10/1/2022

EAuction Project

AWS Cloud Depolyment Architecture

Thaw Dar Shwe, Ngu War (Cognizant)
NGUWAR.THAWDARSHWE@COGNIZANT.COM

Table of Contents

1. Introduction 2
   1.1. Purpose 2
   1.2. Scope 2
2. Architecture 3
   2.1. Elastic Container Service (ECS) Deployment 3
   2.2. Elastic Beanstalk Deployment 4
   2.3. Lamda Deployment 5
   2.4. Frontend Application Deployment 6
   2.5. Technologies used 6

1. Introduction

1.1. Purpose

This document provides a comprehensive deployment architecture of EAuction application, using a number of AWS services to depict different aspects of microservice world. It is intended to capture and convey the significant deployment plans made on the system.

1.2. Scope

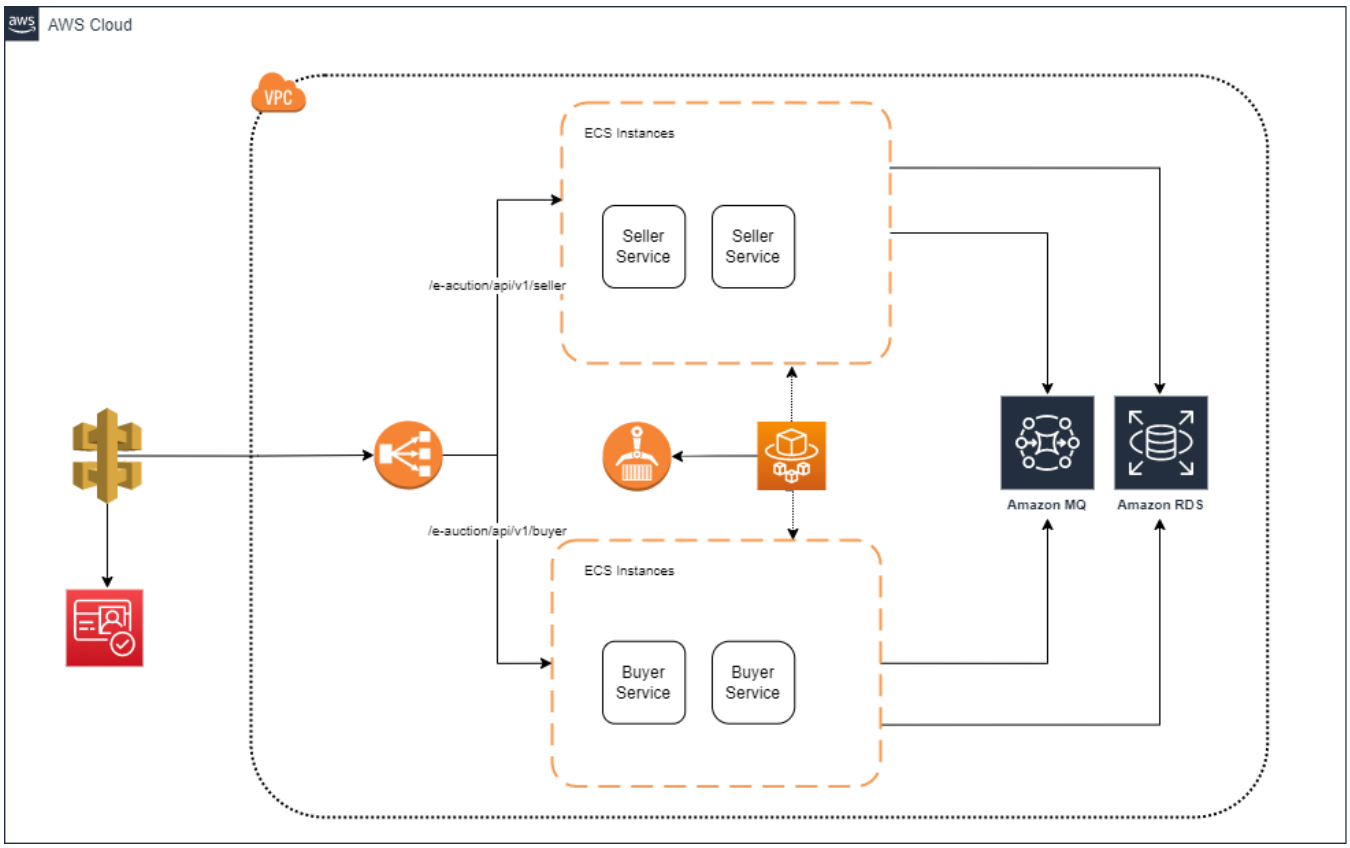
This document provides four deployment plans made on AWS cloud for EAuction project.

1. Elastic Container Service (ECS) Deployment for Seller and Buyer microservices using Docker Image and ELB.
2. Elastic Beanstalk Deployment for placing bid functionality of Buyer service
3. Lamda Deployment for product deletion functionality of Seller service
4. Frontend Angular Application Deployment using S3, EC2 and Route 53

In addition, RDS will be used for storing product and bid data and MQ will be used for event communication between microservices. All rest endpoints will be secured by API Gateway using AWS Cognito and act as a source of truth to access the microservices.

2. Architecture

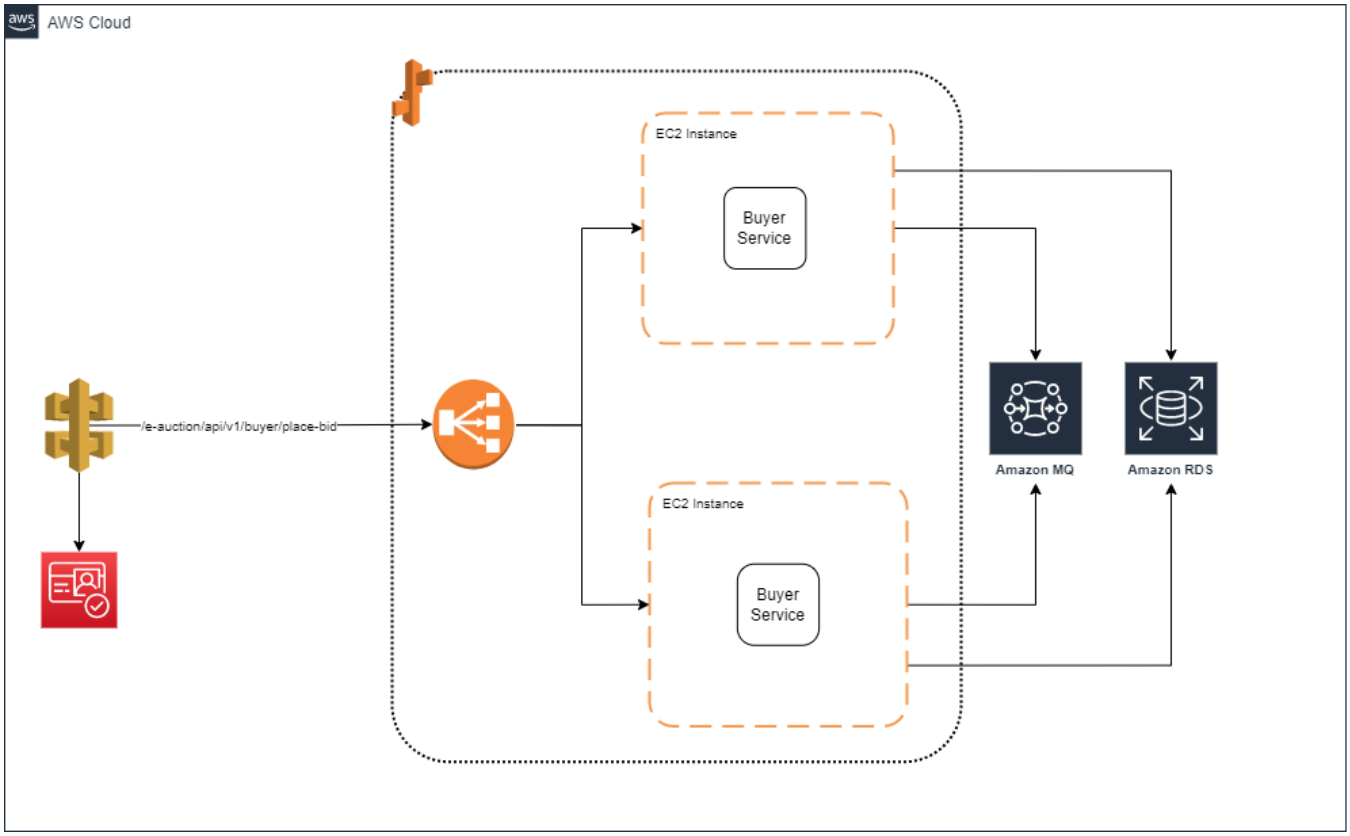
2.1. Elastic Container Service (ECS) Deployment



The following are the detailed description of each component in the diagram.

- Elastic Container Service
  - Seller Service and Buyer Service are written in spring boot and will be exposed as container instances in ECS.
  - Docker images will be uploaded to ECR and which will eventually be pulled by ECS Task Definitions.
  - ECS services will be created using AWS Fargate and it will deploy two container instances to two AZs with elastic load balancer for scaling the application load and improving the throughput.
  - The containers will connect to RDS for data retrieval and updates.
  - The containers will connect to MQ for event communication with other microservices.
- API Gateway
  - API gateway will call ELB as rest API endpoint and secure the end point with AWS Cognito.
- AWS Cognito
  - AWS Cognito will be used for authentication and authorization.
- Amazon RDS
  - PostgreSQL will be deployed on Amazon RDS.
- Amazon MQ
  - Amazon MQ will be used as message broker for event communication between microservices.

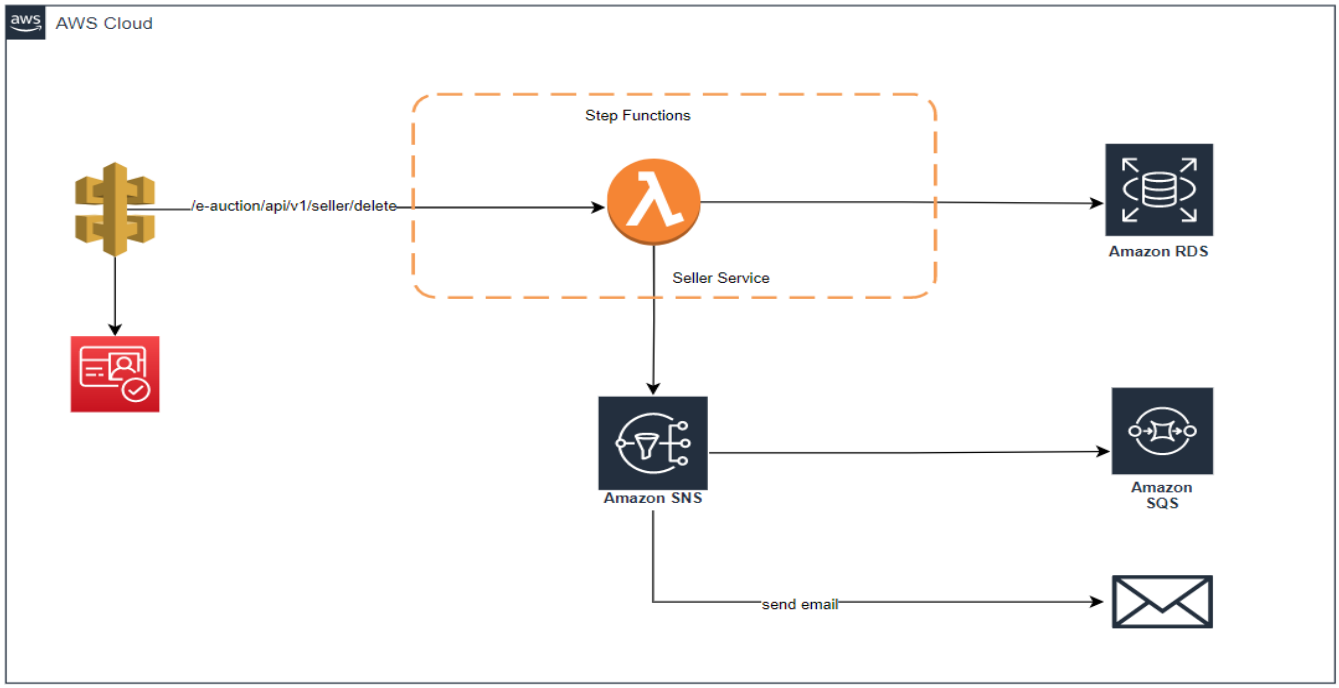
2.2. Elastic Beanstalk Deployment



The following are the detailed description of each component in the diagram.

- Elastic Beanstalk
  - Placing bid functionality is written in spring boot and will be exposed as deployed in AWS Elastic Beanstalk.
  - Two EC2 instances will be deployed to two Availability Zones(AZ) with Elastic Load Balancer for scaling the application load and improving the throughput.
  - The EC2 instances will connect to RDS for data retrieval and updates.
  - The EC2 instances will connect to MQ for event communication with other microservices.
- API Gateway
  - API gateway will call ELB as rest API endpoint and secure the end point with AWS Cognito.
- AWS Cognito
  - AWS Cognito will be used for authentication and authorization.
- Amazon RDS
  - PostgreSQL will be deployed on Amazon RDS.
- Amazon MQ
  - Amazon MQ will be used as message broker for event communication between microservices.

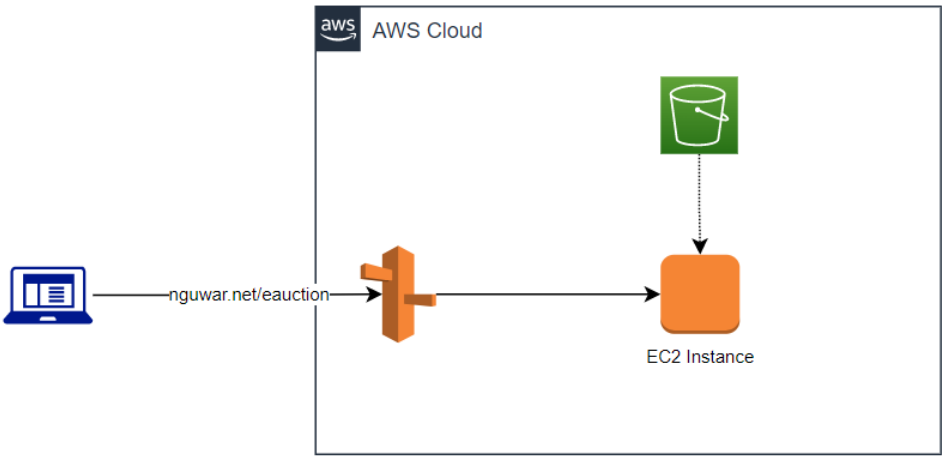
2.3. Lamda Deployment



The following are the detailed description of each component in the diagram.

- Lamda Function
  - Product deletion functionality is written in spring boot and will be exposed as Lamda Function.
  - Lamda Function will connect to RDS for data retrieval and updates.
  - Lamda Function will have the ability to run concurrently up to 100 instances.
- API Gateway
  - API gateway will call Lamda Function as rest API endpoint and secure the end point with AWS Cognito.
- AWS Cognito
  - AWS Cognito will be used for authentication and authorization.
- Amazon RDS
  - PostgreSQL will be deployed on Amazon RDS.
- AWS Step Functions
  - State Machine will be implemented to handle the exception thrown by Lamda Function.
- Amazon SNS
  - SNS topics will be sent after Lamda is invoked or unexpected error occurred at Lamda Function. Finally, it will send email notification to the administrator.
- Amazon SQS
  - Amazon SQS will receive the message by SNS when Lamda is invoked.

2.4. Frontend Application Deployment



The following are the detailed description of each component in the diagram.

- Amazon S3
  - Angular application bundle will be uploaded to S3 bucket.
- EC2 Instance
  - Web server will be installed on EC2 Instance and then frontend application will be downloaded from S3 bucket and deployed at the web server.
- Route 53
  - Rout 53 will be used for Domain Name Registration and users will eventually be able to access the frontend application with respective domain name.

2.5. Technologies used

The following are the AWS service used to deploy the microservices and frontend application.

- Amazon ECS
- AWS Elastic Beanstalk
- AWS Lamda
- Amazon MQ
- Amazon RDS
- Amazon S3
- Amazon EC2
- Amazon Route 53
- AWS Elastic Load Balancing
- Amazon API Gateway
- AWS Step Functions
- Amazon SNS
- Amazon SQS
- Aws Cognito