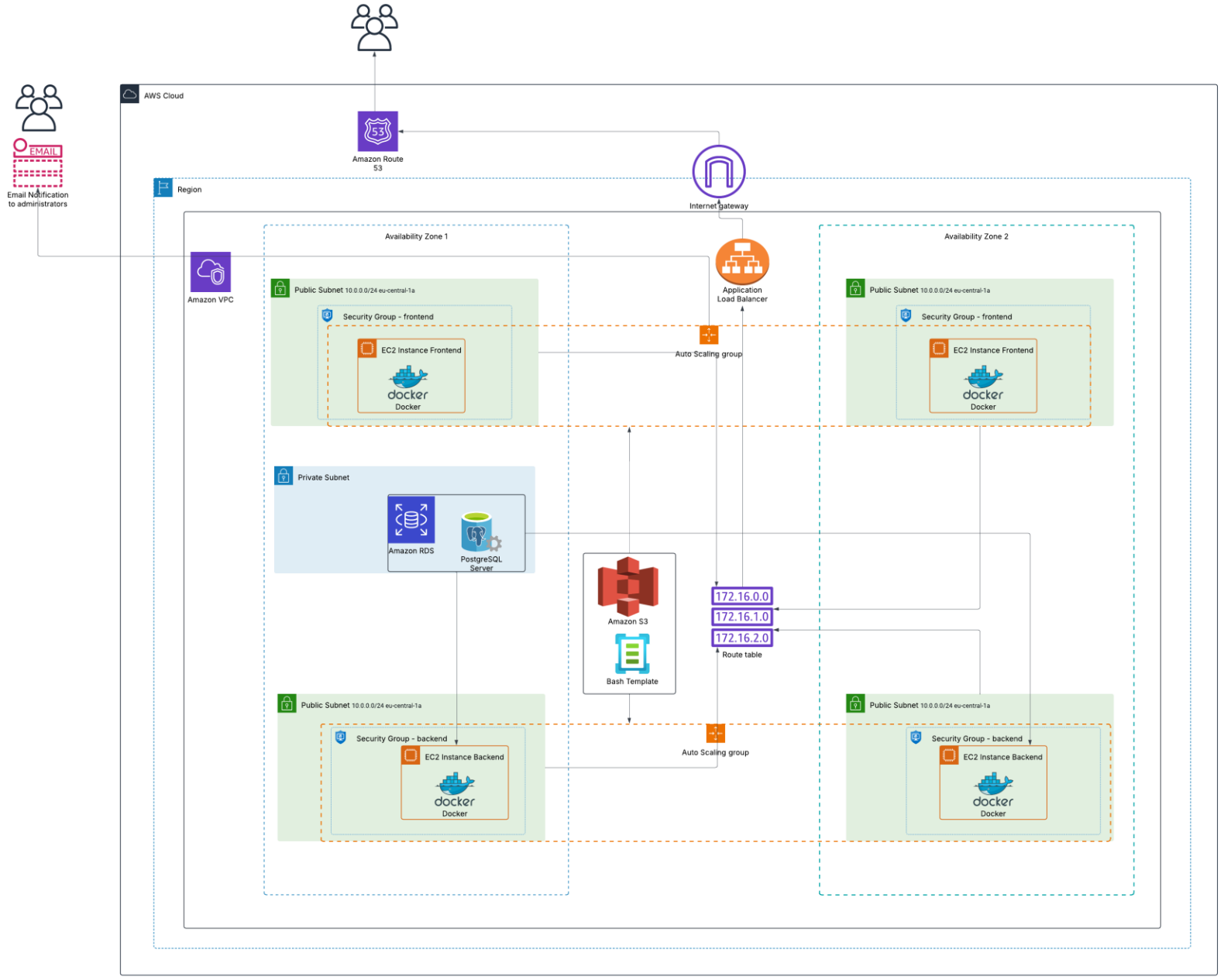
AWS-PROJECT: K´OOBEN CLOUD

THE GOAL OF THIS PROJECT IS TO DEPLOY A SCALABLE AND RELIABLE INFRASTRUCTURE ON AWS, ENSURING HIGH AVAILABILITY, SECURITY, AND EFFICIENT RESOURCE MANAGEMENT. THE SYSTEM IS DESIGNED TO HANDLE INCREASING DEMAND WHILE MAINTAINING PERFORMANCE AND STABILITY.

AWS ARCHITECTURE DIAGRAM:

Key Components:

- **EC2 Instances:** Separate instances for frontend (**Next.js**) and backend (**NestJS**), each running in a **Dockerized environment.**
- **Auto Scaling Group:** Ensures availability and scalability for backend instances.
- **Launch Templates:** Automate the provisioning of backend and frontend instances.
- **Application Load Balancer (ALB):**
  - **Frontend:** Routes traffic from **kooben.guitarrasargentinas.com** to frontend instances.
  - **Backend:** Frontend connects **directly via ALB ARN,** no subdomain required.
- **Amazon RDS (PostgreSQL):** The backend now connects to a **managed PostgreSQL database** instead of a local containerized instance.
- **Route 53:** Manages domain name routing for the frontend.
- **Security Groups & VPC:** Configured to allow only necessary traffic



APPLICATION PROVISIONING & DEPLOYMENT

Provisioning Steps:

1.Application Preparation:

1. Created a **Dockerfile** and a **Docker Compose** file to containerize the application.
2. Built and pushed the application image to **Docker Hub**.

2.AWS Setup:

1. Created a **Launch Template** to automate instance provisioning.
2. Launched separate **EC2 instances** for **frontend and backend** services.
3. Configured an **Auto Scaling Group** for backend scalability.
4. Implemented an **Application Load Balancer (ALB)** to distribute traffic.
5. **Frontend (kooben.guitarrasargentinas.com) connects to backend using the ALB ARN** (no subdomain).
6. Migrated **PostgreSQL database from EC2 to Amazon RDS** for scalability.

Domain & Networking:

1. Configured **Route 53** for frontend domain management.
2. Secured backend access using **VPC and Security Groups.**

Tools & Services Used:

- **AWS**: EC2, Auto Scaling Group, ALB, Route 53, RDS, Security Groups
- **Containerization**: Docker, Docker Compose
- **Database: Amazon RDS (PostgreSQL)**
- **Version Control & Hosting**: GitHub, Docker Hub
- **Networking**: Load Balancer **ARN** for backend API calls

Tools & Services Used:

POTENTIAL ENHANCEMENTS:

Migrate Database to RDS:
Move PostgreSQL from EC2 to AWS RDS to improve scalability, reliability, and data persistence.

CI/CD Integration:
Automate deployments using GitHub Actions for a more efficient development workflow.

Expand System Capabilities:
Introduce additional AWS services and performance optimizations to enhance functionality.

Improve Monitoring & Security:
AWS CloudWatch: Enable real-time monitoring, metrics, and logs for better observability.
AWS CloudTrail: Track API activity and security events for compliance and auditing.