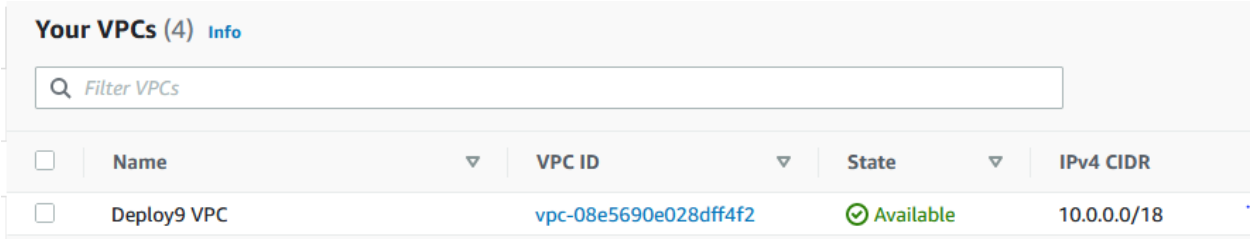
Deploy 9 - Terraform Challenge

Kenneth Tan

Part 1:

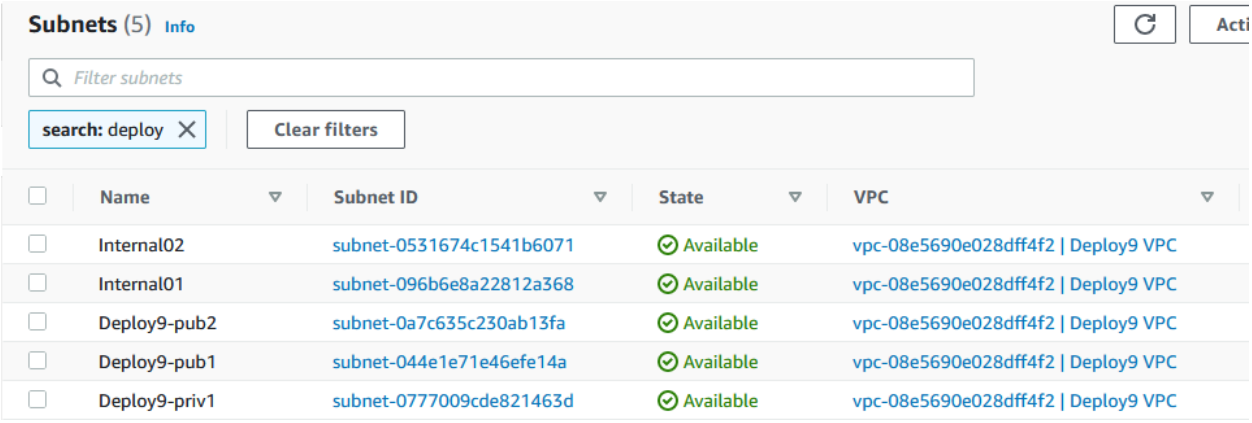
1. Create new VPC with:
    - 5 subnets (2 public, 1 private, 2 internal)
    - 2 route tables (public & private)
    - an Internet Gateway
    - and 1 NAT Gateway (in 1 of the private subnets)



Your VPCs (4) Info

Filter VPCs

Name	VPC ID	State	IPv4 CIDR
Deploy9 VPC	vpc-08e5690e028dff4f2	Available	10.0.0.0/18

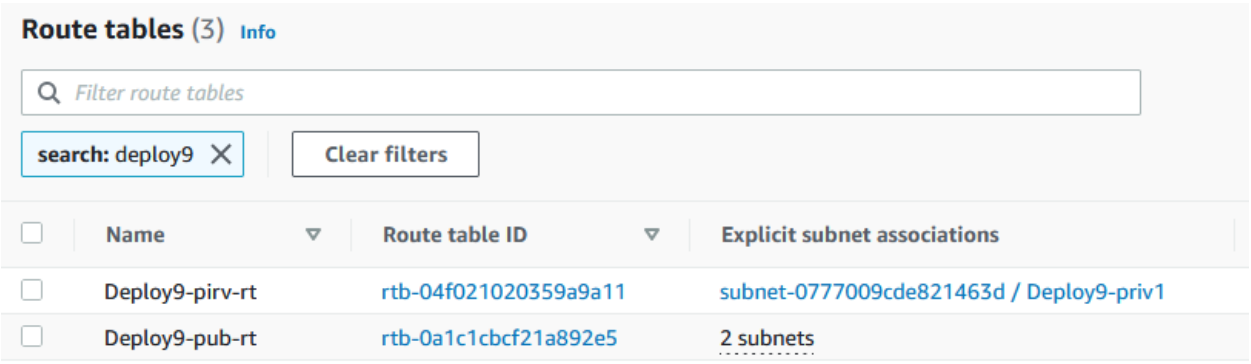


Subnets (5) Info

Filter subnets

search: deploy | Clear filters

Name	Subnet ID	State	VPC
Internal02	subnet-0531674c1541b6071	Available	vpc-08e5690e028dff4f2 | Deploy9 VPC
Internal01	subnet-096b6e8a22812a368	Available	vpc-08e5690e028dff4f2 | Deploy9 VPC
Deploy9-pub2	subnet-0a7c635c230ab13fa	Available	vpc-08e5690e028dff4f2 | Deploy9 VPC
Deploy9-pub1	subnet-044e1e71e46efe14a	Available	vpc-08e5690e028dff4f2 | Deploy9 VPC
Deploy9-priv1	subnet-0777009cde821463d	Available	vpc-08e5690e028dff4f2 | Deploy9 VPC

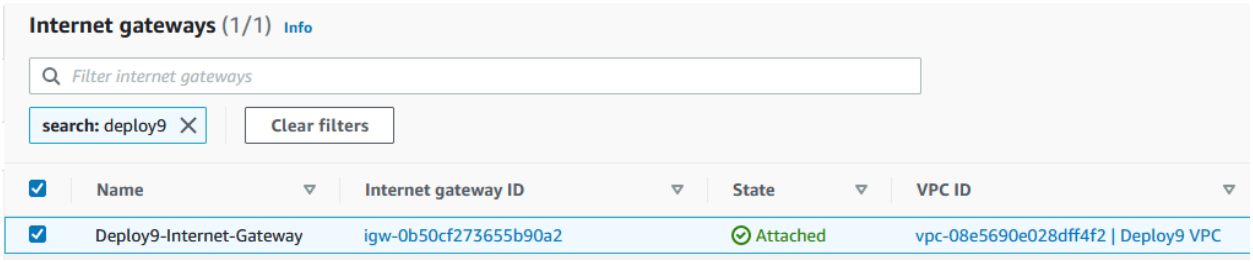


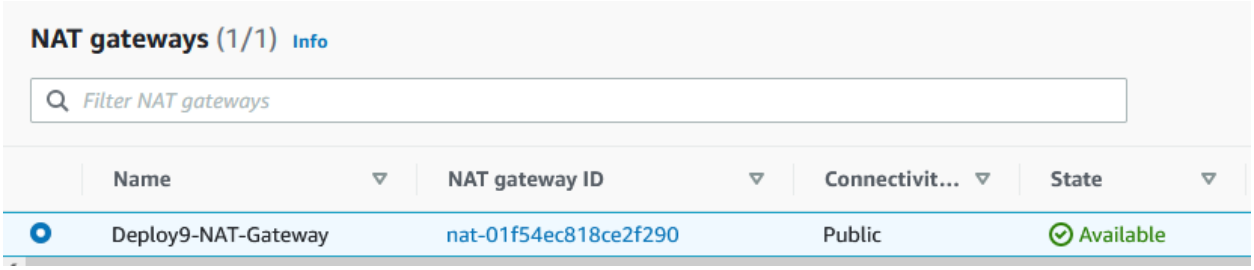
Route tables (3) Info

Filter route tables

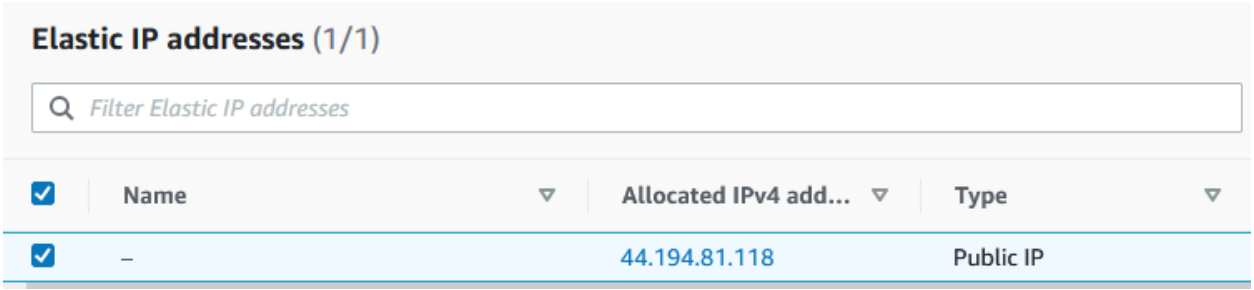
search: deploy9 | Clear filters

Name	Route table ID	Explicit subnet associations
Deploy9-pirv-rt	rtb-04f021020359a9a11	subnet-0777009cde821463d / Deploy9-priv1
Deploy9-pub-rt	rtb-0a1c1cbcf21a892e5	2 subnets



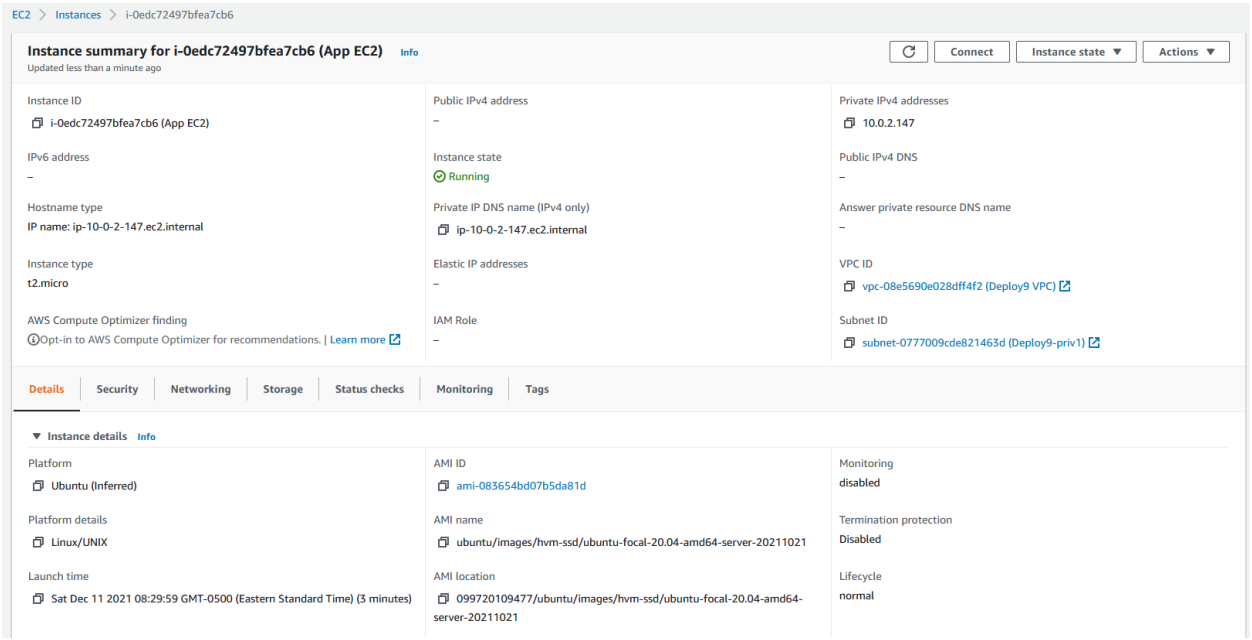


This NAT is in the public subnet and traffic in the private subnet is routed to this. This ill allow for the private EC2 to receive updates and maintenance as needed. An elastic IP was also needed for this NAT as shown below.

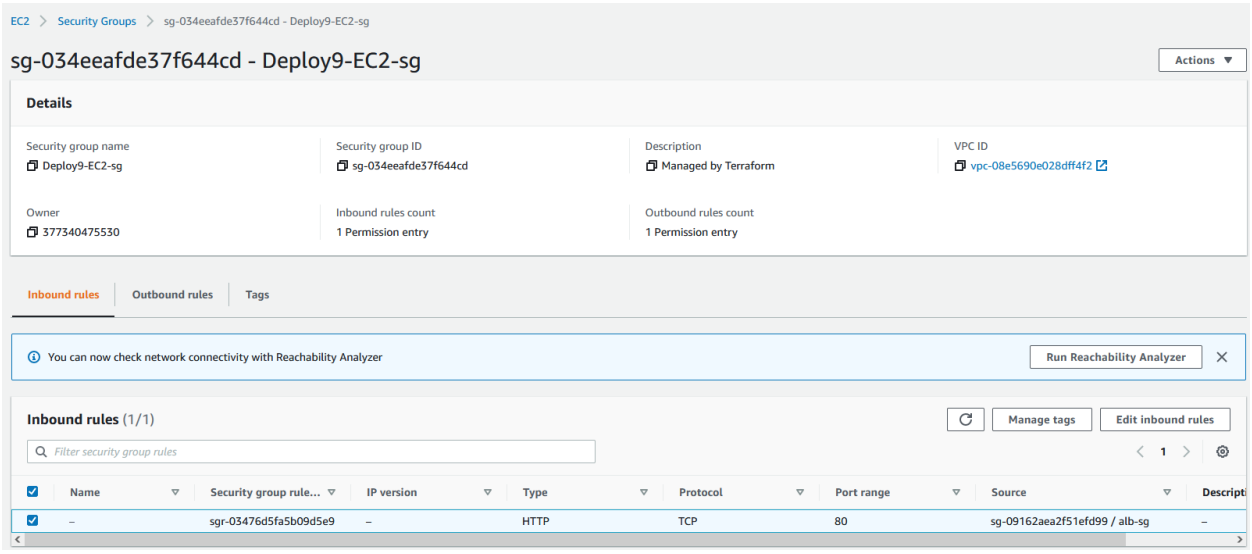


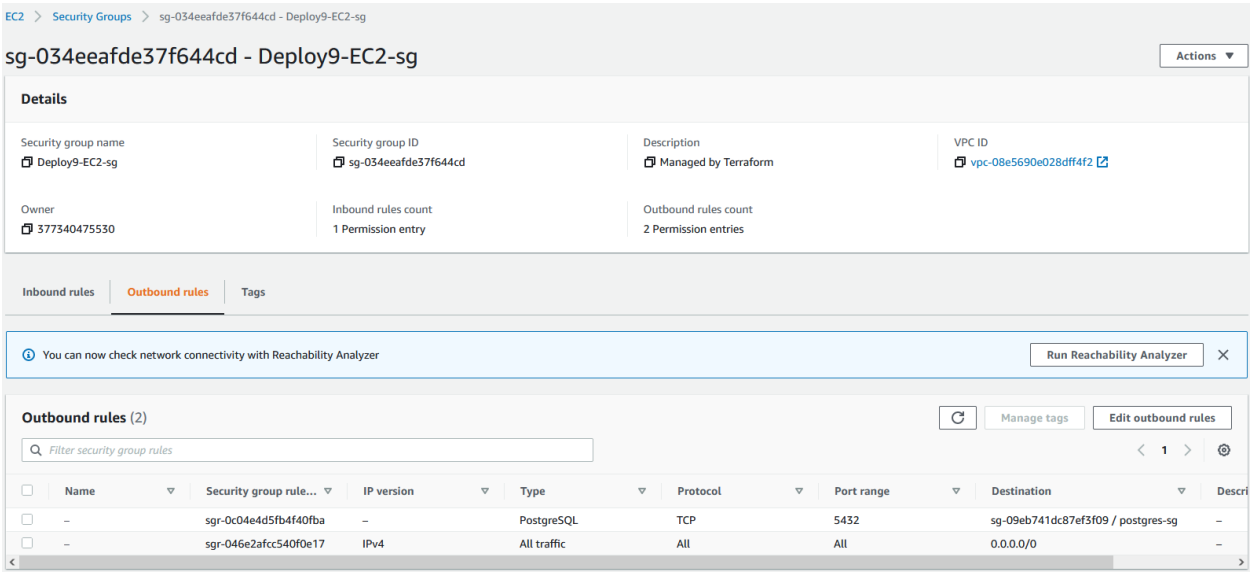
Part 2:

1. Create 1 EC2 instance in the private subnet with:
    - An Ubuntu AMI (version of your choosing)
    - Instance type/size, tags, and other settings of your choosing



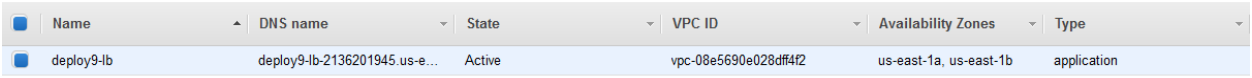
2. Create a security group for the EC2 with the following rules:
    - Ingress: allow port 80 traffic from the ALB security group
    - Egress: allow all outbound traffic to any ipv4 address



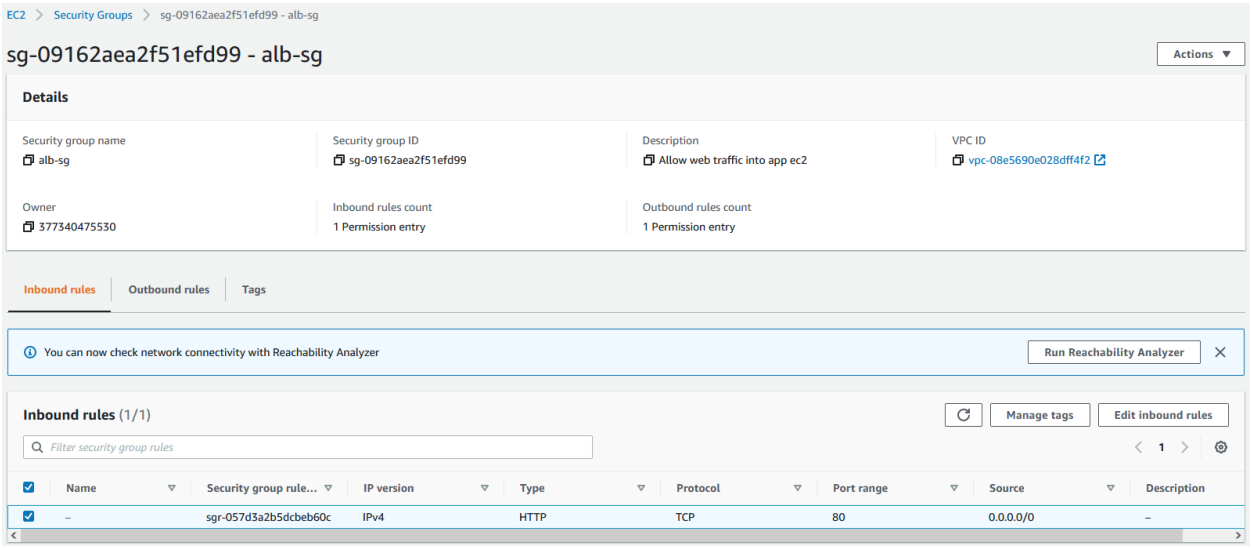


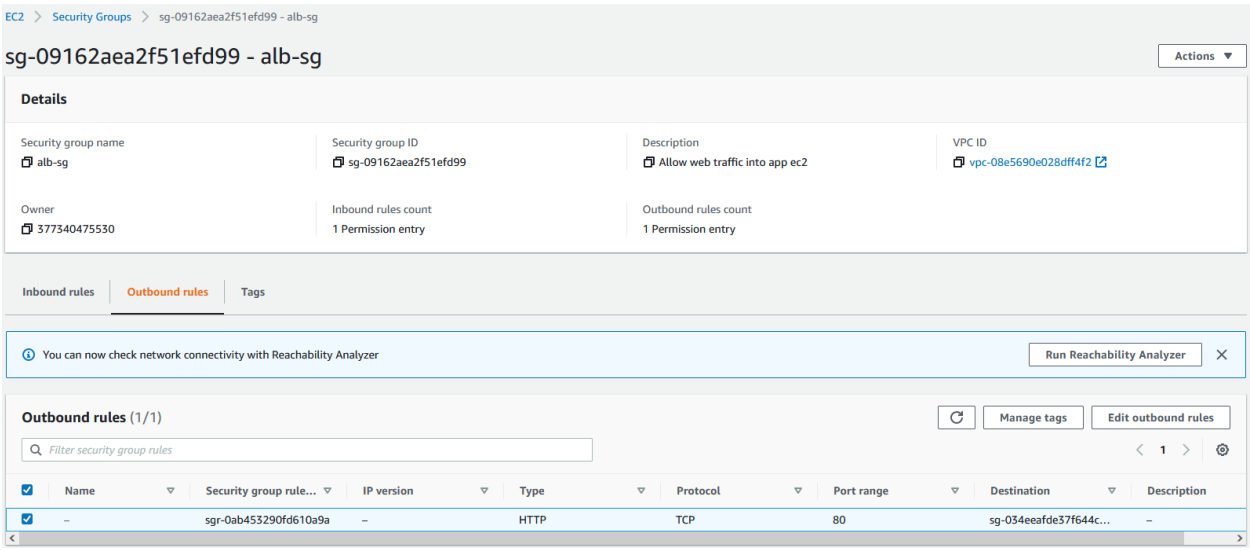
Part 3:

1. Create 1 ALB in the 2 public subnets

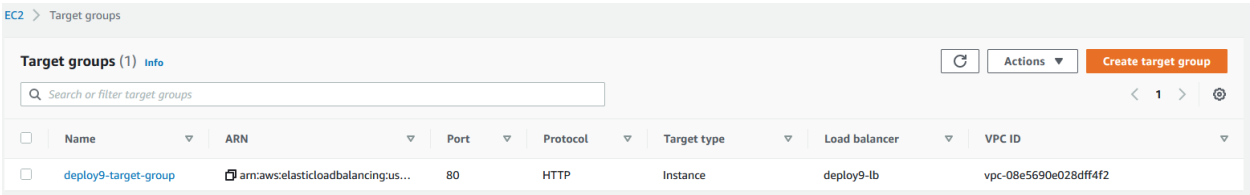


2. Create a security group for the ALB with the following rules:
    - Ingress: allows only port 80 inbound traffic from any ipv4 address
    - Egress: allow only port 80 outbound traffic to the EC2 security group

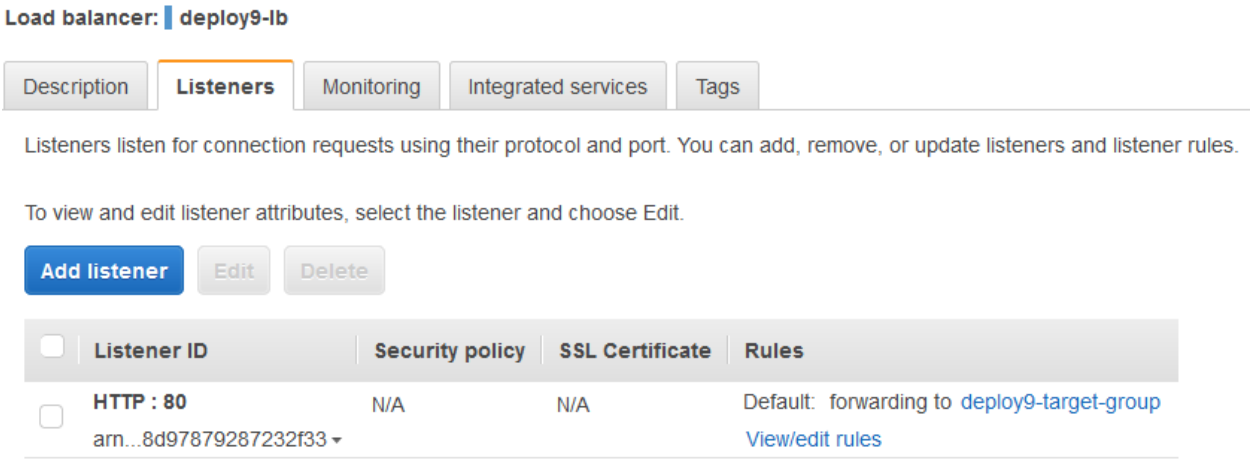




3. Create a target group and add the EC2 instance to the group



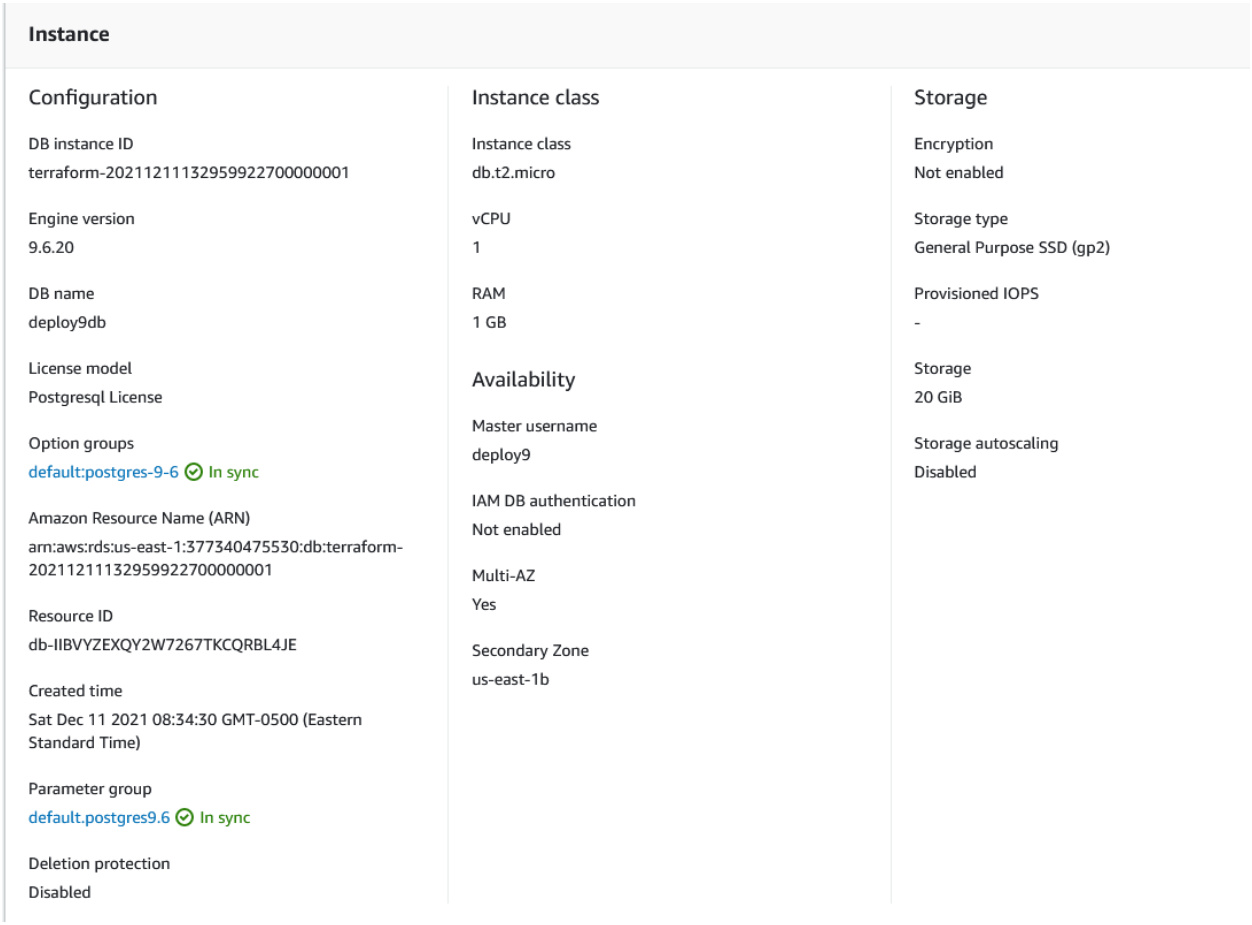
4. Create an ALB listener that forwards traffic to the target group



Part 4:

1. Create 1 PostgreSQL RDS instance
    - Make it multi-az

- Name, instance type/size, tags, db username/password, and other settings of your choosing



Instance

Configuration

DB instance ID
terraform-20211211132959227000000001

Engine version
9.6.20

DB name
deploy9db

License model
Postgresql License

Option groups
default:postgres-9-6 In sync

Amazon Resource Name (ARN)
arn:aws:rds:us-east-1:377340475530:db:terraform-20211211132959227000000001

Resource ID
db-IIBVYZEXQY2W7267TKCQRBL4JE

Created time
Sat Dec 11 2021 08:34:30 GMT-0500 (Eastern Standard Time)

Parameter group
default.postgres9.6 In sync

Deletion protection
Disabled

Instance class

Instance class
db.t2.micro

vCPU
1

RAM
1 GB

Availability

Master username
deploy9

IAM DB authentication
Not enabled

Multi-AZ
Yes

Secondary Zone
us-east-1b

Storage

Encryption
Not enabled

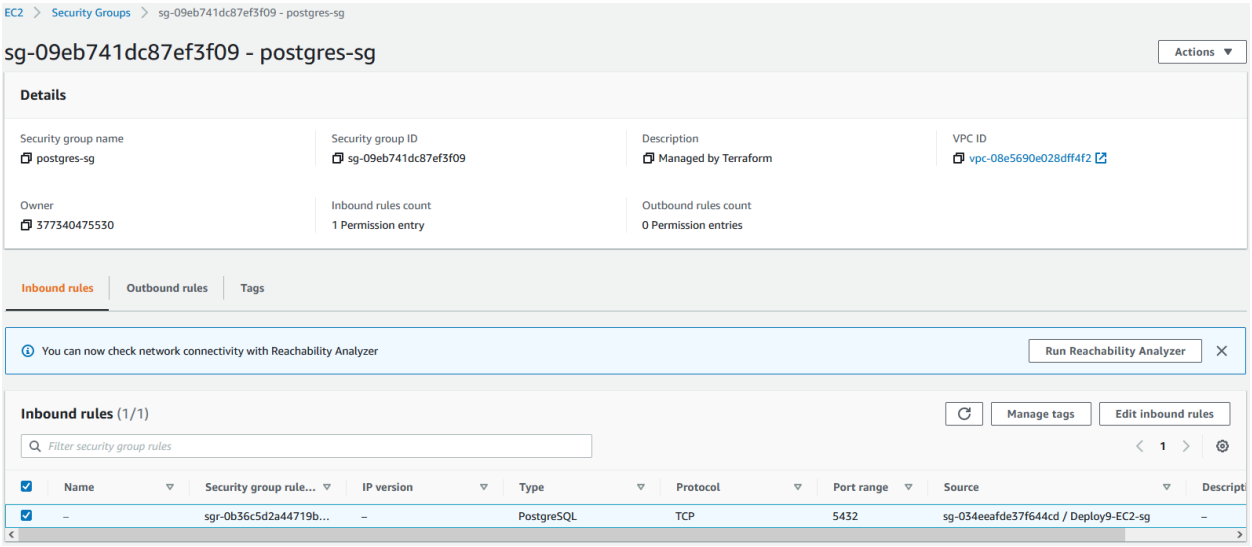
Storage type
General Purpose SSD (gp2)

Provisioned IOPS
-

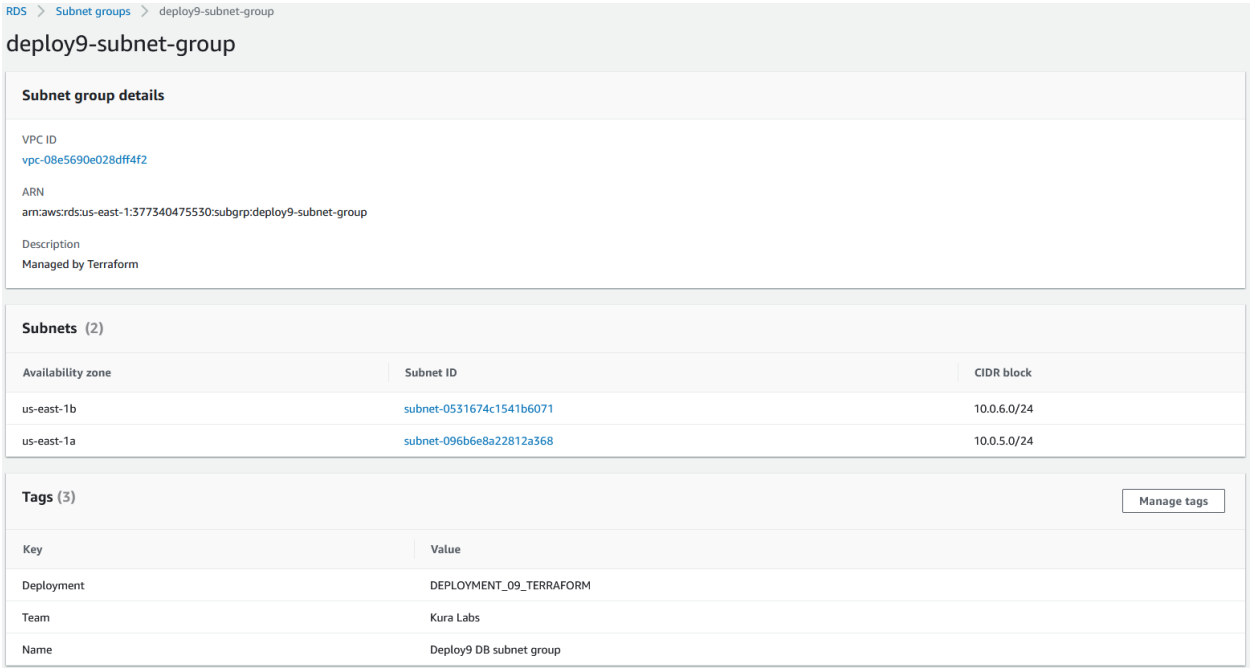
Storage
20 GiB

Storage autoscaling
Disabled

2. Create a security group for the RDS with the following rule:
    - Ingress: allow traffic to its port from the EC2 security group



3. Create a DB subnet group for the RDS consisting of the 2 internal subnets



Challenges:

One of the interesting challenges I ran into when creating the infrastructure with terraform was needing to work around cycling issues when creating security groups. Ultimately, the security groups were to only take traffic from each other but when doing so, I ran into a cycling error. This cycling error came up when trying to create 2 resources that depended on the existence of the other. The work around I found was using the aws_security_group_rule resource that was able to add on a rule for either inbound or outbound traffic.

Another challenge I faced was creating routing tables. Association of the tables was a completely separate function that needed to be called in the form of aws_subnet_association.