**Final Task**

1. Create one simple file uploader application in any programming language with following requirements.  
  1. Uploaded files should be store in S3 bucket  
  2. Your application should keep log of following thing in postgres db  
   1. IP of the user  
   2. Filename  
   3. Extension  
   4. Size of the file  
2. Application should be deployed in a containerized env  
3. Ensure that the application deployment is highly available and can handle scaling.  
4. Set up a CI/CD pipeline using a tool like Jenkins, GitLab CI/CD, or AWS CodePipeline.

**Networking and Security**

1. Configure appropriate security groups and network ACLs to control inbound and outbound traffic.  
2. Ensure that sensitive information (e.g., database credentials) are securely managed.  
3. create appropriate VPC and subnets   
4. implement necessary IAM roles and policy  
5. Database should not be publicly accessible

Github code repos-

1. Frontend - <https://github.com/dangar0090/final-task-frontend>
2. Backend - <https://github.com/dangar0090/final-task-backend>

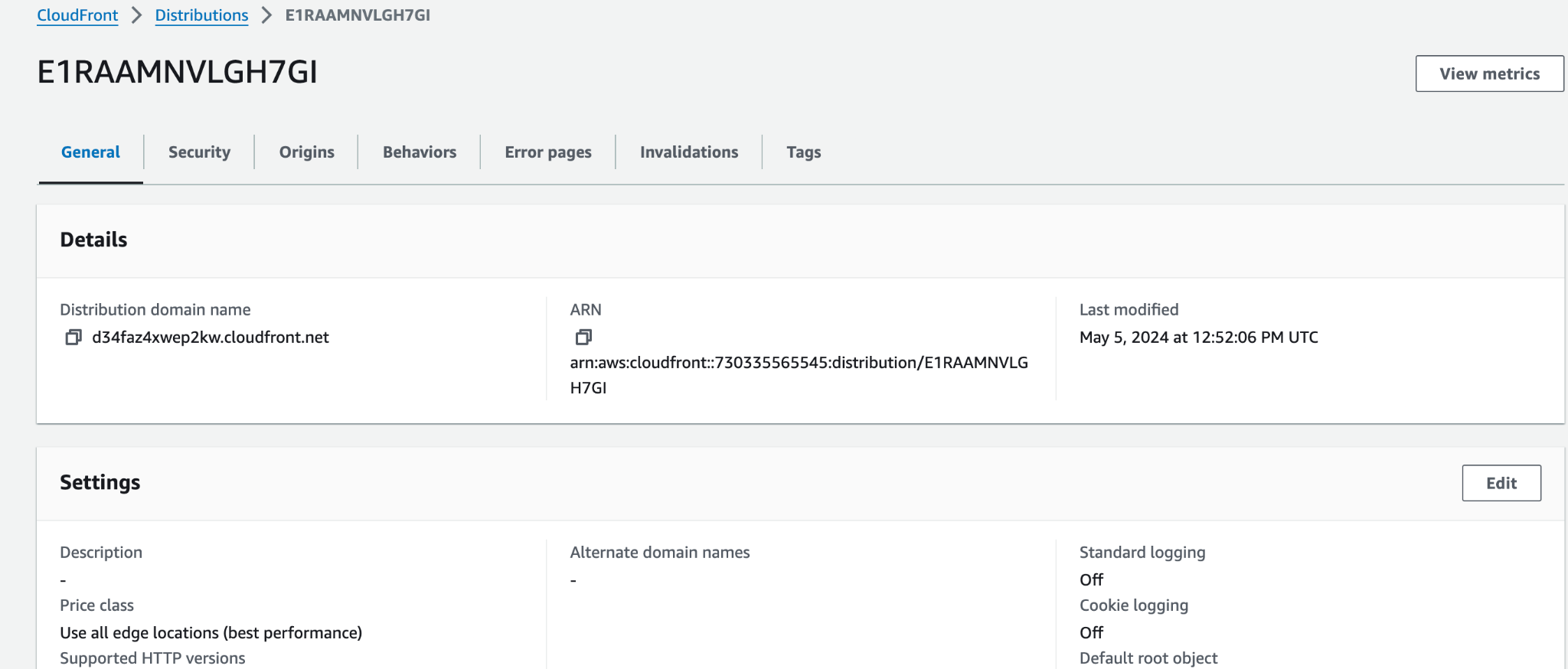
**Frontend :**

Services used :

1. AWS Codebuild
2. S3
3. Cloudfront

The frontend pulls the code on each push and merge detected in the frontend github link

Cloudfront distribution that hosts the static website from S3



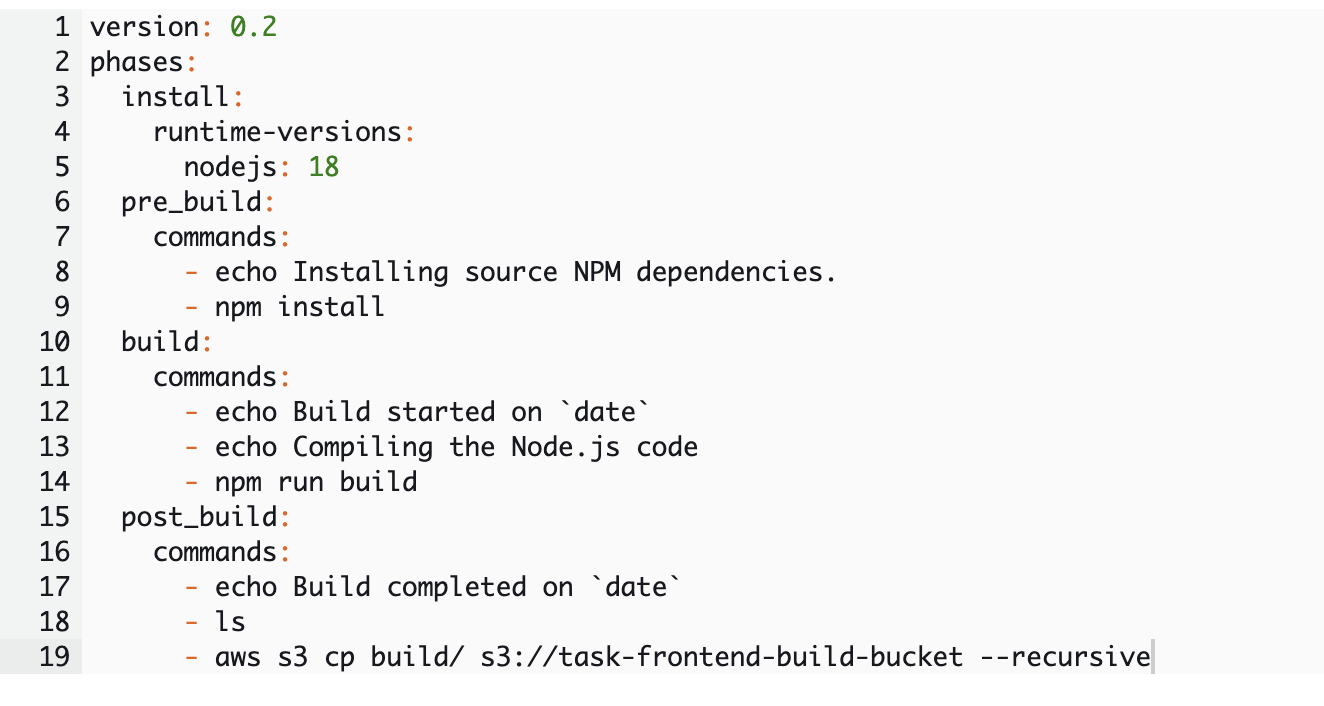
Make sure when running the Distribution dns on your browser for example on chrome , you go to -

In chrome, go to settings and in content, go to site details, select the distribution domain , scroll down and go to insecure content and set it to allow. Or just use the link -   
chrome://settings/content/siteDetails?site=https%3A%2F%2Fd34faz4xwep2kw.cloudfront.net%3A443

Or we can make the cloudfront run on http protocol by specifying it explicitly like - http://d34faz4xwep2kw.cloudfront.net/

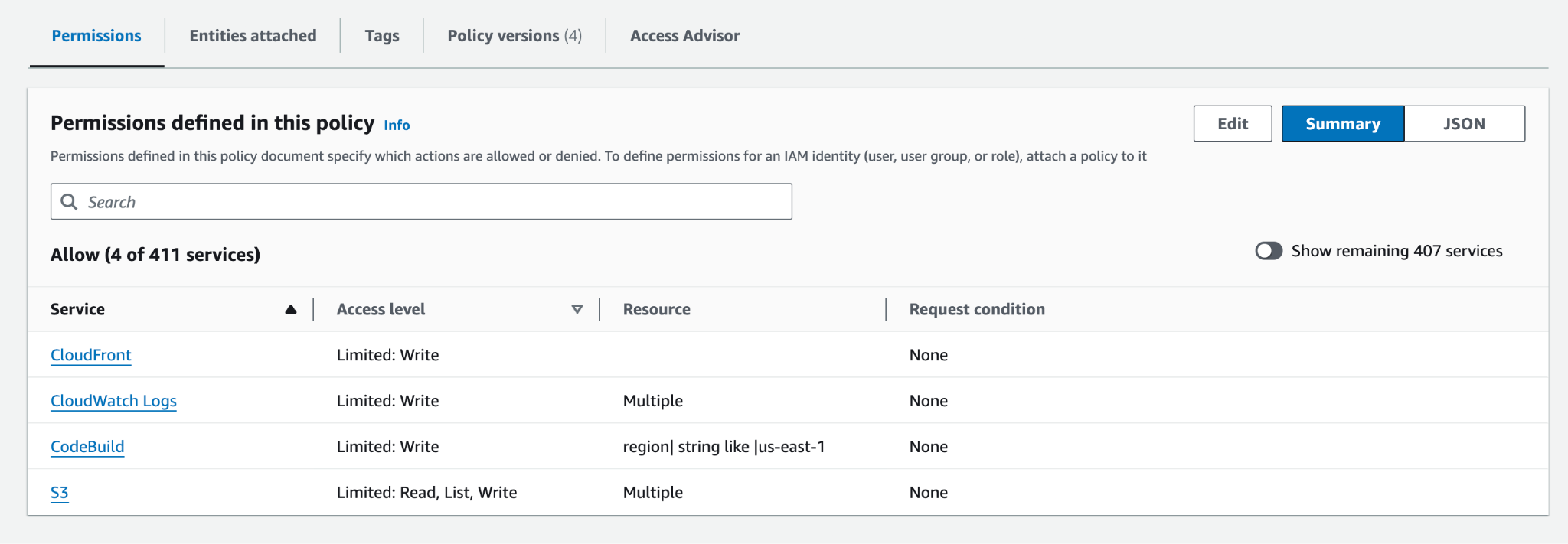
This is necessary because we haven’t got a ssl certificate as we don’t have any domain and our cloudfront is using an https request while we don’t have a certificate yet.

The buildspec.yaml of codebuild -



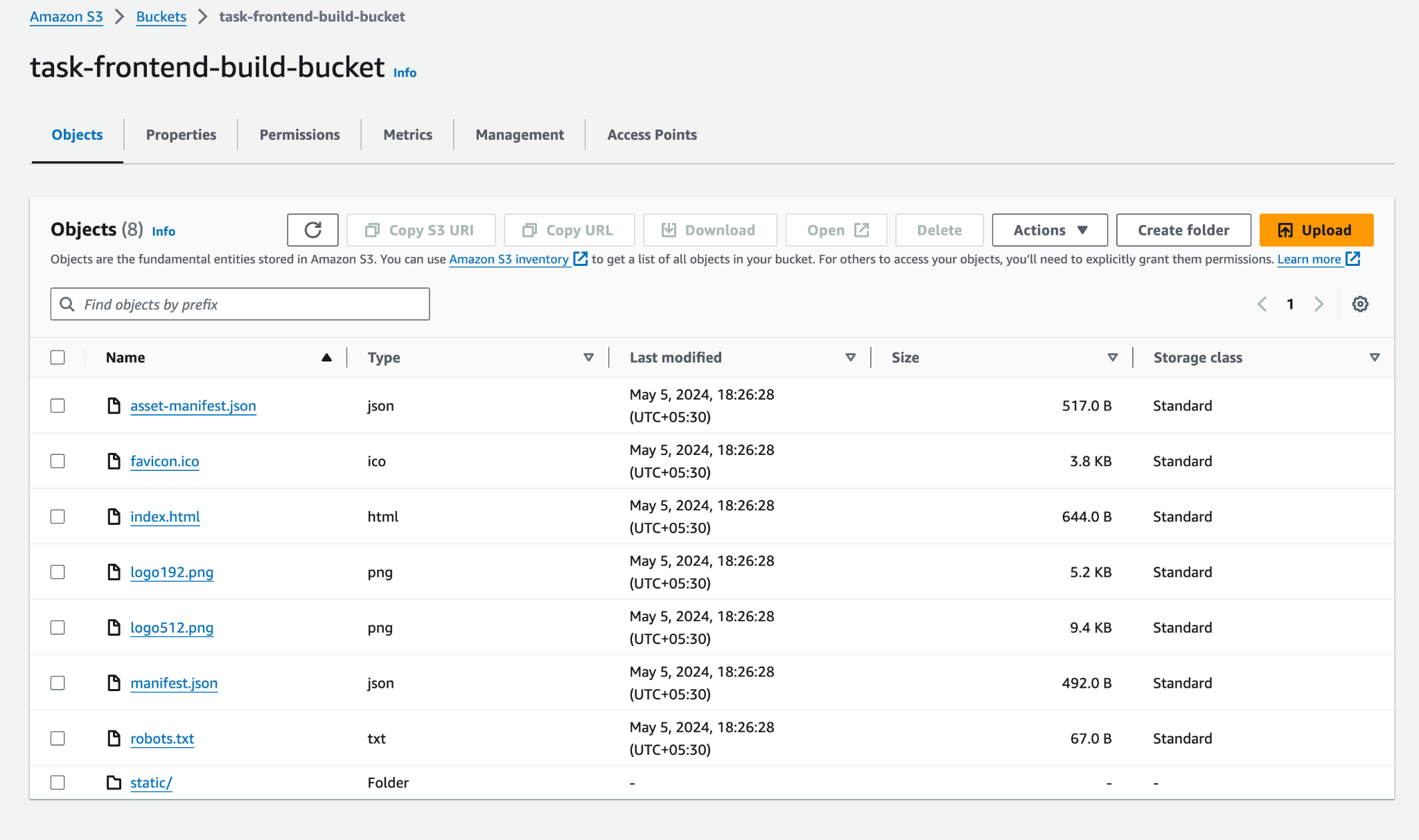
The above buildspec compiles the code and puts it into the S3 bucket called -   
task-frontend-build-bucket

Permissions defined in the policy for the codebuild service role of frontend codebuild



S3 frontend build bucket called -

Task-frontend-build-bucket -



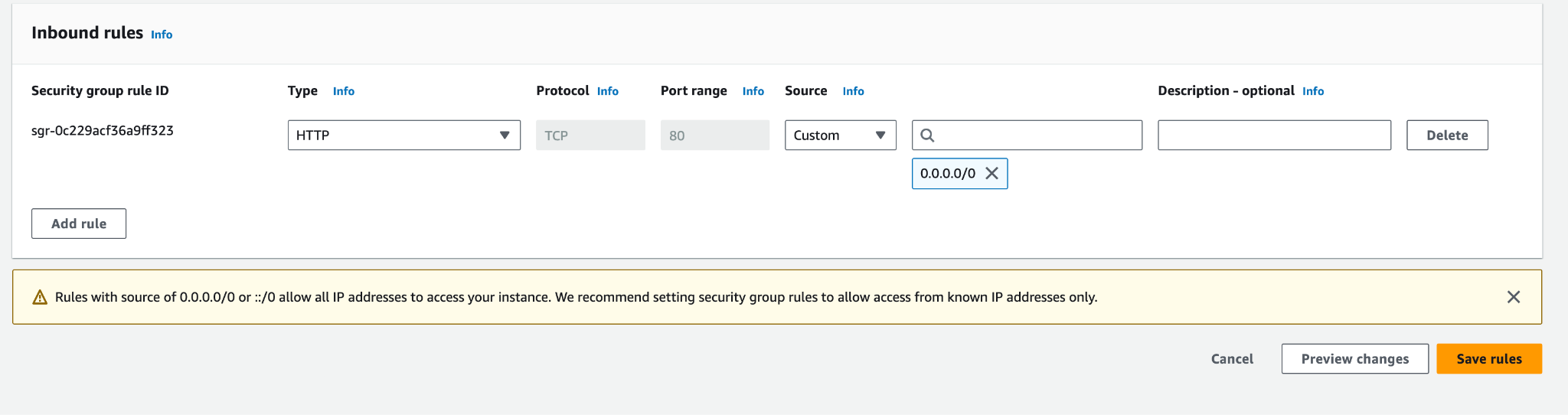
This is generated from the codebuild process.

**Backend :**

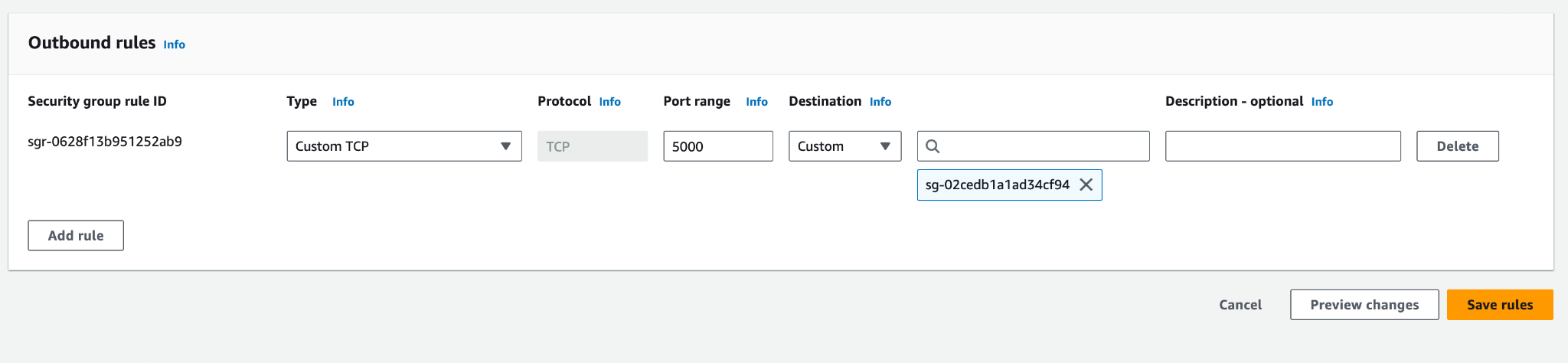
Services used :

1. AWS Codepipeline- includes codebuild and ECS deploy
2. AWS Secrets Manager
3. S3
4. ECS
5. ECR
6. Cloudwatch
7. ELB - Application Load balancer

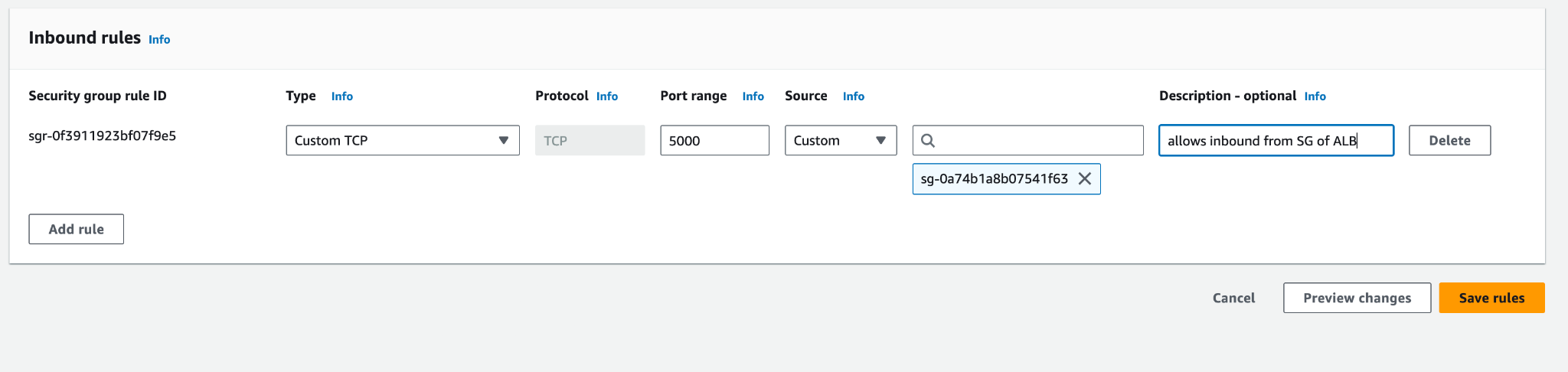
Load balancer security group inbound -



Load balancer security group outbound which does port mapping to the ECS service from port 80 to port 5000:

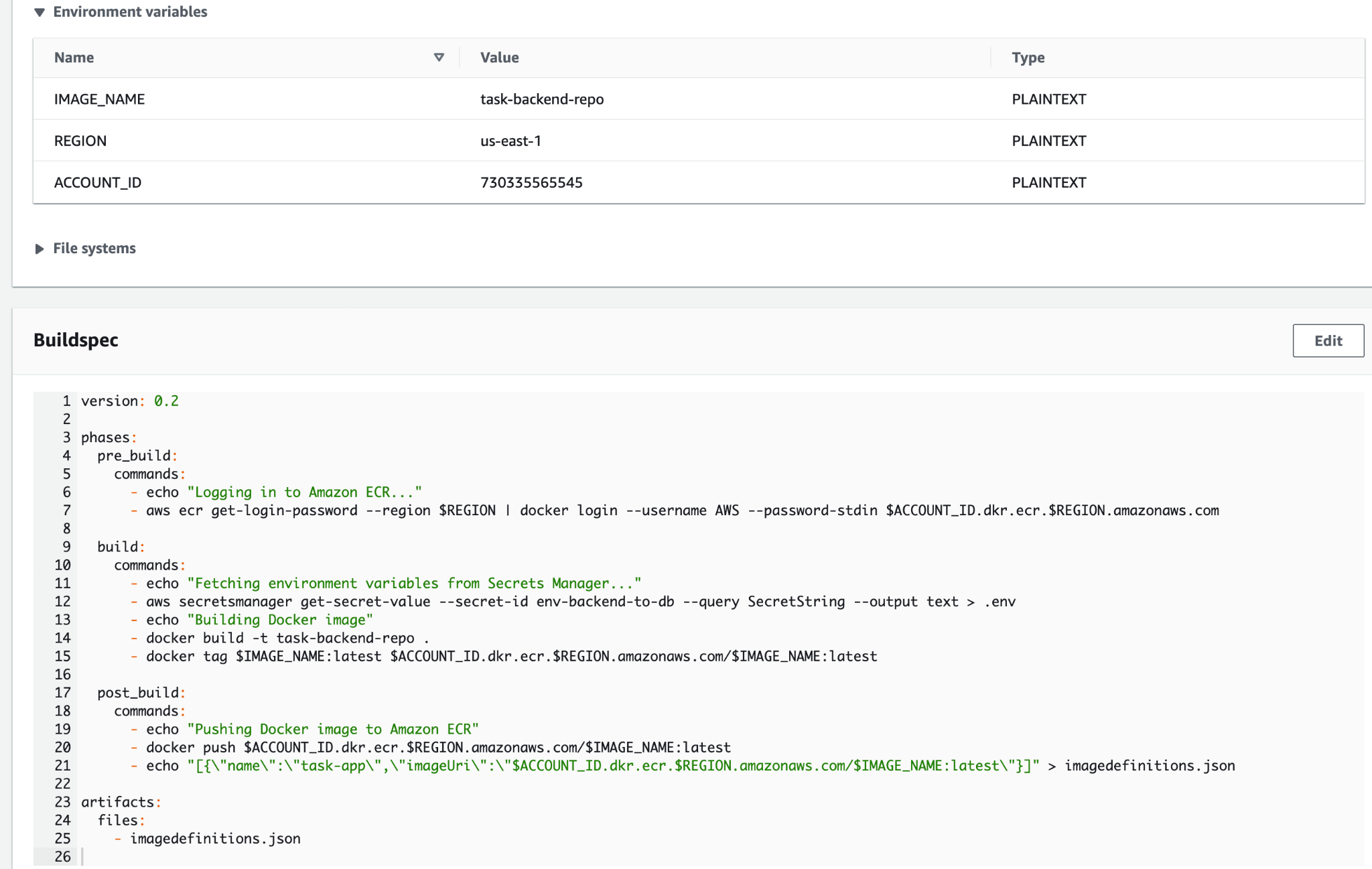


ECS task security group allows inbound from Security group of ALB



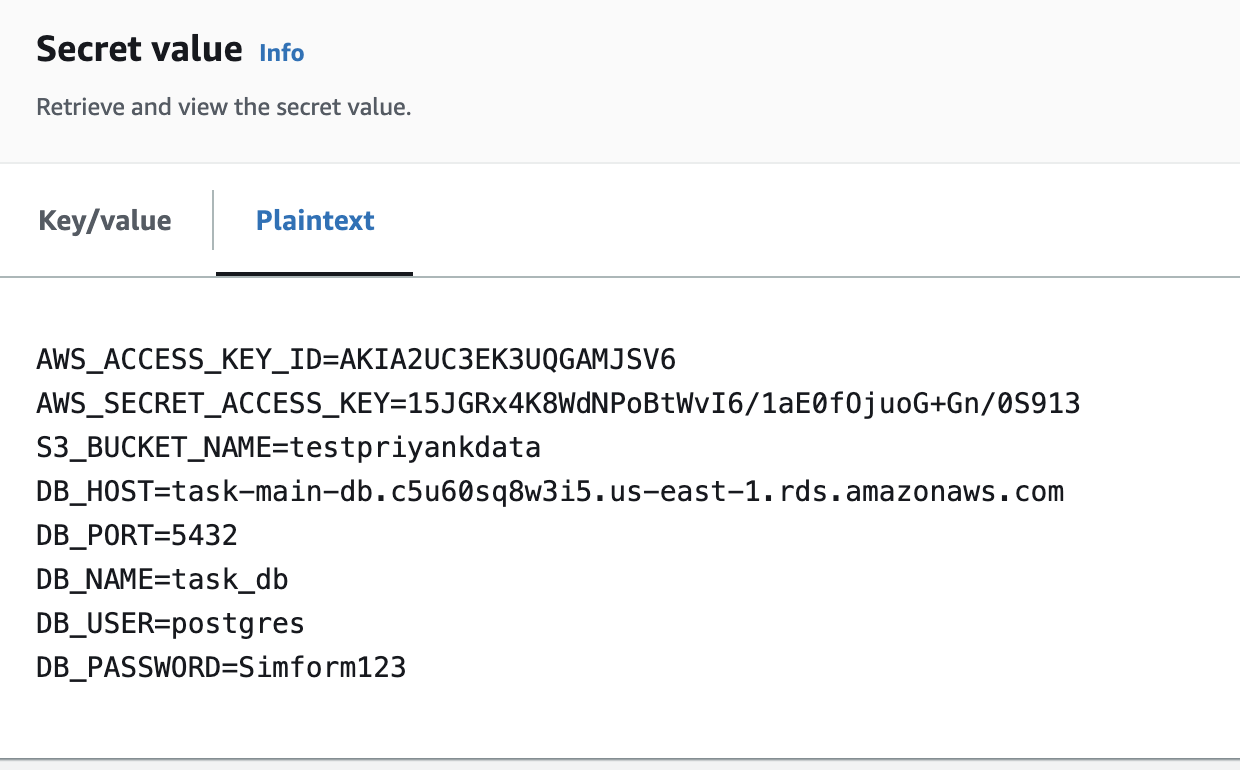
The backend codebuild pulls image from the backend code github link

Codebuild buildspec.yml of backend -   
The env variables used for the buildspec.yml are also mentioned



This buildspec gets the secret value in plain text format and stores it in the built image so that when we run the code(container) , it can fetch the env from its environment. It also pushes the image to the ECR

Secret value in Secret manager

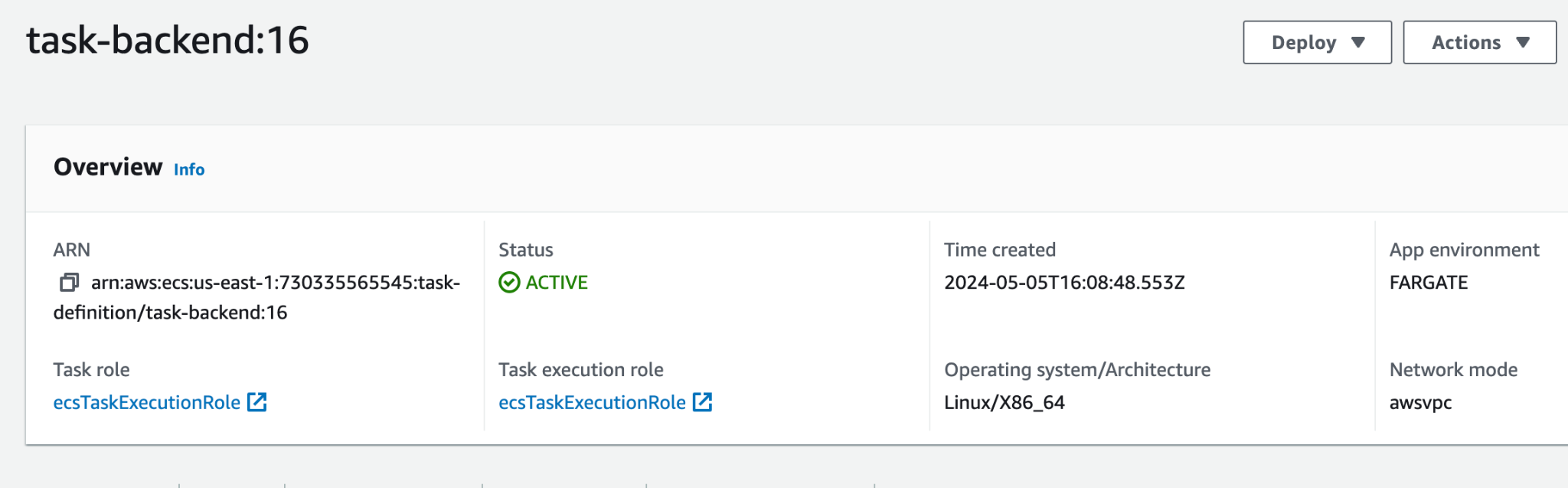


Application load balancer is in all 3 AZs for high availability and is in a public subnet to allow incoming requests from outside.

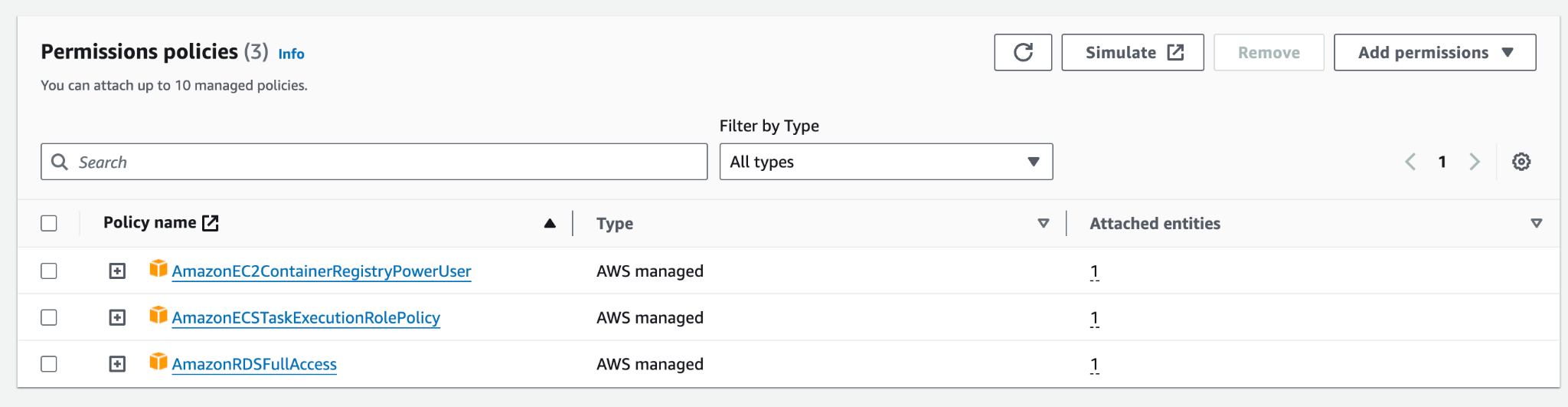
Load balancer accepts port 80 from all inbound IPs. I have kept this open for everyone to test it.

ECS accepts inbound on port 5000 from SG of load balancer.

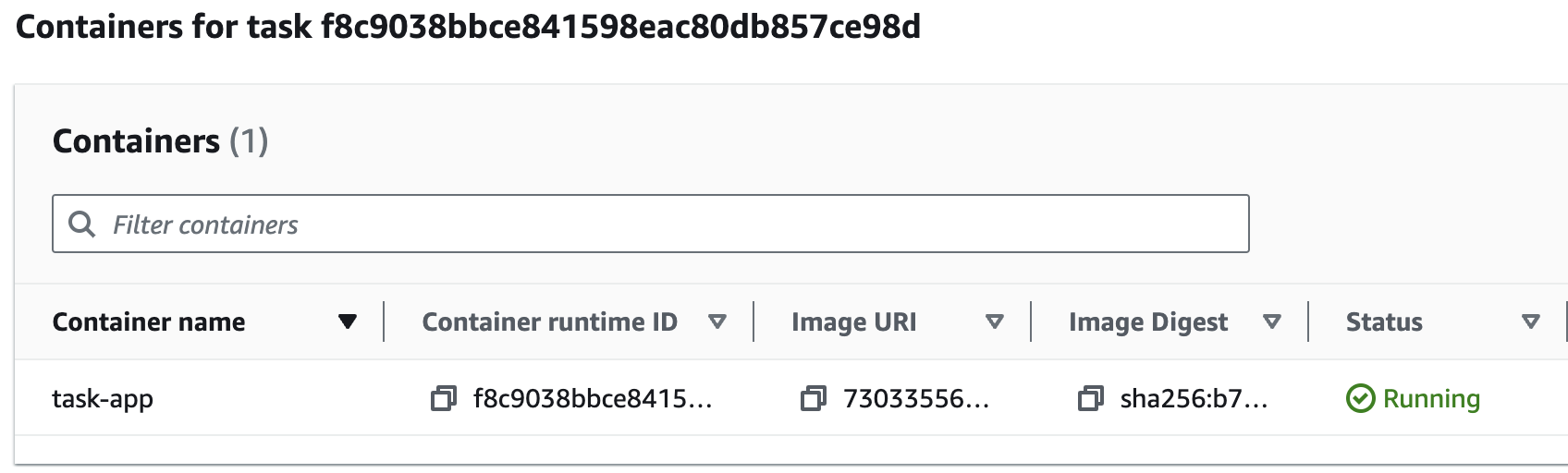
ECS service task definition -



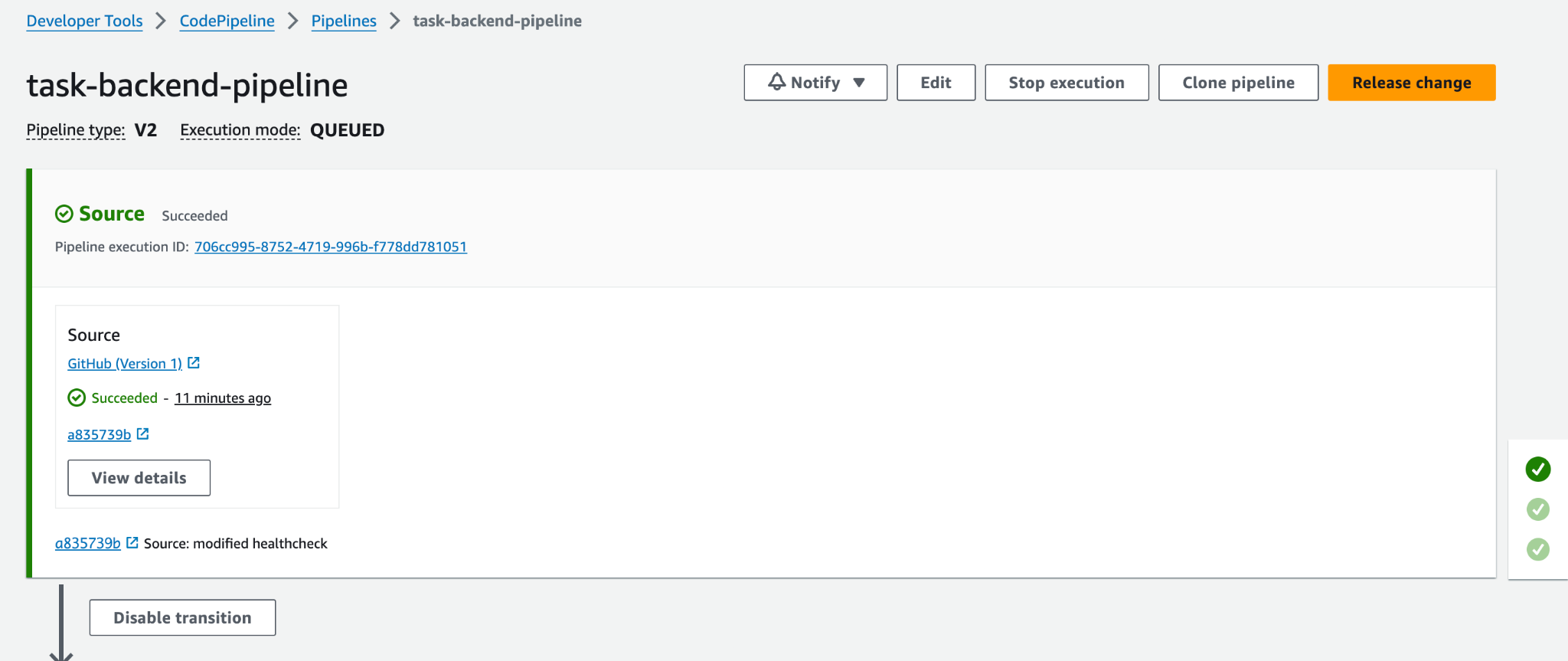
ECS task definition role given the access of ECR, RDS

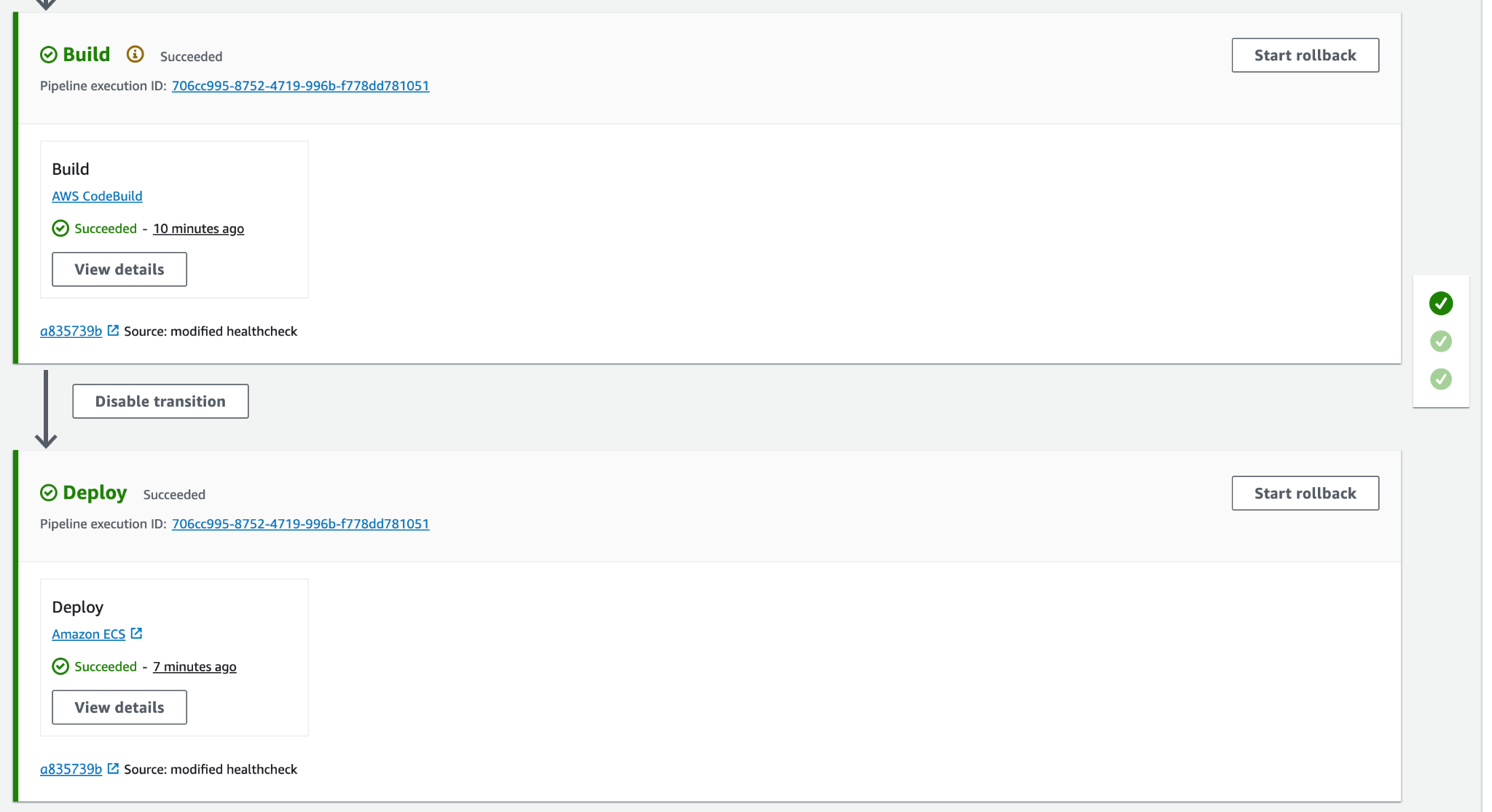


ECS service task running -



**Codepipeline in its executing state:**

****

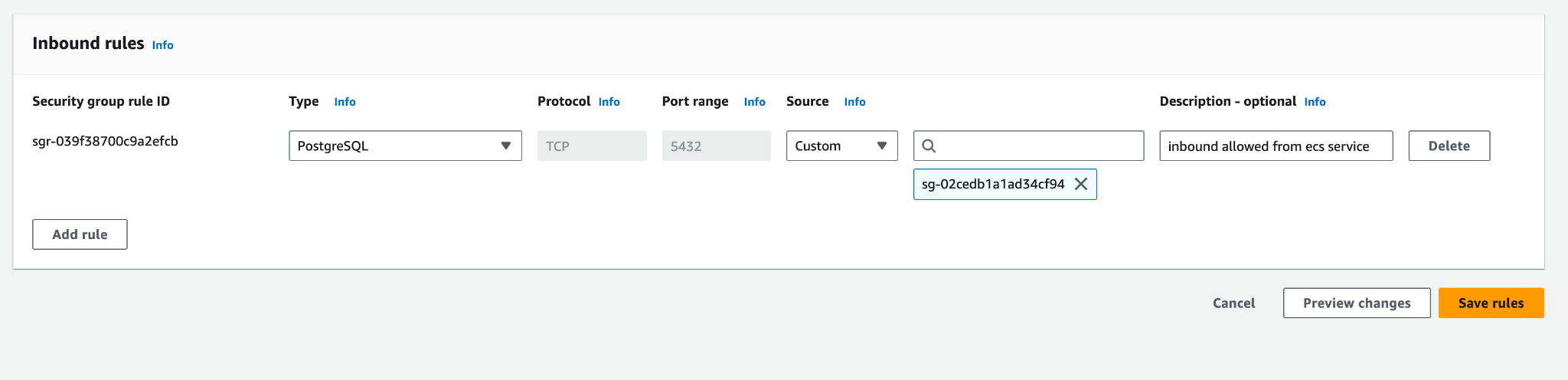
****

**Database** :

Databaseis stored in RDS Postgres which is in a private subnet so not publicly accessible, but as ECS and Database are in the same VPC and they have been given proper IAM roles and permissions, they can talk to each other.

Database is in 2 subnets in 2 different AZs for maintaining high availability and medium costs of data transfer.

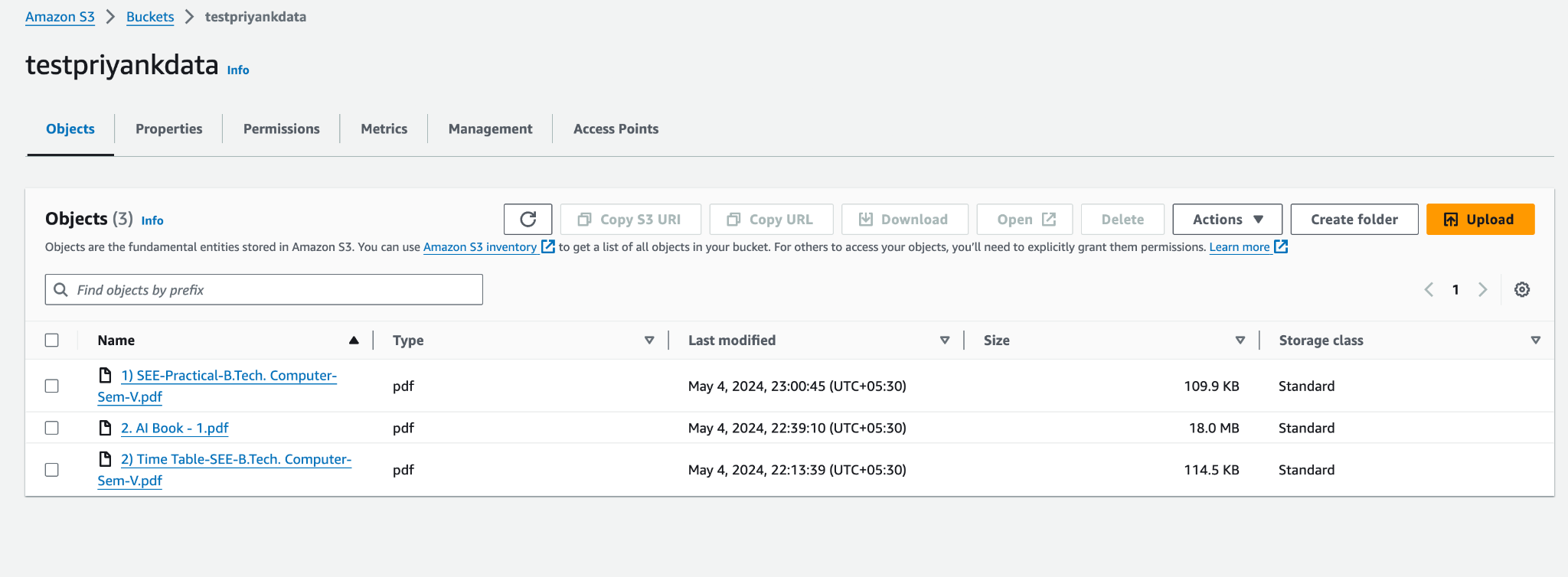
Database’s security group allows traffic only from the security group of ECS service.



Application load balancer is in all 3 AZs for high availability and is in a public subnet to allow incoming requests from outside.

Load balancer accepts port 80 from all inbound IPs. I have kept this open for everyone to test it.

ECS accepts inbound on port 5000 from SG of load balancer.

Screenshot of S3 bucket where the documents are being uploaded -   
 ****

We can also check the database which is private which is also receiving a log of the following things in a table -   
   1. IP of the user

   2. Filename

   3. Extension

   4. Size of the file

We can do this by creating an ssh tunnel to the private RDS instance by creating an EC2 in the public subnet in the same VPC as the RDS instance and then using it as the middle-man (ssh tunnel) to connect to the RDS instance from our local system or host by using a tool like pgadmin .

We would then need to create a table by running a query -

CREATE TABLE public.users

(

filename character varying(255) NOT NULL,

"user-ip" cidr NOT NULL,

extension character varying(255) NOT NULL,

"file-size" integer NOT NULL,

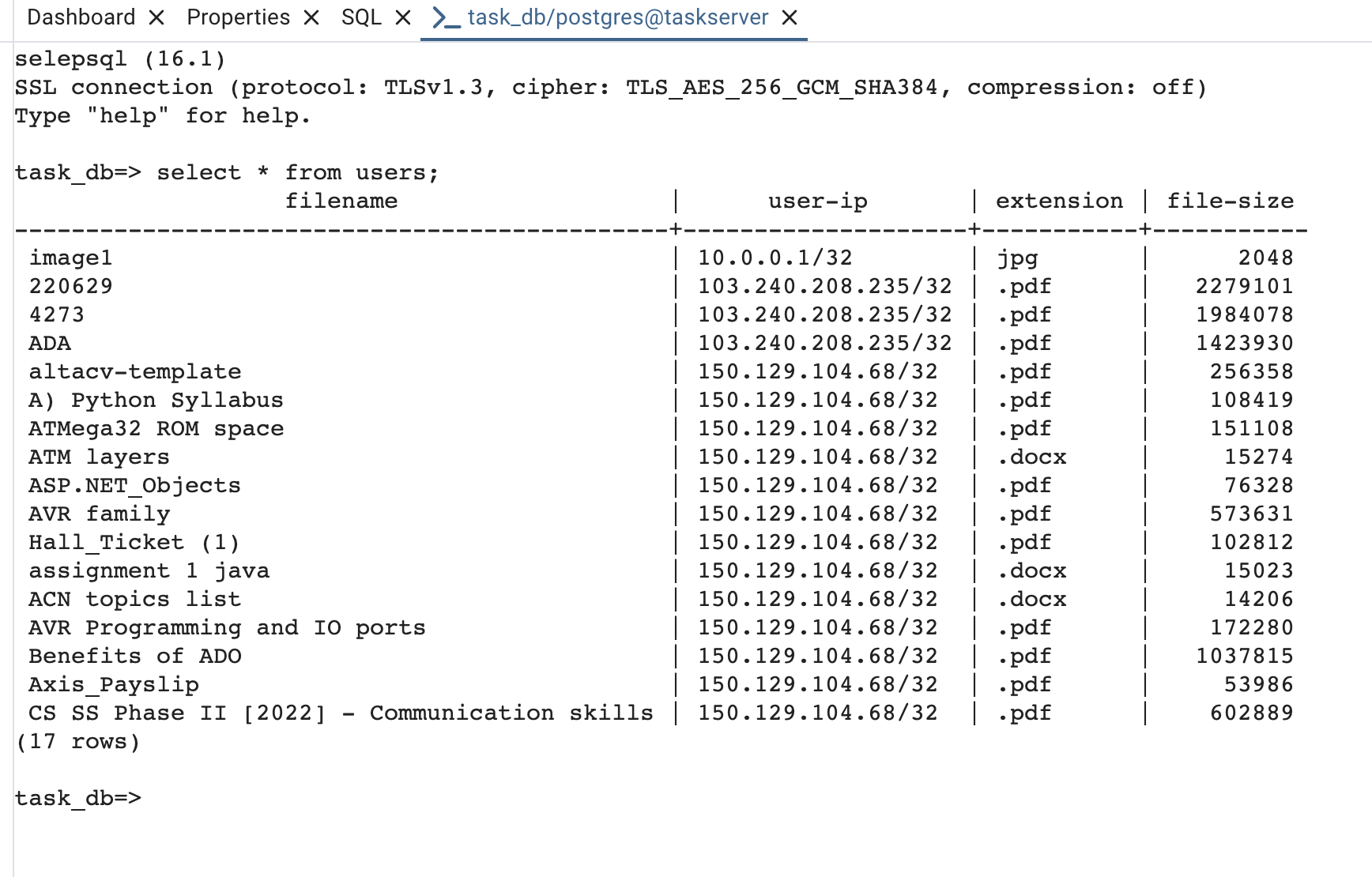
PRIMARY KEY (filename),

UNIQUE (filename)

);

Then after uploading many files from our frontend (cloudfront), we will get the required table values.

By running a query in the psql server , we see the output that we need.

****

**Architecture for basic understanding of the flow .**

