Roulettech Assignment

Project Link: https://github.com/gauthamkv07/Roulettech

Demo video LInk: Roulettech-assignment.mp4

Project overview

I created a project to manage user names and details. The frontend, built with React.js, includes two components: one for collecting data and another for displaying a list of users and their emails. The backend, developed using Django, consists of two APIs: one for retrieving data and another for storing it. I hosted the Django application on AWS EC2 using Docker and deployed the React.js application with CloudFront and S3.

Tech Stack

React.js: Frontend
Django: Backend
SQLite3: Database
AWS S3: Storage
AWS CloudFront: CDN
AWS EC2: Hosting
Docker: Containerization
Nginx: Proxy

Backend Application

I used Docker to build the project and pushed the image to Docker Hub. I created a VPC with one private and one public subnet. To provide internet access to the EC2 instance in the private subnet, I opted to create a NAT instance in the public subnet, as I found the NAT Gateway to be a bit costly. A NAT instance is essentially an EC2 instance within the same VPC that allows the private subnet to access the internet.

Additionally, I created another instance (Bastion Host) in the same VPC to SSH into the EC2 instance. Using the Bastion Host, I installed Docker and hosted the application. I enabled HTTP requests on the Bastion Host to access the application. I used Nginx to direct traffic from the Bastion Host to port 8000 on the private EC2 instance. This setup allowed me to access the application via the Bastion Host.

However, I realized that enabling HTTP on the Bastion Host is a bad practice due to security risks. Instead, I should have used a load balancer, gateways, or other secure methods to access the application. Allowing HTTP traffic on the Bastion Host exposes it to potential security vulnerabilities.

According to best practices, I should have set up a single private subnet, used a NAT Gateway for internet access, and created a Virtual Private Gateway for secure access to the instance. A load balancer should have been used to access the application. Furthermore, I apologize for not meeting the requirement of creating two API endpoints; I only created one.

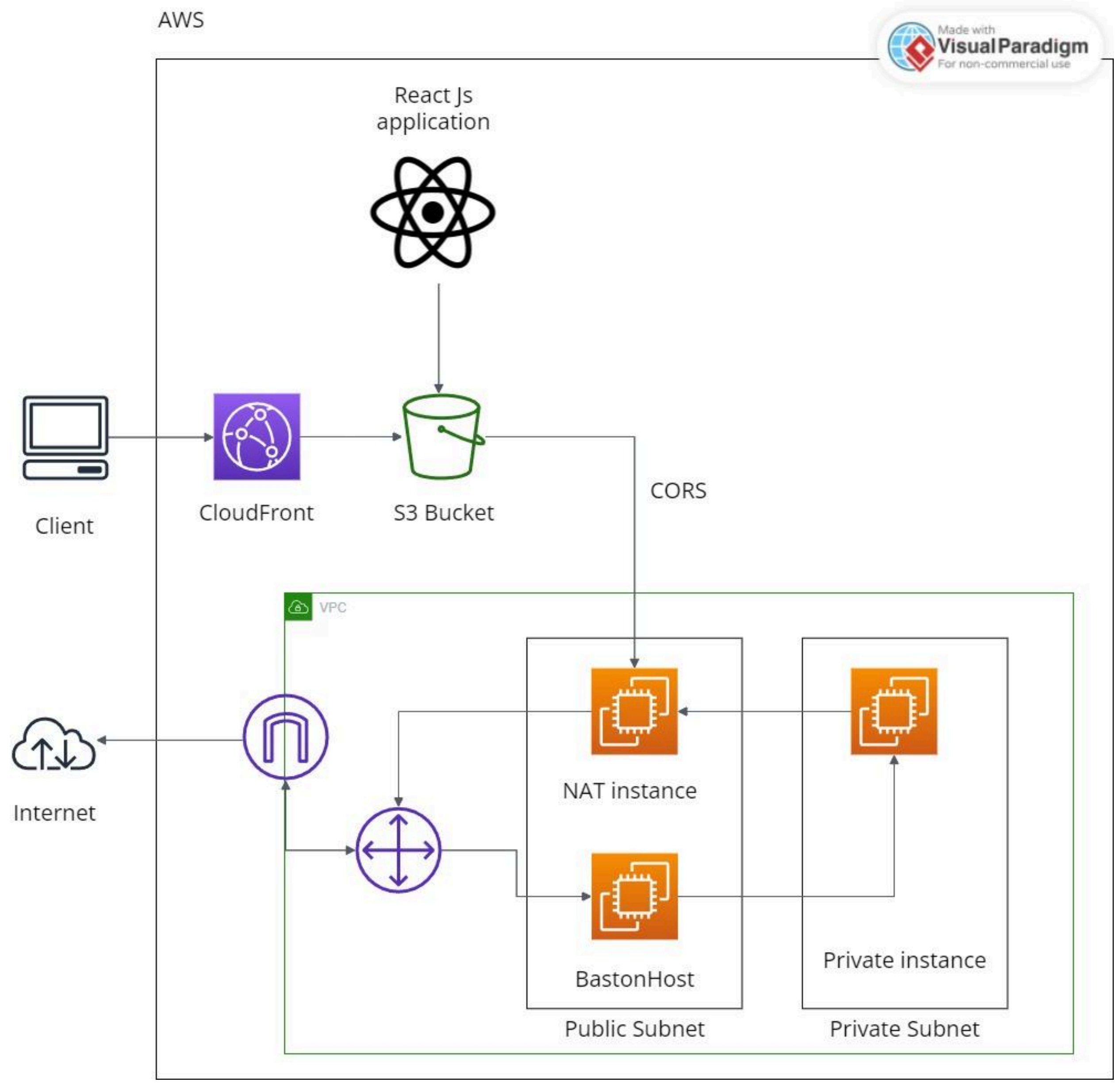
Frontend Application

I built the project using npm, which generated a `build` directory with the necessary files. I uploaded these files to an AWS S3 bucket and configured CORS permissions to allow access. I then set up a CloudFront distribution to serve the content from the S3 bucket and specified `index.html` from the build directory as the default document, providing an entry point to the application. For API connectivity, I used the IP address of the Bastion Host.

Conclusion

With this setup, I successfully developed and deployed a full-stack application. I took great satisfaction in the effort I invested, viewing it not just as an assignment but as a personal project to create and learn something new. It was an enjoyable week, and I’m thrilled to see the application fully operational.

Architecture


AWS
Made with Visual Paradigm For non-commercial use
React Js application
Client
CloudFront
S3 Bucket
CORS
VPC
Internet
NAT instance
BastonHost
Private instance
Public Subnet
Private Subnet