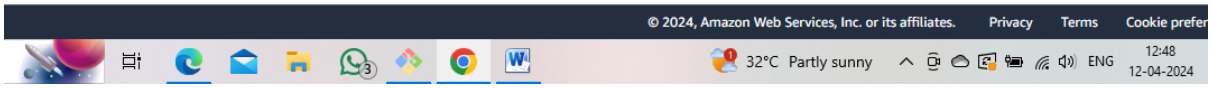
✓ **Create database_sg.tf and add the below code to it:**

```
# creating database

resource "aws_security_group" "database-sg" {

name  = "database sg"

description = "allow inbound traffic from application layer"

vpc_id  = aws_vpc.demovpc.id

ingress {

description  = "allow traffic from application layer"

from_port  = 3306
```

```
  to_port  = 3306

  protocol = "tcp"

  security_groups = [aws_security_group.demosg.id]

 }

  egress {

  from_port   = 32768

  to_port     = 65535

  protocol    = "tcp"

  cidr_blocks = ["0.0.0.0/0"]

 }

 tags = {

   Name = "database sg"

 }

}
```

```
aws_instance.demoinstance1[0]: Destroying... [id=i-0d28b1c1aef5f1e9b]
aws_instance.demoinstance[0]: Destroying... [id=i-001313e9b265f7076]
aws_security_group.database-sg: Creating...
aws_security_group.database-sg: Creation complete after 1s [id=sg-0b8eff9670282c06a]
aws_instance.demoinstance1[0]: Still destroying... [id=i-0d28b1c1aef5f1e9b, 10s elapsed]
aws_instance.demoinstance[0]: Still destroying... [id=i-001313e9b265f7076, 10s elapsed]
aws_instance.demoinstance[0]: Destruction complete after 14s
aws_instance.demoinstance[0]: Creating...
aws_instance.demoinstance1[0]: Destruction complete after 14s
aws_instance.demoinstance1[0]: Creating...
aws_instance.demoinstance[0]: Still creating... [10s elapsed]
aws_instance.demoinstance1[0]: Still creating... [10s elapsed]
aws_instance.demoinstance[0]: Still creating... [20s elapsed]
aws_instance.demoinstance1[0]: Still creating... [20s elapsed]
aws_instance.demoinstance[0]: Still creating... [30s elapsed]
aws_instance.demoinstance1[0]: Still creating... [30s elapsed]
aws_instance.demoinstance[0]: Creation complete after 31s [id=i-0271abcdc5257f9be]
aws_instance.demoinstance1[0]: Creation complete after 31s [id=i-03e7e147c6cd5c58e]

Apply complete! Resources: 3 added, 0 changed, 2 destroyed.
[ec2-user@ip-172-30-3-38 ~]$
```



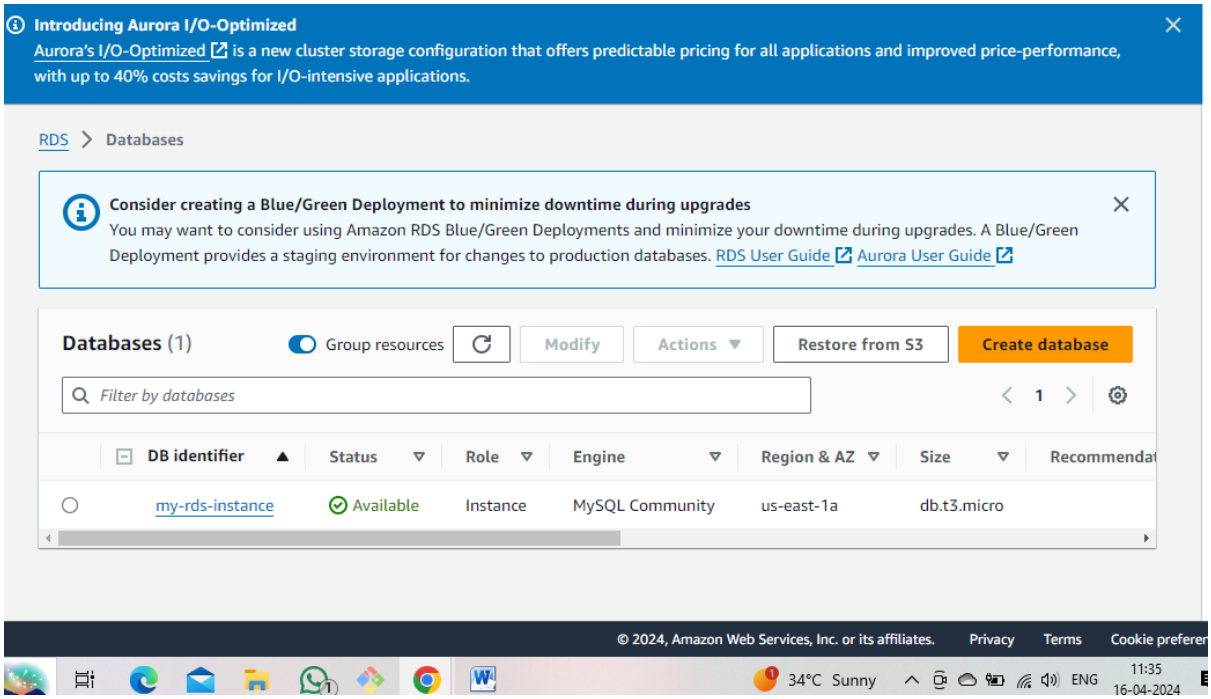


Step11: Create a file for RDS instance

✓ **Create a rds.tf file and add the below code to it:**

```
#Creating RDS instance

    resource "aws_db_subnet_group" "default" {

    name = "main"

    subnet_ids = [aws_subnet.database_subnet-1.id,
aws_subnet.database_subnet-2.id]

    tags = {

       Name = "my db_subnet_group"

  }

}
```

```
resource "aws_db_instance" "my_rds" {
identifier = "my-rds-instance"
allocated_storage = 10
db_subnet_group_name = aws_db_subnet_group.default.name
engine = "mysql"
engine_version = "8.0.34"
instance_class = "db.t3.micro"
multi_az = "true"
username = "Pavani"
password = "Pavani123#"
skip_final_snapshot = true
vpc_security_group_ids = [aws_security_group.database-sg.id]
}
```



Step12: Create a file for output file

✓ **Create outputs.tf file and add the below code to it:**

```
#getting DNS of load balancer

  output "lb_dns_name" {

  description = " the DNS name of the load balancer"

  value = aws_lb.LB-elb.dns_name

}
```

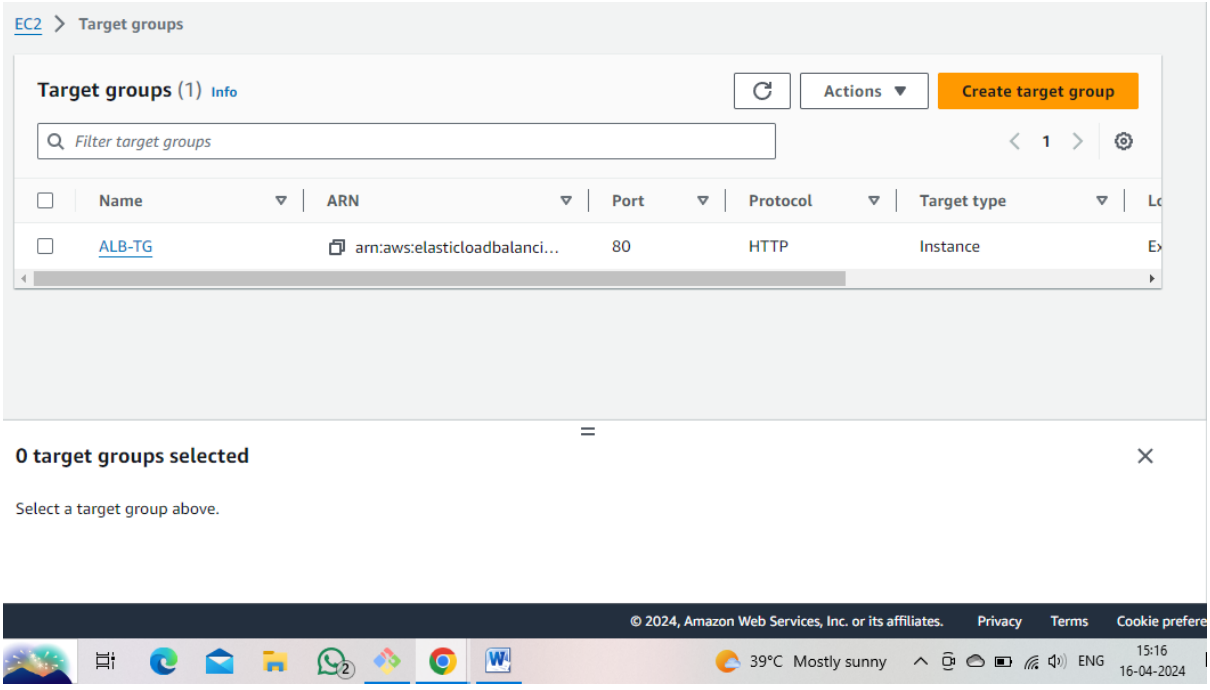
Step13: Create a file application load balancer

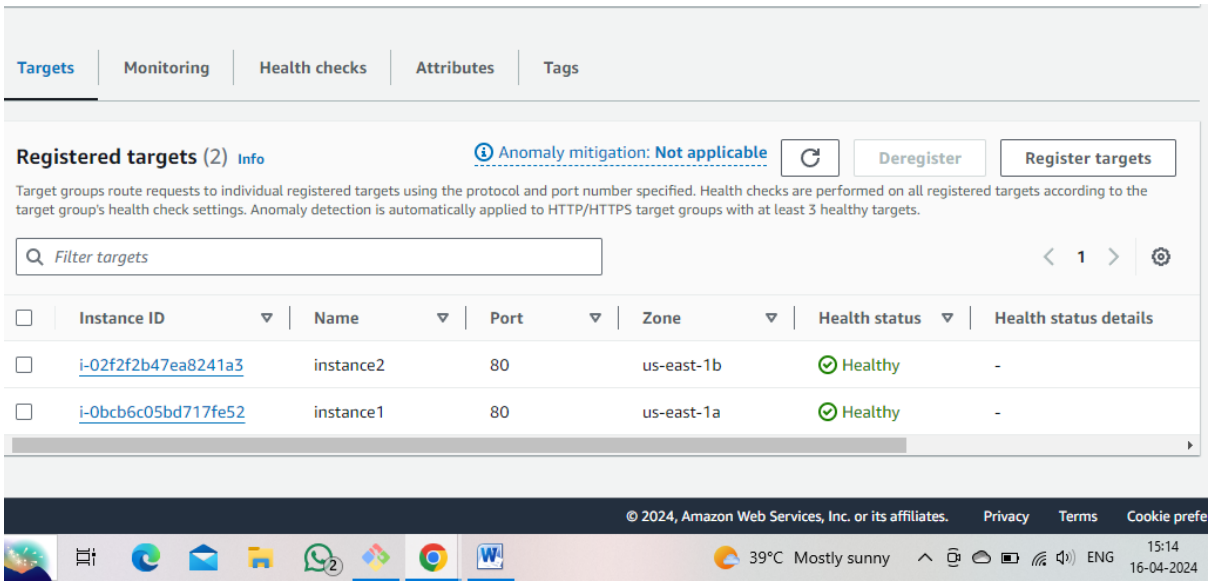
✓ **Create alb.tf file and add the below code to it:**

```
# Creating external load balancer
 resource "aws_lb" "LB-elb" {
  name = "External-LB"
  internal  = false
 load_balancer_type = "application"
 security_groups = [aws_security_group.demosg.id]
 subnets = [aws_subnet.public_subnet-1.id,
aws_subnet.public_subnet-2.id]
}
resource "aws_lb_target_group" "tg-elb" {
 name = "ALB-TG"
 port = 80
 protocol = "HTTP"
 vpc_id   = aws_vpc.demovpc.id
}
resource "aws_lb_target_group_attachment" "attachment" {
 count = length (aws_instance.demo-instance)
 target_group_arn = aws_lb_target_group.tg-elb.arn
 target_id = aws_instance.demo-instance [count.index].id
 port = 80
 depends_on = [aws_instance.demo-instance]
}
```

```
 resource "aws_lb_target_group_attachment" "attachment1" {
 count = length (aws_instance.demo-instance1)
 target_group_arn = aws_lb_target_group.tg-elb.arn
 target_id = aws_instance.demo-instance1 [count.index].id
 port = 80
 depends_on = [aws_instance.demo-instance1]
}
resource "aws_lb_listener" "LB-elb" {
 load_balancer_arn = aws_lb.LB-elb.arn
 port  = 80
 protocol = "HTTP"
 default_action {
  type = "forward"
  target_group_arn = aws_lb_target_group.tg-elb.arn
 }
}
```

```
ws_instance.demo-instance[0]: Creating...
ws_instance.demo-instance1[0]: Creating...
ws_instance.demo-instance[0]: Still creating... [10s elapsed]
ws_instance.demo-instance1[0]: Still creating... [10s elapsed]
ws_instance.demo-instance[0]: Still creating... [20s elapsed]
ws_instance.demo-instance1[0]: Still creating... [20s elapsed]
ws_instance.demo-instance[0]: Creation complete after 22s [id=i-0bcb6c05bd717fe52]
ws_lb_target_group_attachment.attachment[0]: Creating...
ws_lb_target_group_attachment.attachment[0]: Creation complete after 0s [id=arn:aws:elasticloadbalanci
g:us-east-1:905418428108:targetgroup/ALB-TG/a5711365c9363e8f-20240416093726759200000003]
ws_instance.demo-instance1[0]: Still creating... [30s elapsed]
ws_instance.demo-instance1[0]: Creation complete after 32s [id=i-02f2f2b47ea8241a3]
ws_lb_target_group_attachment.attachment1[0]: Creating...
ws_lb_target_group_attachment.attachment1[0]: Creation complete after 0s [id=arn:aws:elasticloadbalanc
ng:us-east-1:905418428108:targetgroup/ALB-TG/a5711365c9363e8f-20240416093736899500000004]

pply complete! Resources: 4 added, 0 changed, 0 destroyed.

utputs:

b_dns_name = "External-LB-2066826496.us-east-1.elb.amazonaws.com"
ec2-user@ip-172-30-3-38 ~]$
```

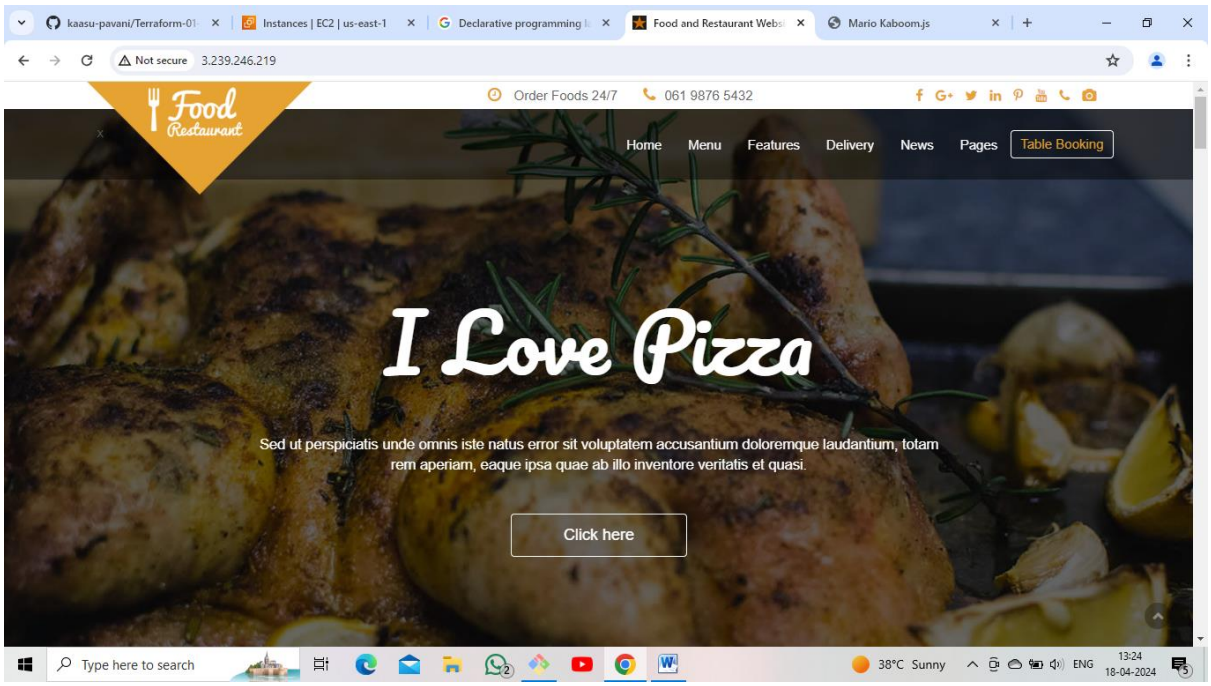




hello world from ip-10-0-1-211.ec2.internal

Deploying an Application1:

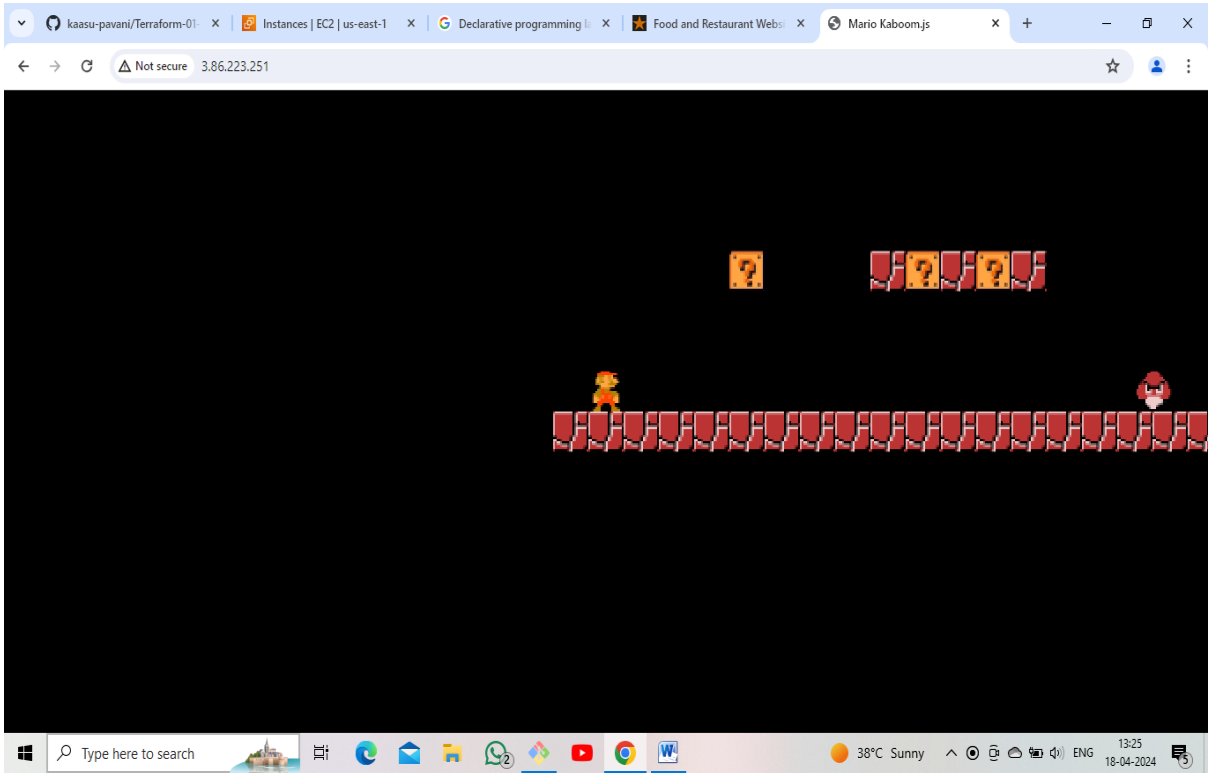
```
#!bin/bash
yum update -y
yum install -y git
yum install -y httpd.x86_64
systemctl start httpd.service
systemctl enable httpd.service
git clone https://github.com/GOUSERABBANI44/food.git /var/www/html/
~
```



Deploying an Application2:

```
#!bin/bash
yum update -y
yum install -y git
yum install -y httpd.x86_64
systemctl start httpd.service
systemctl enable httpd.service
git clone https://github.com/GOUSERABBANI44/Mario.git /var/www/html/
~
```

Deploying an Application 1



**********THE END**********