Smart Salary

Team: Techno Spartan

Introduction

Smart Salary is a web application for an enterprise, where the employees can view their salary, department and reporting manager on a dashboard.
Managers can see employees reporting to them and their own information.
HR can update employee information and have access to HR department specific dashboard that provides insights into employee demographics based on gender, age and salary difference by age.

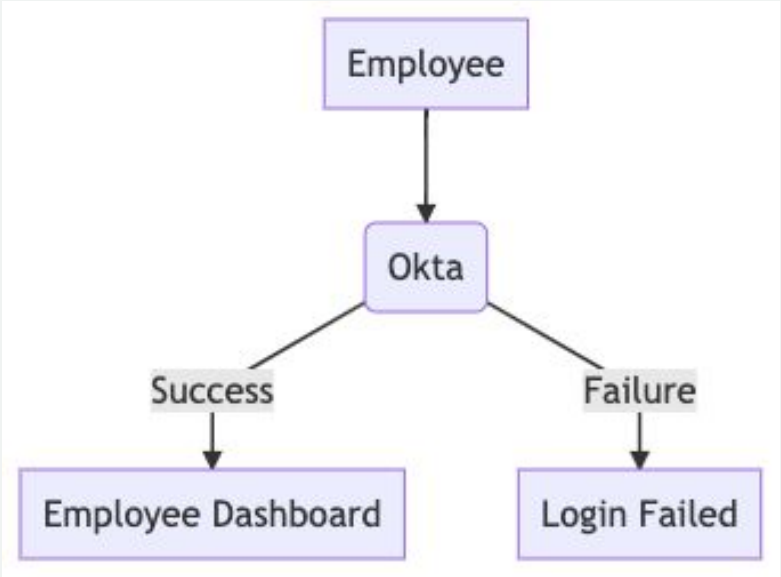
We have used the default project which has a large enterprise database. Our project majorly focuses on generating insights from the large employee data set using Databricks and Amazon Elastic Container Service which we have used for application deployment.

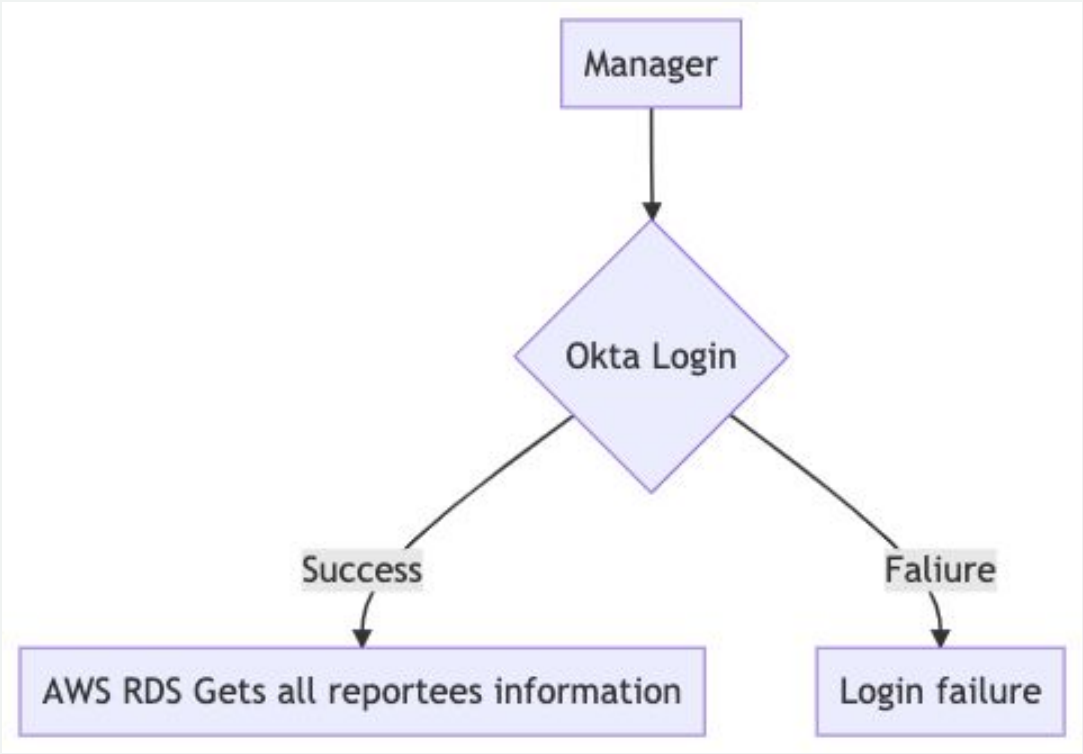
3 roles: Employee, Manager and HR.

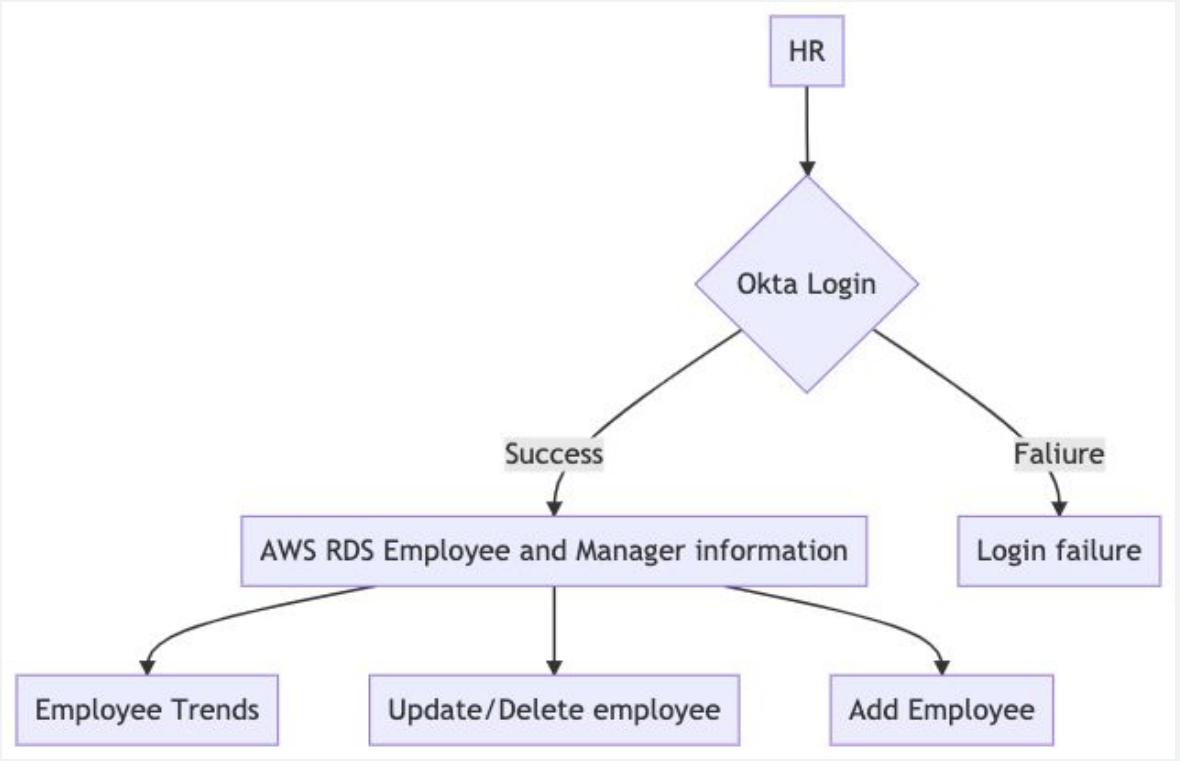
Our application has following features –

- Employee Dashboard – Employee information like name, salary, reporting manager.
- Manager - List of employees reporting to manager
- HR - List of Employees, perform actions like update/delete and can view visualizations generated from data analysis using databricks.

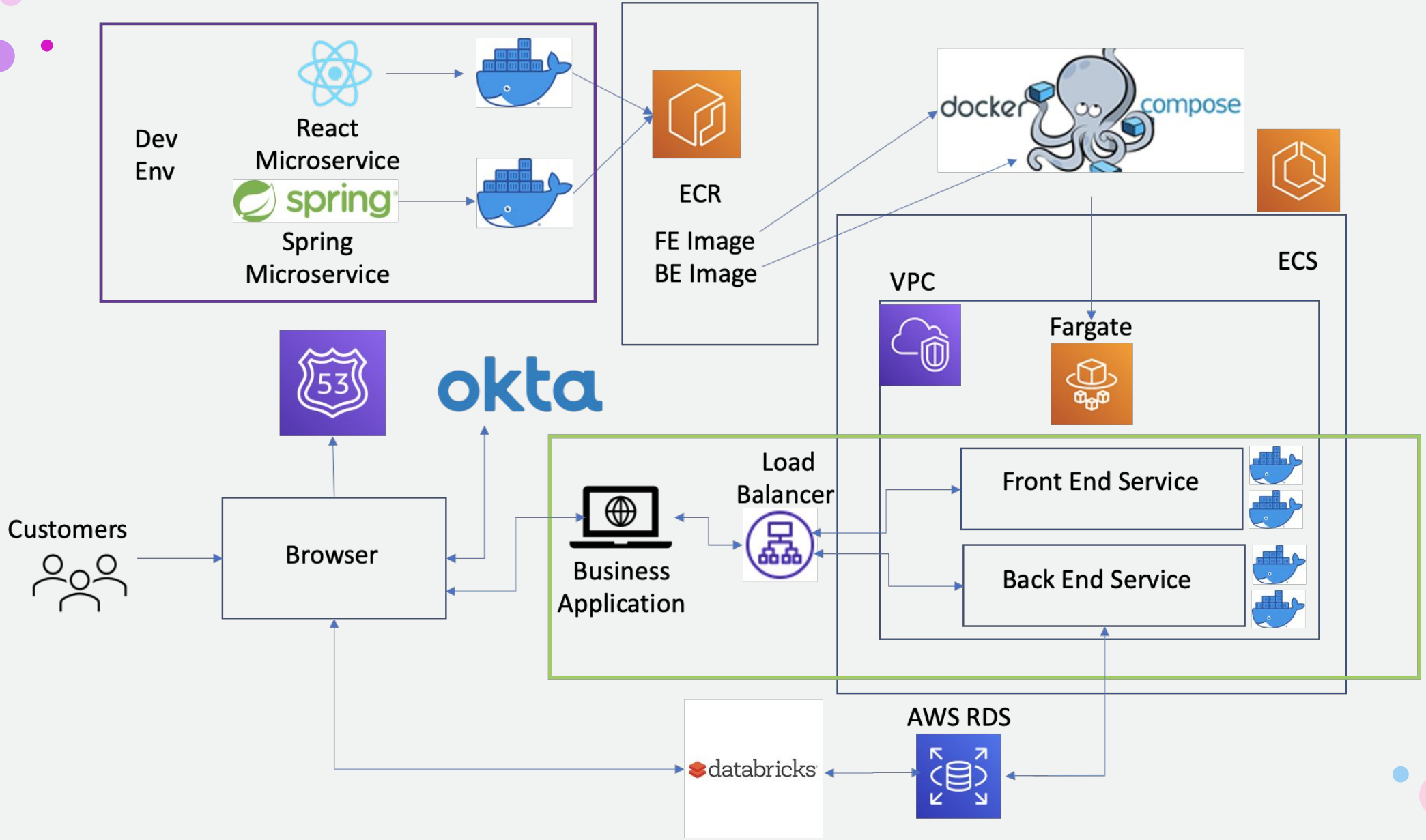
Use Cases


Employee
Okta
Success
Failure
Employee Dashboard
Login Failed


Manager
Okta Login
Success
Faliure
AWS RDS Gets all reportees information
Login failure


HR
Okta Login
Success
Faliure
AWS RDS Employee and Manager information
Login failure
Employee Trends
Update/Delete employee
Add Employee

Architecture Diagram

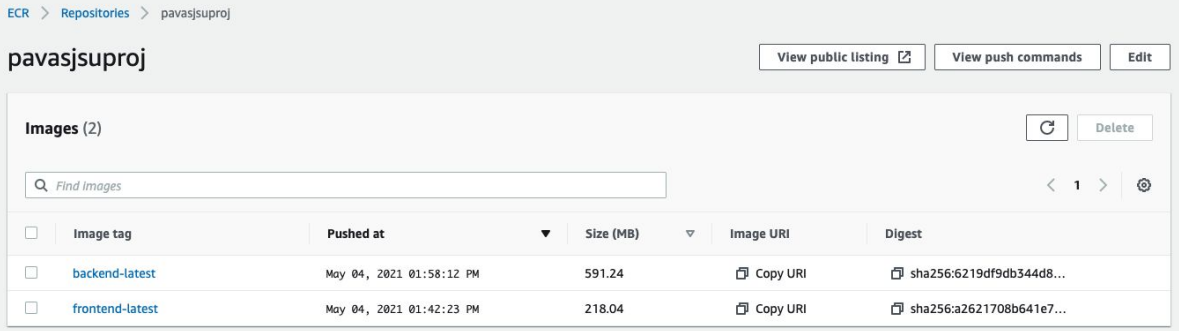


Tech Stack and Why this stack?

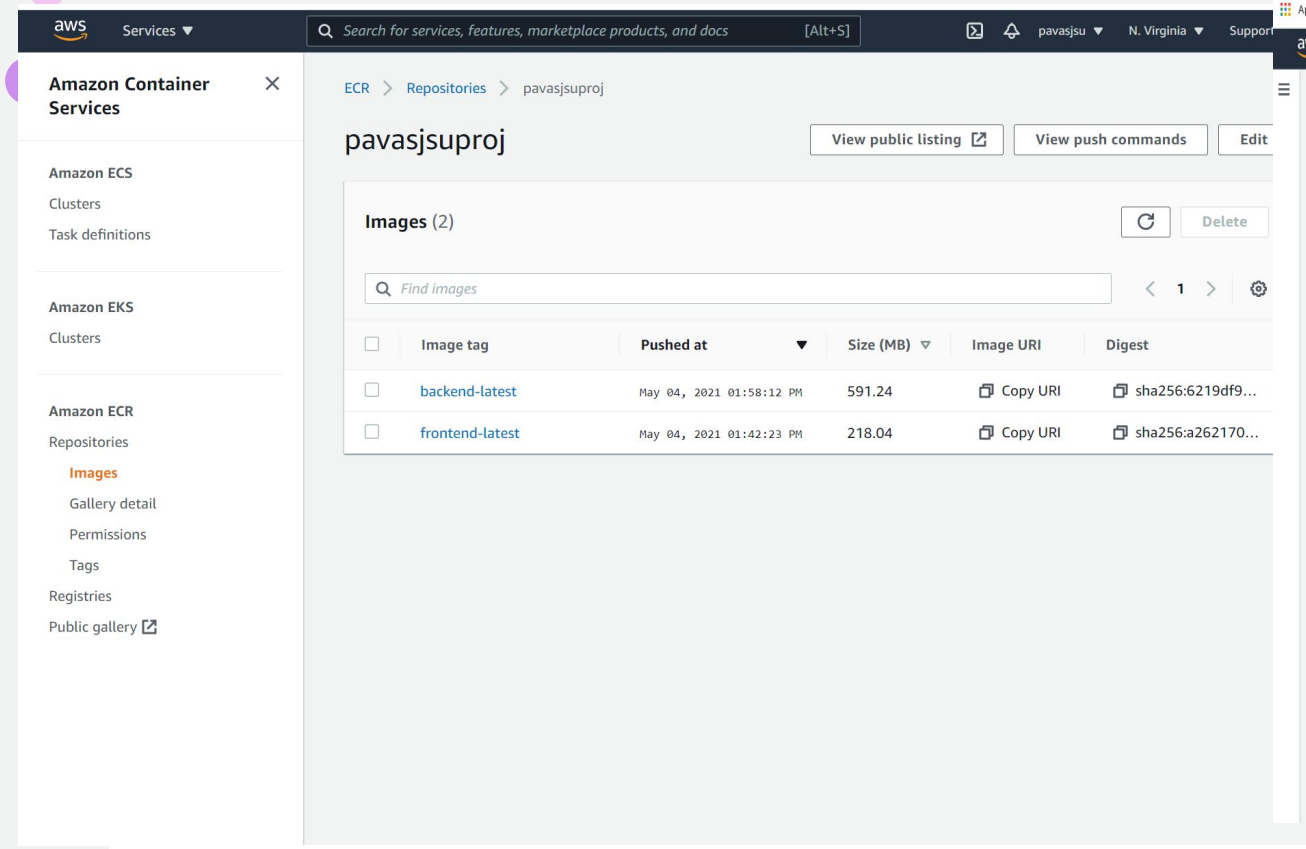
- Okta SSO - Single Page Application
- AWS RDS – MySQL
- Databricks – Python, Pandas, Matplotlib (Community edition it's free to use)
- Deployment
  - Docker
  - Docker Compose
  - AWS ECS (Better isolation between services)
  - AWS ECR (Made use of public repository which is almost free)

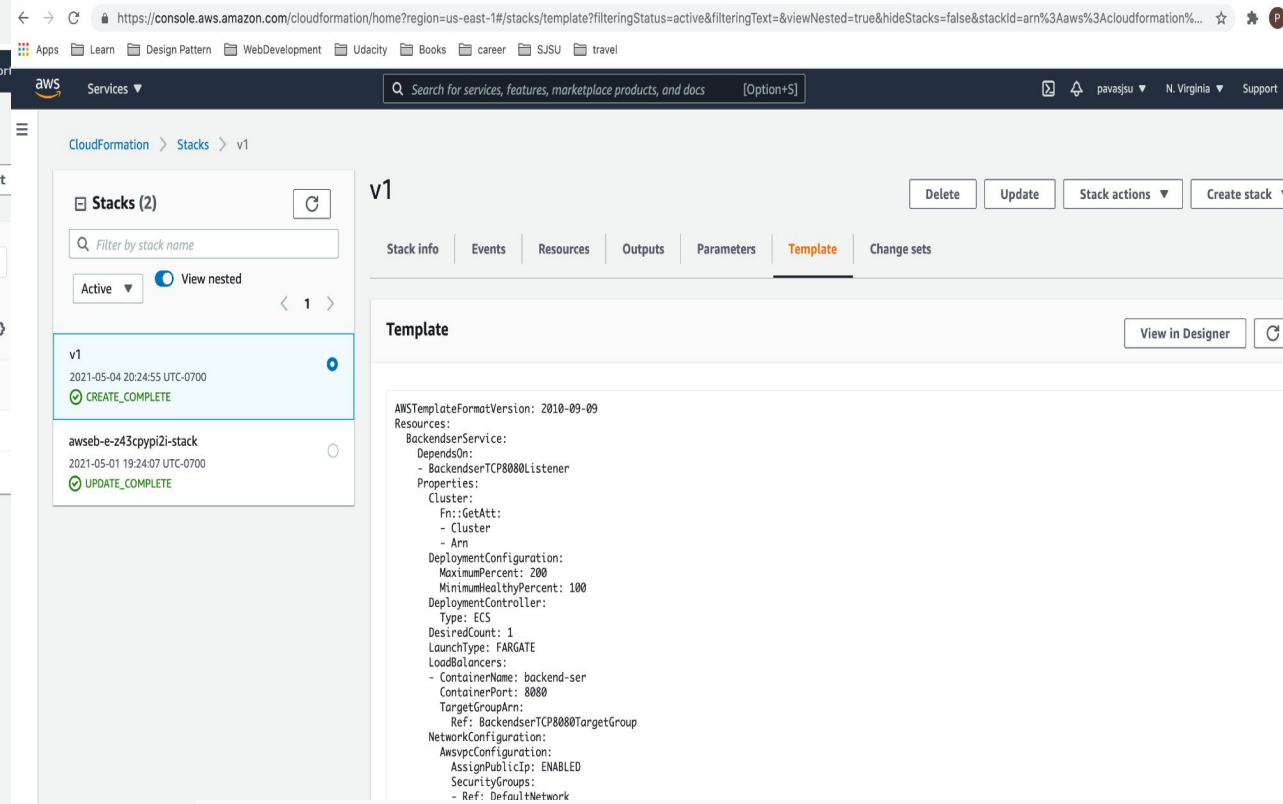
- We chose ECS fargate because the individual container services can be scaled independently.
- Last semester we made use of elastic beanstalk where we realized that the services' frontend and backend containers cannot be scaled independently.
- For students with limited network bandwidth, the ECR docker push is an expensive operation. Hope we can leverage the university network and bandwidth in the future for academic experiments.
- AWS compresses the images stored in ECR!!

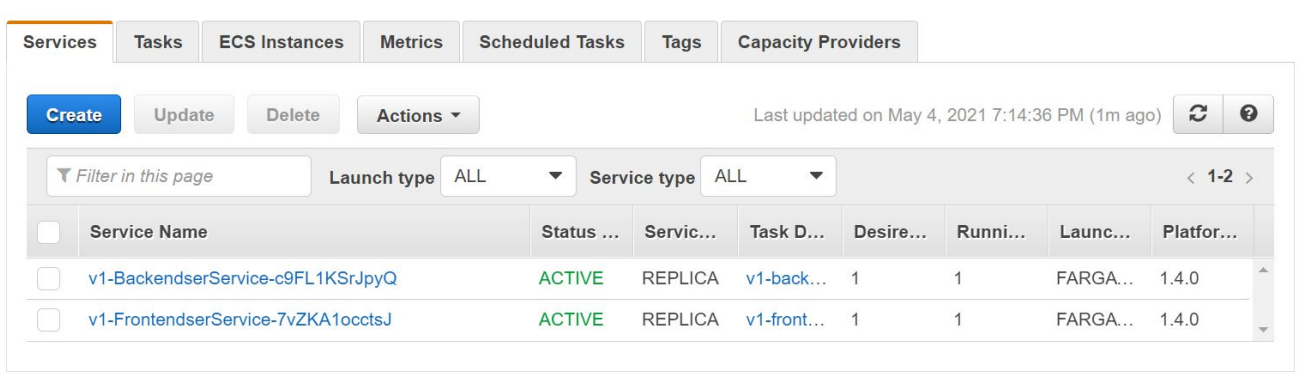
```
~ docker images
REPOSITORY                                  TAG              IMAGE ID       CREATED         SIZE
v1_frontend-ser                             latest           6faa70985b1c   24 hours ago    720MB
public.ecr.aws/x4y9x2u7/pavasjsuproj        frontend-latest  6faa70985b1c   24 hours ago    720MB
v1_backend-ser                              latest           324c3211144c   24 hours ago    912MB
public.ecr.aws/x4y9x2u7/pavasjsuproj        backend-latest   324c3211144c   24 hours ago    912MB
```

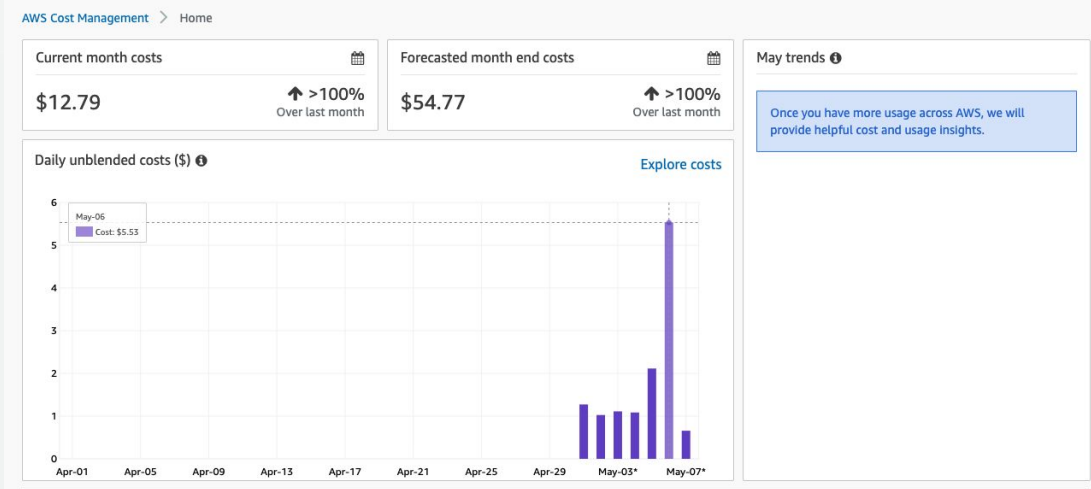


AWS configurations and cost









Thank You!!