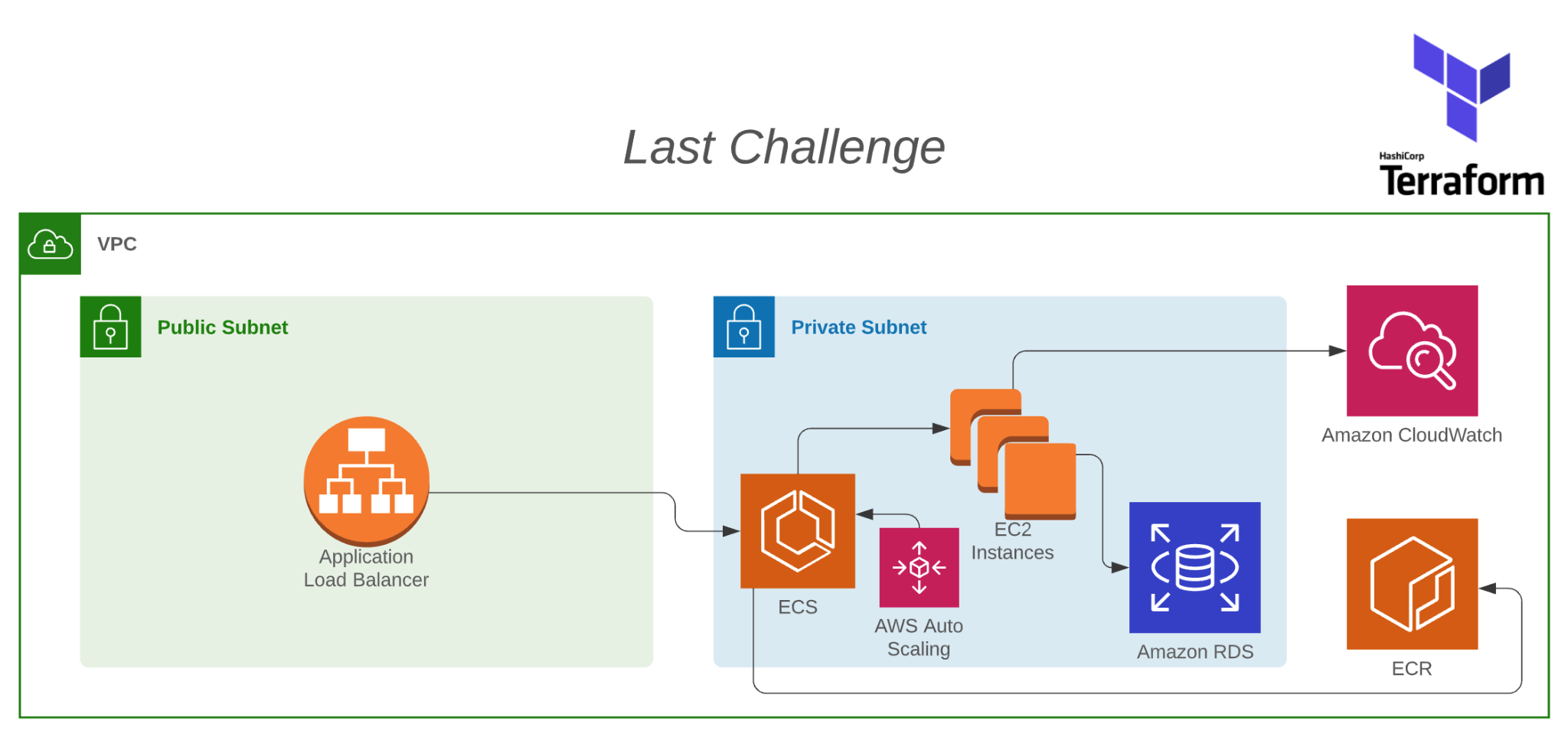
## horizontal line



Last challenge

11.03.2022

**─**  
Felipe Gabardo de Castilho

# Overview

For the last challenge we are going to be including ECR and ECS resources and also start using infra as code.

# Tasks

1. Move our infra resources to terraform
2. Include an ECR and ECS in your terraform code
3. Make the necessary changes in your Load Balancer and Auto Scaling
4. Our ECS will use our Docker image, where the image should be available in ECR

# 1 . Prerequisites

The first step for this project is a series of prerequisites: AWS CLI, Docker, Terraform

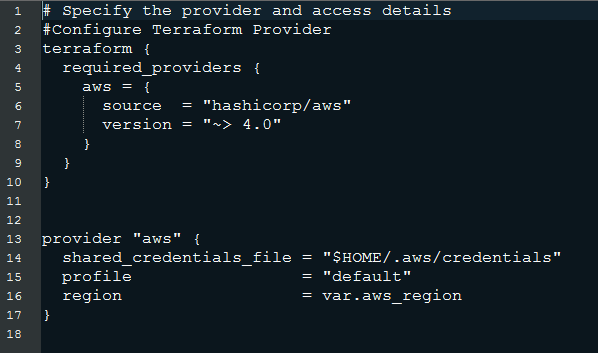
## Let’s get started

We begin by creating the project and setting up the file structure that we want to use. I’ve decided to break up the code into files handling each of the different parts of our infrastructure, making it more clear to understand.



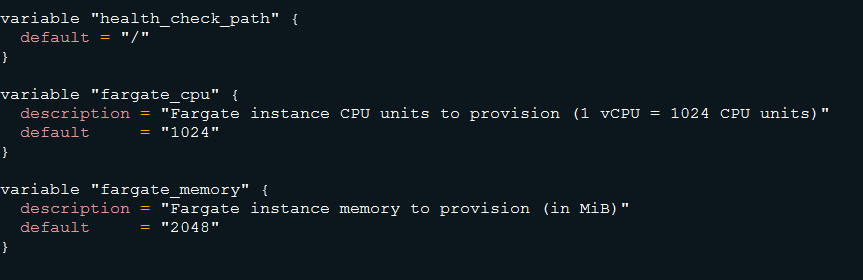
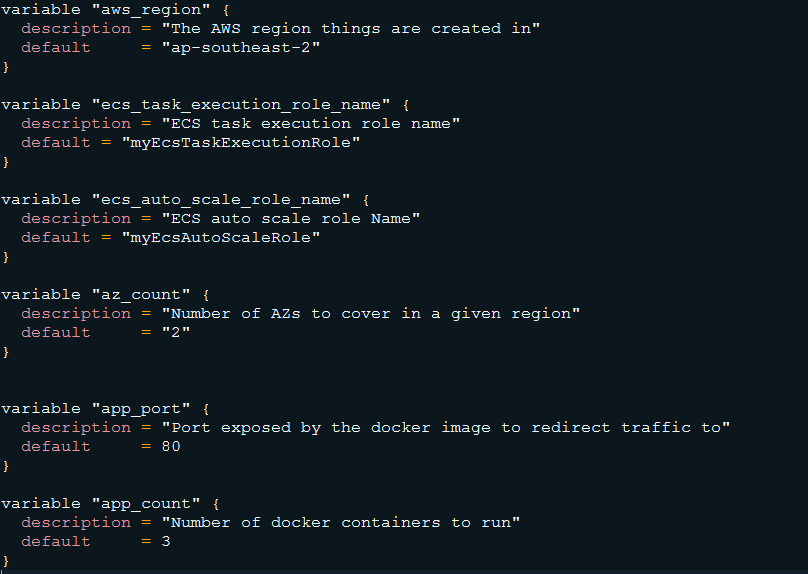
## Provider.tf

The fist file that we are working on is the *provider.tf,*  where we are telling Terraform that our provider is AWS and the location where our credentials can be found.



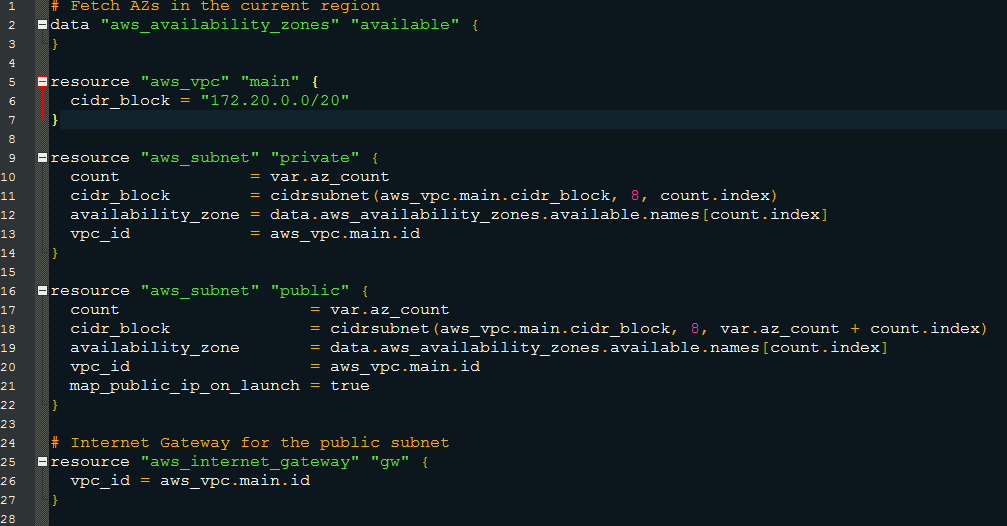
## Variables.tf

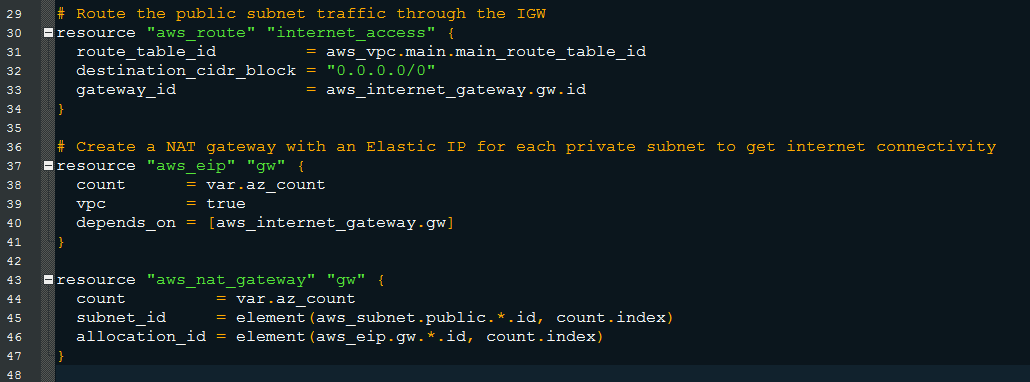
In the variables.tf file we set up our variables for the application. Terraform will look here for these values anytime it sees *var.VARIABLE\_NAME*

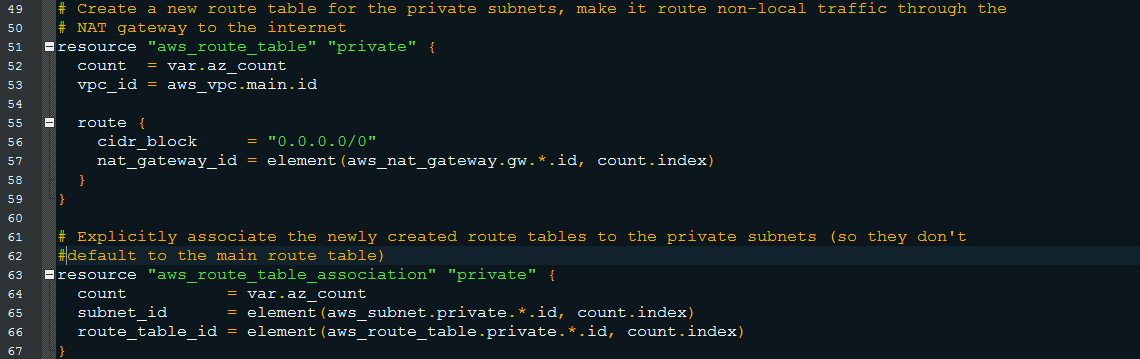


## Network.tf

For our network, to get the minimal amount of high availability, I deployed our ECS cluster to run on at least two Availability Zones, as our load balancer also need at least 2 public subnets in different AZs

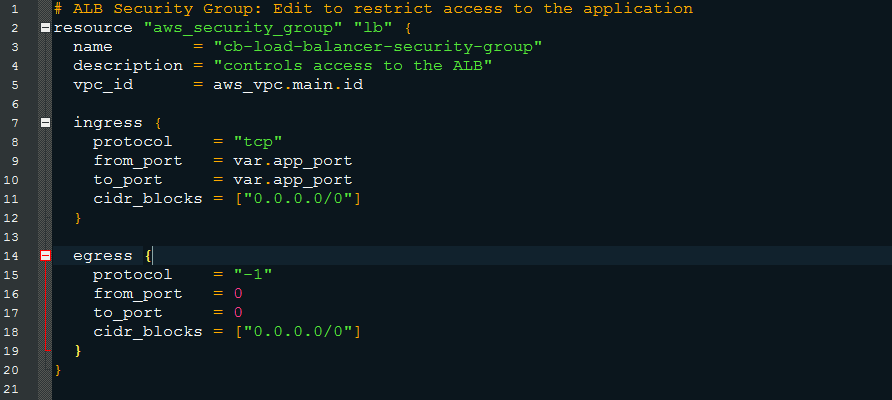


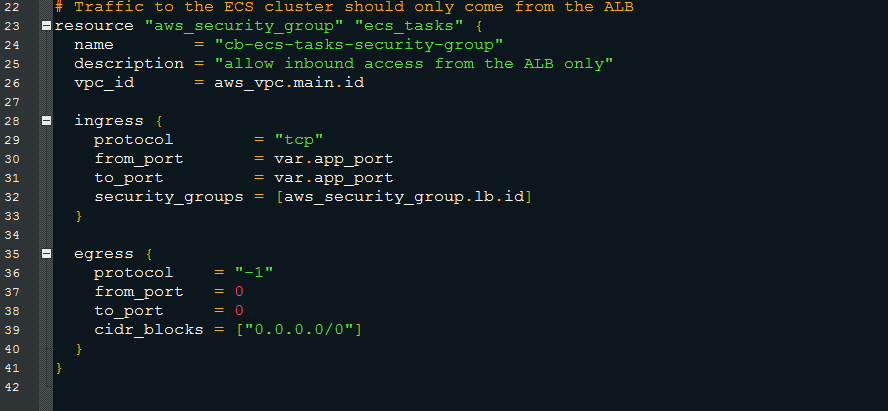




## Security.tf

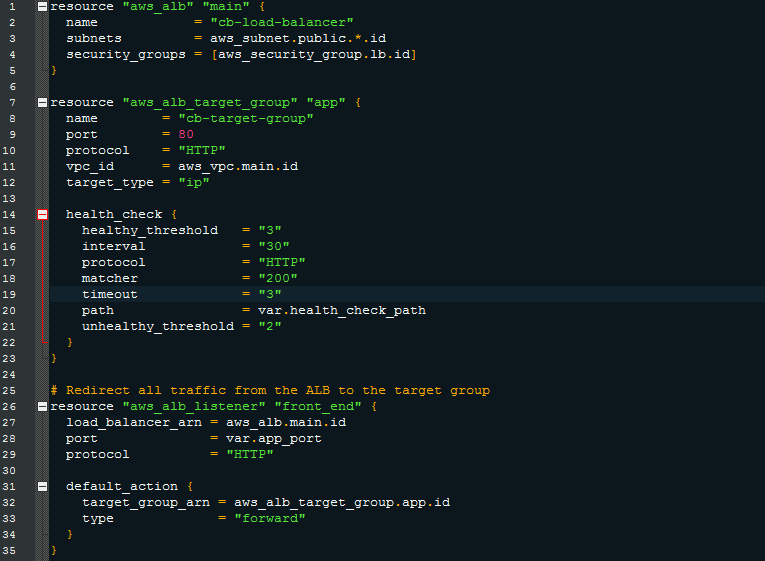
With the network created, we set up the security groups for our load balancer and ECS





# 2. ECS and Load Balancer

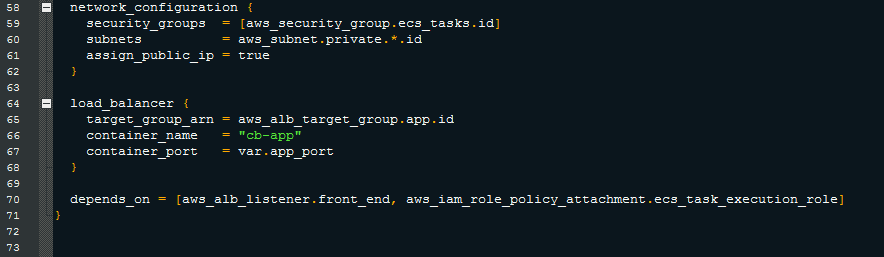
After creating all the files above, it is time to build our application load balancer and ECS cluster.   
Alb.tf  
Our load balancer was created with a health check, listening for HTTP requests on port 80



Ecs.tf

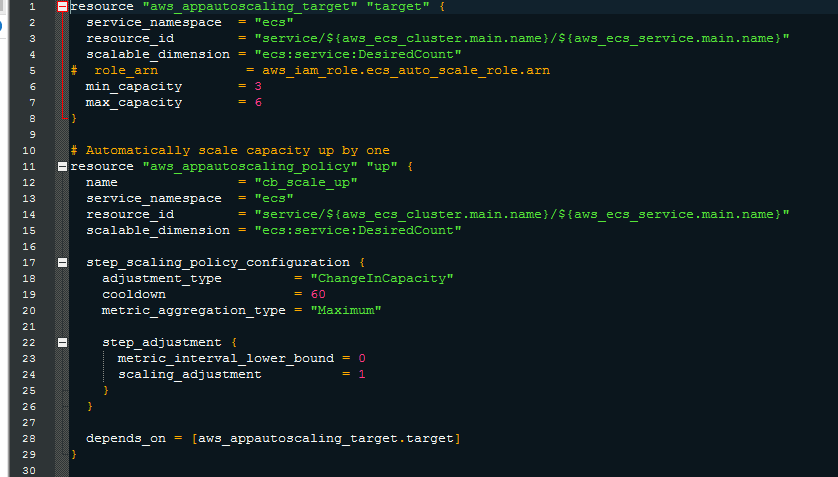
Then for the ecs.tf, it was the most time consume file to create, as this is where first we created our ECR repository, we are going to push our Docker image. Also, we create our ECS cluster, Tasks and ECR service

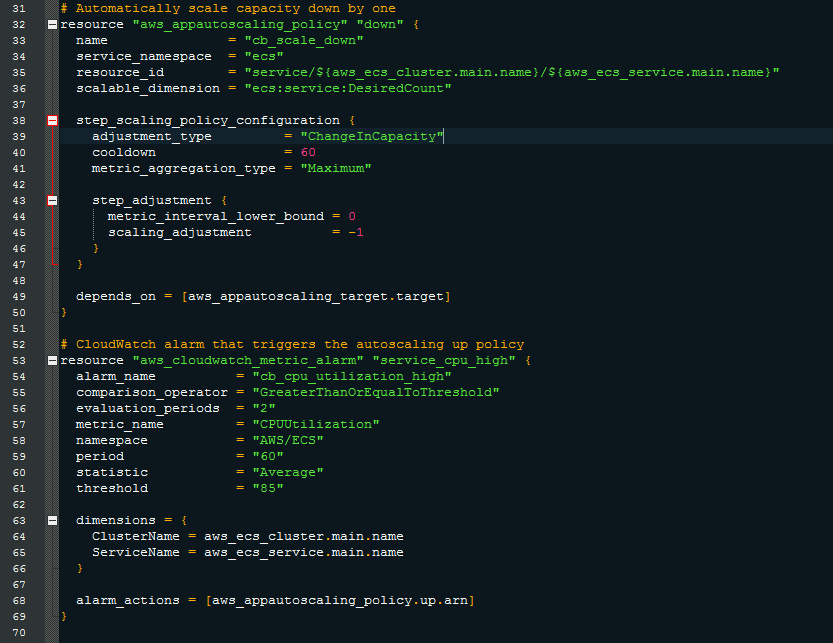


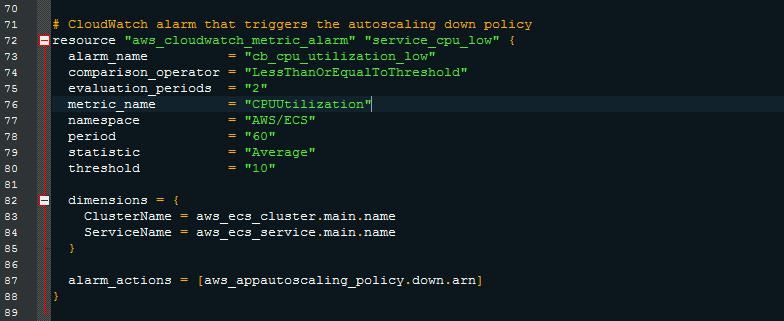


Auto\_scaling.tf

The next was the auto\_scaling.tf, where we get an auto scaling set up to monitor our application and adjust the capacity, setting up the minimum capacity to 3 and maximum to 6 tasks. Also, we set up a CloudWatch alarm that trigger the autoscaling up and down policy for CPU Utilization.

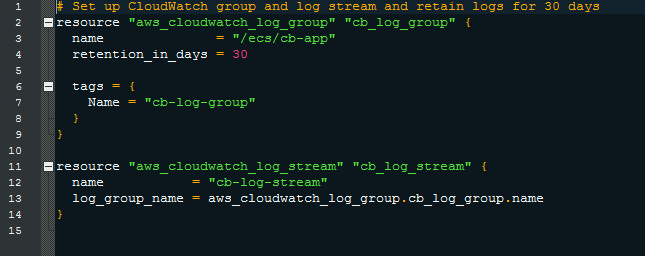






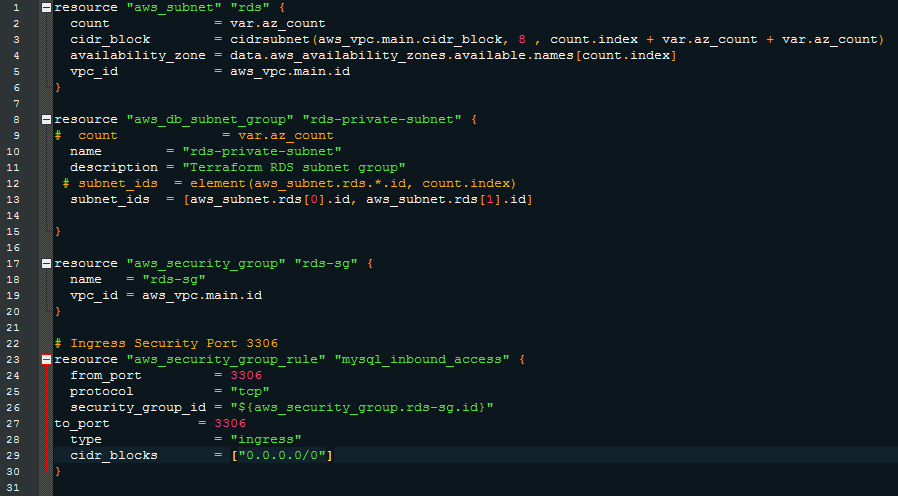
Logs.tf

Then we set up our CLoudWatch group and log stream



RDS.tf

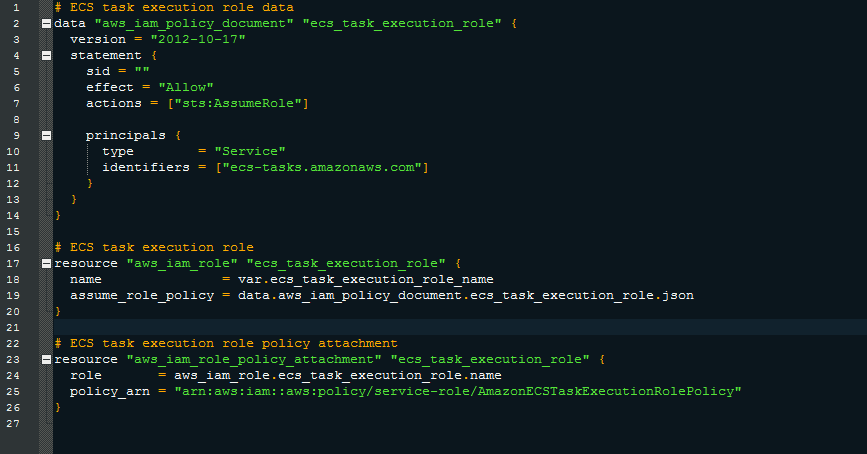
It is in the RDS.tf that we created our RDS database, setting up the variables for the db\_instance, as also the private subnets, subnet group and security group.





Roles.tf

We also have the IAM roles and policies for our ECS tasks

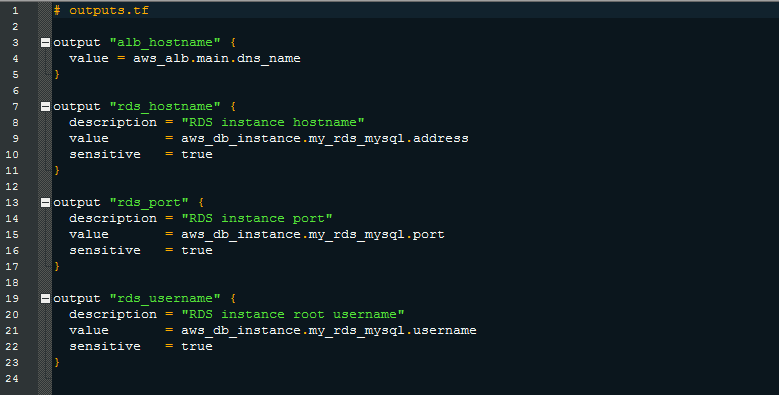


# 

# 

# 3. Deploy

With the infrastructure built for our application, it is time to deploy. Last but not least, we have our *outputs.tf* file to get data back after a successful deployment.



# 4 . Project update - RDS Endpoint

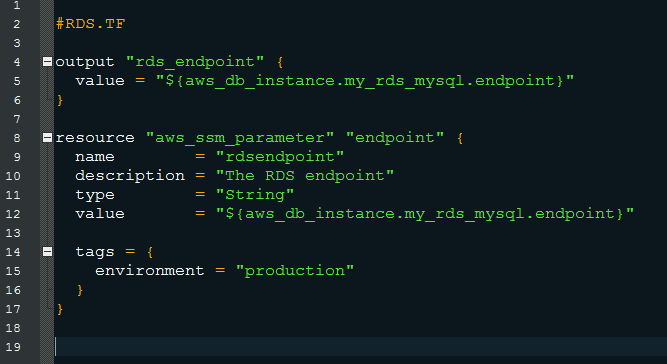
With the infrastructure built for our application, an improvement was yet needed. The situation was: Every time that the terraform code was deployed and a new RDS database was created, with that a new endpoint was generated.

This endpoint was set under the .env file on the db\_host parameter, with a new and updated endpoint being generated, there was the need to manually change the .env file, updating DB\_HOST endpoint.

With the aim to find a solution for this situation, the following tasks were executed:

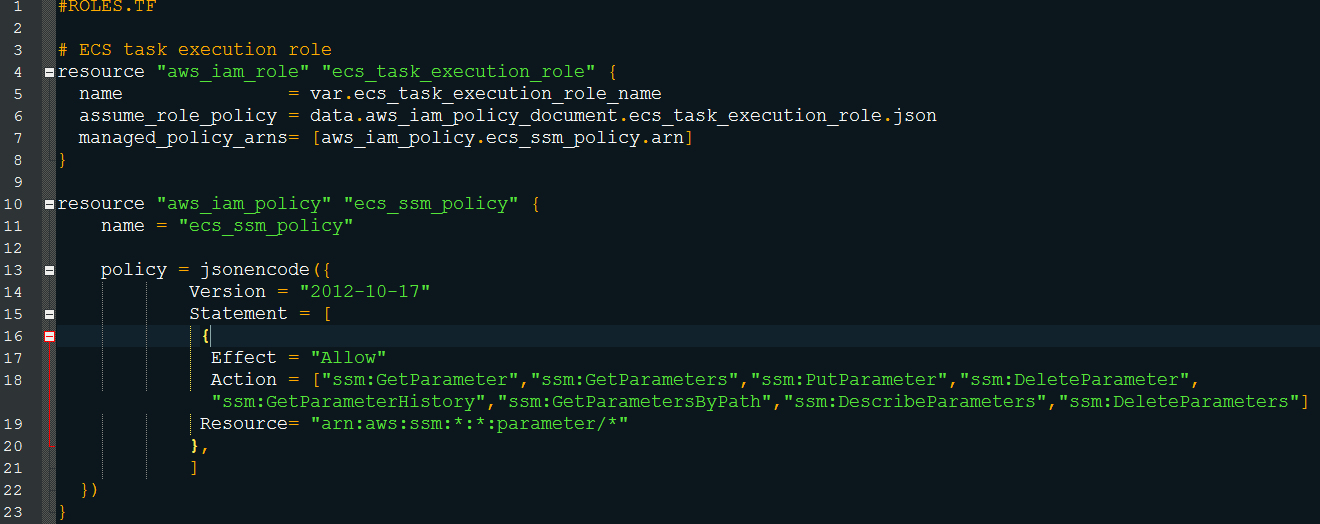
rds.tf

Created a AWS\_SSM\_PARAMETER from the resource “aws\_db\_instance.my\_rds\_mysql.endpoint”



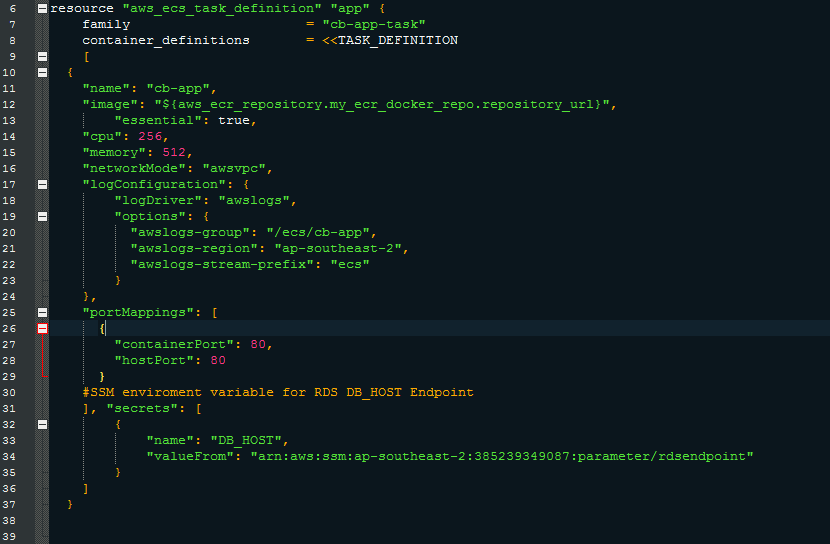
Roles.tf

With a new SSM parameter created, it was time to update the IAM policy to allow access to the SSM register.



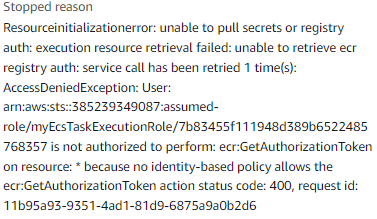
Ecs.tf

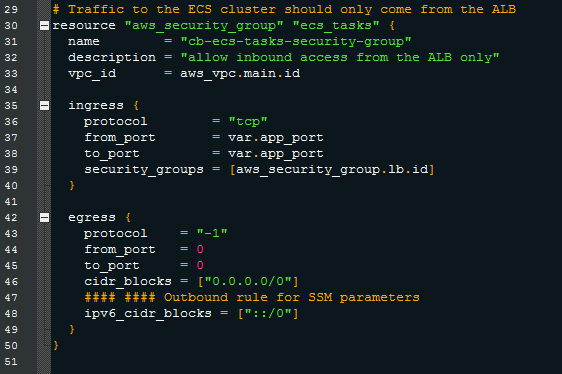
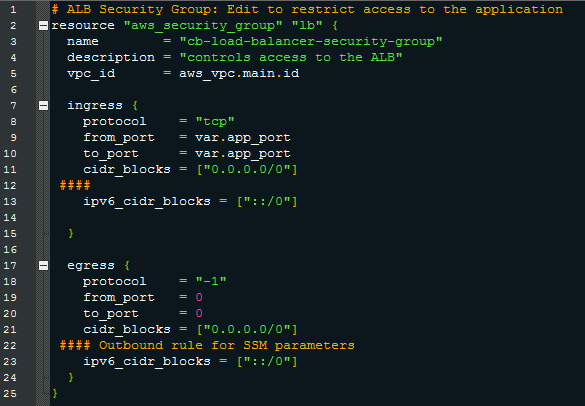
After the configurations above are running successfully, it was time to set the environment variable for our ECS tasks by adding the ARN path in our *aws\_ecs\_task\_definition*  under *secretes*

**

Security.tf

A new configuration for security.tf was needed due to the following error when deploying our task.



For this, a new outbound rule ( ***ipv6\_cidr\_blocks = ["::/0"]*** ) had to be included under resource "aws\_security\_group" for the load balancer and for the ecs tasks

# 4 . Project solution - RDS Endpoint

With all the new configuration above, our ECS tasks are using the updated RDS endpoint provided in our SSM\_PARAMETER, where we were able to deploy our application through terraform and push our docker image without the need of any change in our previous docker image.

