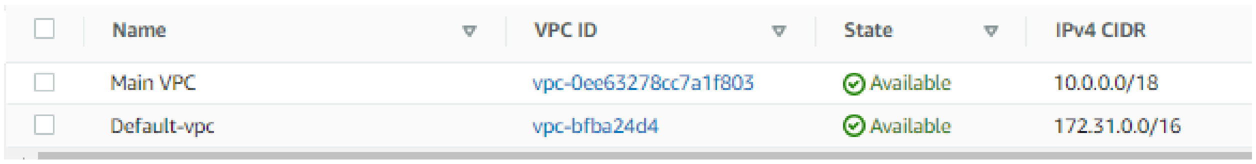
**Part 1: VPC**
**Create a new VPC with:**
*** 5 subnets (2 public, 1 private, 2 internal)**
*** 2 route tables (public & private)**
*** an Internet Gateway**
*** and 1 NAT Gateway (in 1 of the private subnets)**

- First step is to create provider.tf to configure

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-2"
  default_tags {
    tags = {
      Deployment = "DEPLOYMENT_09_TERRAFORM"
      Team       = "Kura Labs"
    }
  }
}
```

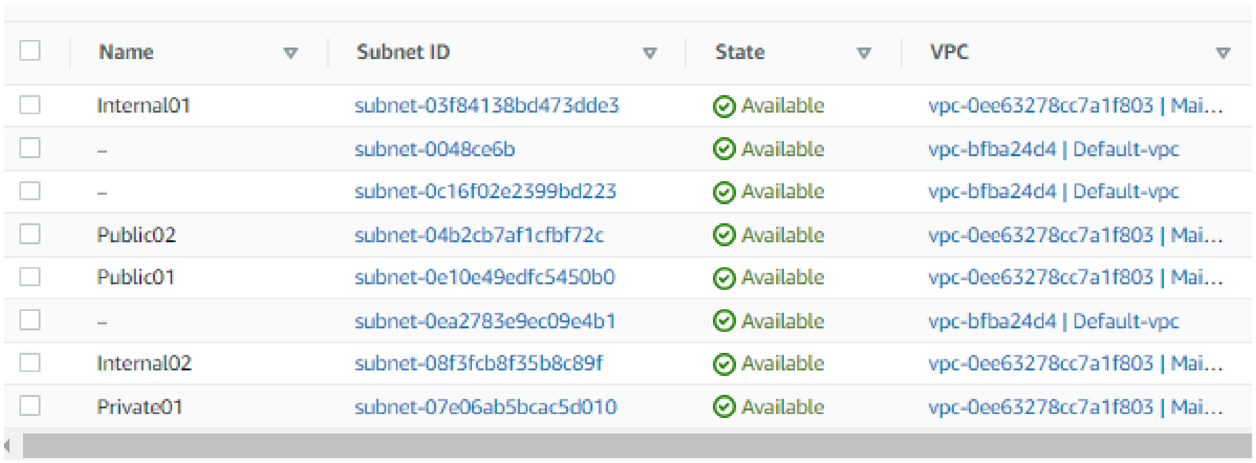
Then we created the resources

- Created a main vpc



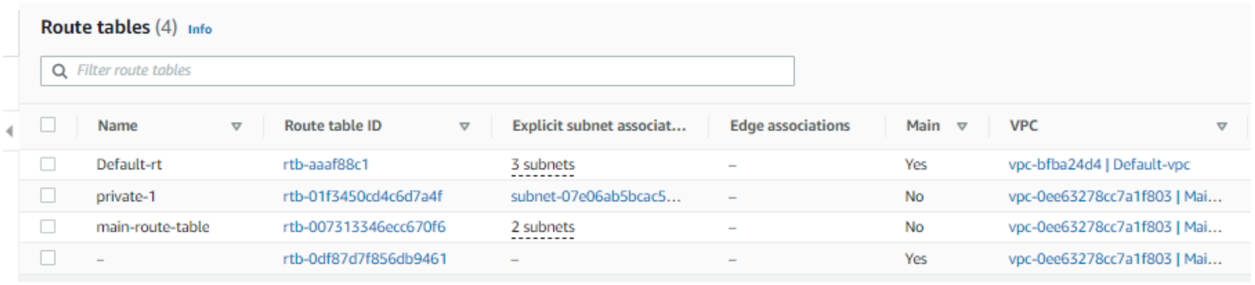
| Name | VPC ID | State | IPv4 CIDR |
|---|---|---|---|
| Main VPC | vpc-0ee63278cc7a1f803 | Available | 10.0.0.0/18 |
| Default-vpc | vpc-bfba24d4 | Available | 172.31.0.0/16 |

- Created 5 subnets ( 2 public, 2 internal, and 1 private).



| Name | Subnet ID | State | VPC |
|---|---|---|---|
| Internal01 | subnet-03f84138bd473dde3 | ✓ Available | vpc-0ee63278cc7a1f803 \| Mai… |
| – | subnet-0048ce6b | ✓ Available | vpc-bfba24d4 \| Default-vpc |
| – | subnet-0c16f02e2399bd223 | ✓ Available | vpc-bfba24d4 \| Default-vpc |
| Public02 | subnet-04b2cb7af1cfbf72c | ✓ Available | vpc-0ee63278cc7a1f803 \| Mai… |
| Public01 | subnet-0e10e49edfc5450b0 | ✓ Available | vpc-0ee63278cc7a1f803 \| Mai… |
| – | subnet-0ea2783e9ec09e4b1 | ✓ Available | vpc-bfba24d4 \| Default-vpc |
| Internal02 | subnet-08f3fcb8f35b8c89f | ✓ Available | vpc-0ee63278cc7a1f803 \| Mai… |
| Private01 | subnet-07e06ab5bcac5d010 | ✓ Available | vpc-0ee63278cc7a1f803 \| Mai… |

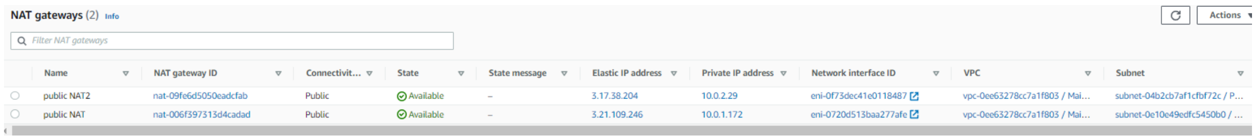
- 2 route tables (main and privates)



**Route tables** (4) Info

Filter route tables

| Name | Route table ID | Explicit subnet associat… | Edge associations | Main | VPC |
|---|---|---|---|---|---|
| Default-rt | rtb-aaaf88c1 | 3 subnets | – | Yes | vpc-bfba24d4 \| Default-vpc |
| private-1 | rtb-01f3450cd4c6d7a4f | subnet-07e06ab5bcac5… | – | No | vpc-0ee63278cc7a1f803 \| Mai… |
| main-route-table | rtb-007313346ecc670f6 | 2 subnets | – | No | vpc-0ee63278cc7a1f803 \| Mai… |
| – | rtb-0df87d7f856db9461 | – | – | Yes | vpc-0ee63278cc7a1f803 \| Mai… |

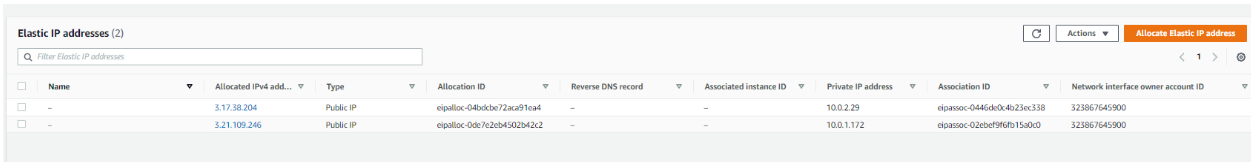


**NAT gateways** (2) Info

Filter NAT gateways

| Name | NAT gateway ID | Connectivit… | State | State message | Elastic IP address | Private IP address | Network interface ID | VPC | Subnet |
|---|---|---|---|---|---|---|---|---|---|
| public NAT2 | nat-09fe6d5050eadcfab | Public | ✓ Available | – | 3.17.38.204 | 10.0.2.29 | eni-0f73dec41e0118487 | vpc-0ee63278cc7a1f803 / Mai… | subnet-04b2cb7af1cfbf72c / P… |
| public NAT | nat-006f397313d4cadad | Public | ✓ Available | – | 3.21.109.246 | 10.0.1.172 | eni-0720d513baa277afe | vpc-0ee63278cc7a1f803 / Mai… | subnet-0e10e49edfc5450b0 / … |

- Created 2elastic ips for the nat gateways.

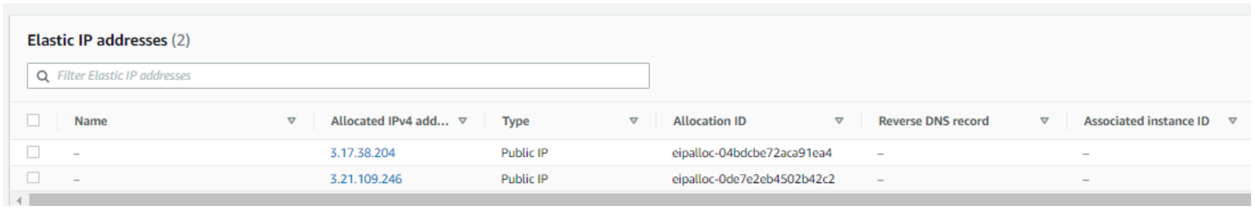


**Elastic IP addresses** (2)

Filter Elastic IP addresses

| Name | Allocated IPv4 add… | Type | Allocation ID | Reverse DNS record | Associated instance ID | Private IP address | Association ID | Network interface owner account ID |
|---|---|---|---|---|---|---|---|---|
| – | 3.17.38.204 | Public IP | eipalloc-04bdcbe72aca91ea4 | – | – | 10.0.2.29 | eipassoc-0446de0c4b23ec338 | 323867645900 |
| – | 3.21.109.246 | Public IP | eipalloc-0de7e2eb4502b42c2 | – | – | 10.0.1.172 | eipassoc-02ebef9f6fb15a0c0 | 323867645900 |

- Created 2 elastic ips



**Elastic IP addresses** (2)

Filter Elastic IP addresses

| Name | Allocated IPv4 add… | Type | Allocation ID | Reverse DNS record | Associated instance ID |
|---|---|---|---|---|---|
| – | 3.17.38.204 | Public IP | eipalloc-04bdcbe72aca91ea4 | – | – |
| – | 3.21.109.246 | Public IP | eipalloc-0de7e2eb4502b42c2 | – | – |

My code for vpc.tf

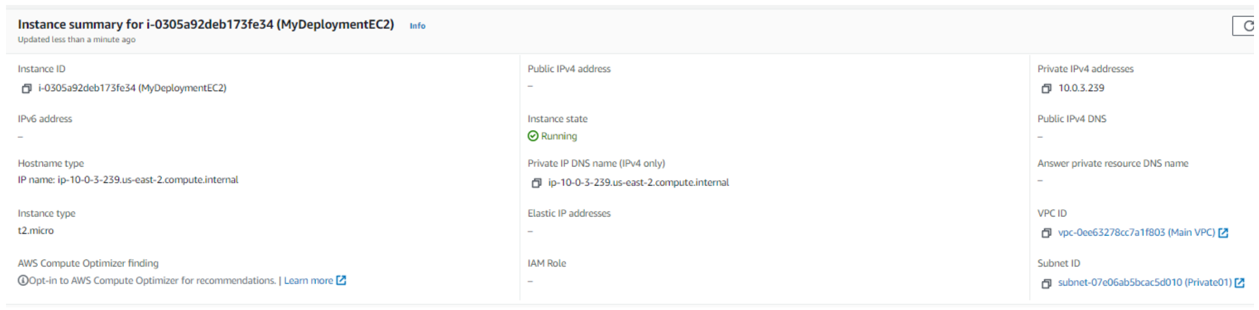
```
# 1VPC
#
https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resource
s/vpc
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/18"

  tags = {
    Name = "Main VPC"
  }
}

#public1
resource "aws_subnet" "public01" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "us-east-2a"
  map_public_ip_on_launch = true
  tags = {
    "Name" = "Public01"
  }
}

#public2
resource "aws_subnet" "public02" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.2.0/24"
  availability_zone       = "us-east-2b"
  map_public_ip_on_launch = true

  tags = {
    "Name" = "Public02"
  }
}

#Private1
resource "aws_subnet" "private01" {
  vpc_id            = aws_vpc.main.id
  cidr_block         = "10.0.3.0/24"
  availability_zone = "us-east-2a"
  tags = {
    Name = "Private01"
  }
}

#Internal1
resource "aws_subnet" "internal01" {
  vpc_id            = aws_vpc.main.id
  cidr_block         = "10.0.4.0/24"
  availability_zone = "us-east-2a"
  tags = {
    Name = "Internal01"
  }
}

#Internal2
resource "aws_subnet" "internal02" {
  vpc_id            = aws_vpc.main.id
  cidr_block         = "10.0.5.0/24"
  availability_zone = "us-east-2b"
  tags = {
    Name = "Internal02"
  }
}




#2. create internet gatewat
resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "ig1"
  }
}

resource "aws_eip" "nat1" {
  # EIP may require IGW to exist prior to association.
  # Use depends_on to set an explicit dependency on the IGW.
  depends_on = [aws_internet_gateway.main]
}

resource "aws_eip" "nat2" {
  # EIP may require IGW to exist prior to association.
  # Use depends_on to set an explicit dependency on the IGW.
  depends_on = [aws_internet_gateway.main]
}

#3. Create a public Nat gateways for private subnet, created 2 for the
failover
resource "aws_nat_gateway" "gw1" {
  allocation_id = aws_eip.nat1.id
  subnet_id     = aws_subnet.public01.id #public subnet

  tags = {
    Name = "public NAT"
  }
}

resource "aws_nat_gateway" "gw2" {
  allocation_id = aws_eip.nat2.id
  subnet_id     = aws_subnet.public02.id #public subnet

  tags = {
    Name = "public NAT2"
  }


  # To ensure proper ordering, it is recommended to add an explicit
dependency
  # on the Internet Gateway for the VPC.
  depends_on = [aws_internet_gateway.main]
}



# 4. Create  Route Tables
```

```
#main route table/public route table
resource "aws_route_table" "main-route-table" {
  vpc_id = aws_vpc.main.id

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

  tags = {
    "Name" = "main-route-table"
  }
}


resource "aws_route_table" "private-rt-01" {
  # The VPC ID.
  vpc_id = aws_vpc.main.id

  route {
    # The CIDR block of the route.
    cidr_block = "0.0.0.0/0"

    # Identifier of a VPC NAT gateway.
    nat_gateway_id = aws_nat_gateway.gw1.id
  }

  # A map of tags to assign to the resource.
  tags = {
    Name = "private-1"
  }
}

# 5. Create  Route Tables associations
resource "aws_route_table_association" "public1" {
  subnet_id = aws_subnet.public01.id
  # The ID of the routing table to associate with.
  route_table_id = aws_route_table.main-route-table.id
}
```

```
resource "aws_route_table_association" "public2" {
  subnet_id = aws_subnet.public02.id
  # The ID of the routing table to associate with.
  route_table_id = aws_route_table.main-route-table.id
}

resource "aws_route_table_association" "private1" {
  # The subnet ID to create an association.
  subnet_id = aws_subnet.private01.id

  # The ID of the routing table to associate with.
  route_table_id = aws_route_table.private-rt-01.id
}
```

**Part 2 Ec2**

- Created a ec2 instance that was an Ubuntu ami
  - It had a security group with the following rules:
    - Ingress: allow port 80 traffic from the ALB security group'
    - Egress: allow all outbound traffic to any ipv4 address



**Instance summary for i-0305a92deb173fe34 (MyDeploymentEC2)** Info
Updated less than a minute ago

Instance ID
i-0305a92deb173fe34 (MyDeploymentEC2)

IPv6 address
–

Hostname type
IP name: ip-10-0-3-239.us-east-2.compute.internal

Instance type
t2.micro

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Public IPv4 address
–

Instance state
Running

Private IP DNS name (IPv4 only)
ip-10-0-3-239.us-east-2.compute.internal

Elastic IP addresses
–

IAM Role
–

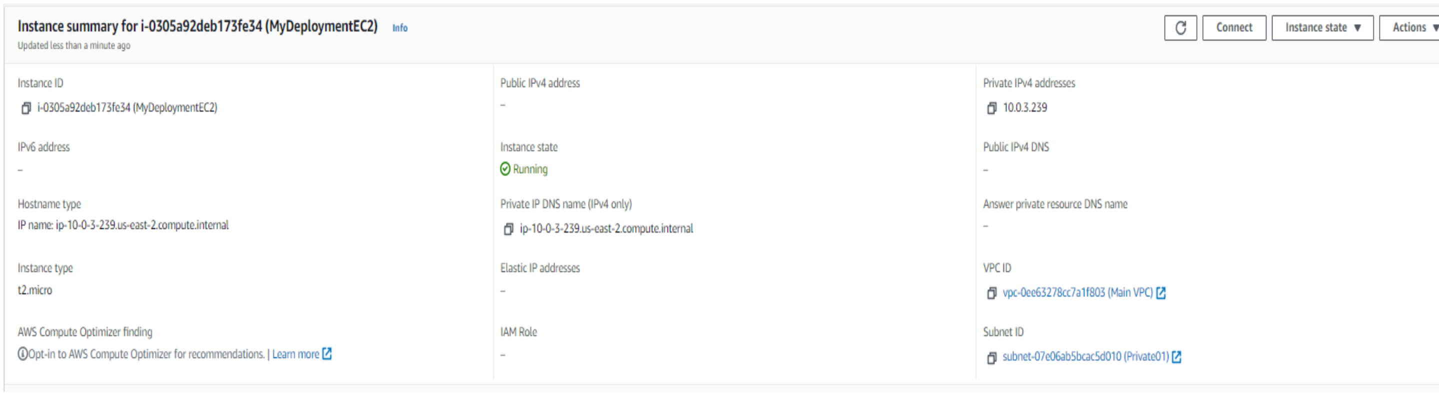
Private IPv4 addresses
10.0.3.239

Public IPv4 DNS
–

Answer private resource DNS name
–

VPC ID
vpc-0ee63278cc7a1f803 (Main VPC)

Subnet ID
subnet-07e06ab5bcac5d010 (Private01)



Instance summary for i-0305a92deb173fe34 (MyDeploymentEC2) Info

Updated less than a minute ago

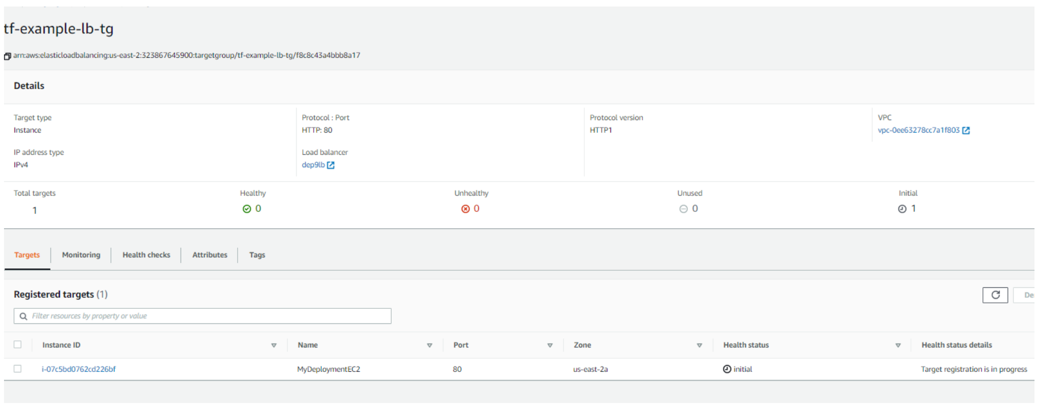
| | | |
|---|---|---|
| Instance ID | Public IPv4 address | Private IPv4 addresses |
| i-0305a92deb173fe34 (MyDeploymentEC2) | – | 10.0.3.239 |
| IPv6 address | Instance state | Public IPv4 DNS |
| – | Running | – |
| Hostname type | Private IP DNS name (IPv4 only) | Answer private resource DNS name |
| IP name: ip-10-0-3-239.us-east-2.compute.internal | ip-10-0-3-239.us-east-2.compute.internal | – |
| Instance type | Elastic IP addresses | VPC ID |
| t2.micro | – | vpc-0ee63278cc7a1f803 (Main VPC) |
| AWS Compute Optimizer finding | IAM Role | Subnet ID |
| Opt-in to AWS Compute Optimizer for recommendations. \| Learn more | – | subnet-07e06ab5bcac5d010 (Private01) |

```
#security group for ubuntu ec2
resource "aws_security_group" "ubuntu_ec2" {
  name        = "ubuntu_ec2"
  description = "Allow port 80 inbound traffic"
  vpc_id      = aws_vpc.main.id

  ingress {
    description = "TCP from VPC"
    from_port   = 80
    to_port     = 80
    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 = "MyDep9UbuntuEc2"
  }
}

data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "Dep9EC2" {
  ami             = "ami-0629230e074c580f2"
  instance_type   = "t2.micro"
  key_name        = "Python"
  security_groups = [aws_security_group.ubuntu_ec2.id]
  subnet_id       = aws_subnet.private01.id

  tags = {
    Name = "MyDeploymentEC2"
  }
}
```

**Part 3 :ALB**

created a security group

- Created a load balancer security group that allows inbound port 80 ipv4 traffic
- The  load balancer security group ingress rules that allow only port 80 access from any ipv4 traffic.  And it’s egress rules were that it allowed outbound traffic to port 80 only to the deployment 9 Ec2 instance security group.



tf-example-lb-tg

arn:aws:elasticloadbalancing:us-east-2:323867645900:targetgroup/tf-example-lb-tg/f8c8c43a4bbb8a17

**Details**

| | | | |
|---|---|---|---|
| Target type<br>Instance | Protocol : Port<br>HTTP: 80 | Protocol version<br>HTTP1 | VPC<br>vpc-0ee63278cc7a1f803 |
| IP address type<br>IPv4 | Load balancer<br>dep9lb | | |

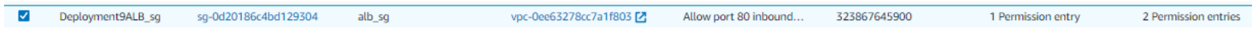
| Total targets | Healthy | Unhealthy | Unused | Initial |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 |

Targets | Monitoring | Health checks | Attributes | Tags

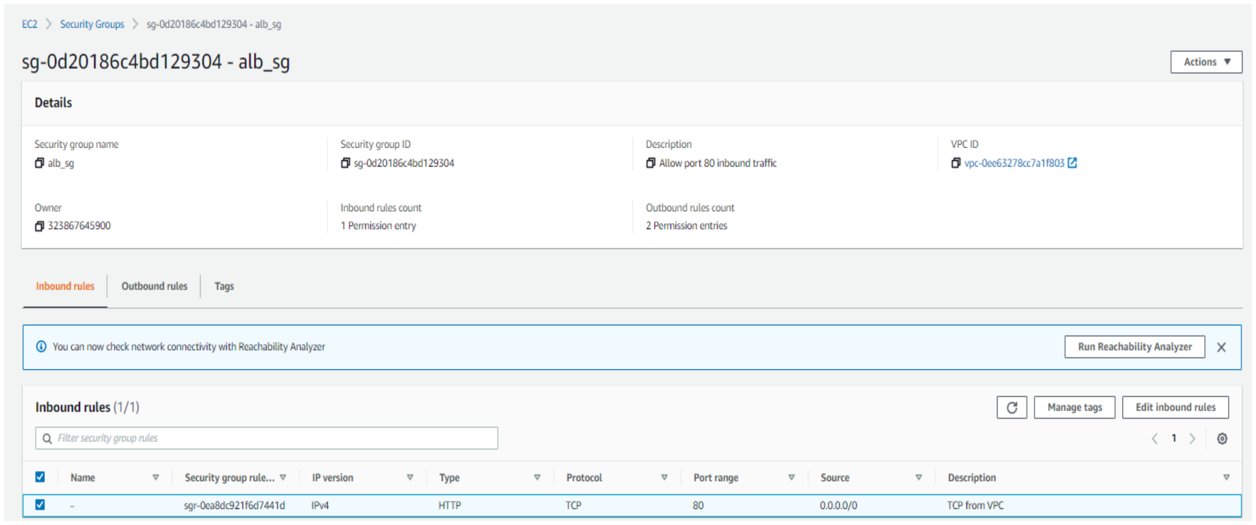
**Registered targets** (1)

Filter resources by property or value

| Instance ID | Name | Port | Zone | Health status | Health status details |
|---|---|---|---|---|---|
| i-07c5bd0762cd226bf | MyDeploymentEC2 | 80 | us-east-2a | initial | Target registration is in progress |



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ✓ | Deployment9ALB_sg | sg-0d20186c4bd129304 | alb_sg | vpc-0ee63278cc7a1f803 | Allow port 80 inbound… | 323867645900 | 1 Permission entry | 2 Permission entries |



EC2 > Security Groups > sg-0d20186c4bd129304 - alb_sg

sg-0d20186c4bd129304 - alb_sg

Actions ▼

**Details**

| | | | |
|---|---|---|---|
| Security group name<br>alb_sg | Security group ID<br>sg-0d20186c4bd129304 | Description<br>Allow port 80 inbound traffic | VPC ID<br>vpc-0ee63278cc7a1f803 |
| Owner<br>323867645900 | Inbound rules count<br>1 Permission entry | Outbound rules count<br>2 Permission entries | |

Inbound rules | Outbound rules | Tags

You can now check network connectivity with Reachability Analyzer

Run Reachability Analyzer

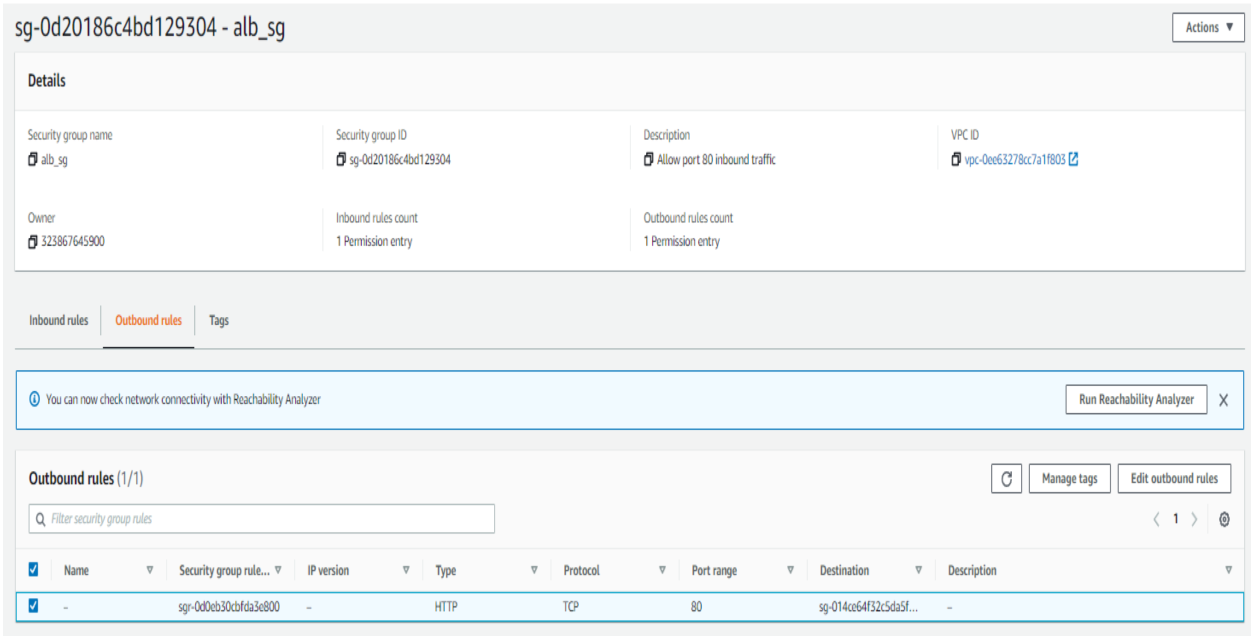
**Inbound rules** (1/1)

Manage tags | Edit inbound rules

Filter security group rules

| ✓ | Name | Security group rule... | IP version | Type | Protocol | Port range | Source | Description |
|---|---|---|---|---|---|---|---|---|
| ✓ | – | sgr-0ea8dc921f6d7441d | IPv4 | HTTP | TCP | 80 | 0.0.0.0/0 | TCP from VPC |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ✓ | Deployment9ALB_sg | sg-0d20186c4bd129304 | alb_sg | vpc-0ee63278cc7a1f803 | Allow port 80 inbound… | 323867645900 | 1 Permission entry | 2 Permission entries |


sg-0d20186c4bd129304 - alb_sg

Details

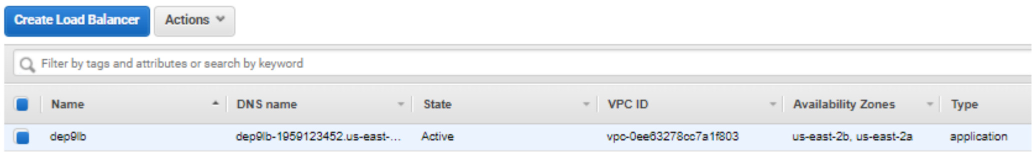
| Security group name | Security group ID | Description | VPC ID |
|---|---|---|---|
| alb_sg | sg-0d20186c4bd129304 | Allow port 80 inbound traffic | vpc-0ee63278cc7a1f803 |

| Owner | Inbound rules count | Outbound rules count |
|---|---|---|
| 323867645900 | 1 Permission entry | 1 Permission entry |

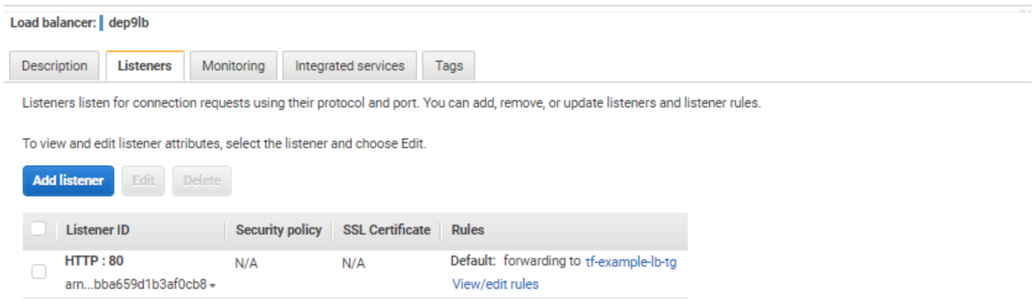
You can now check network connectivity with Reachability Analyzer

Outbound rules (1/1)

| Name | Security group rule... | IP version | Type | Protocol | Port range | Destination | Description |
|---|---|---|---|---|---|---|---|
| - | sgr-0d0eb30cbfda3e800 | - | HTTP | TCP | 80 | sg-014ce64f32c5da5f... | - |


Filter by tags and attributes or search by keyword

| Name | DNS name | State | VPC ID | Availability Zones | Type |
|---|---|---|---|---|---|
| dep9lb | dep9lb-1959123452.us-east-... | Active | vpc-0ee63278cc7a1f803 | us-east-2b, us-east-2a | application |


Load balancer: dep9lb

Listeners listen for connection requests using their protocol and port. You can add, remove, or update listeners and listener rules.

To view and edit listener attributes, select the listener and choose Edit.

| Listener ID | Security policy | SSL Certificate | Rules |
|---|---|---|---|
| HTTP : 80<br>arn...bba659d1b3af0cb8 | N/A | N/A | Default: forwarding to tf-example-lb-tg<br>View/edit rules |

My code for alb.tf

```
resource "aws_security_group" "alb_sg" {
  name        = "alb_sg"
  description = "Allow port 80 inbound traffic"
  vpc_id      = aws_vpc.main.id
  ingress {
    description = "TCP from VPC"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port = 80
    to_port   = 80
    protocol  = "tcp"
    #protocol         = "-1"  removed it because this is equivalent to all

    security_groups = [aws_security_group.ubuntu_ec2.id]
  }

  tags = {
    Name = "Deployment9ALB_sg"
  }
}

resource "aws_lb_target_group" "dep9tg" {
  name     = "tf-example-lb-tg"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.main.id
}

resource "aws_lb_target_group_attachment" "dep9tgat" {
  target_group_arn = aws_lb_target_group.dep9tg.arn
  target_id        = aws_instance.Dep9EC2.id
  port             = 80
}


resource "aws_lb" "dep9lb" {
  name               = "dep9lb"
  internal           = false
  load_balancer_type = "application"
  security_groups     = [aws_security_group.alb_sg.id]
  subnets             = [aws_subnet.public01.id, aws_subnet.public02.id]

  enable_deletion_protection = false

  tags = {
    Environment = "Deployment9"
  }
}

resource "aws_lb_listener" "dep9lbl" {
  load_balancer_arn = aws_lb.dep9lb.arn
  port              = "80"
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.dep9tg.arn
  }
}
```

## Part 4 - RDS (Not finished): I created a rds.tf for the postgresql database. This is what I have so far for the rds.tf but it doesn't seem to work when I run terraform apply.

My code for rds.tf

```
resource "aws_security_group" "rds_sg" {
  name        = "rds_sg"
  description = "Allow TLS inbound traffic"
  vpc_id      = aws_vpc.main.id

  ingress {
    description      = "TLS from VPC"
    from_port        = 80
    to_port         = 80
    protocol        = "tcp"
    security_groups = [aws_security_group.ubuntu_ec2.id]
  }
  tags = {
    Name = "SG-for-RDS"
  }
}


resource "aws_db_instance" "rds" {
  allocated_storage      = 20
  engine                 = "postgres"
  engine_version         = "9.6.20-R1"
  instance_class         = "db.t2.micro"
  multi_az               = "true"
  name                   = "mydb"
  username               = "bishajit"
  password               = "kura123"
  vpc_security_group_ids = [aws_security_group.rds_sg.id]
  skip_final_snapshot    = true
}


resource "aws_db_subnet_group" "default" {
  name       = "main"
  subnet_ids = [aws_subnet.internal01.id, aws_subnet.internal02.id]

  tags = {
    Name = "My DB subnet group"
  }
}
```

It gives me this error

```
│
│ Error: Error creating DB Instance: InvalidParameterValue: Invalid DB engine
│       status code: 400, request id: d9a9bfdb-3656-4cd5-b91f-39f5956cb303, {
│   AllocatedStorage: 20,
│   AutoMinorVersionUpgrade: true,
│   BackupRetentionPeriod: 0,
│   CopyTagsToSnapshot: false,
│   DBInstanceClass: "db.t2.micro",
│   DBInstanceIdentifier: "terraform-20211215041104490800000001",
│   DBName: "mydb",
│   DeletionProtection: false,
│   Engine: "postgresSQL",
│   EngineVersion: "9.6.20-R1",
│   MasterUserPassword: "********",
│   MasterUsername: "bishajit",
│   MultiAZ: true,
│   PubliclyAccessible: false,
│   StorageEncrypted: false,
│   Tags: [{
│       Key: "Team",
│       Value: "Kura Labs"
│     },{
│       Key: "Deployment",
│       Value: "DEPLOYMENT_09_TERRAFORM"
│     }],
│   VpcSecurityGroupIds: ["sg-088d4b5bb37f5472f"]
│ }
│
│   with aws_db_instance.rds,
│   on rds.tf line 19, in resource "aws_db_instance" "rds":
│   19: resource "aws_db_instance" "rds" {
│
╵
```