AO Technical Task 2 - Dhanuka

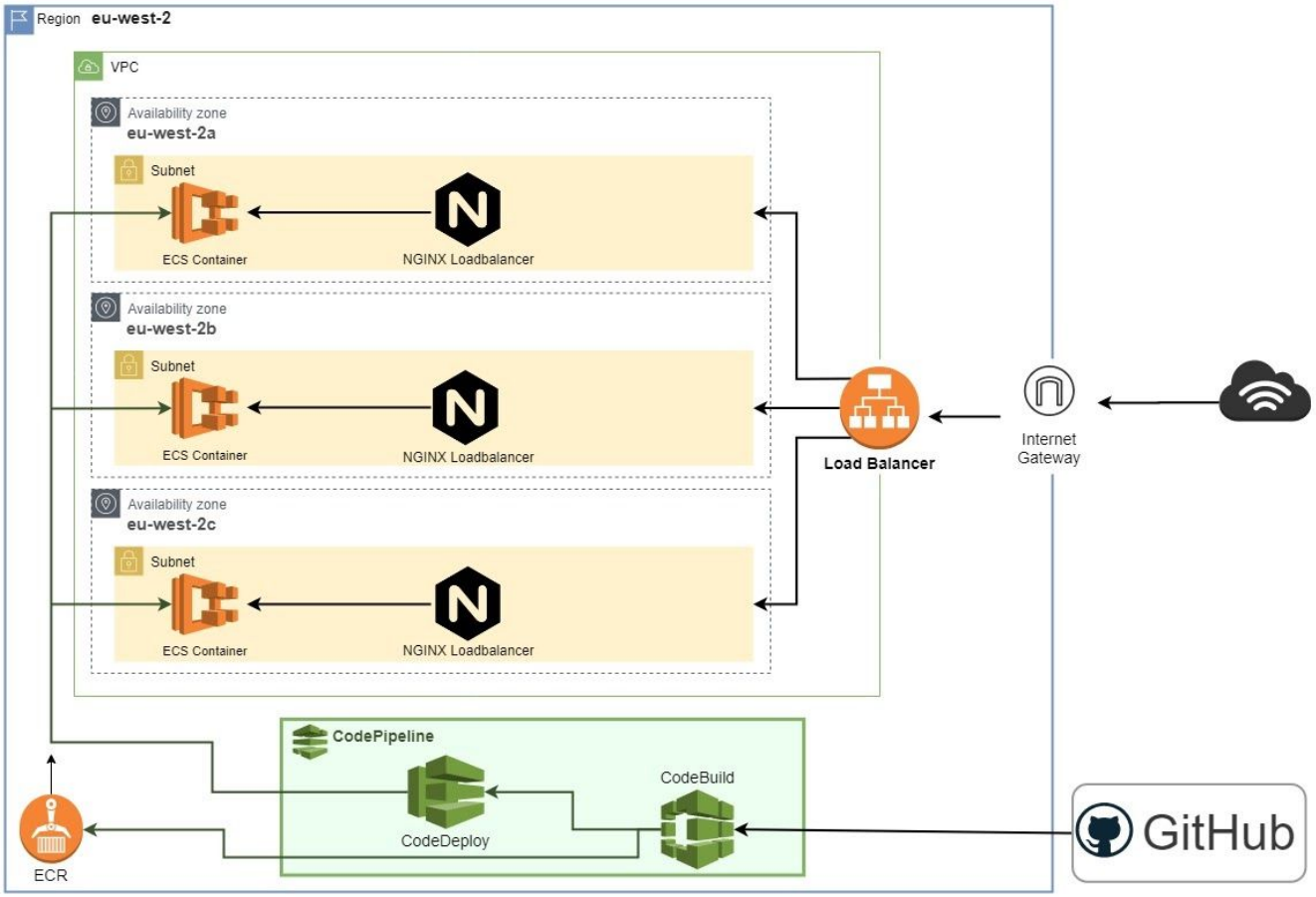
NGINX Load Balancer

As instructed in the Task I have created an Nginx load balancer for handling traffic to the app. The “nginx.conf” can be configured to redirect traffic to multiple servers, in this case I have used it to load balance between containers “aspnetapp” image is running on. I have also made a “docker-compose.yml” file to build the main application container image as well as the nginx server container image automatically. I have used the local host remote ip address with the addition of a container port for testing purposes in the local machine.

How to test-

1. Build docker image of the app with the tag "aspnetapp"
2. run "docker compose up" to run nginx container and app container.
3. connect to localhost:8080 on your browser.

This connects to the nginx load balancer and will serve the application container.



App Environment and Deploy pipeline

Environment (ECS Fargate)

IAM Roles - I created the IAM roles necessary for the code pipeline and the AWS Environment the app will be deployed to.

AWS ECR - Created an ECR repository to store container images which can then be pulled from the ECS service.

AWS ECS cluster - Created a cluster for the ECS containers to be in.

AWS ECS service - Created the ECS services and task definitions for the containers.

Load balancer - Created a AWS load balancer to direct traffic to different subnets the app will be deployed to.

VPC and Subnets - Used default AWS subnets and VPC

Deploy pipeline

CodePipeline

- Code pipeline requires quite a large set of IAM roles and policies which can be found in "/deploymen-iam.tf"
- The codepipeline uses github as a source and pulls the files after authenticating with a github auth token.
- Codebuild uses a S3 bucket for building artifacts which is also created in the terraform environment. Codebuild builds and uploads the docker images into the AWS ECR.
- The deployment step consists of the pipeline deploying the created images into our AWS environment.
- Deployment groups - Created deployment group for the application with blue green deployment

Buildspec.yml - Created build spec file to log into ecr, build and upload the images to the ecr repository. This file is needed for codebuild and includes the build commands needed in order to build the app and upload to ECR.

Future Improvements

- Make use of modules in terraform for better readability and re-usability
- Improve IAM roles with least privilege access to the AWS resources
- Setup a new VPC with private subnets for the application containers so it's only route to the public internet is through the load balancer.
- Change the capacity provider to EC2 if it benefits other services in your environment or if there is a cost benefit.
- Add more outputs depending on what's necessary
- Deploy the tf environment within the pipeline.
- Implement Cloudwatch logs to receive logs from the containers.
- Use AWS trusted advisor to inspect the environment and recommend security changes .