**AWS(Amazon Web Services) Notes/Key Points:**

On-Premise Deployment:

It is a physical location and is called a data center.

Load Balancer- Which handles primary and secondary servers. - Distributed Environment

Challenges:

1. Capacity, configuration management is expensive
2. Maintenance required
3. Security - Non Negotiate(100 % guarantee)
4. Resource/Employee management (24/7)
5. Cost (on-premise location rent, electricity,time)
6. DR(Disaster Recovery) Management – Dead end

Above challenges Managed by a **public cloud provider**.

Computers on the cloud are called Virtual Machines(Computers).

**Amazon AWS**

**Microsoft Azure**

**Google compute cloud platform(GCP)**

Alibaba - China

IBM

Oracle

URL : <https://aws.amazon.com/free>

**Full Stack Developer or Software Engineer:**

**12 Months Free:**

**Amazon EC2** (Elastic Compute Cloud)

**Amazon S3**

API gateway - It will help to create an endpoint URL

Amazon Elasticache

**Always Free:**

**Amazon DynamoDB**

**Amazon Lambda**

Amazon SNS (Simple Notification Service)

Amazon Cloudwatch

Amazon SQS (Simple Queue Service)

AWS System Manager

AWS Global Infrastructure:

<https://aws.amazon.com/about-aws/global-infrastructure/regions_az/>

Region - Physical Location

Example : Mumbai

Each Region consists of 2 or more data centers(Availability Zones or AZs).

HA- High Availability of your application

Region –

Default Region - us-east-1(N.Virginia)

Please refer the AWS official document from the below:

Graphical user interface, text, application

Description automatically generated

AWS Official Documentation:

<https://docs.aws.amazon.com/>

Example:

EC2 - <https://docs.aws.amazon.com/ec2/index.html>

EC2 (Elastic Cloud Compute) - It is the Virtual Server( Create a server on the AWS Cloud)

Usage based(Pay Per Use)

3 hours / day - charge is only for 3 hours

EBS (Elastic Block Store) - It is Virtual Hard disk

Root Value

It is virtual hard disk – 10 GB (Dynamic upgrade)

GP2(General Purpose) SSD - 8 GB data can be stored

Physical harddisk - C:, D: Partinitioning

ssh -i "mykey.pem" [ec2-user@ec2-13-127-37-174.ap-south-1.compute.amazonaws.com](mailto:ec2-user@ec2-13-127-37-174.ap-south-1.compute.amazonaws.com)

IAM - Identity Access Management

It is global level and not specific to any region

*IAM role - this helps developers to communicate between AWS services using IAM ARN(URL).*

There are 3 components:

Users - End User

Groups

Roles

AWS CLI - Command Line Interface

<https://aws.amazon.com/cli/>

Installation(Mac, Windows and Linux) : <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

AWS SDK library:

1. Install AWS-SDK library - npm install aws-sdk or yarn add aws-sdk
2. Example AWS- S3 Programmetically use <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-example-creating-buckets.html>

Root Admin - By Default, All access (Create user, group and roles)

EngineerGrp:

IAM User 1 - EC2 (IAM policy Document: level of permission - read/write or both)

IAM User 2 - DynamoDB

MarketingGrp :

IAM User 3 : S3

IAM User 4 : Lambda

3 ways to access AWS services:

1. Manual - through AWS console -> Only Admin can login into console.

Users, groups and role

1. Programmatically access -> Developer
2. through AWS CLI

> aws s3 create-bucket <bucket-name>

> aws ec2 read-server <ec2-instance> url

1. through AWS SDK library

*aws-sdk* library

npm install aws-sdk

mainly focus to create an IAM role for assigning the AWS services from one another.

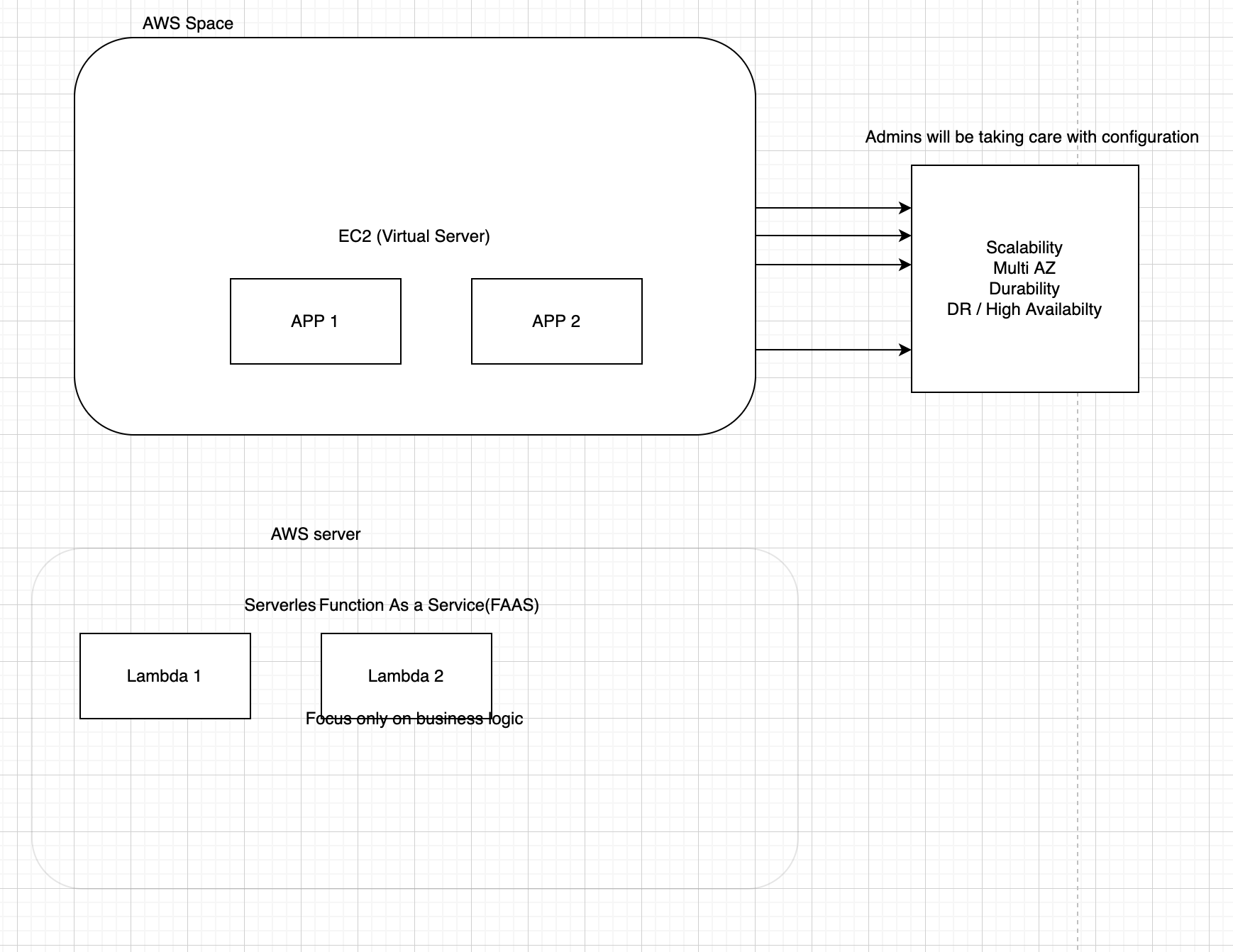
IAM Policy Document:

It is a JSON format script. This helps to provide granular permission to the AWS services which you want to use by IAM role.

Example: Lambda permission - Read, Write or Read and Write

For Execute your code - we need Server

Lambda Diagram:



1 API = 1 microservice = 1 Lambda

2nd API = 2nd microservice = 2nd Lambda

UI Code - S3

Backend Code - Lambda

Lambda triggering

exports.handler {

//business logic

}

Cloudwatch:

Application Monitoring tool - It is used to troubleshoot the application whenever facing an issue.

Today = Netlify (UI Code) + Render (Application Code) + MongoDB (DB)

Cloud AWS = S3(UI Code) + Lambda (Application Code) + DynamoDB (DB)

DynamoDB = MongoDB (Hands-on Lab)

No-SQL DB ⇒ It can be document based storage

S3 - Object based storage

OS AWS(S3)

—- —-----

Folder - Bucket

File - Key

Error Code:

—-------------

200 - Successfully Created

401 - Un Authorized (4XX)

403 - Forbidden (4XX)

500 - Internal Server error (5XX)

404 - Not Found (4XX)

S3 NodeJS Code Example:

<https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/getting-started-nodejs.html>

Secure website:

2 ways to protect the data

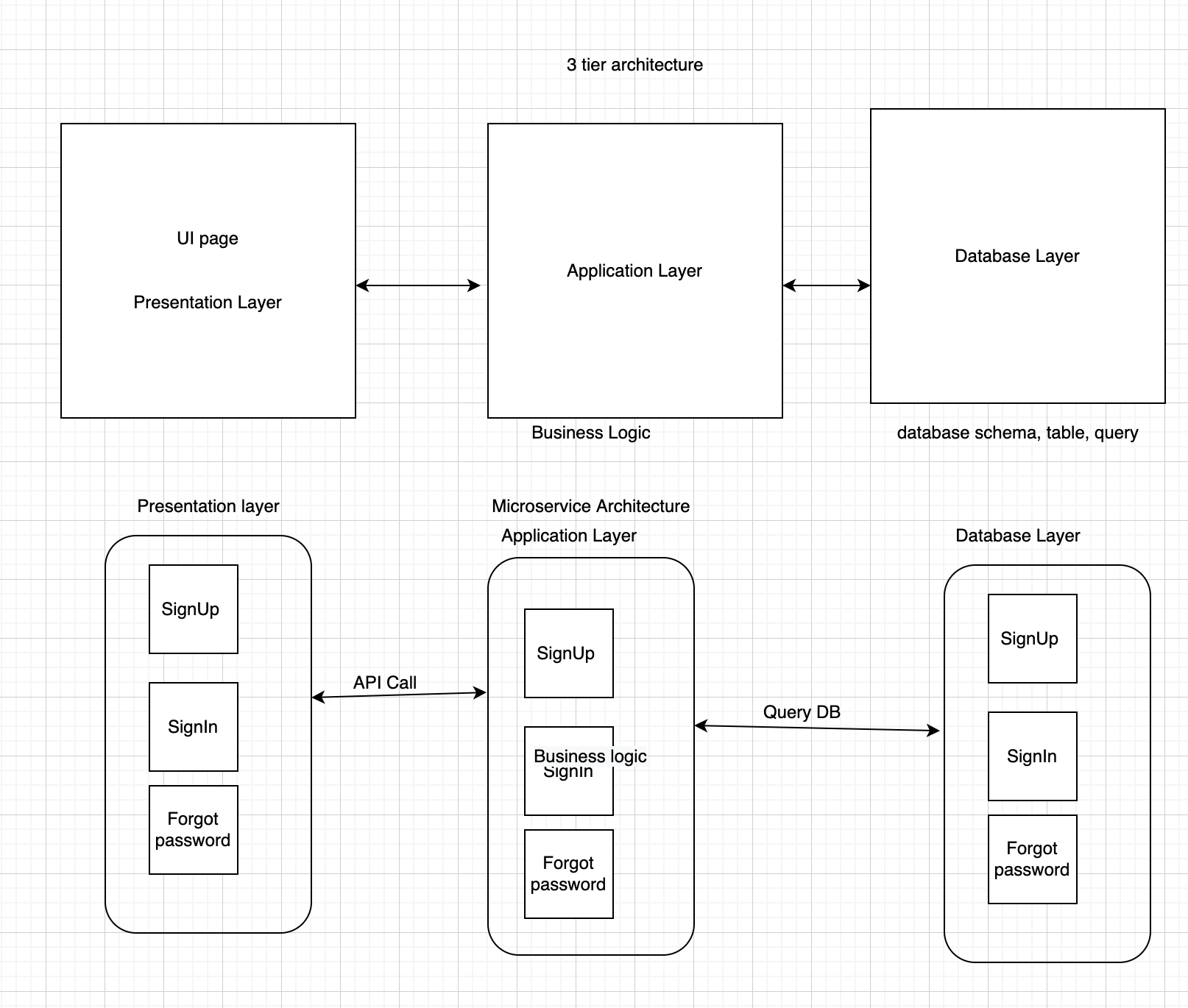
1. During the transit the data (endpoint url) – https, WAF(Web Application Firewall)
2. At rest the data (Database -mongoDB or DynamoDB) – Hashing algorithm(256), Encrypt and Decrypt(ES - DES)

Security Attack:

DDOS

SQL Injection

**Microservice Architecture:**



1 Microservice - 1 API - 1 Lambda

2 Microservice - 2 API - 2 Lambda

SDLC:(Software Development Life Cycle)

* Waterfall Model (Top-Down Approach)
* Requirements Gathering Phase - Product Manager
* Analysis and Design Phase - Product Owner / Solution Architect
* **Implementation and Dev Phase - Developer(Full stack)**
* Testing Phase - QA(Quality Analyst)
* Release Phase - Production Deployment
* EBSO Production Support
* Agile Methodology (Current Approach)
* Requirement/UserStory
* Break into Multiple task
* Task convert into design and dev
* Once Dev complete testing phase
* release phase and then production support

Repo Source Code(ShoppingCart):

Src

* UI - S3
  + UserForm (Html, JS, CSS, Images)
    - signUp.js
    - userRegistration.js
      * Name
      * MobileNo
      * Age
      * Gender
      * Location
      * Address

- Model (Application Layer) Lambda

- UserModel

- signUpModel.js

if(mobileNo startswith +91){

call Db layer country\_India

}else if (mobileNo startswith +65) {

call Db layer country\_singapore

}

- userRegModel.js

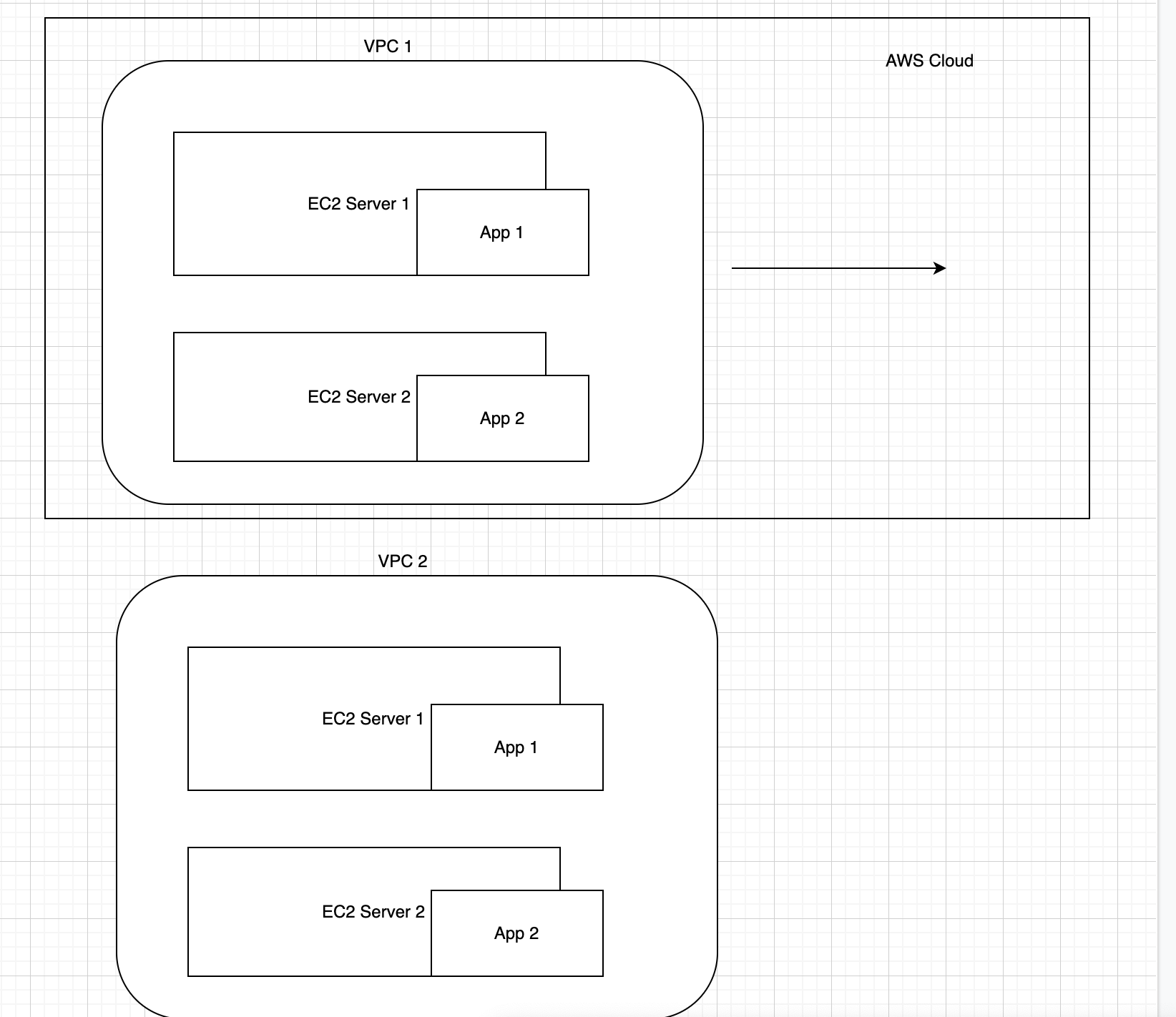
- DB (DB layer)

- User

- writeUser.js

- retrieveUser.js

VPC - Virtual Private Cloud



AWS Doc Reference:

[What is Amazon VPC? - Amazon Virtual Private Cloud](https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html)

WaterFall Model:

==============

Software Development Life Cycle:

—----------------------------------------

Requirement Analysis - SME or Product Managers

Design (HLD and LLD) - Architects

Implementation and Coding – Full stack or MERN

Development:

Desktop at Home == > started developing app → create server -> run the app

Too many request can not handle

Setup Infra (it can be building room)

* Multiple servers to handle more request and traffics
* Primary, secondary and so on
* Distributed with Load balancers
* Scaling
* Dedicated room for maintaining the servers (Datacentre)

Testing - QA

Delivery - Manager will handle

UI

Application Logic

Backend Code

Microservice - API

Client → API -> Authentication → Lambda(Backend Service)

**DynamoDB :**

​​Clone

URL :git clone <https://github.com/referbruv/dynamodb-nodejs-sample.git>

aws configure

Run the nodejs server

node server.js

Error:

UnrecognizedClientException: The security token included in the request is invalid

at Request.extractError (/Users/muthurm/ADSK-WS/Learning/dynamodb-nodejs-sample/node\_modules/aws-sdk/lib/protocol/json.js:51:27)

at Request.callListeners (/Users/muthurm/ADSK-WS/Learning/dynamodb-nodejs-sample/node\_modules/aws-sdk/lib/sequential\_executor.js:106:20)

at Request.emit (/Users/muthurm/ADSK-WS/Learning/dynamodb-nodejs-sample/node\_modules/aws-sdk/lib/sequential\_executor.js:78:10)

Solution:

vi ~/.aws/credentials

remove access\_token entry

IAM - User, Group and Roles

Company A:

AWS Account – managed by **Super Admin → granular permission(Read, Write or Both)**

**CRUD - Create, Read, Update and Delete**

Team 1: (Project : HR related) → AWS services : S3, EC2

Dev1

Dev2

Team 2: (Project : Sales & Marketing related) → AWS services : Lambda, Dynamodb

Dev 3

Dev4

Message Services:

===============

1. SQS - Simple Queue Service

— lambda 1(producer) – queue –- lambda 2(consumer) → queue →lambda 3

it ll be in json payload

– pull based model

1. SNS - Simple Notification Service

→ used for sending notification (OTP) to customers

– push based model

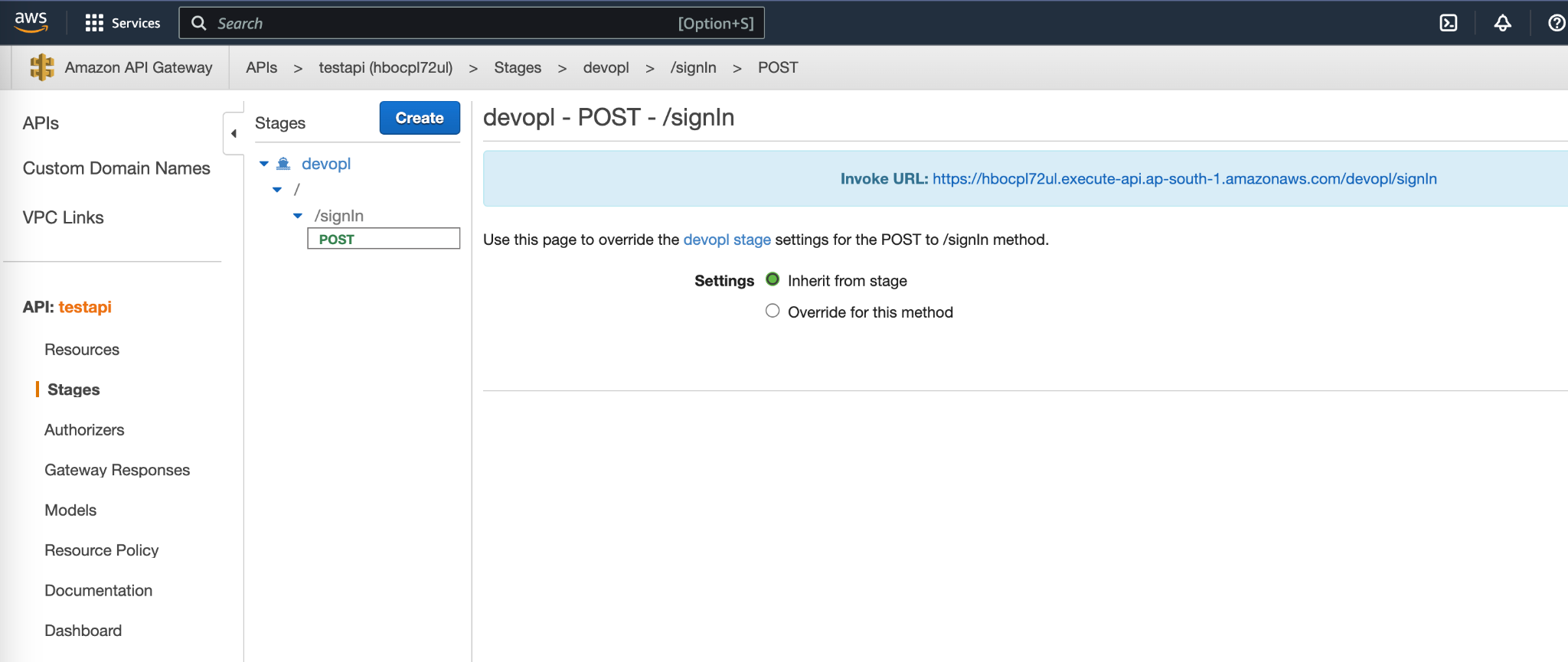
**End-End Test:**

1.Git clone :<https://github.com/muthuramanathanm/service_layer>

2. Create Lambda function

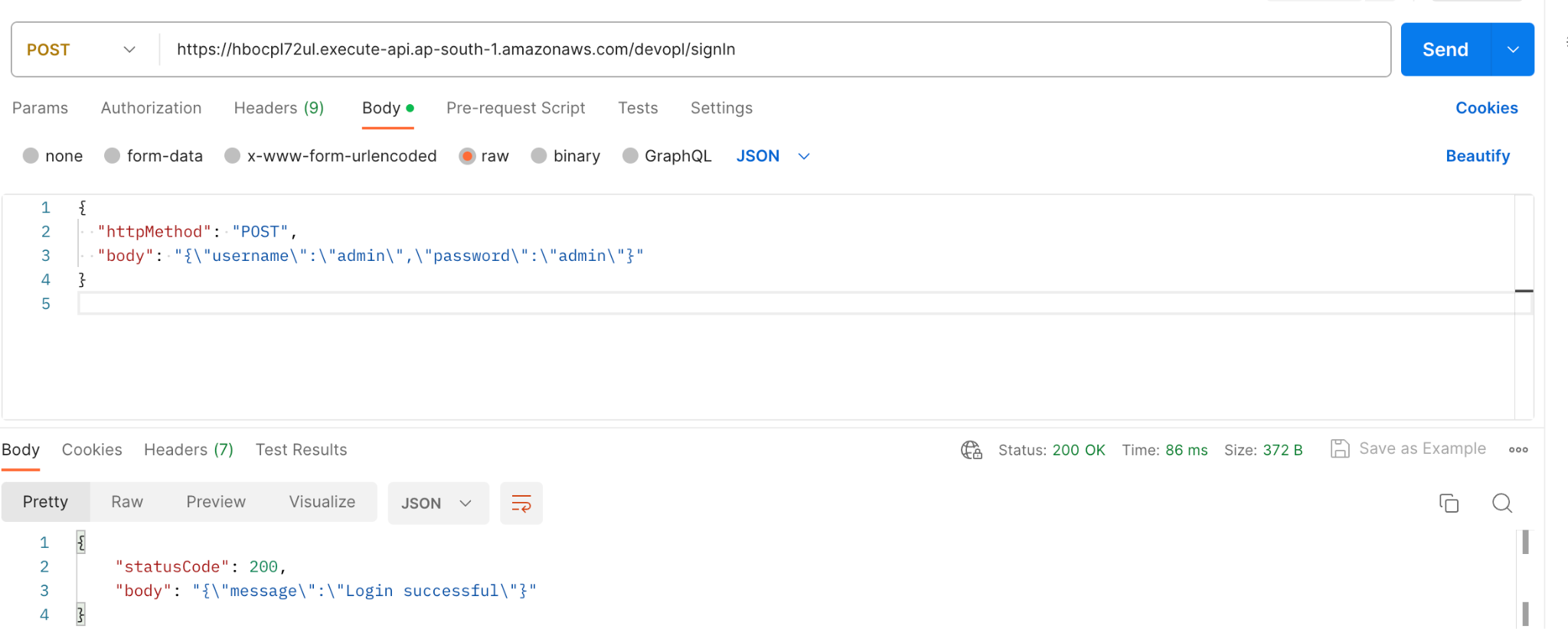
3. Upload zip folder and test the backend changes(lambda)

4. create API gateway -> POST -> resource path=signIn



5. Go to Postman and test API

POST <https://hbocpl72ul.execute-api.ap-south-1.amazonaws.com/devopl/signIn>



6. Git clone : <https://github.com/muthuramanathanm/my-app.git>

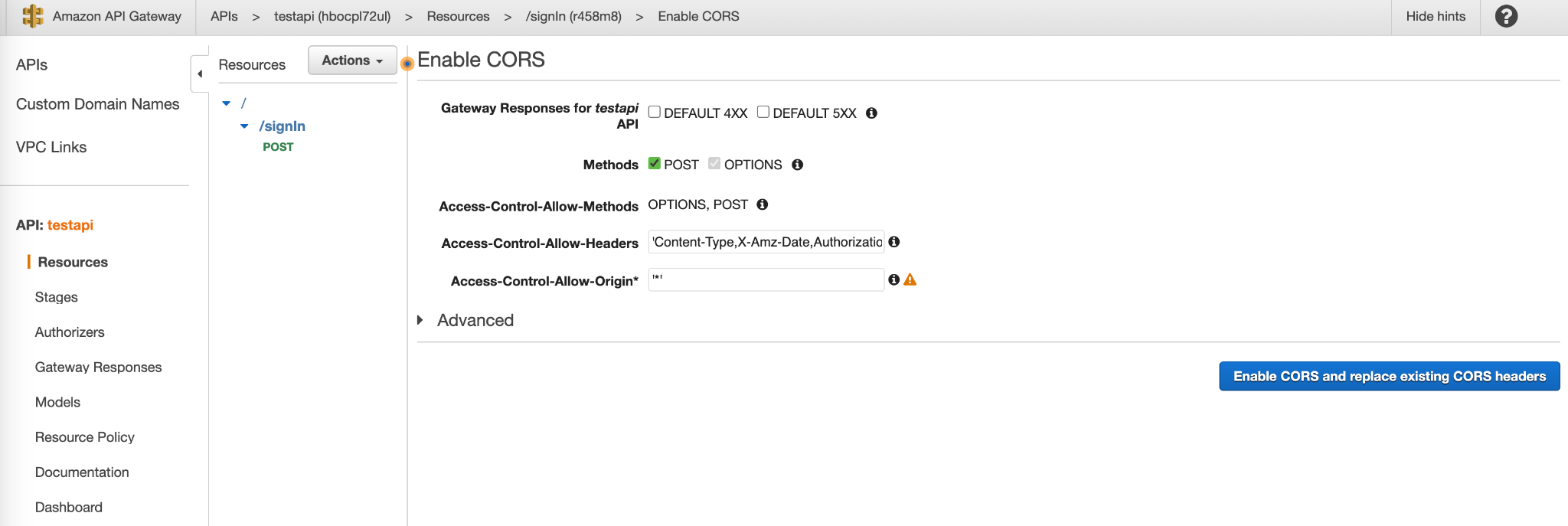
7. npm install and Update the API url in route.js

8. npm start

**Troubleshoot:**

If we get this CORS error while running the UI:  
Access to fetch at 'https://hbocpl72ul.execute-api.ap-south-1.amazonaws.com/devopl/signIn' from origin 'http://localhost:3000' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

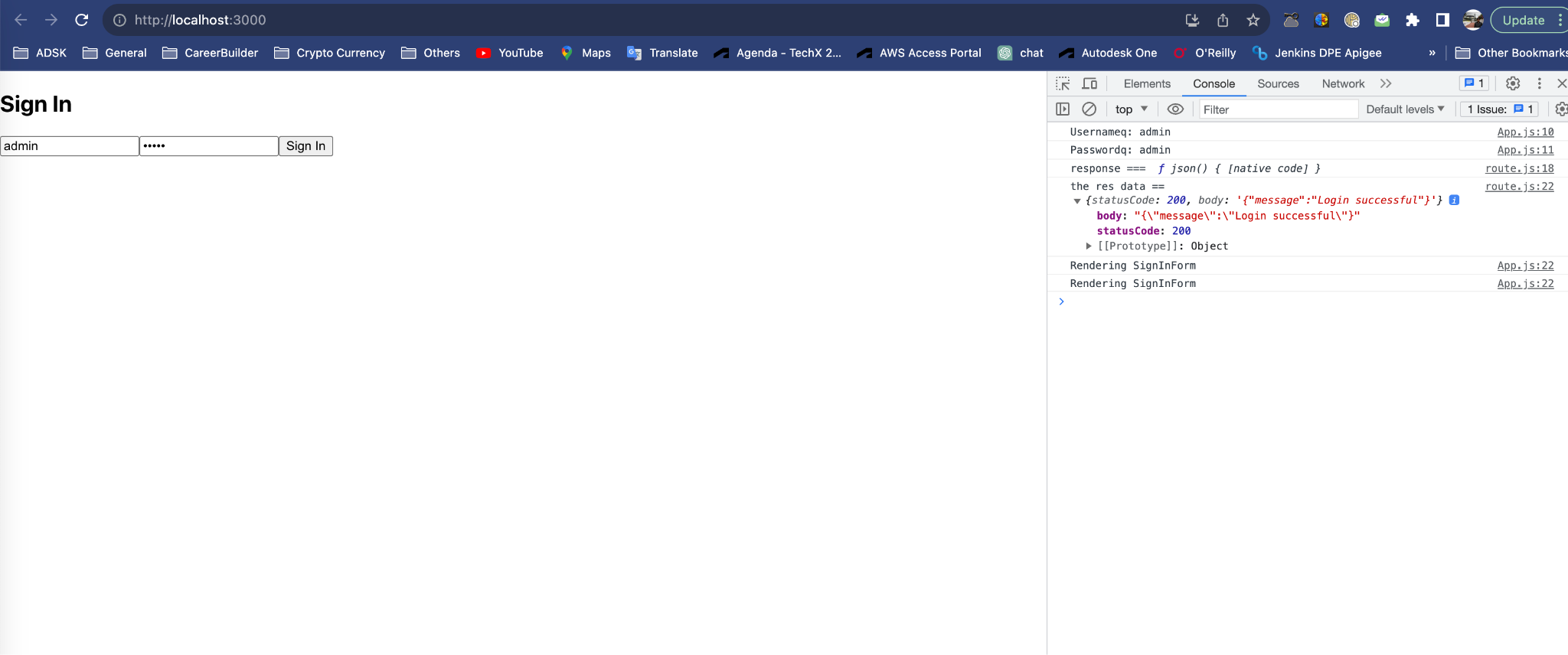
Solution:

Actions -> enable CORS

one click on the enable CORS button as above and then go to actions -> deploy API.

**Test UI:**

Username and password match:



Username and password mismatch:

