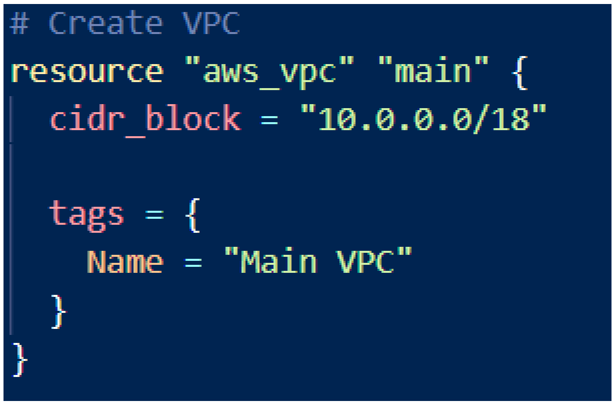
Part 1 – VPC

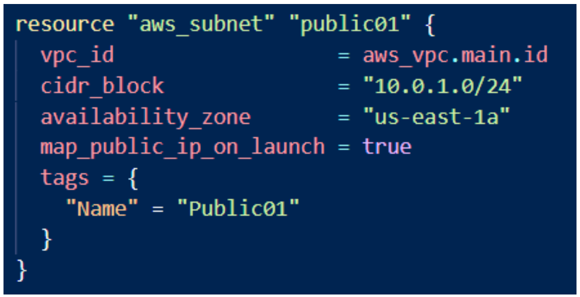
VPCs are virtual private clouds which are used for creating isolated networks. Inside my vpc.tf file, I used the resource module called “aws_vpc” and named my resource “main”. In addition, I gave my vpc a tag with the name “Main VPC” and a cidr block of “10.0.0.0/18”.



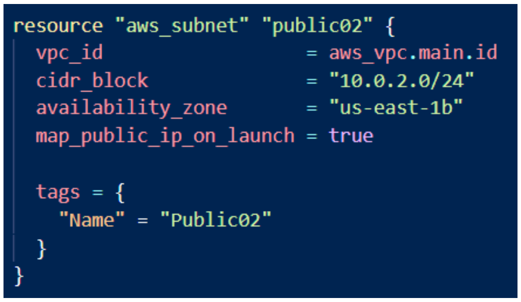
```
# Create VPC
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/18"

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

Subnets

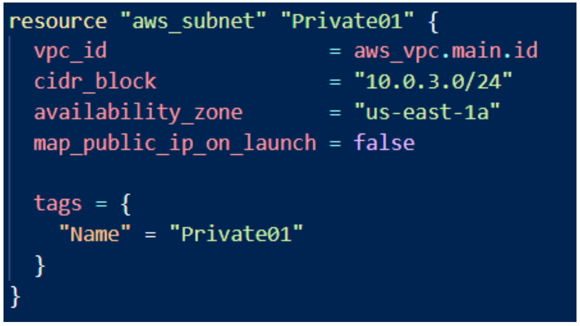
After creating the vpc, the next thing is to create the subnets. In short, subnets are networks inside a network. For this assignment we created 2 public subnets, 1 private and 2 internal subnets.



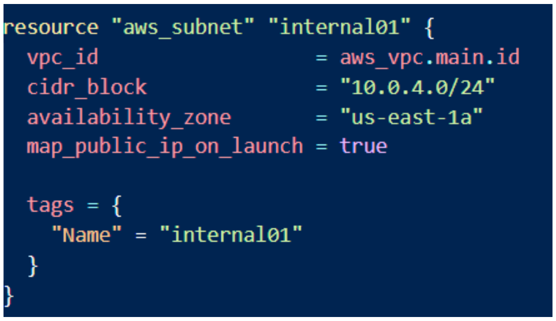
```
resource "aws_subnet" "public01" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true
  tags = {
    "Name" = "Public01"
  }
}
```



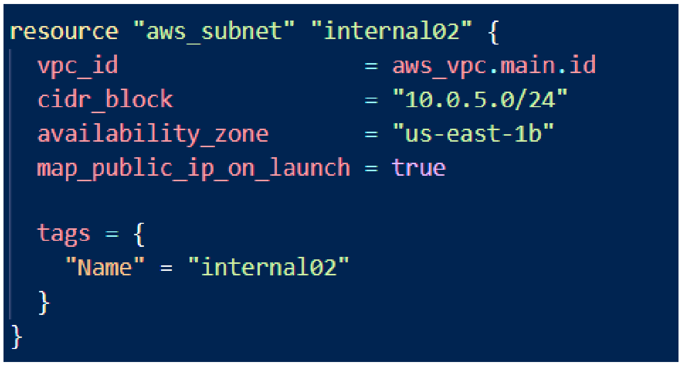
```
resource "aws_subnet" "public02" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.2.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true

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



```
resource "aws_subnet" "Private01" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.3.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = false

  tags = {
    "Name" = "Private01"
  }
}
```



```
resource "aws_subnet" "internal01" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.4.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true

  tags = {
    "Name" = "internal01"
  }
}
```



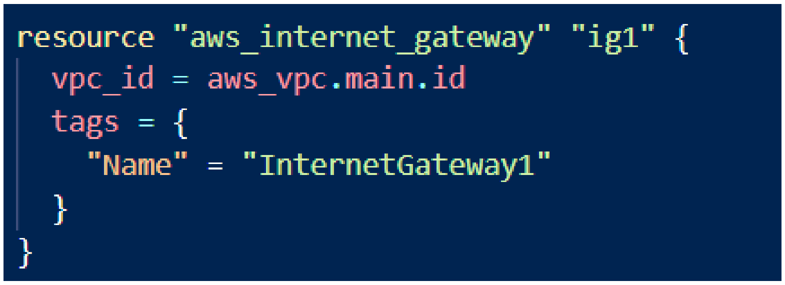
```
resource "aws_subnet" "internal02" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.5.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true

  tags = {
    "Name" = "internal02"
  }
}
```

Note: configured the vpc that each subnet would use as well as gave each subnet a unique cidr block. Also gave them availability zones which would be either us-east-1a or us-east-1b.

In addition, note that, the map_public_ip_on_launch is false for only the private subnet since we don’t want a public ip for the private subnet.

Internet Gateway

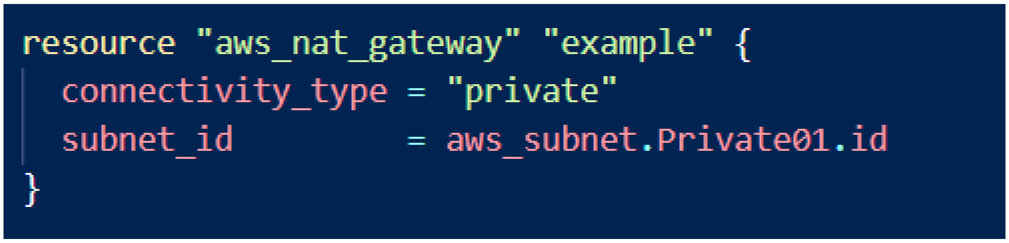
Now create an internet gateway which will control the traffic (outside and inside) the vpc.



```
resource "aws_internet_gateway" "ig1" {
  vpc_id = aws_vpc.main.id
  tags = {
    "Name" = "InternetGateway1"
  }
}
```

Nat Gateway

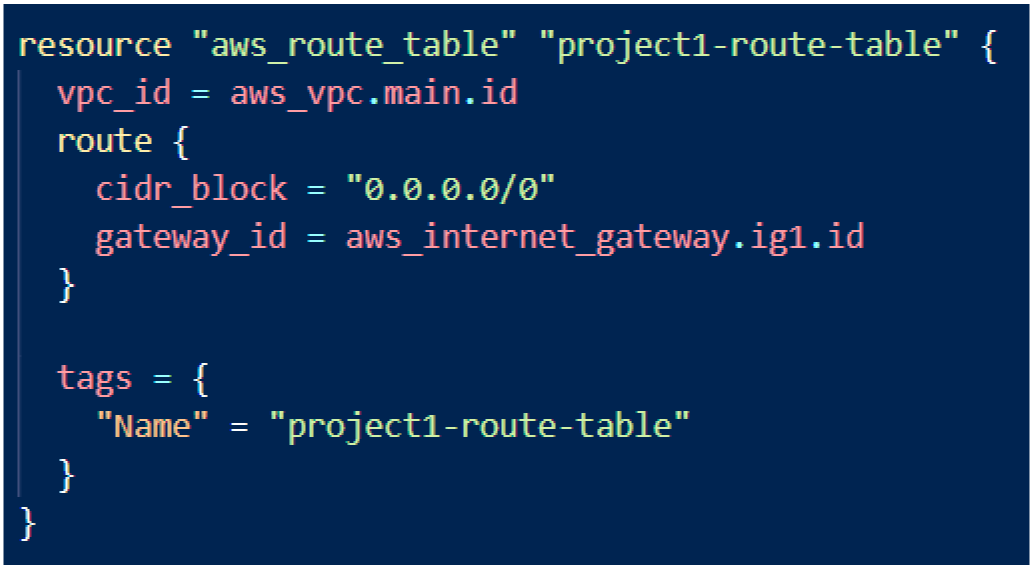
Using the Nat Gateway, it allows for our private resources to interact with the internet to get updates. Since our private resources don’t have direct access to the internet, it will speak to the nat gateway which is located inside the public subnet.



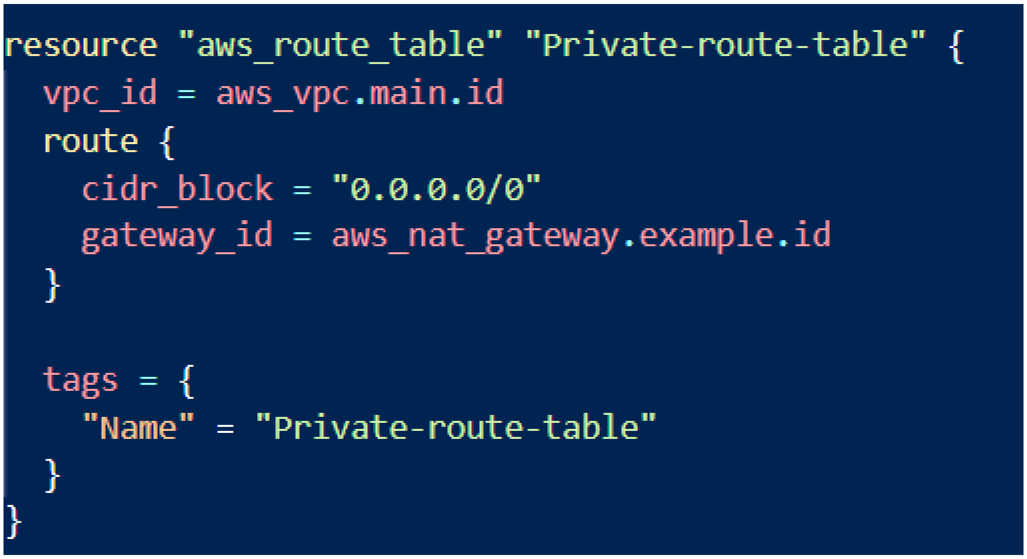
```
resource "aws_nat_gateway" "example" {
  connectivity_type = "private"
  subnet_id         = aws_subnet.Private01.id
}
```

Route Table

Now create public and private routing tables which will determine where the network traffic from the subnets are directed.



```
resource "aws_route_table" "project1-route-table" {
  vpc_id = aws_vpc.main.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.ig1.id
  }

  tags = {
    "Name" = "project1-route-table"
  }
}
```

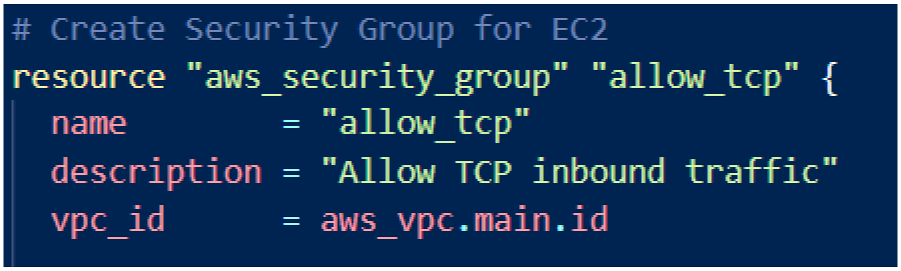


```
resource "aws_route_table" "Private-route-table" {
  vpc_id = aws_vpc.main.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.example.id
  }

  tags = {
    "Name" = "Private-route-table"
  }
}
```

Part 2 – EC2

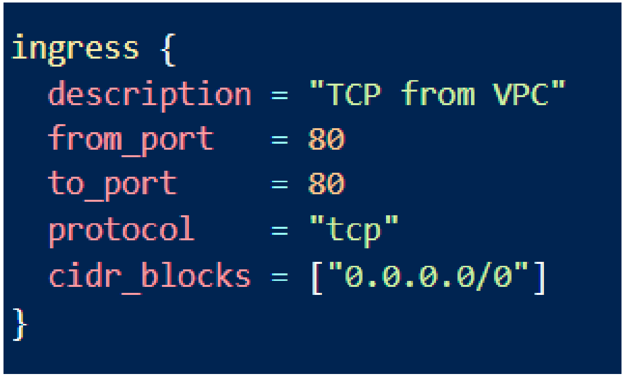
Now create an EC2 instance in a private subnet. Configure it so that it’s using an Ubuntu AMI, instance type, size and tags. For security, create a security group with certain ingress and egress rules.

The first thing that I decided to do is create a security group using the resource “aws_security_group”. Inside this resource, I named my security group, gave it a description, and linked it to my vpc.

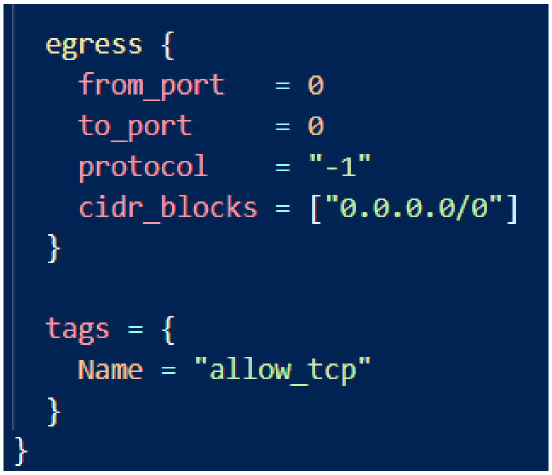


```
# Create Security Group for EC2
resource "aws_security_group" "allow_tcp" {
  name        = "allow_tcp"
  description = "Allow TCP inbound traffic"
  vpc_id      = aws_vpc.main.id
```

Ingress had to allow port 80 traffic from the ALB security group.

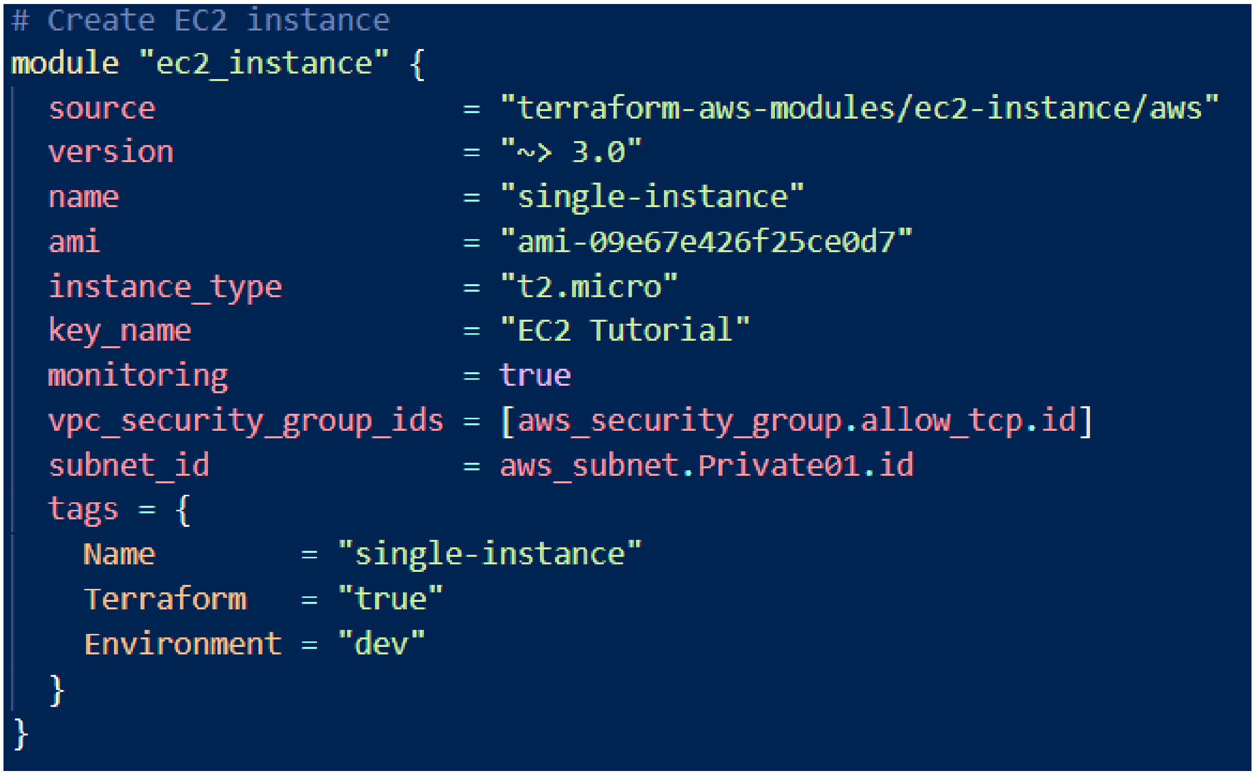


```
ingress {
  description = "TCP from VPC"
  from_port   = 80
  to_port     = 80
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
```



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

  tags = {
    Name = "allow_tcp"
  }
}
```

Then create EC2

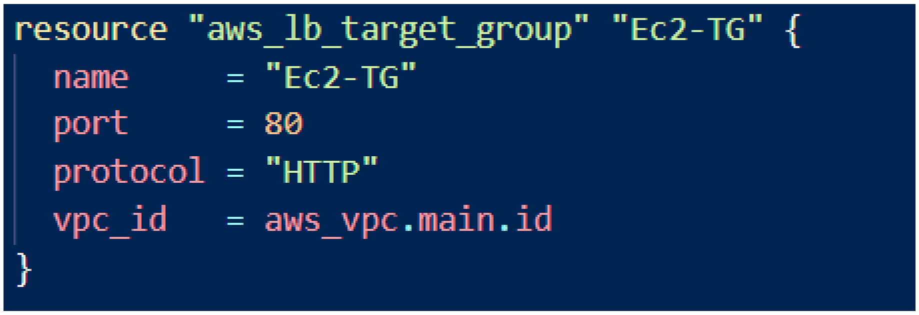


```
# Create EC2 instance
module "ec2_instance" {
  source                 = "terraform-aws-modules/ec2-instance/aws"
  version                = "~> 3.0"
  name                   = "single-instance"
  ami                    = "ami-09e67e426f25ce0d7"
  instance_type          = "t2.micro"
  key_name               = "EC2 Tutorial"
  monitoring             = true
  vpc_security_group_ids = [aws_security_group.allow_tcp.id]
  subnet_id              = aws_subnet.Private01.id
  tags = {
    Name        = "single-instance"
    Terraform   = "true"
    Environment = "dev"
  }
}
```

Part 3 – ALB

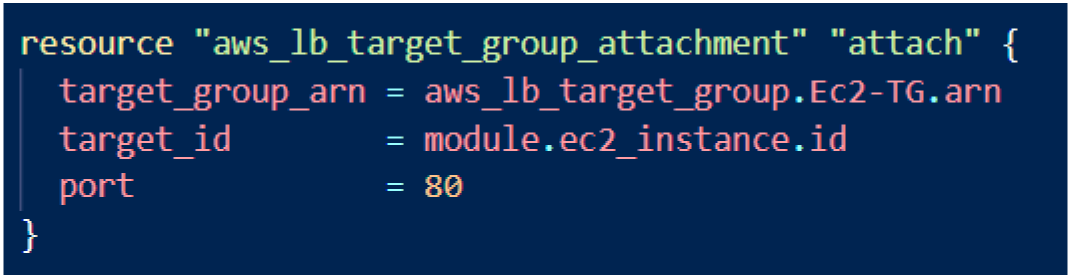
Create an application load balancer in 2 public subnets. The ALB needs a security group with certain required ingress and egress rules. After we create the security group, we have to develop a target group and link it to the ec2 instance we created in part 2.
The ALB needs a listener that forwards traffic to the target group. While creating this, we have to accept HTTP traffic only.

I decided on creating the target group first since the ALB depends on it. I used the “aws_lb_target_group” resource and simply named it, linked the vpc, choose what type of port and protocol it should listen for.



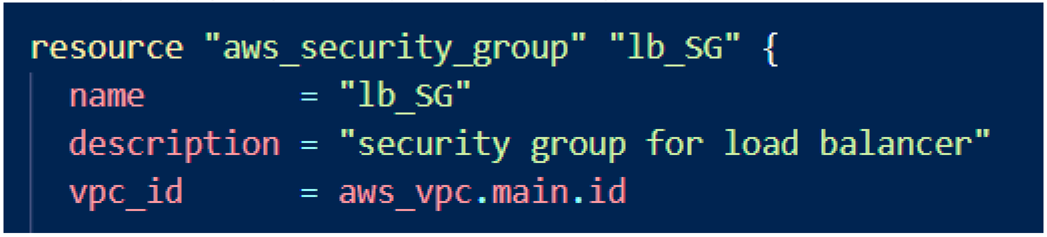
```
resource "aws_lb_target_group" "Ec2-TG" {
  name     = "Ec2-TG"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.main.id
}
```

I then needed to attach the target group so I used the resource “aws_lb_target_group_attachment” which links the target group to the load balancer. You would then need to configure the ec2 that will be targeted and a specific port.



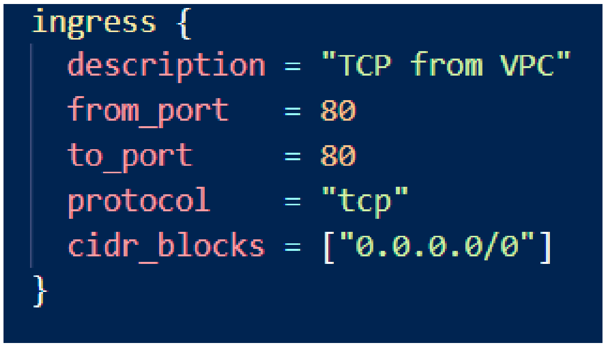
```
resource "aws_lb_target_group_attachment" "attach" {
  target_group_arn = aws_lb_target_group.Ec2-TG.arn
  target_id        = module.ec2_instance.id
  port             = 80
}
```

We can then create the security group for the load balancer. We start off by naming the security group, giving a description, and linking the vpc.



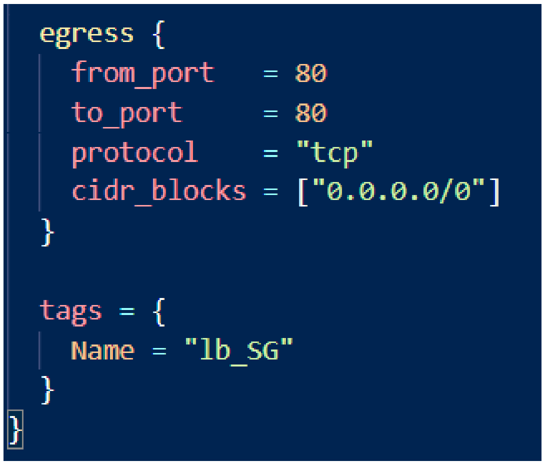
```
resource "aws_security_group" "lb_SG" {
  name        = "lb_SG"
  description = "security group for load balancer"
  vpc_id      = aws_vpc.main.id
```

For the ingress rules, we are simply allowing only port 80 inbound from any ipv4.



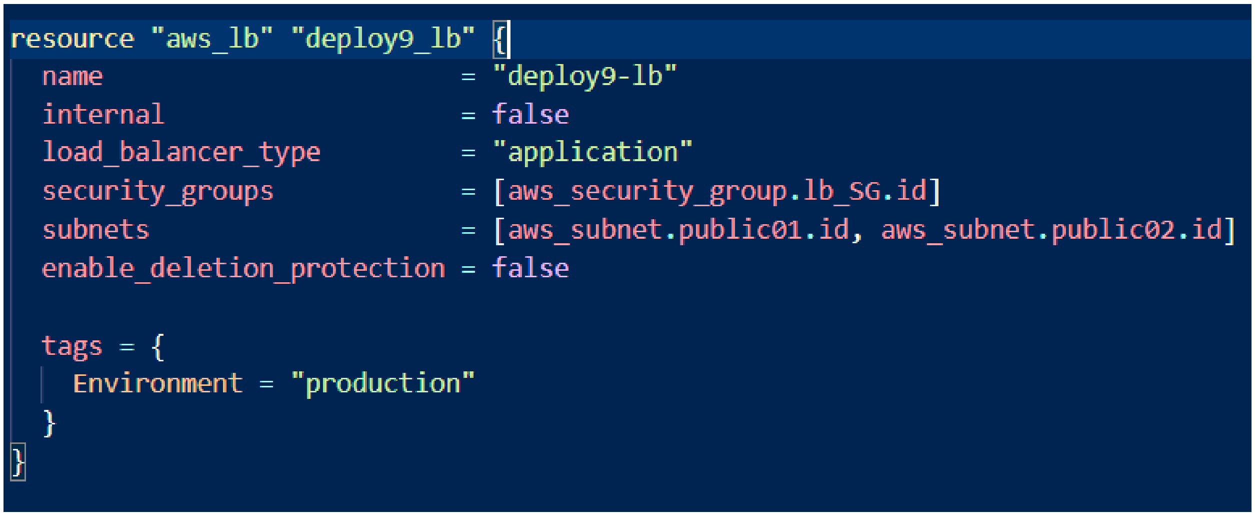
```
  ingress {
    description = "TCP from VPC"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
```

For egress, we are allowing only port 80 outbound traffic to the EC2 security group. This will direct all http outbound traffic to the ec2 instance.



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

  tags = {
    Name = "lb_SG"
  }
}
```

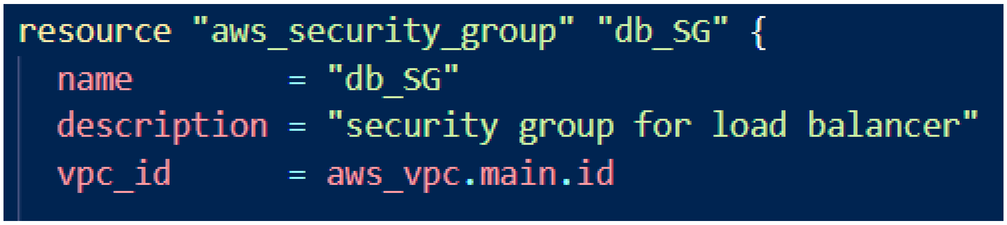
Create the load balancer using the resource "aws_lb". While creating the load balancer, there are important configurations such as load_balancer_type which tells the load balancer what type it is. We need it to be an application load balancer so we will select that. We then need to attach the security group and subnet.



```
resource "aws_lb" "deploy9_lb" {
  name                       = "deploy9-lb"
  internal                   = false
  load_balancer_type         = "application"
  security_groups            = [aws_security_group.lb_SG.id]
  subnets                    = [aws_subnet.public01.id, aws_subnet.public02.id]
  enable_deletion_protection = false

  tags = {
    Environment = "production"
  }
}
```

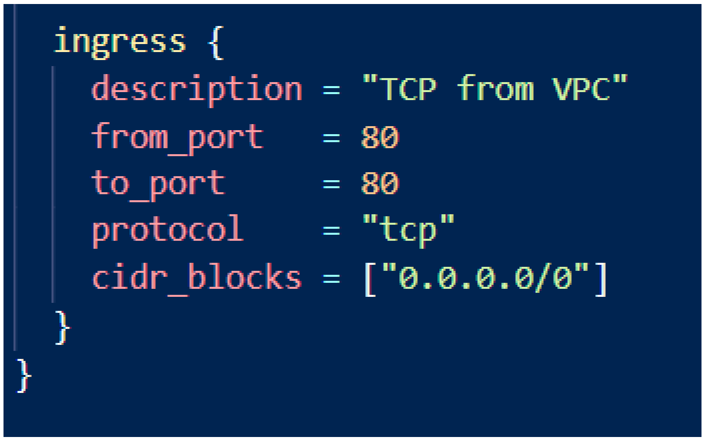
Part 4 – RDS

Create the security group for the database



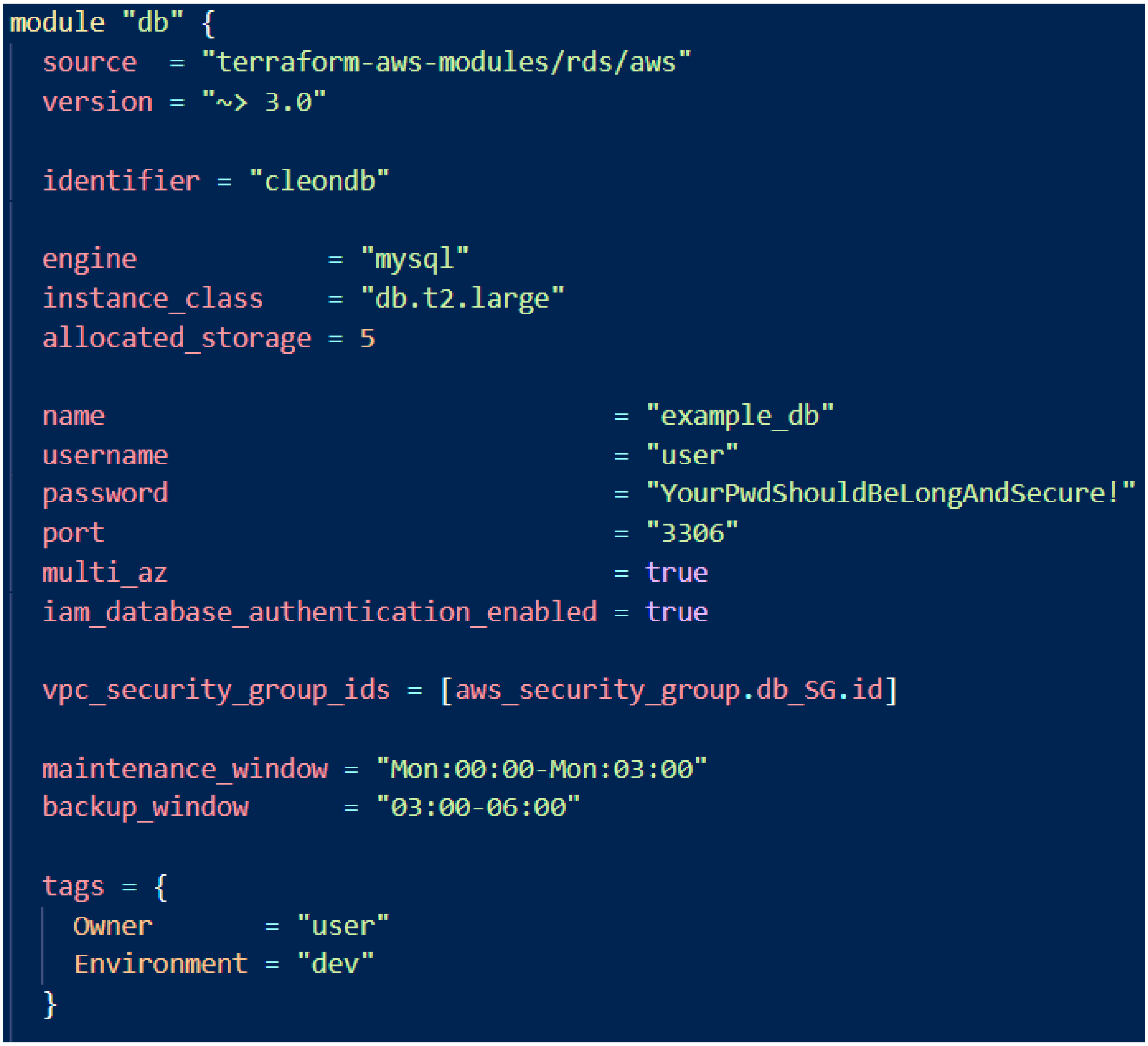
```
resource "aws_security_group" "db_SG" {
  name        = "db_SG"
  description = "security group for load balancer"
  vpc_id      = aws_vpc.main.id
```

For the ingress rules, we had to allow port 80 traffic from the EC2 security group.

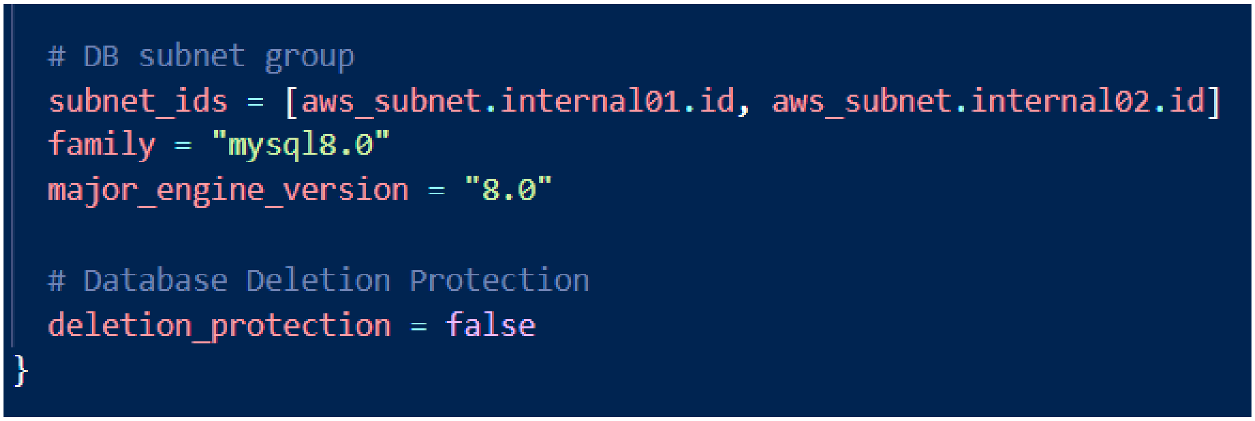


```
  ingress {
    description = "TCP from VPC"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

Once the security group was established, I could make the database using the module “db” .



```
module "db" {
  source  = "terraform-aws-modules/rds/aws"
  version = "~> 3.0"

  identifier = "cleondb"

  engine            = "mysql"
  instance_class    = "db.t2.large"
  allocated_storage = 5

  name                                = "example_db"
  username                            = "user"
  password                            = "YourPwdShouldBeLongAndSecure!"
  port                                = "3306"
  multi_az                            = true
  iam_database_authentication_enabled = true

  vpc_security_group_ids = [aws_security_group.db_SG.id]

  maintenance_window = "Mon:00:00-Mon:03:00"
  backup_window      = "03:00-06:00"

  tags = {
    Owner       = "user"
    Environment = "dev"
  }
```



```
  # DB subnet group
  subnet_ids = [aws_subnet.internal01.id, aws_subnet.internal02.id]
  family = "mysql8.0"
  major_engine_version = "8.0"

  # Database Deletion Protection
  deletion_protection = false
}
```

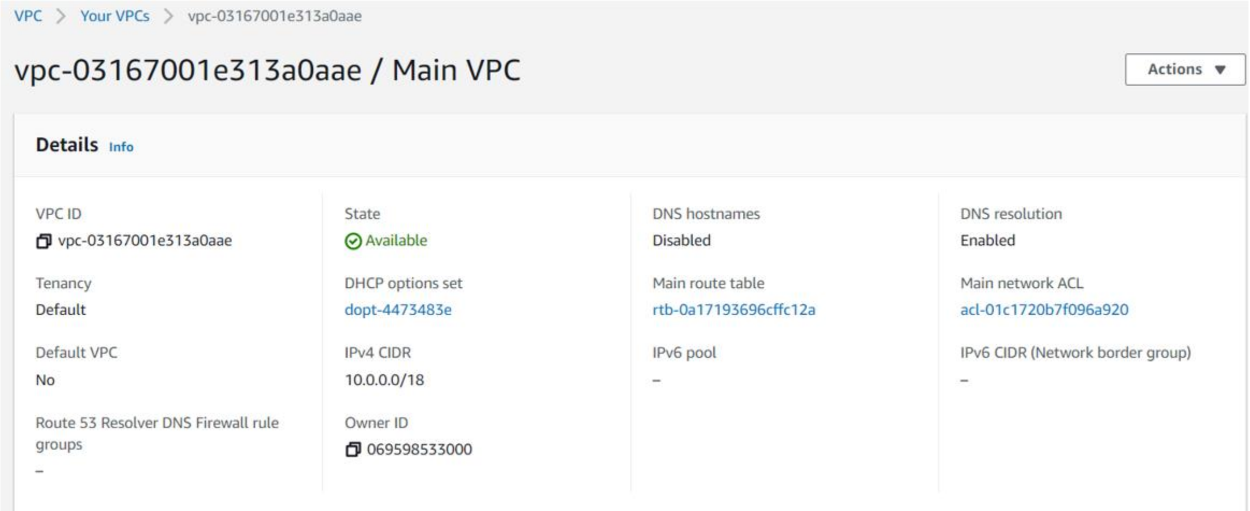
Once everything was set up I ran a few helpful commands

```
terraform fmt -recursive
terraform init
terraform plan
terraform apply -auto-approve
terraform destroy -auto-approve
```

Note: auto approve was used so that I didn’t have to repeatedly type yes to approve

Results

VPC



vpc-03167001e313a0aae / Main VPC

Details Info

VPC ID
vpc-03167001e313a0aae

State
Available

DNS hostnames
Disabled

DNS resolution
Enabled

Tenancy
Default

DHCP options set
dopt-4473483e

Main route table
rtb-0a17193696cffc12a

Main network ACL
acl-01c1720b7f096a920

Default VPC
No

IPv4 CIDR
10.0.0.0/18

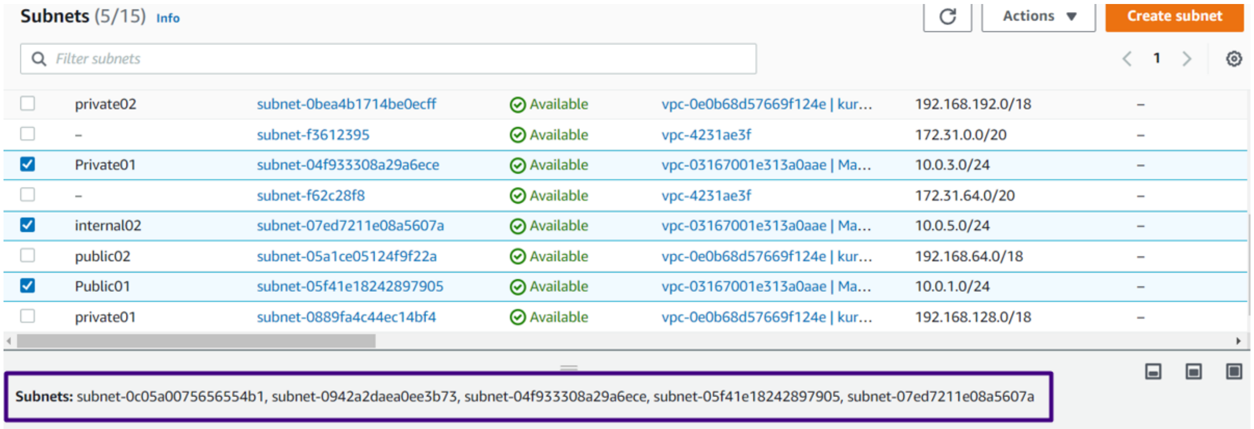
IPv6 pool
–

IPv6 CIDR (Network border group)
–

Route 53 Resolver DNS Firewall rule groups
–

Owner ID
069598533000

Subnets

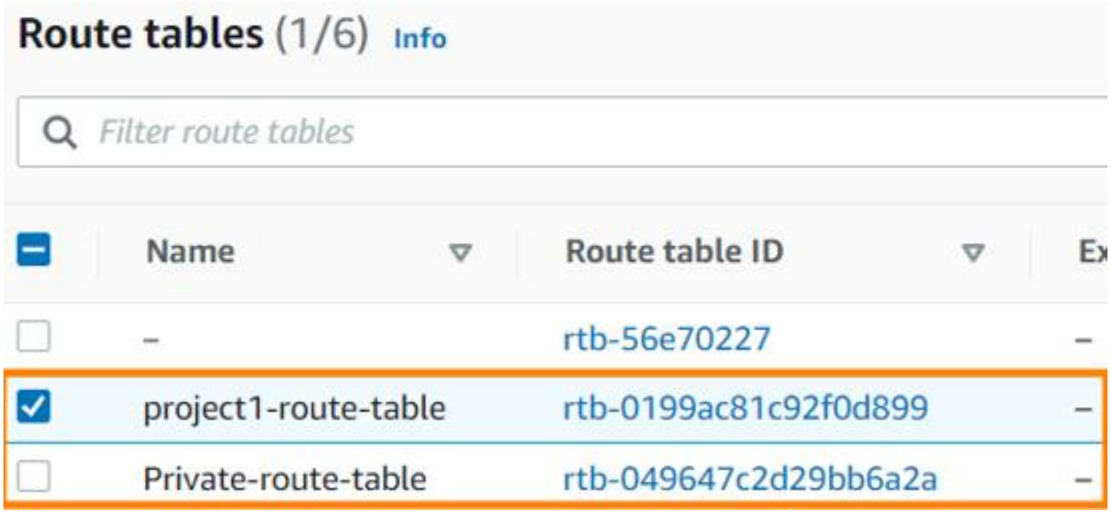


Subnets (5/15) Info

☐	private02	subnet-0bea4b1714be0ecff	Available	vpc-0e0b68d57669f124e | kur…	192.168.192.0/18	–
☐	–	subnet-f3612395	Available	vpc-4231ae3f	172.31.0.0/20	–
☑	Private01	subnet-04f933308a29a6ece	Available	vpc-03167001e313a0aae | Ma…	10.0.3.0/24	–
☐	–	subnet-f62c28f8	Available	vpc-4231ae3f	172.31.64.0/20	–
☑	internal02	subnet-07ed7211e08a5607a	Available	vpc-03167001e313a0aae | Ma…	10.0.5.0/24	–
☐	public02	subnet-05a1ce05124f9f22a	Available	vpc-0e0b68d57669f124e | kur…	192.168.64.0/18	–
☑	Public01	subnet-05f41e18242897905	Available	vpc-03167001e313a0aae | Ma…	10.0.1.0/24	–
☐	private01	subnet-0889fa4c44ec14bf4	Available	vpc-0e0b68d57669f124e | kur…	192.168.128.0/18	–

Subnets: subnet-0c05a0075656554b1, subnet-0942a2daea0ee3b73, subnet-04f933308a29a6ece, subnet-05f41e18242897905, subnet-07ed7211e08a5607a

Route Tables

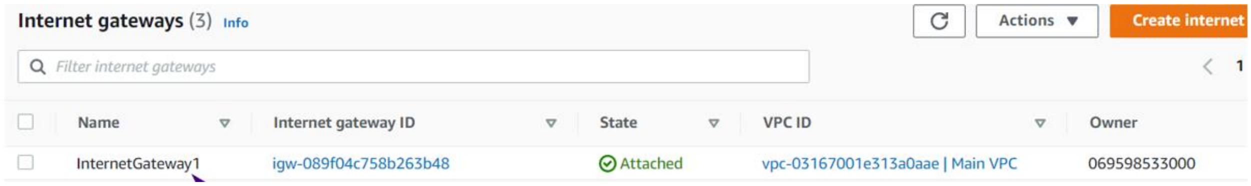


Route tables (1/6) Info

Filter route tables

☐	Name	Route table ID	Ex
□	–	rtb-56e70227	–
☑	project1-route-table	rtb-0199ac81c92f0d899	–
□	Private-route-table	rtb-049647c2d29bb6a2a	–

Internet Gateway

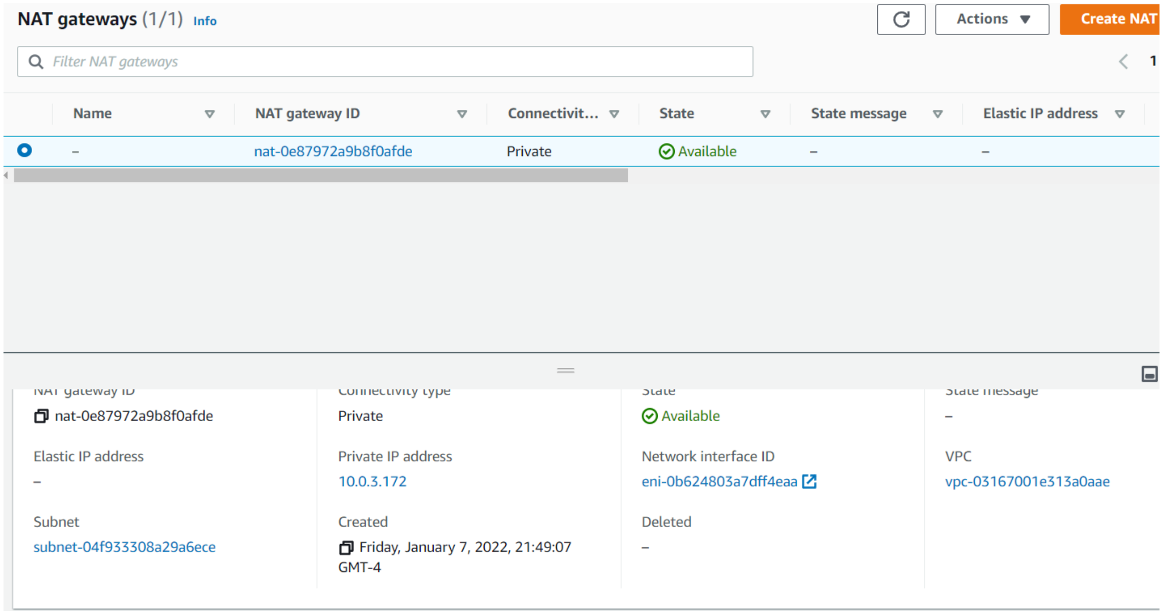


Internet gateways (3) Info

Filter internet gateways

□	Name	Internet gateway ID	State	VPC ID	Owner
□	InternetGateway1	igw-089f04c758b263b48	✓ Attached	vpc-03167001e313a0aae | Main VPC	069598533000

Nat Gateway



NAT gateways (1/1) Info

Filter NAT gateways

	Name	NAT gateway ID	Connectivit...	State	State message	Elastic IP address
●	–	nat-0e87972a9b8f0afde	Private	✓ Available	–	–

NAT gateway ID
nat-0e87972a9b8f0afde

Connectivity type
Private

State
✓ Available

State message
–

Elastic IP address
–

Private IP address
10.0.3.172

Network interface ID
eni-0b624803a7dff4eaa

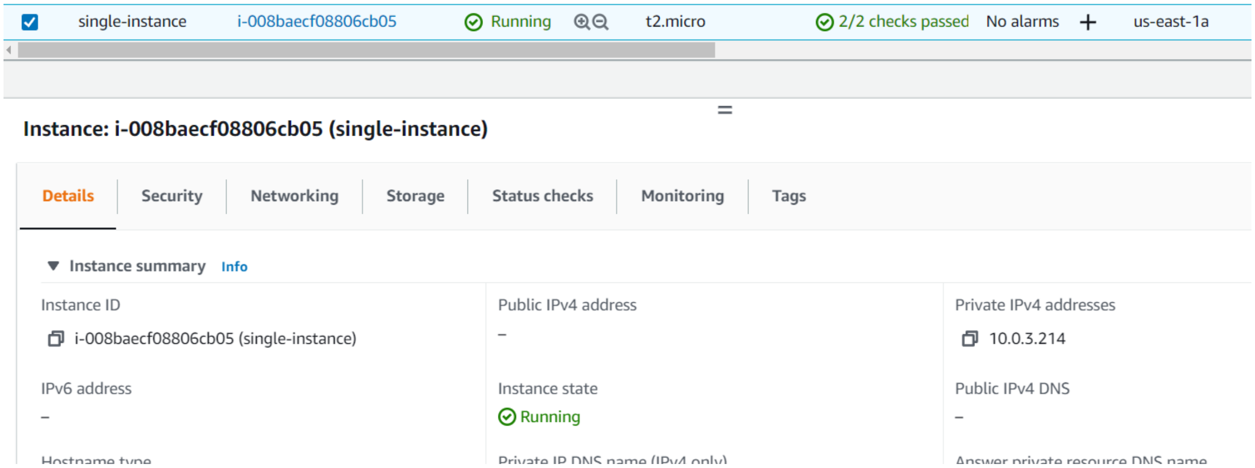
VPC
vpc-03167001e313a0aae

Subnet
subnet-04f933308a29a6ece

Created
Friday, January 7, 2022, 21:49:07 GMT-4

Deleted
–

EC2



	Name	Instance ID	Instance state		Instance type	Status check	Alarm status		Availability Zone
☑	single-instance	i-008baecf08806cb05	✓ Running		t2.micro	✓ 2/2 checks passed	No alarms	+	us-east-1a

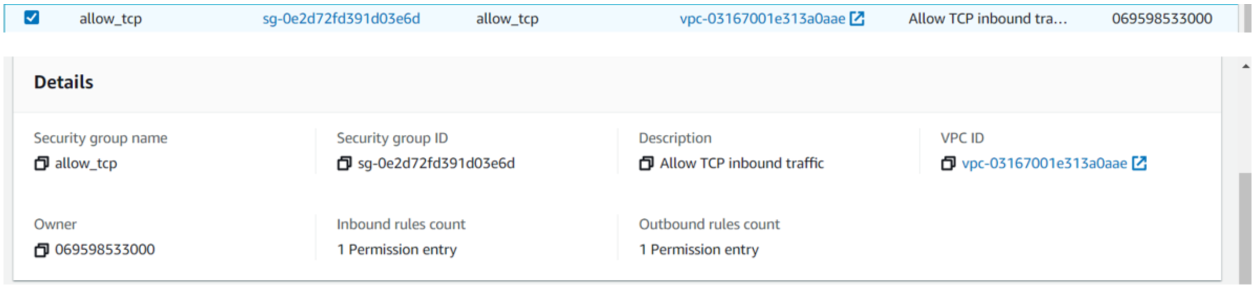
Instance: i-008baecf08806cb05 (single-instance)

▼ **Instance summary** Info

Instance ID i-008baecf08806cb05 (single-instance)	Public IPv4 address –	Private IPv4 addresses 10.0.3.214
IPv6 address –	Instance state ✓ Running	Public IPv4 DNS –
Hostname type	Private IP DNS name (IPv4 only)	Answer private resource DNS name

Security Groups

EC2:

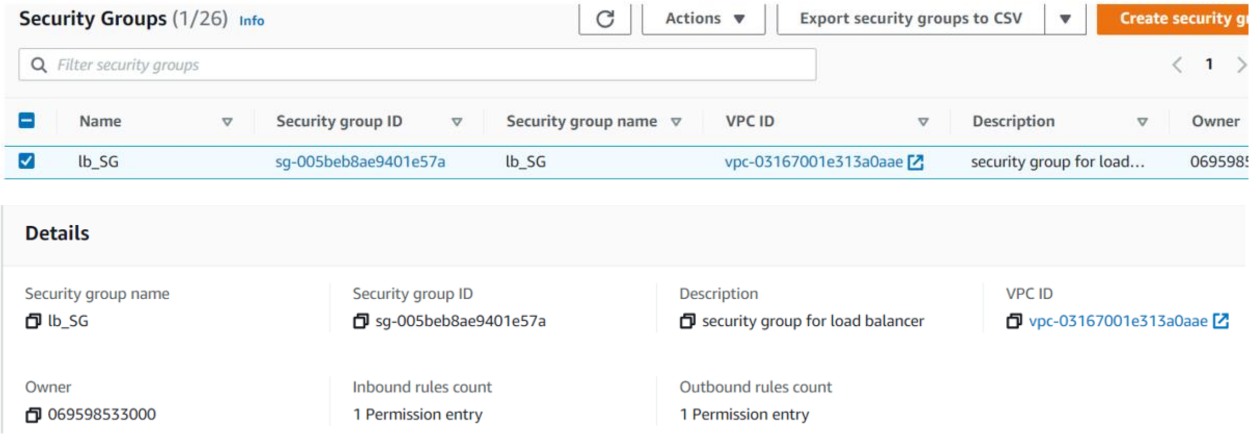


☑	allow_tcp	sg-0e2d72fd391d03e6d	allow_tcp	vpc-03167001e313a0aae	Allow TCP inbound tra...	069598533000

Details

Security group name allow_tcp	Security group ID sg-0e2d72fd391d03e6d	Description Allow TCP inbound traffic	VPC ID vpc-03167001e313a0aae
Owner 069598533000	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

ALB:



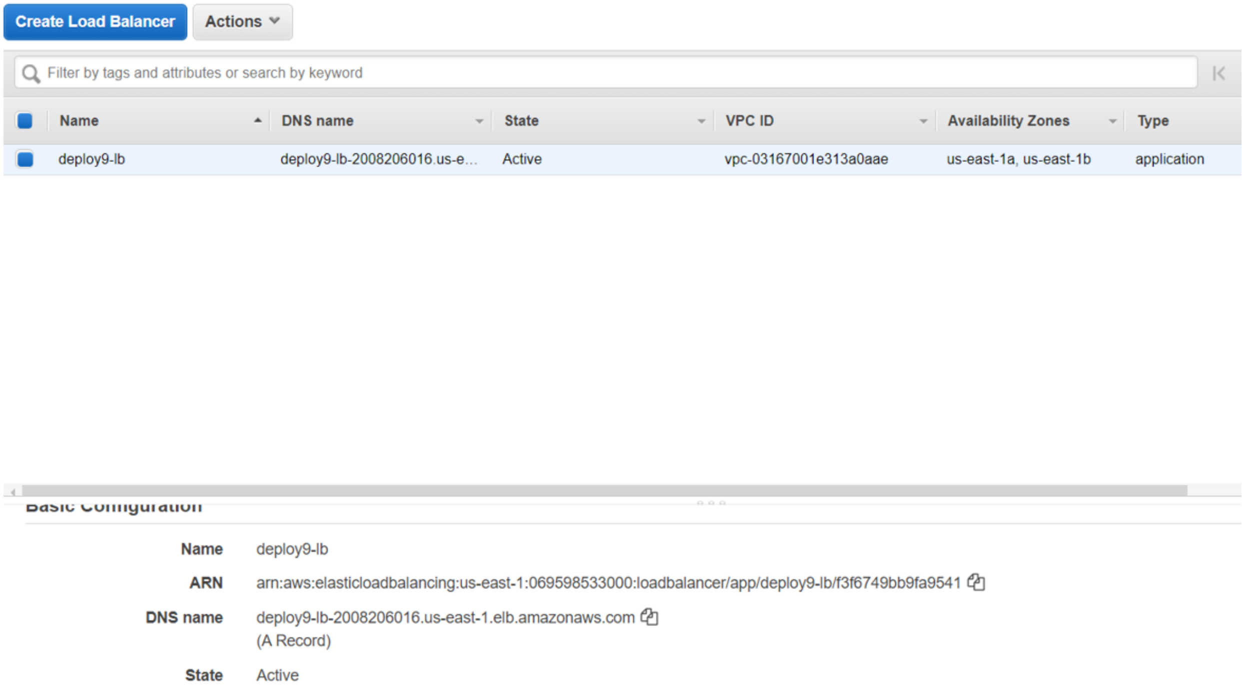
Security Groups (1/26) Info

☐	Name	Security group ID	Security group name	VPC ID	Description	Owner
☑	lb_SG	sg-005beb8ae9401e57a	lb_SG	vpc-03167001e313a0aae	security group for load...	0695985

Details

Security group name lb_SG	Security group ID sg-005beb8ae9401e57a	Description security group for load balancer	VPC ID vpc-03167001e313a0aae
Owner 069598533000	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

ALB

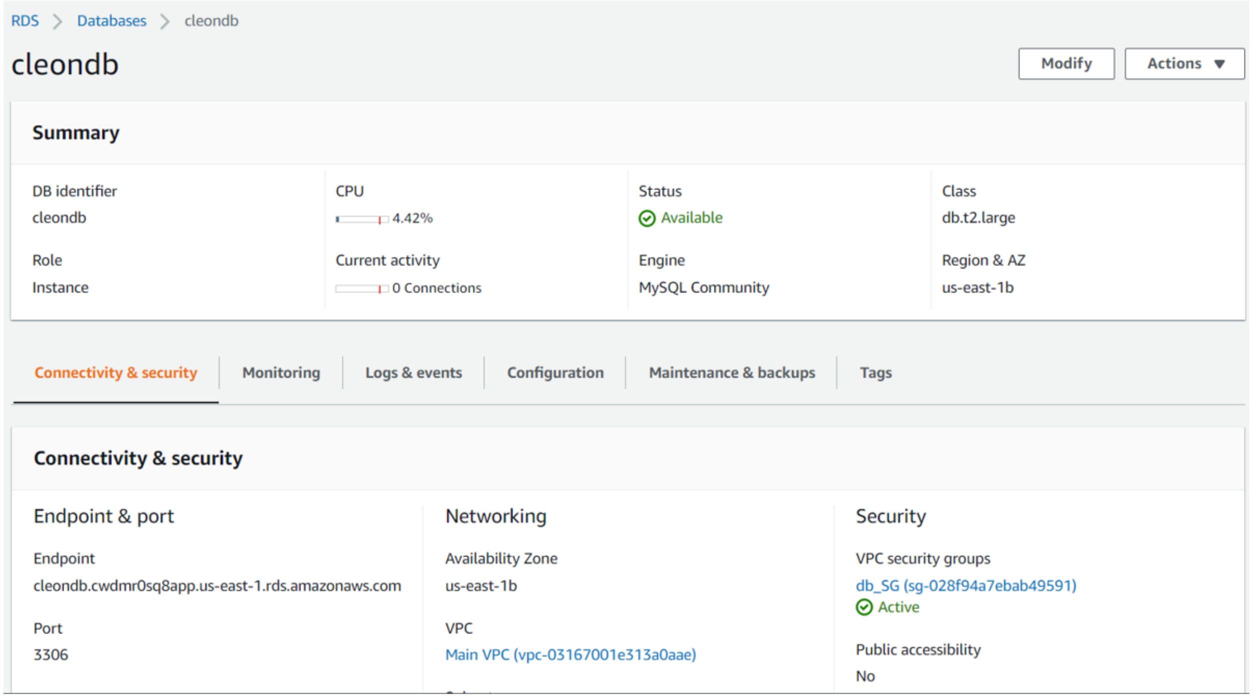


	Name	DNS name	State	VPC ID	Availability Zones	Type
☑	deploy9-lb	deploy9-lb-2008206016.us-e...	Active	vpc-03167001e313a0aae	us-east-1a, us-east-1b	application

Basic Configuration

Name	deploy9-lb
ARN	arn:aws:elasticloadbalancing:us-east-1:069598533000:loadbalancer/app/deploy9-lb/f3f6749bb9fa9541
DNS name	deploy9-lb-2008206016.us-east-1.elb.amazonaws.com (A Record)
State	Active

RDS Database



cleondb

Summary

DB identifier cleondb	CPU 4.42%	Status ✓ Available	Class db.t2.large
Role Instance	Current activity 0 Connections	Engine MySQL Community	Region & AZ us-east-1b

Connectivity & security

Endpoint & port	Networking	Security
Endpoint cleondb.cwdmr0sq8app.us-east-1.rds.amazonaws.com	Availability Zone us-east-1b	VPC security groups db_SG (sg-028f94a7ebab49591) ✓ Active
Port 3306	VPC Main VPC (vpc-03167001e313a0aae)	Public accessibility No