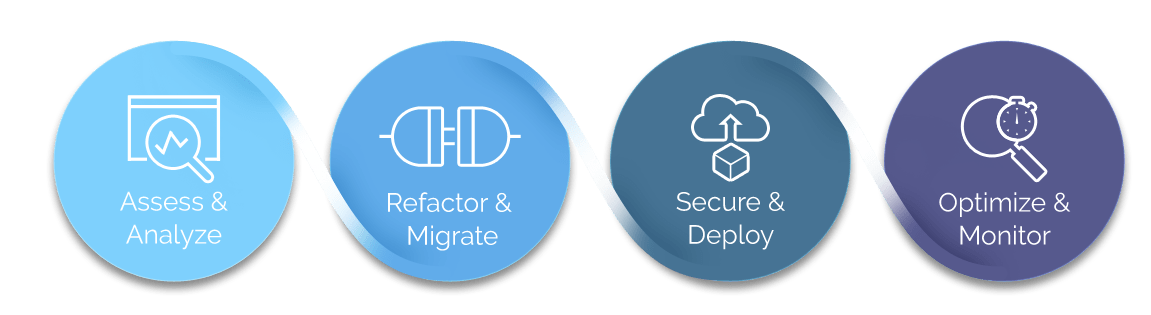
Name: Tan Jun Heng Admin No. P2004217 Class: DIT/FT/2A/01

**Web Application Deployment Assessment Report**

**Enterprise Systems Development**

Assignment CA1

26 July 2021



Web Application Deployment Report

By Tan Jun Heng

GitHub Repo URL : <https://github.com/kisekixcel/ESDE-CA2-p2004217-TanJunHeng>

Table of Contents

[1. Introduction 5](#_Toc79177039)

[1.1 Overview 5](#_Toc79177040)

[1.2 Scope 5](#_Toc79177041)

[1.3 Project Team 5](#_Toc79177042)

[2. Prerequisites 6](#_Toc79177043)

[3. EC2 Instances Deployment 7](#_Toc79177044)

[3.1 Creating Amazon EC2 Instance and setting security group 7](#_Toc79177045)

[3.2 Setting up EC2 Key Pair 10](#_Toc79177046)

[3.3 Connecting to EC2 Instance using WinSCP 12](#_Toc79177047)

[3.3.1 Generating EC2 Private Key 13](#_Toc79177048)

[3.3.2 Connecting to Amazon EC2 Instance 15](#_Toc79177049)

[3.4 Allocating Elastic IP addresses to Amazon EC2 16](#_Toc79177050)

[4. Setting up FrontEnd and BackEnd Server in Amazon EC2 17](#_Toc79177051)

[4.1 Uploading FrontEnd & BackEnd project for Hosting 17](#_Toc79177052)

[4.2 Connecting to EC2 Instance using PuTTY 19](#_Toc79177053)

[4.3 Hosting FrontEnd & BackEnd Web Application 20](#_Toc79177054)

[4.3.1 Install nodejs on your Amazon EC2 20](#_Toc79177055)

[4.3.2 Running FrontEnd Web Application on Amazon EC2 20](#_Toc79177056)

[4.3.3 Running BackEnd Web Application on Amazon EC2 21](#_Toc79177057)

[5. AWS Cloud9 22](#_Toc79177058)

[5.1 Creating AWS Cloud9 Environment 22](#_Toc79177059)

[6. AWS RDS for MySQL 23](#_Toc79177060)

[6.1 Creating AWS RDS 23](#_Toc79177061)

[6.2 Enabling Port 3306 for AWS RDS Security Group to be accessed anywhere 25](#_Toc79177062)

[6.3 Connecting to RDS Database from MySQL 26](#_Toc79177063)

[6.4 Creating Database Schema 28](#_Toc79177064)

[6.5 Inserting records into AWS RDS tables 29](#_Toc79177065)

[7. AWS API Gateway & AWS Lambda 31](#_Toc79177066)

[7.1 Creating API Gateway 31](#_Toc79177067)

[7.2 Creating Lambda Function for update\_design.html 32](#_Toc79177068)

[7.3 Creating API Method for esde\_beedesign\_getDesignByid function 35](#_Toc79177069)

[7.4 Verify that the API route is being used 42](#_Toc79177070)

[8. Creating DynamoDB database 43](#_Toc79177071)

[8.1 Exporting user table data from MySQL 43](#_Toc79177072)

[8.2 Exporting file table data from MySQL 44](#_Toc79177073)

[8.3 Exporting role table data from MySQL 45](#_Toc79177074)

[8.4 Uploading MySQL database to DynamoDB using Cloud9 47](#_Toc79177075)

[8.4.1 Uploading user Table 47](#_Toc79177076)

[8.4.2 Uploading files Table 50](#_Toc79177077)

[8.4.3 Uploading role Table 53](#_Toc79177078)

[9. AWS CloudWatch 56](#_Toc79177079)

[9.1 Centralized logging for Web API Access 56](#_Toc79177080)

[9.1.1 Create an IAM role for logging to CloudWatch 56](#_Toc79177081)

[9.1.2 Adding the IAM role in the API Gateway console 58](#_Toc79177082)

[9.1.3 Turn on logging for API and stage 59](#_Toc79177083)

[9.1.4 Test the logging setup 60](#_Toc79177084)

[9.2 Viewing Lambda Function CloudWatch logs 61](#_Toc79177085)

[10. MySQL RDS Database security 62](#_Toc79177086)

[10.1 Ensure only EC2 Instance running the web application can have access to RDS 62](#_Toc79177087)

[10.1.1 Verify that only the EC2 Instance can connect to the database. 63](#_Toc79177088)

[10.2 Encrypting RDS Database Snapshot using KMS keys 64](#_Toc79177089)

[10.3 Connecting to Encrypted RDS Database. 67](#_Toc79177090)

[10.4 Publishing MySQL logs to CloudWatch - Database logging 68](#_Toc79177091)

[10.4.1 Change value of log\_output parameter 69](#_Toc79177092)

[10.4.2 Verifying logs has been logged 70](#_Toc79177093)

[10.5 Lease Privilege for Database credentials 71](#_Toc79177094)

[11. CloudTrail 73](#_Toc79177095)

[11.1 Creating CloudTrail 73](#_Toc79177096)

[11.2 Viewing AWS CloudTrail logs 74](#_Toc79177097)

[11.3 Creating CloudWatch Logs for CloudTrail 75](#_Toc79177098)

[11.3.1 Verify that CloudWatch logs has been created for CloudTrail 76](#_Toc79177099)

[12. getProfile AWS Lambda Function 78](#_Toc79177100)

[12.1 Create getProfile Lambda Function 78](#_Toc79177101)

[12.2 getProfile source code 78](#_Toc79177102)

[12.3 Setting up getProfile API Gateway 81](#_Toc79177103)

[12.4 Setting IAM role permissions 85](#_Toc79177104)

[12.5 Test getProfile Lambda Function on Postmen 86](#_Toc79177105)

[12.6 Test that the Function works for the website. 87](#_Toc79177106)

[13. updateDesigns AWS Lambda Function 89](#_Toc79177107)

[13.1 Create updateDesigns Lambda Function 89](#_Toc79177108)

[13.2 updateDesigns source code 89](#_Toc79177109)

[13.3 Setting up updateDesign API Gateway 92](#_Toc79177110)

[13.4 Setting IAM role permissions 96](#_Toc79177111)

[13.5 Verify that the Lambda Function works. 98](#_Toc79177112)

[14. manageSubmissions AWS Lambda Function 100](#_Toc79177113)

[14.1 Create manageSubmissions Lambda Function 100](#_Toc79177114)

[14.2 manageSubmissions source code 100](#_Toc79177115)

[14.3 Setting up managesubmissions API Gateway 103](#_Toc79177116)

[14.4 Setting IAM role permissions 108](#_Toc79177117)

[14.5 Verify that managesubmissions Lambda Function works 108](#_Toc79177118)

# Introduction

* 1. Overview  
     This report documents the step-by-step deployment of the web applications on AWS for Bee Design Award Competition System.
  2. Scope  
     - Troubleshooting and testing during deployment stages  
     - Step-by-step on how applications was refactored using serverless approach on AWS  
     - Cloud Security features that was enhanced in the cloud, such as Security groups, WEB API, Logging and etc.  
     - Database Security  
     - Evidence on the results for the features that was implemented and explored.
  3. Project Team

|  |  |
| --- | --- |
| **Role** | **Name** |
| Project Team Member | Tan Jun Heng |

# Prerequisites

Please download WinSCP as it is required for the deployment of the AWS services.

<https://winscp.net/eng/download.php>

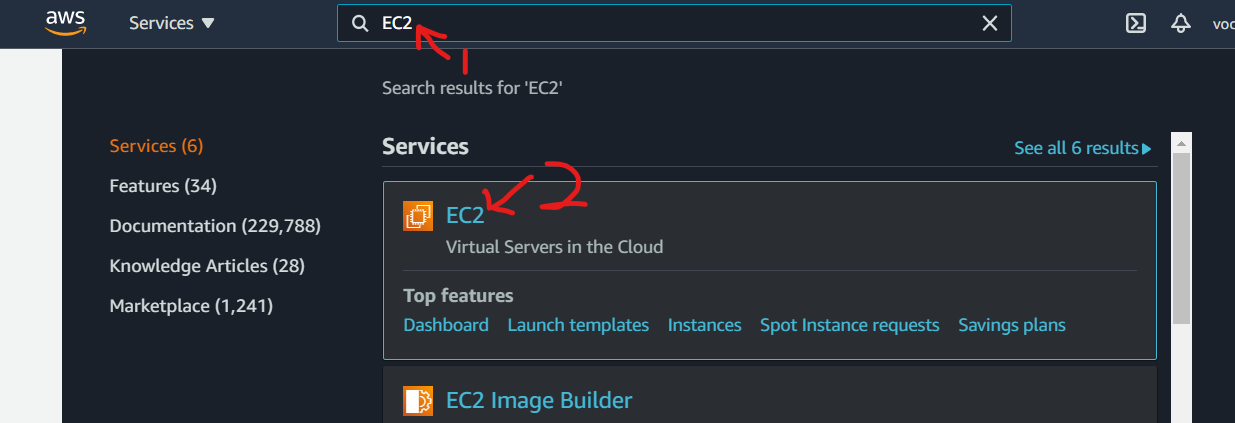


# EC2 Instances Deployment

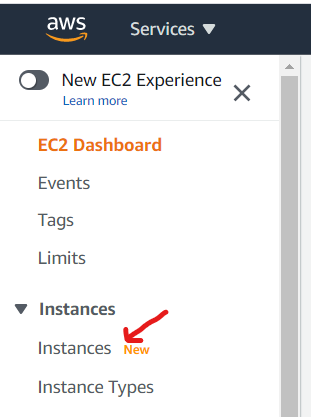
## 3.1 Creating Amazon EC2 Instance and setting security group

This section shows the steps on how an EC2 instances is deployed.

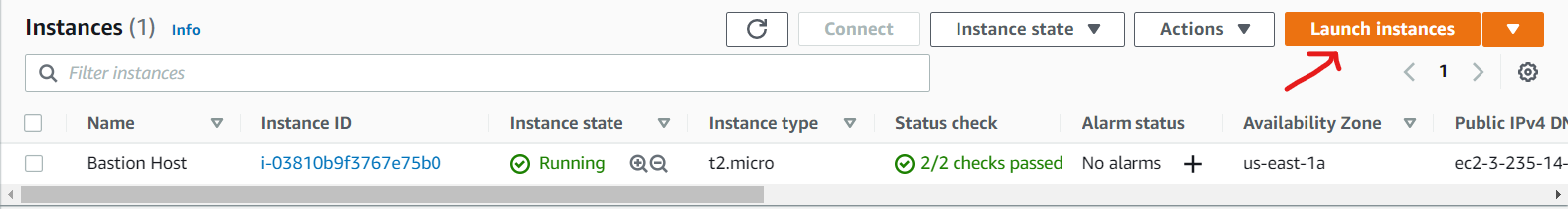
On AWS Management Console

Click on the Search bar on top of the page and search for “EC2” service.

Click on the EC2 service.

On the left side of the EC2 Dashboard, click on “Instances”

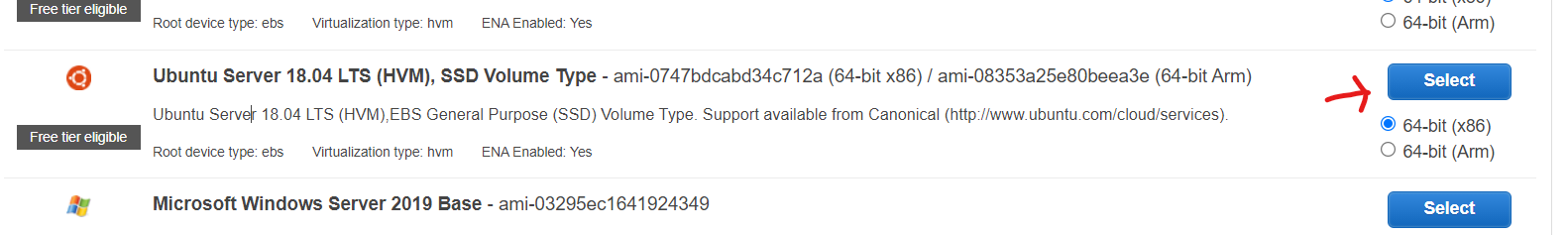
On the instances page, Click on “Launch instances” button.



After clicking on the “Launch instances” button, it would direct you to the instances creation page.

Step 1: Choose an Amazon Machine Image (AMI)

Look for Ubuntu Server 18.04 LTS (HVM), SSD Volume Type



Click “Select”

Step 2: Choose an Instance Type

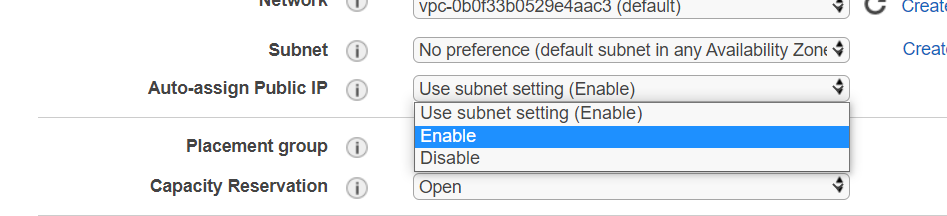
Leave the settings as it as and click “Next: Configure Instance Details”



Step 3: Configure Instance Details

On the Auto-assign Public IP setting,

Click “Enable”



Click on “Next: Add Storage” button.



Step 4: Add Storage

Leave the settings as it as and click “Next: Add Tags”



Step 5: Add Tags

Leave the settings as it as and click “Next: Configure Security Group”.



Step 6: Configure Security Group

On Assign a security group, select “Create a new security group” radio button.

Name the security group on Security group name as “ESDE-EC2-SecurityGroup”

With the description as “ESDE-EC2-SecurityGroup created 2021-07-26”

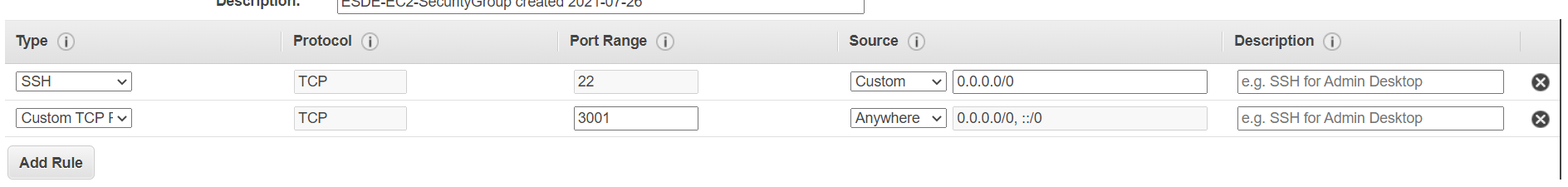
Click on “Add Rule”



Choose Type: Custom TCP Protocol

Choose Port Range: 3001

Source: Anywhere

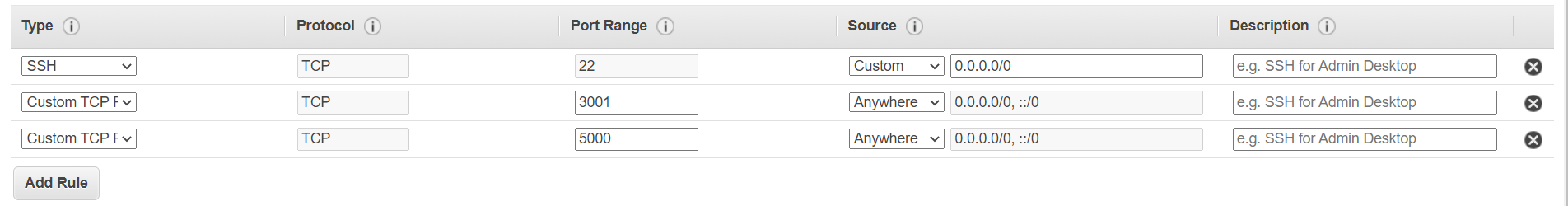


Add another rule by clicking on the “Add Rule” button.

Choose Type: Custom TCP Protocol

Choose Port Range: 5000

Source: Anywhere



Click “Review and Launch”



Step 7: Review Instance Launch

Review the settings and click “Launch”



## 3.2 Setting up EC2 Key Pair

The EC2 Key Pair is required to access the AWS EC2 Instance. Without the key, access to the EC2 Instance will be denied.

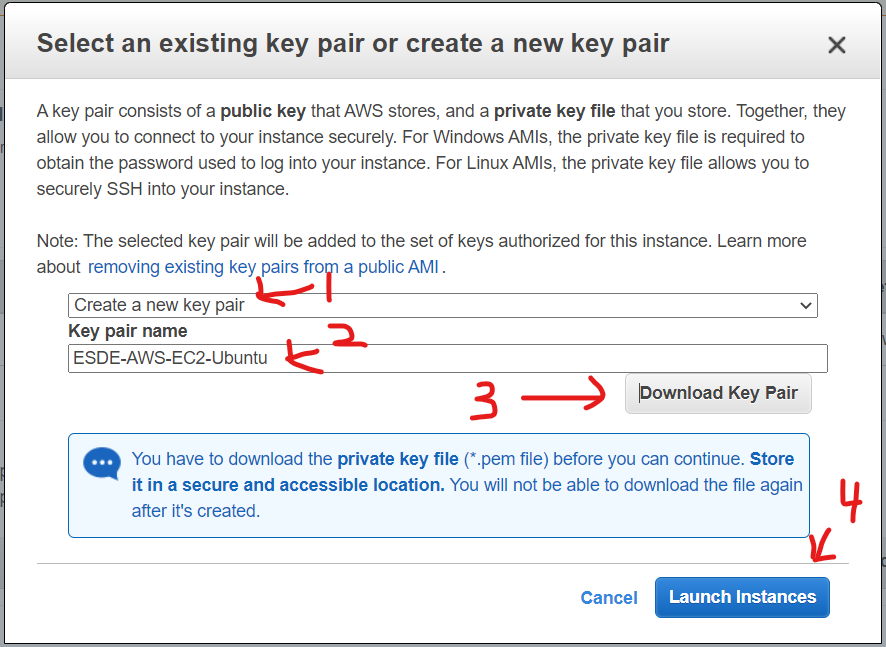
A prompt will appear asking you to Select an existing key pair or create a new key pair.

Choose “Create a new key pair”

Key pair name: “ESDE-AWS-EC2-Ubuntu”

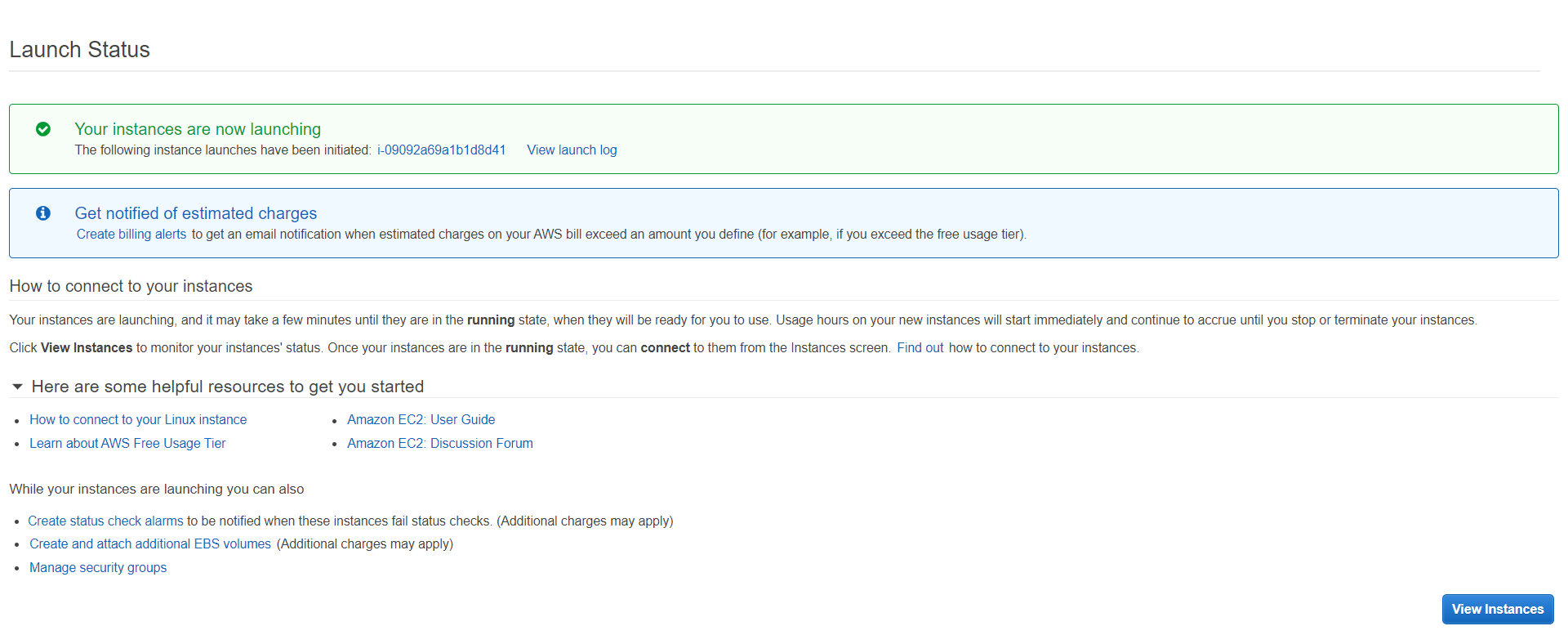
Click “Download Key Pair”

After downloading the Key Pair, click “Launch instances”



After clicking on “Launch Instances”

A prompt should say that the instances is being launched.

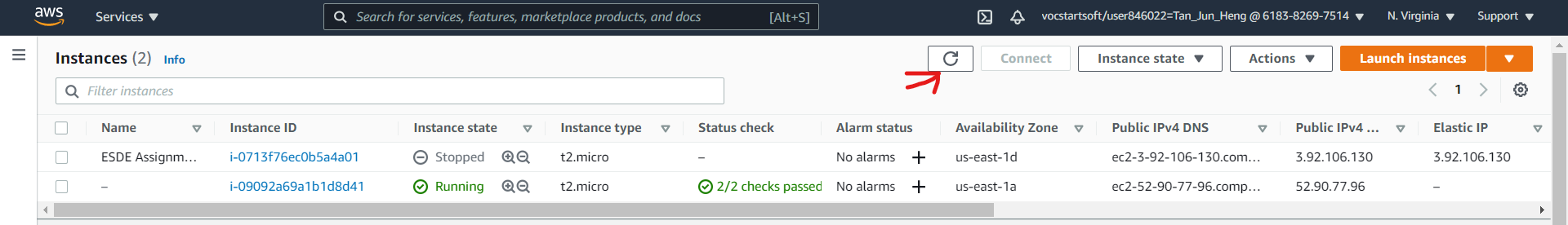


Click “View Instances”



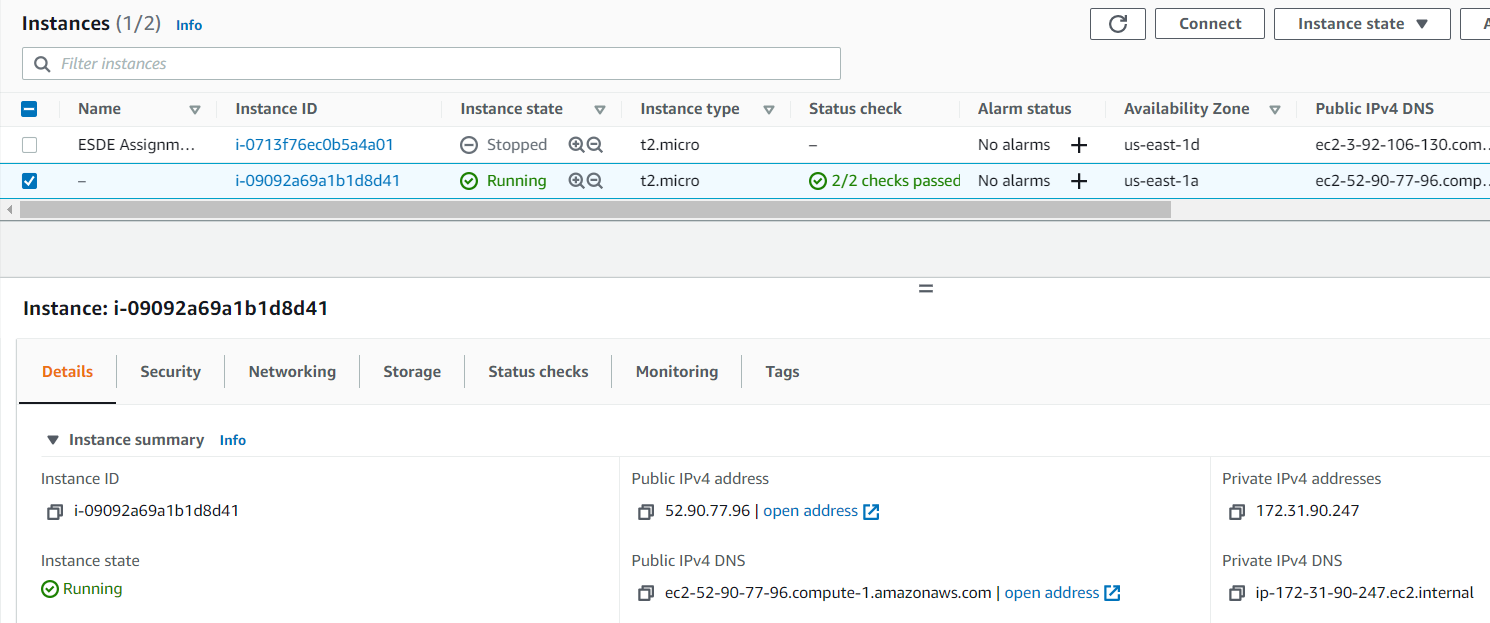
It would redirect you back to the Instances page.

The instance should show that it is initializing, wait for a few mins and click on the “Refresh button”



And you should see that the Instance state is “Running” and the Status check is “2/2 checks passed”

Click on the running instance and the instance details should pop up.



## 3.3 Connecting to EC2 Instance using WinSCP

**WinSCP** is needed to access the folders and command prompt of the EC2 Instance.

Open up **WinSCP**.

On the top left of the software, click “New Session” button.

On the Login Prompt,

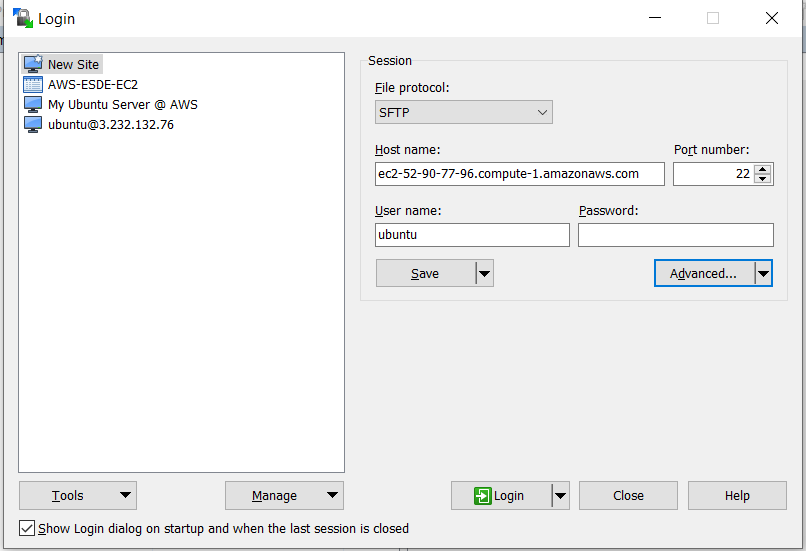
change **File protocol** to “**SCP**”.

On the **Instance details** page,

copy the **Public IPv4 DNS**: “**ec2-52-90-77-96.compute-1.amazonaws.com**” and paste it on **WinSCP** Hostname.

Ensure that the **port** number on **WinSCP** is 22.

Input the **Username** as “ubuntu”, leave the **password** empty.



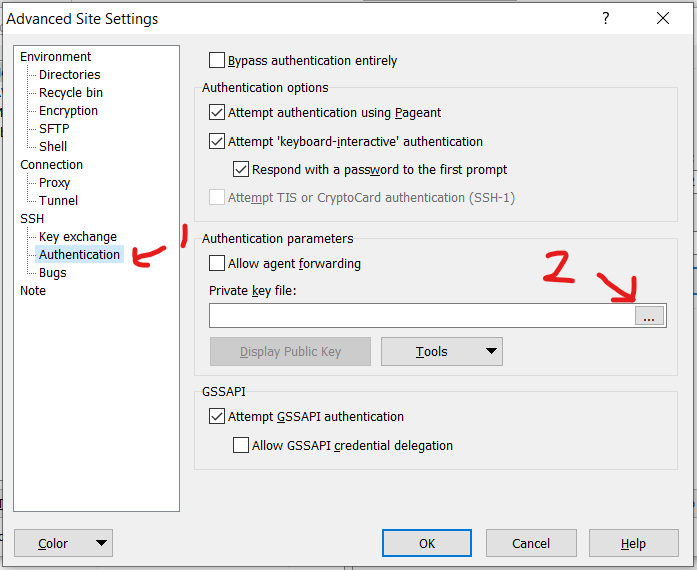
### 3.3.1 Generating EC2 Private Key

A Private Key is needed in order to access the EC2 Instance.

Click “Advanced”

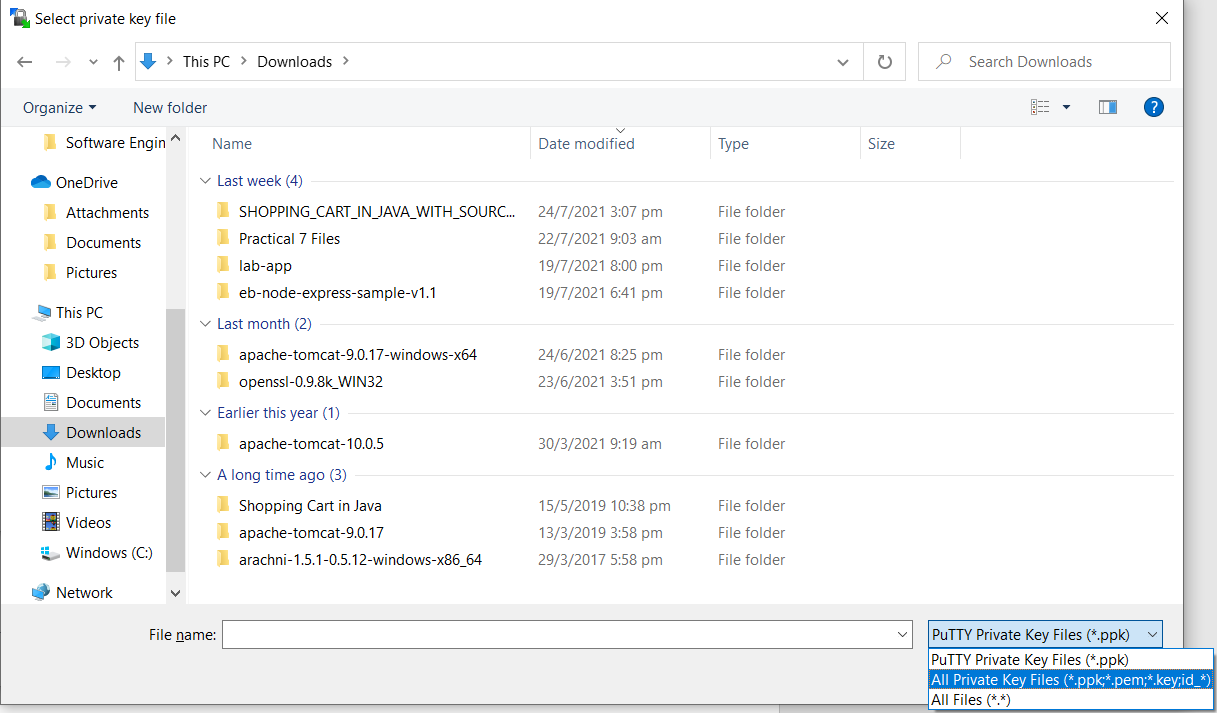
On the left side of Advanced Site Settings

Click Authentication under SSH



On the Private key file, Click on the triple dot button.

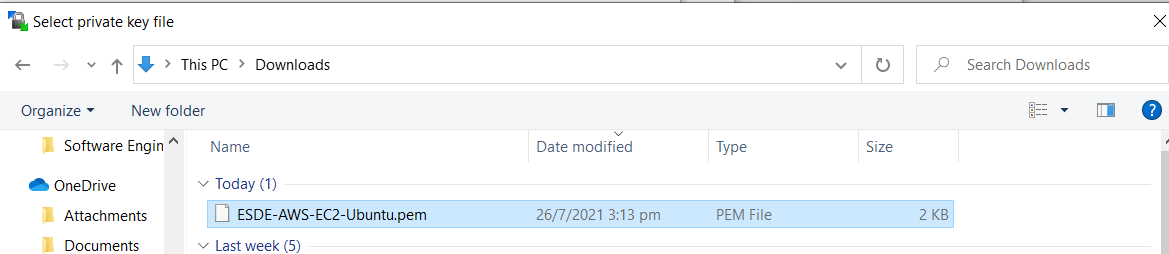
The “Select private key file” prompt would pop up, navigate to where you downloaded your key pair folder.



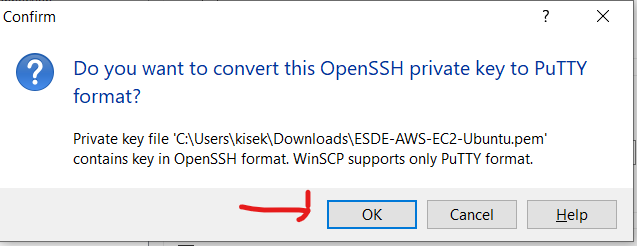
On the bottom right of the prompt, select file type as “All Private Key Files”

After changing the file type, you would be able tot see the private key.

Select the private key by double clicking on it.



There would be a prompt asking you to confirm.



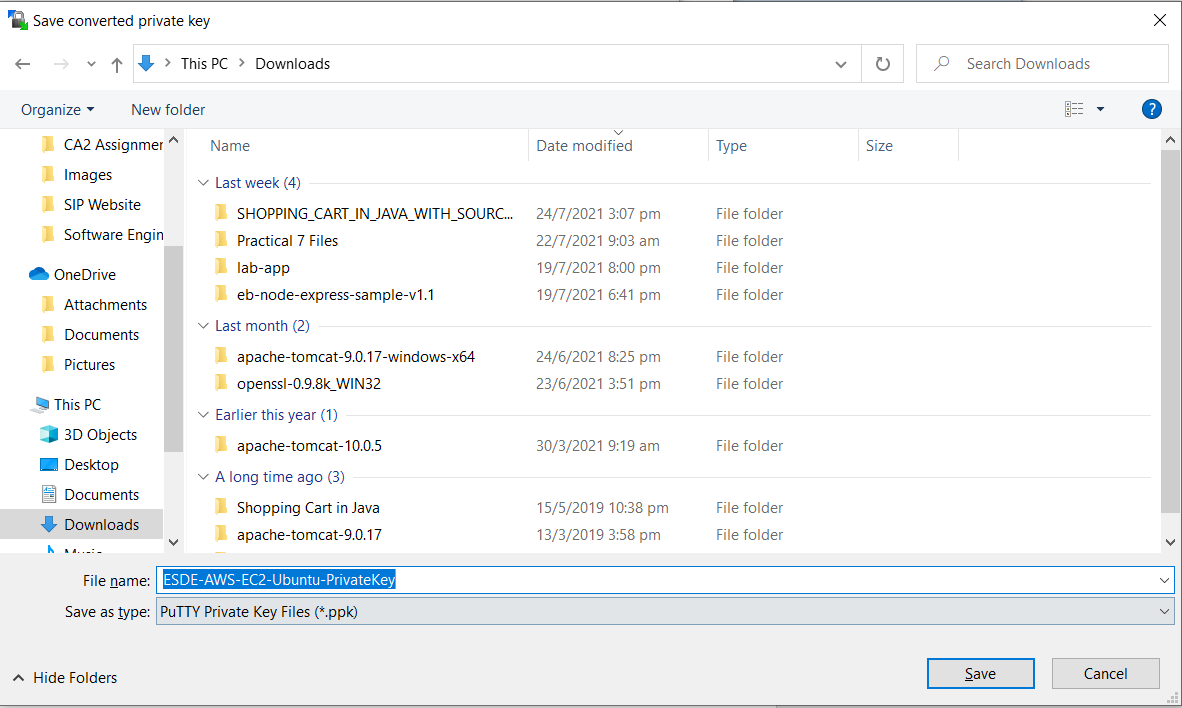
Click “OK”

A prompt would pop up asking you to save the converted private key.

Ensure that the file is saved as a PuTTY Private Key File.

Navigate into your desired folder and save file name as “ESDE-AWS-EC2-Ubuntu-PrivateKey”.

Click “Save”.



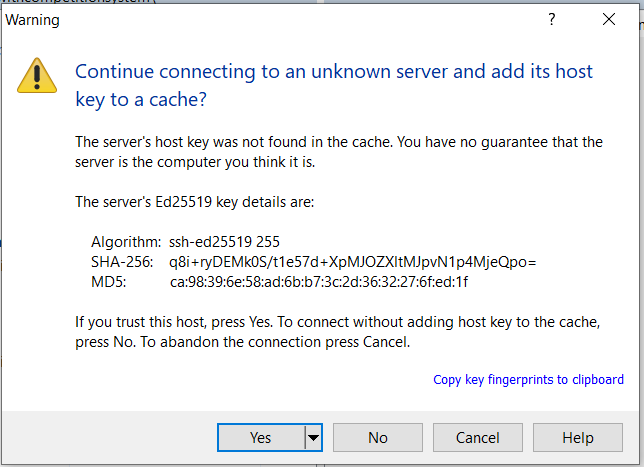
After clicking Save, an information prompt would pop up.

Click “OK”.

Click “OK” again.

### 3.3.2 Connecting to Amazon EC2 Instance

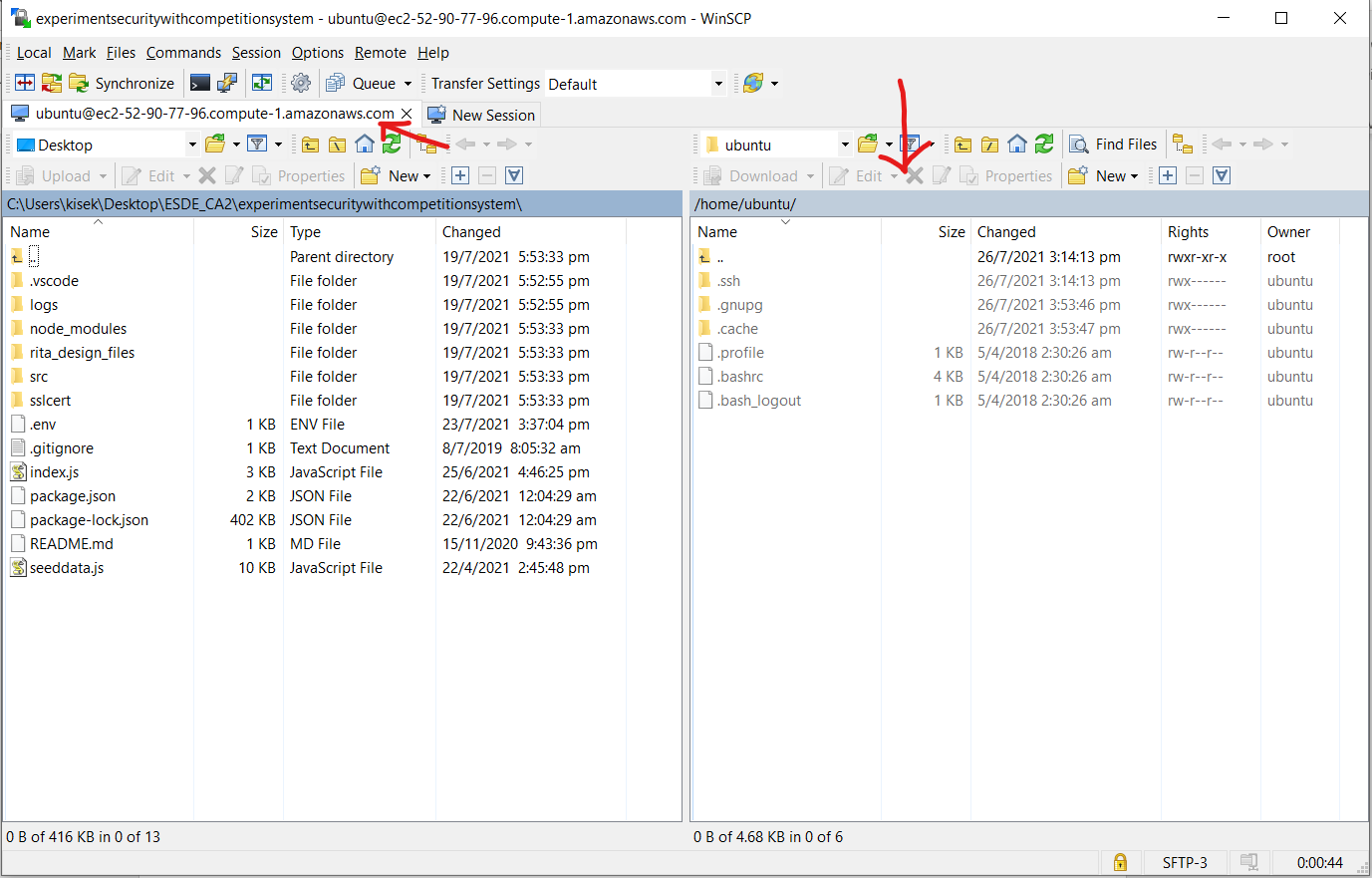
On the Login Page, click “Login”.



A prompt would ask you if you want to continue connecting to the instance.

Click “Yes”.

If the SSH is successful, you would be able to see the instances folders.

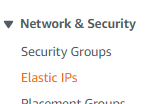


## 3.4 Allocating Elastic IP addresses to Amazon EC2

By allocating an EIP to an EC2 Instance, the EIP will continue to be attached to the EC2 instance even after it has been restarted. This helps to reduce the need to constantly change IP addresses to access the website.

On the side bar of the Instances Dashboard,

Go to Elastic Ips



Click “Allocate Elastic IP address”



Leave the settings as it is and click “Allocate”



An Elastic IP address will be given to you and you start now associate it with the Amazon EC2

Click on the newly allocated IP address.

Click “Actions”

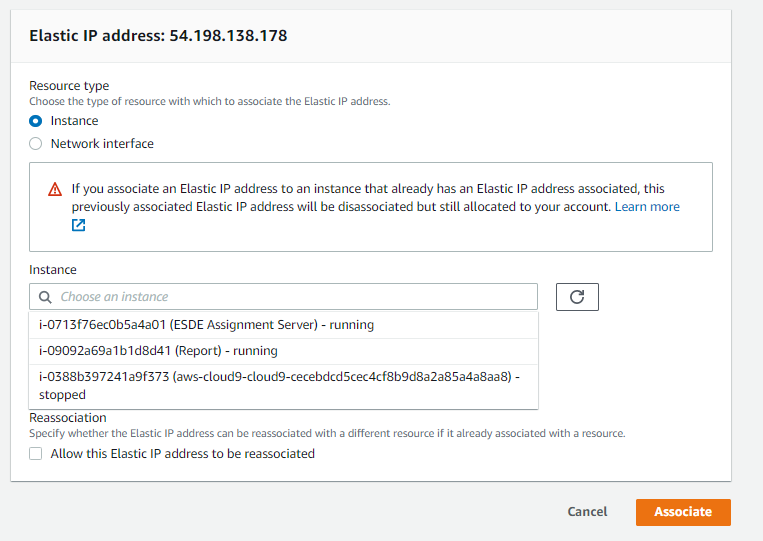


Click “Associate Elastic IP address”

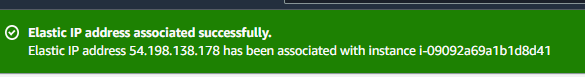


Under the “Instance” section,

Choose “ESDE Assignment Server”



Click “Associate”



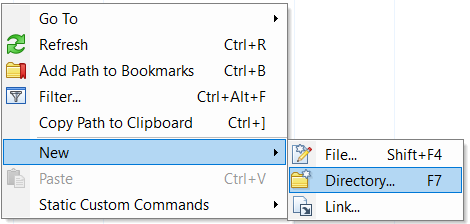
If you are able to see this message, Congratulations ! You have successfully associated an Elastic IP Address to your Amazon EC2 Instance.

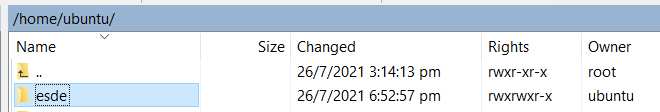
# 4. Setting up FrontEnd and BackEnd Server in Amazon EC2

## 4.1 Uploading FrontEnd & BackEnd project for Hosting

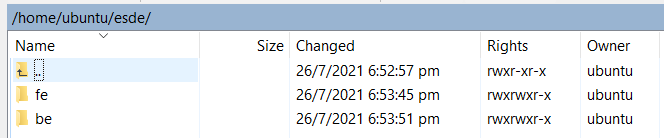
To set up the project server for hosting so that users can start using the website.

In the EC2 Instance folder, create a folder named “esde”

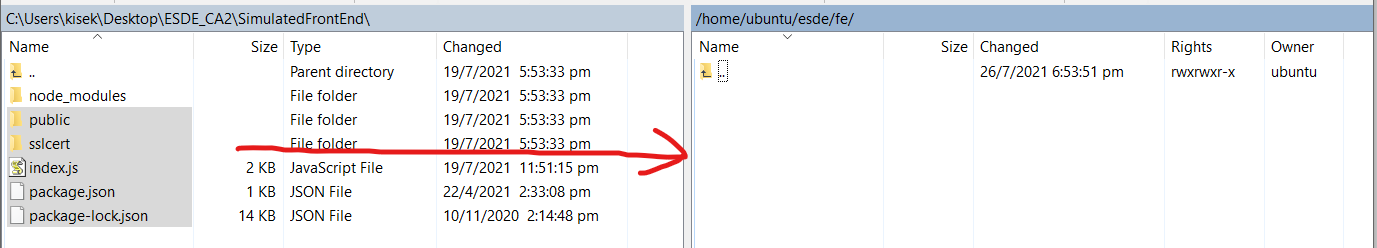




Double click into the folder and create 2 more folders named “fe” and “be”.



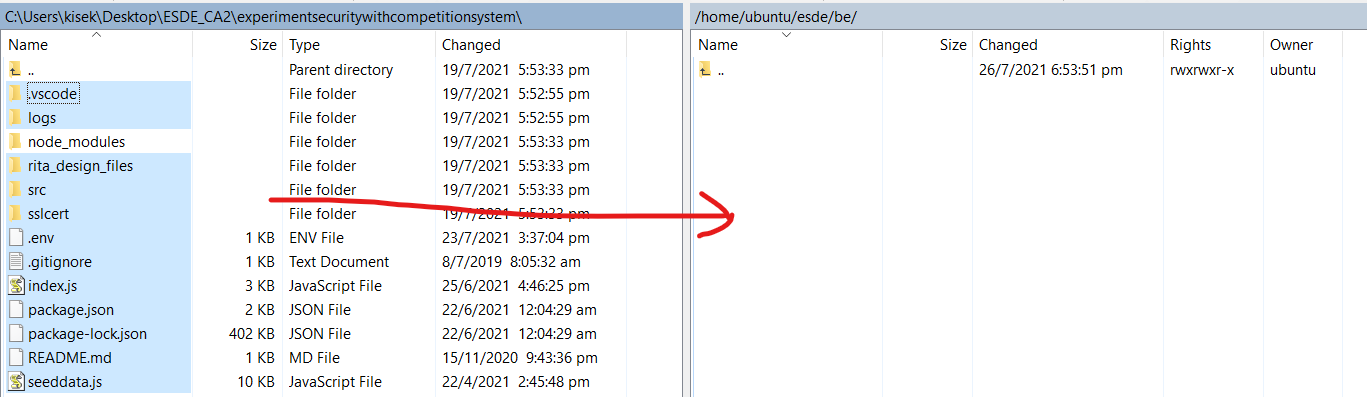
Navigate into the “fe” folder and upload the FrontEnd Web Application folders except for the node\_modules in the folder as highlighted.



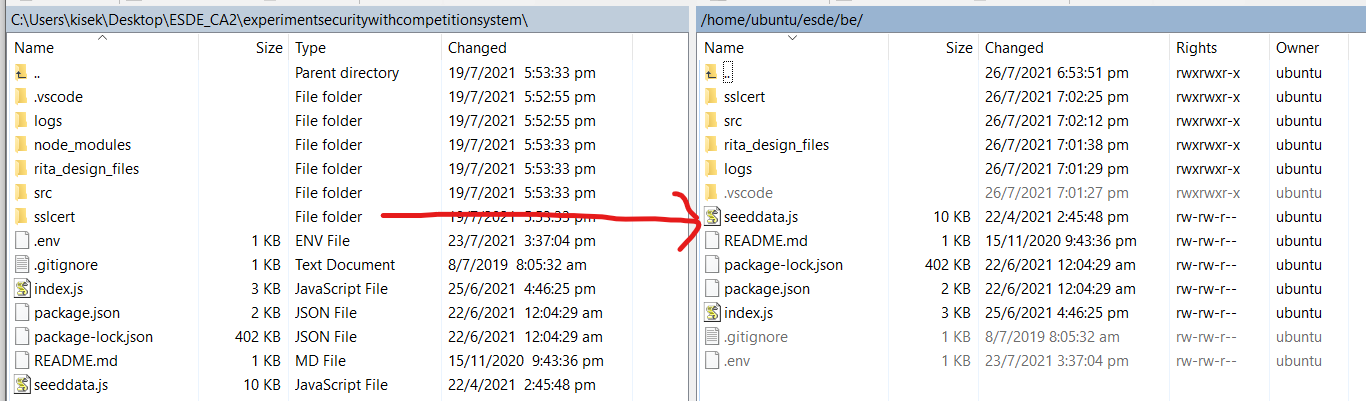
After uploading the folder, it should appear on the Instance’s “fe” folder as shown.



In the EC2 Instance folder, navigate into the “be” folder and upload the BackEnd Web Application Folder except for the node\_modules as highlighted.



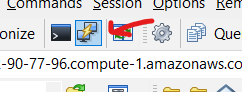
After uploading the folder, it should appear on the Instance’s “be” folder as shown.

****

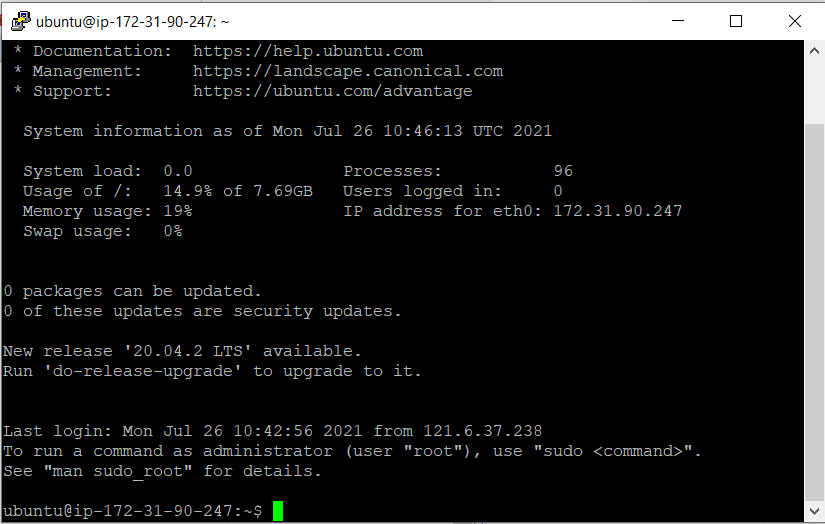
## 4.2 Connecting to EC2 Instance using PuTTY

Connection to the EC2 Instance can be established by using the PuTTY Session in WinSCP.

On the top left section of the WinSCP window, click on the icon



The PuTTY window will be shown.



Type command “ls” in Putty and it will show the web application folder that was uploaded.



## 4.3 Hosting FrontEnd & BackEnd Web Application

### 4.3.1 Install nodejs on your Amazon EC2

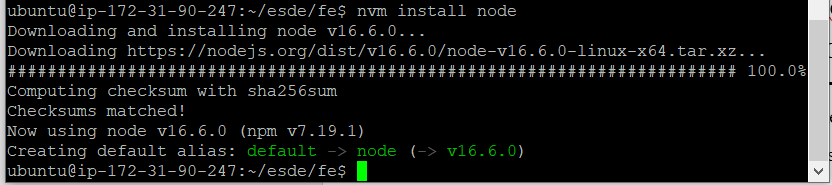
Install node version manager (nvm) by typing the following commands on PuTTY

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash

. ~/.nvm/nvm.sh

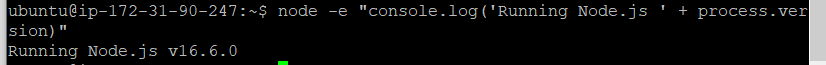
nvm install node

If the installation is successful, you should be able to see the following the version of nodejs



Test that your nodejs is installed and running correctly by typing the following command

node -e "console.log('Running Node.js ' + process.version)"



### 4.3.2 Running FrontEnd Web Application on Amazon EC2

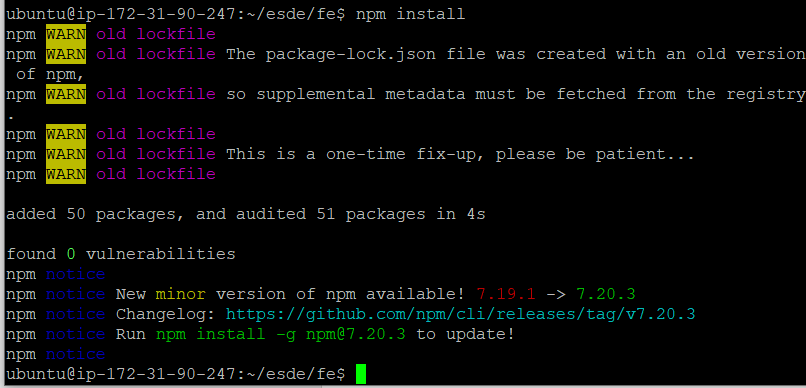
To start running the FrontEnd server, we will need to install node modules into the into FrontEnd folder.

Type the following commands.

“cd esde/fe”

Type “npm install” to install the node modules.

After running the command, you should see that the node modules has been installed.



Type ls to verify that the node\_modules folder has been installed.



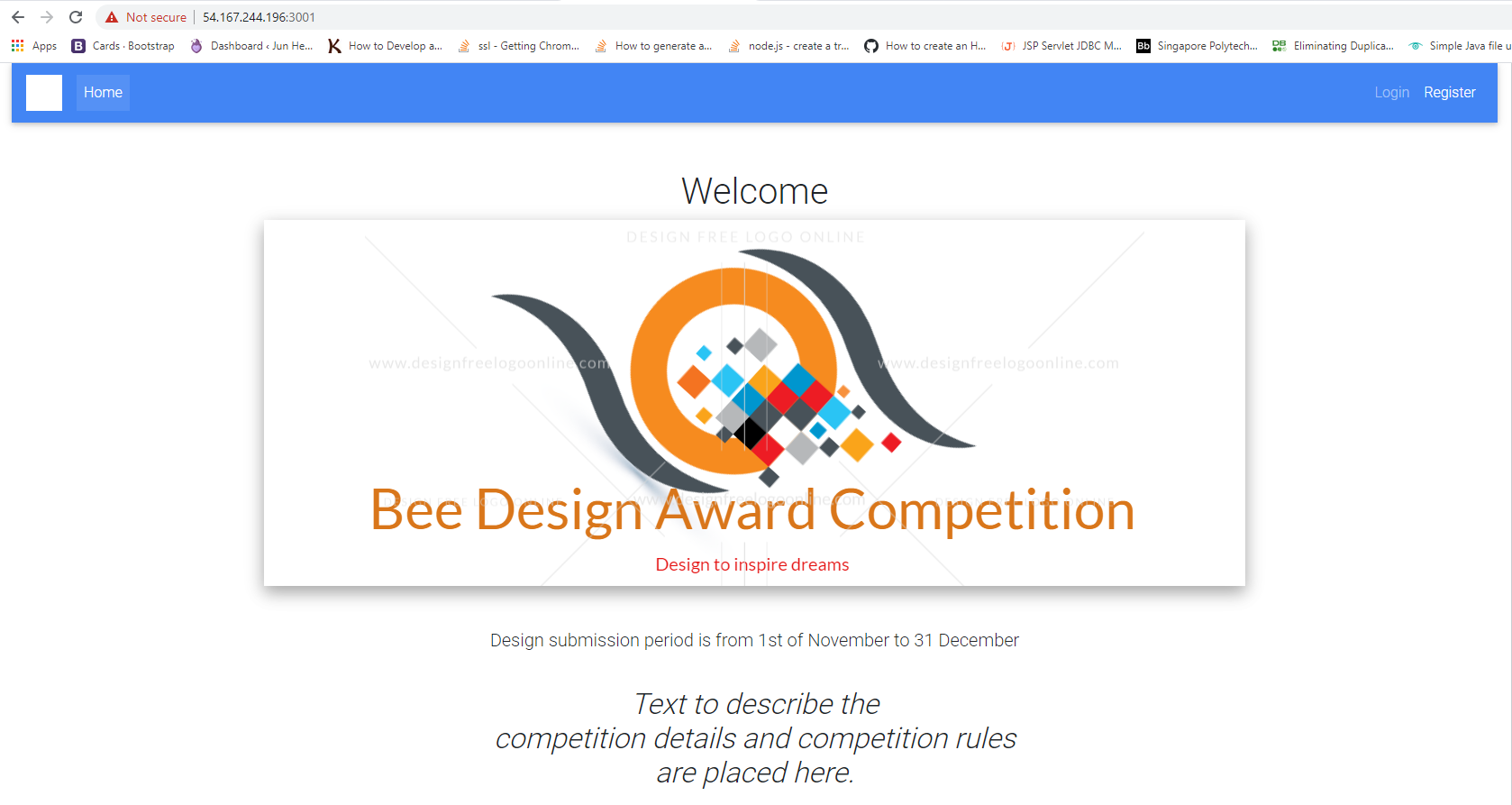
To start hosting the FrontEnd website, run the following command

“node index.js”

You should see that the server has been started at port 3001.



Open your web browser and go to <https://54.167.244.196:3001/>



Congratulations, you should successfully started the Front End web application.

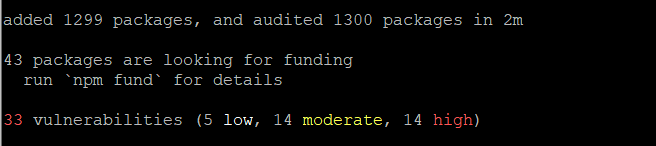
### 4.3.3 Running BackEnd Web Application on Amazon EC2

Open another PuTTY session on WinSCP,

Cd into the BackEnd folder.

“cd esde/be”

Do “npm install” to install the node modules.



To start the BackEnd server, run “node index.js”



Congratulations, you have successfully started the BackEnd web application.

# 5. AWS Cloud9

|  |
| --- |
| **What is an AWS Cloud9 ?**  AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run. And debug your code with just a browser. |

## 5.1 Creating AWS Cloud9 Environment

On AWS Management Console

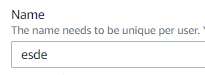
Search for “Cloud9” on the search bar

Click “Create environment”



Step 1: Name environment

Input Name as “esde”



Click “Next step”

****

Step 2: Configure Settings

Leave the settings as it is and click “Next step”

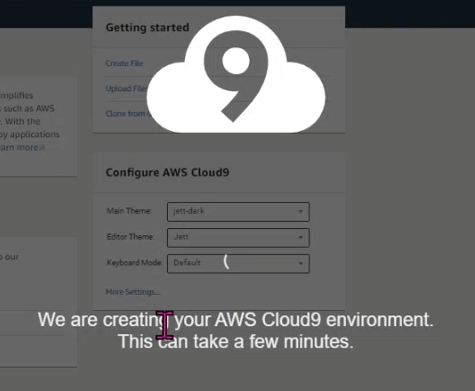
****

Step 3: Review

Ensure that the name and settings are correct and click “Create environment”



Congratulations, you have successfully created a Cloud9 Environment.



# 6. AWS RDS for MySQL

|  |
| --- |
| **What is AWS RDS for MySQL?**  AWS RDS is a database that provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. |

## 6.1 Creating AWS RDS

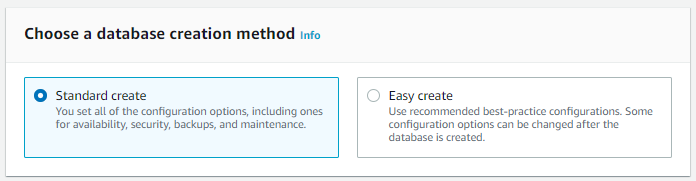
On AWS Dashboard, Go to Amazon RDS

Click “Create database”



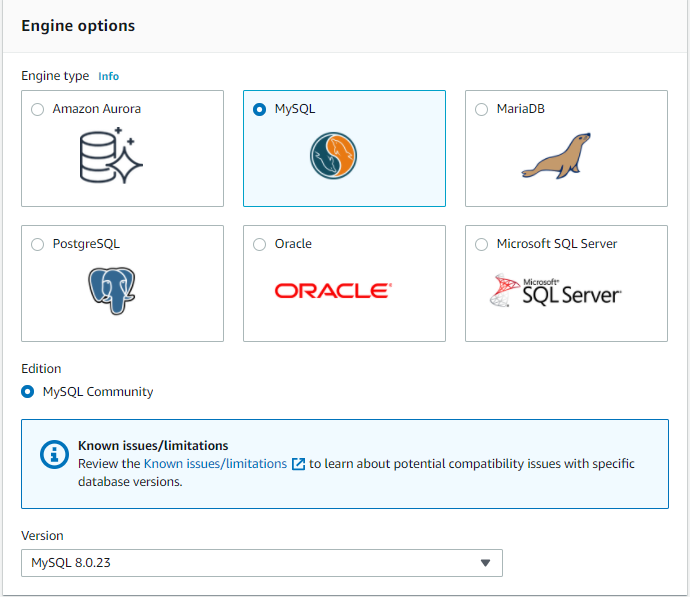
On “Choose a database creation method”

Select “Standard create”



On Engine options section,

Select Engine type as “MySQL”



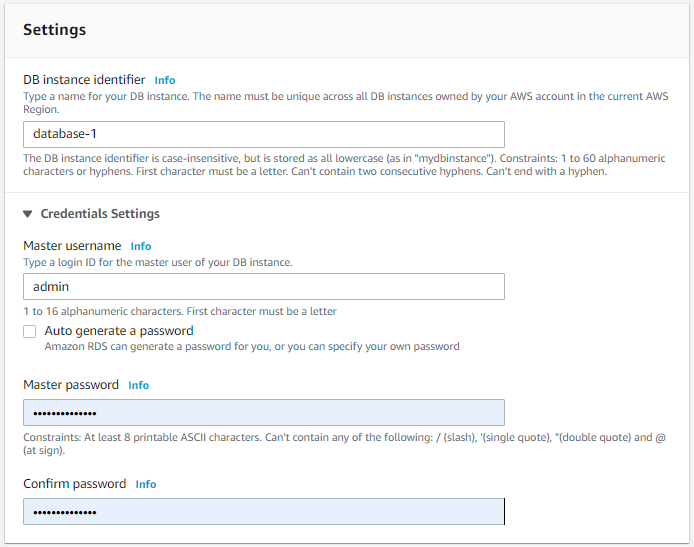
On Templates section,

Select “Free tier”



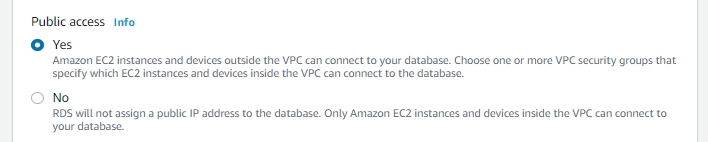
On Settings section,

Name the RDS instance as “database-1” and password as “Junheng01!”



On Connectivity section,

Select “Yes” for public access



Click “Create Database”

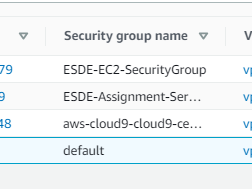


## 6.2 Enabling Port 3306 for AWS RDS Security Group to be accessed anywhere

Go to EC2 Dashboard,

On the side bar, go to Security Groups

Go look for “default” at “Security group name”



Go to Inbound rules and click “Edit inbound rules”



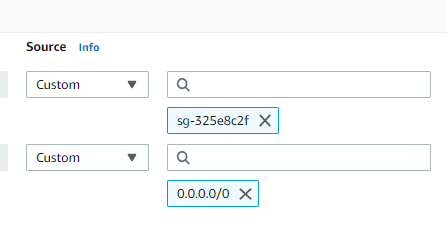
Click “Add rule”



On the Type select “MYSQL/Aurora”



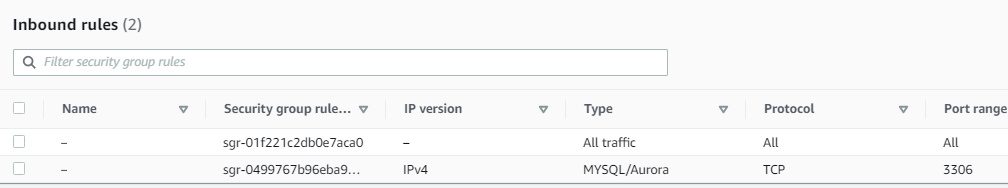
For the source, select “Anywhere-IPv4”



Select “Save rules”

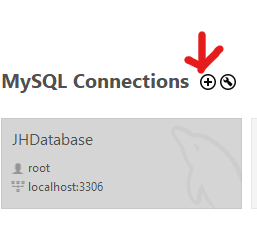


You should be able to see the newly added inbound rules as shown below.



## 6.3 Connecting to RDS Database from MySQL

Open up MySQL Workbench, on the right side, click on the “Plus” icon.

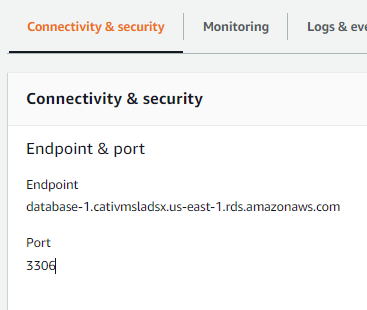


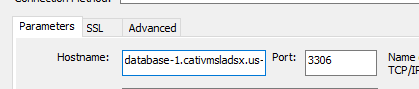
A Setup New Connection prompt will appear.

Set the Connection Name as “AWS-EC2-Database”

Go to Amazon RDS Dashboard and under Connectivity & security,

Copy the Endpoint of the database “database-1.cativmsladsx.us-east-1.rds.amazonaws.com”



Return to MySQL Workbench and input the Endpoint as Hostname as shown. 

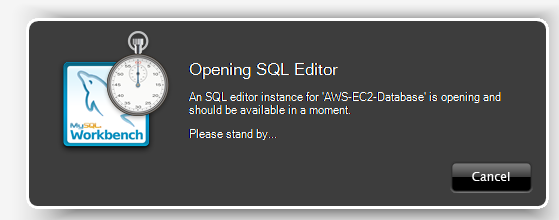
Set the Port as 3306 and Username as admin.

Select “Store in Vault” for password and input “Junheng01!” as password.

Click “OK”

You should be able to see the newly added “AWS-EC2-Database” database connection.

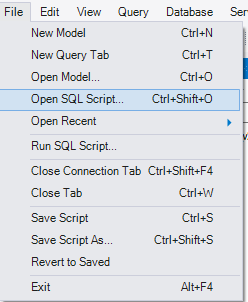
Click on it.



A pop up will appear and the database connect will be established and you will be able to create schemas and insert records into the database.

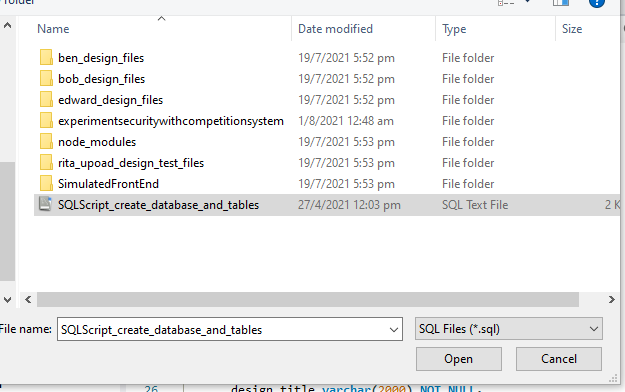
## 6.4 Creating Database Schema

On MySQL Workbench, Go to files and select “Open SQL Script”



A pop up will appear.

Go to the assignments folders and select the SQL Text File as shown.



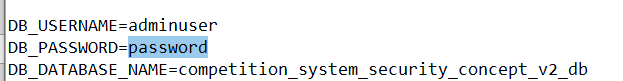
Click on “Open”

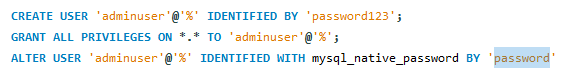
After opening the SQL Text File.

Open up the .env folder in WinSCP.

The folder path is “/home/ubuntu/esde/be”

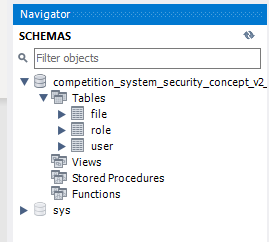
Ensure that the “DB\_PASSWORD” in .env is the same as line 32 of the SQL Text File “IDENTIFIED BY”.





After verifying that the details are correct.

Click on the lightning icon and it will create a schema containing relevant tables as shown.



## 6.5 Inserting records into AWS RDS tables

Open up WinSCP and connect to the EC2 Instance.

After connecting to the EC2

Open Session in PuTTY

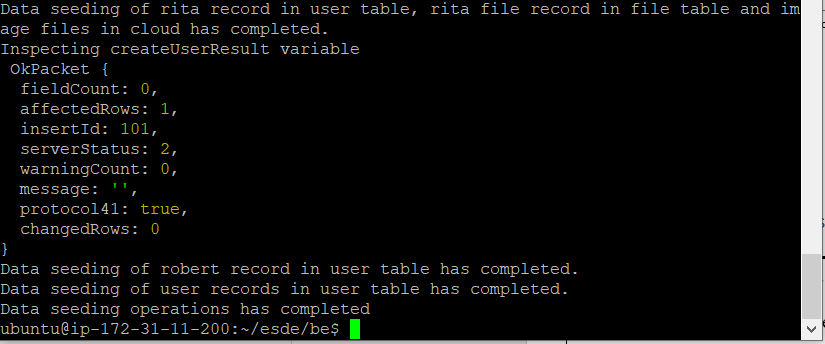
Type the following command.

“cd esde/be”

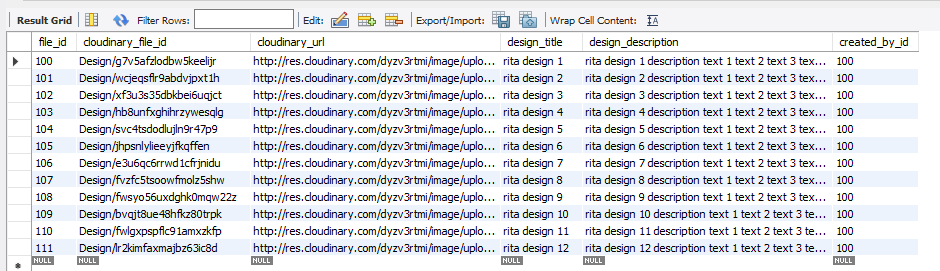
“nodejs seeddata.js”

These commands will insert the record data into the schema tables.

If the data seeding is successful. You will be able to see this following.



Verify that your RDS database has been updated with records.



Congratulations, you have successfully seeded data into the RDS database !

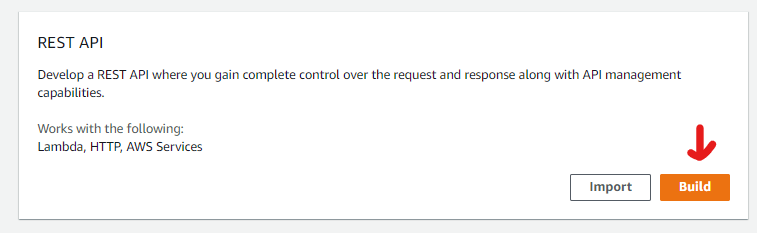
# 7. AWS API Gateway & AWS Lambda

## 7.1 Creating API Gateway

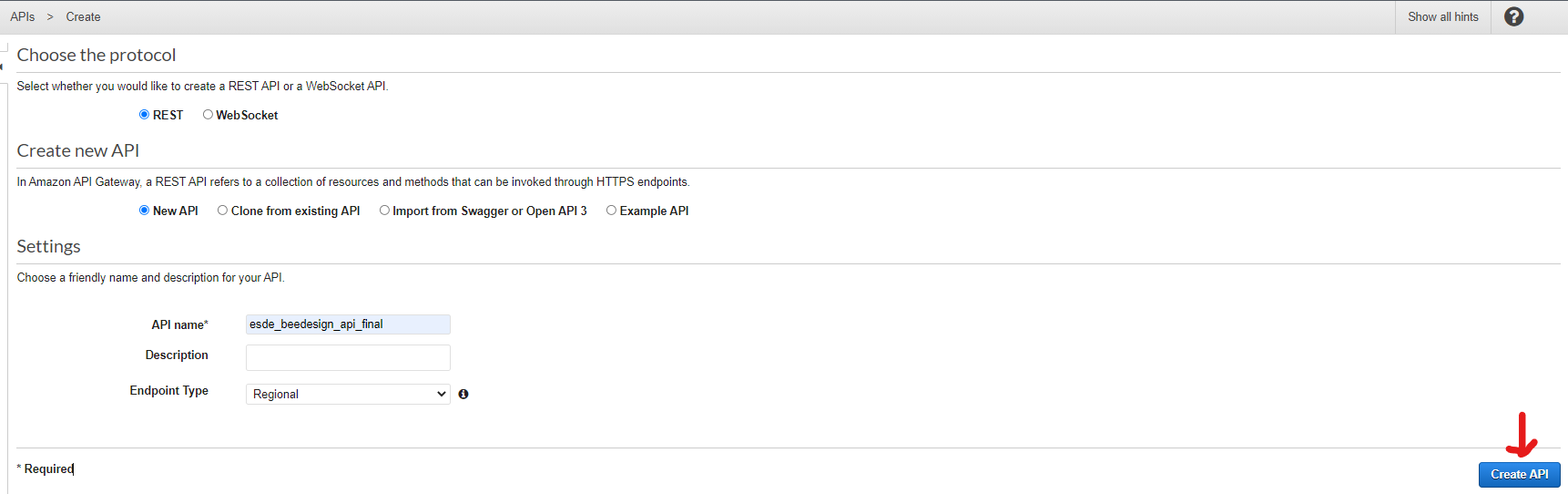
On API Gateway Dashboard, Click “Create API”



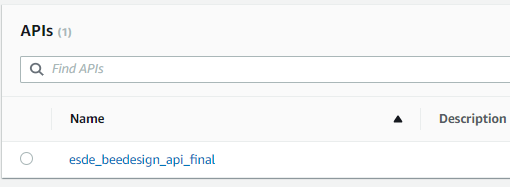
Afterwards, Look for REST API and click “Build”



Name the API as “esde\_beedesign\_api\_final” and click “Create API”



After creating the API Gateway, you should be able to see it on the dashboard as shown



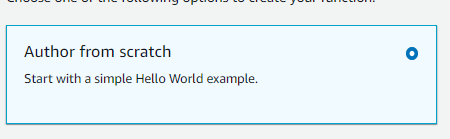
## 7.2 Creating Lambda Function for update\_design.html

We will be creating a Lambda Function for update\_design.html to retrieve design date from DynamoDB.

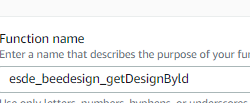
On Lambda Dashboard, click “Create Function”



Ensure that “Author from scratch is selected.”



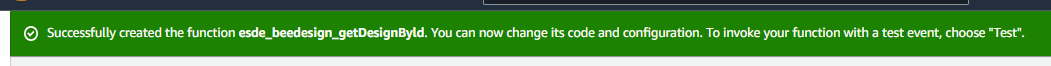
Name the function as “esde\_beedesign\_getDesignByid” and



Click “Create function”



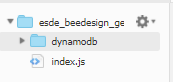
After successfully creating the Lambda Function, you should be able to see the message shown below.



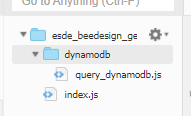
On the Code source section of the function.

Right click on “esde\_beedesign\_getDesignById”

Create a new folder and name it as “dynamodb”



Right click on “dynamodb” folder and create a new file named “query\_dynamodb.js”



**Index.js**

Implement the following code to index.js

var dynamodbQuery = require('dynamodb/query\_dynamodb');

exports.handler = async function(event, context, callback){

    if (event.fid|| (event.queryStringParameters && event.queryStringParameters.fid)) {

        if (event.fid)

            var fileId = parseInt(event.fid);

        else

            var fileId= parseInt(event.queryStringParameters.fid);

            console.log(event)

        var region = "us-east-1"

        var table\_name = "files"

        var expr\_attr\_values = { ":fileid": fileId }

        var key\_cond\_expr = "file\_id=:fileid"

        var proj\_expr = "file\_id,cloudinary\_url,design\_title,design\_description"

        await dynamodbQuery(region, table\_name,expr\_attr\_values,key\_cond\_expr,proj\_expr)

        .then(data => {

                    console.log("Successfully got items from dynamodb.query")

                    var responseCode = 200;

                    var jsonResult = {'filedata': data.Items[0]}

                    let response = {

                            statusCode: responseCode,

                            body: JSON.stringify(jsonResult),

                            headers: {

                                "Access-Control-Allow-Headers" : "Content-Type,user",

                                "Access-Control-Allow-Origin": "\*",

                                "Access-Control-Allow-Methods": "OPTIONS,POST,GET"

                            }

                    }

                    console.log("response: " + JSON.stringify(response))

                    callback(null, response);

            })

            .catch(error => {

                    console.log('There has been a problem with your fetch operation: ' + error.message);

                    var responseCode = 500;

                    let response = {

                        statusCode: responseCode,

                        body: JSON.stringify(error)

                    }

                    console.log("response: " + JSON.stringify(response))

                    callback(null, response);

             });

    } //end if

}

**query\_dynamodb.js**

Implement the following code to “query\_dynamodb.js”

var AWS = require('aws-sdk');

async function queryitems\_dynamodb(region, table\_name,expr\_attr\_values,key\_cond\_expr,proj\_expr) {

    console.log("In the queryitems\_dynamodb method...")

    var dynamodb = new AWS.DynamoDB.DocumentClient({region: region});

    try{

        var params = {  TableName: table\_name,

                        ExpressionAttributeValues: expr\_attr\_values,

                        KeyConditionExpression: key\_cond\_expr ,

                        ProjectionExpression: proj\_expr}

        var items = []

        const results = await dynamodb.query(params).promise()

        console.log("Printing results from  queryitems\_dynamodb " + results)

        return results;

    }

    catch(tryerror) {

        console.log("Error occurred in dynamodb.query..")

        console.log(tryerror, tryerror.stack); // an error occurred

    }

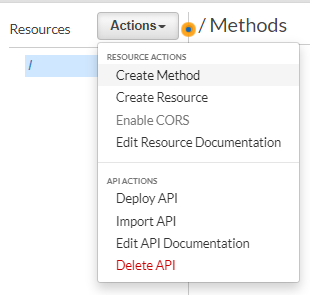
} //end function

module.exports = queryitems\_dynamodb

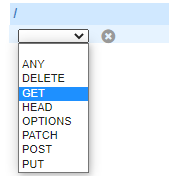
## 7.3 Creating API Method for esde\_beedesign\_getDesignByid function

Click on the newly created API and it will redirect you to the API’s Resources.

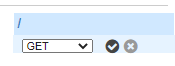
On the Resources Menu, Click on “Actions” button and click “Create Method”



A small dropdown bar will appear, click on it and select “GET”



After selecting the method type, click on the small Tick on the side



A Setup page will appear after clicking on the Tick.

Integration type : Lambda Function

Click on “Use Lambda Proxy integration” checkbox.

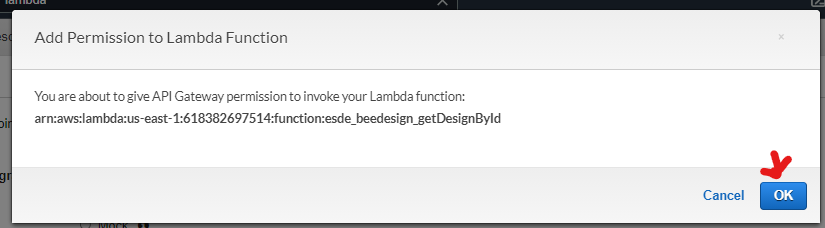
For the Lambda Function section, choose the getDesignById function that was created earlier.



Click “Save”



A pop up will appear asking you to add permission to Lambda Function.



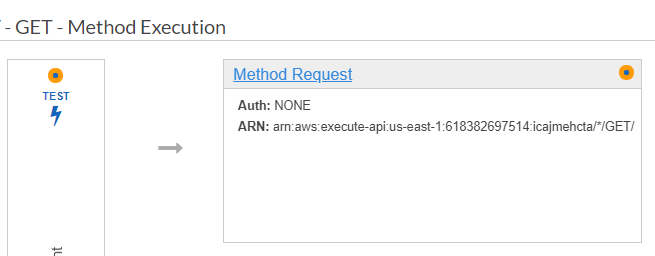
Click “OK”

**Setting up esde\_beedesign\_getDesignById API**

On the Method Execute Dashboard,

**Method Request**

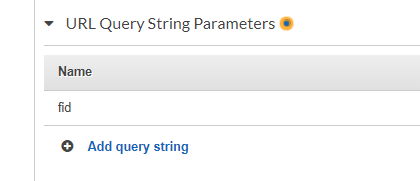
Click on “Method Request”



Select “URL Query String Parameters”

Click “Add query string”

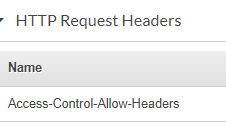
Input “fid” and click on the tick.



Click on “Http Request Headers”

Click “Add header”

Input “Access-Control-Allow-Headers” and click on the Tick

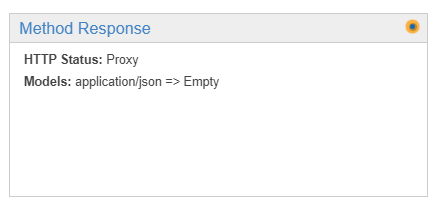


Return back to Method Execution



**Method Response**

Click on “Method Response”



Click on the dropdown bar and under “Response Headers for 200”

Click “Add Header”

Input “Access-Control-Allow-Origin” and click on the tick



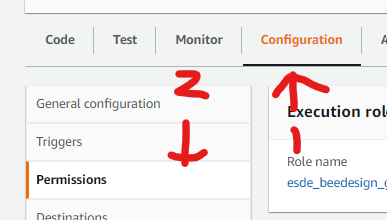
Return back to “Method Execution”



**Integration Request**

To begin setting up the Integration Request, Go to the “esde\_beedesign\_getDesignById” Lambda function dashboard.

Click on “Configuration” and go to “Permissions”



Click on the hyperlink under “Execution role”



It will redirect you to the Lambda Function’s Role permissions.

On the Summary section, copy the “Role ARN” text by clicking on the icon on the right side.



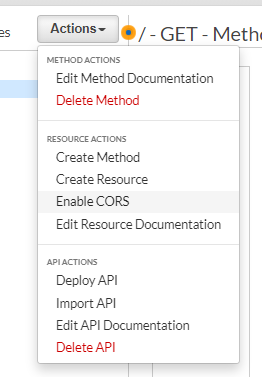
Return back to Integration Request and under “Execution Role” click on the pencil icon and paste the text.



Return back to “Method Execution”

**Enabling CORS**

Under the “Actions” button, click “Enable CORS”



Select both “DEFAULT 4xx” and “DEFAULT 5xx”

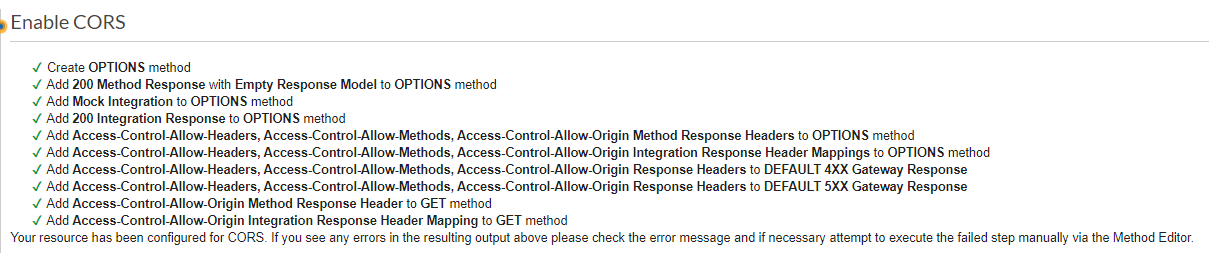
On “Access-Control-Allow-Headers section,

Input the following text

“'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token,user'”

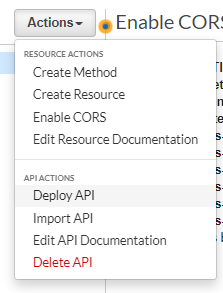
Click on “Enable CORS and replace exiting CORS headers” 

You have successfully enabled CORS if you see the following messages as shown.

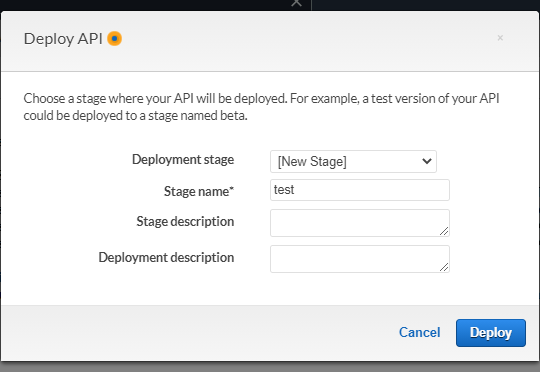


**Deploy API**

Under “Actions” select “Deploy API”



Select New Stage and name it as “test”



Select “Deploy”



After deploying the API you will receive the Invoke URL



Congratulations, you have successfully deployed the API.

**Setting up IAM permissions for esde\_beedesign\_getDesignById**

On the IAM Role Dashboard, Under “Permissions”

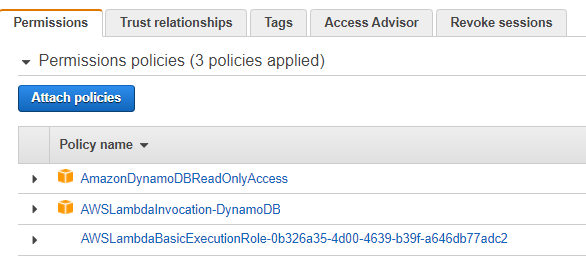
Select “Attach policies”



Attach the following policies

“AmazonDynamoDBReadOnlyAccess”

“AWSLambdaInvocation-DynamoDB”



Under Trust Relationships,

Select “Edit Trust relationship”

Implement the following code under “Service”

"apigateway.amazonaws.com"

"dynamodb.amazonaws.com"

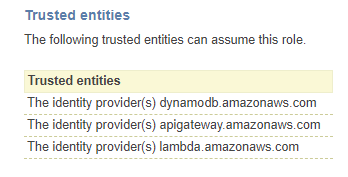
"lambda.amazonaws.com"



Click “Update Trust Policy”



You have successfully edited the trust relationships if you see the following shown below.



**Setting up update\_design.html to use the new API route.**

Go to WinSCP

Go to /home/ubuntu/esde/fe/public/js

Open up update\_design.html folder.

Under the function getOneData()

Change the “url” under axios to https://y2d1tebz53.execute-api.us-east-1.amazonaws.com/test?fid='+ fileId

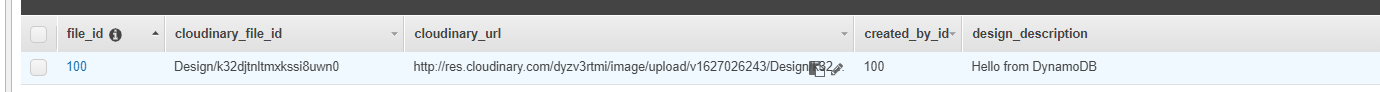


Save the folder

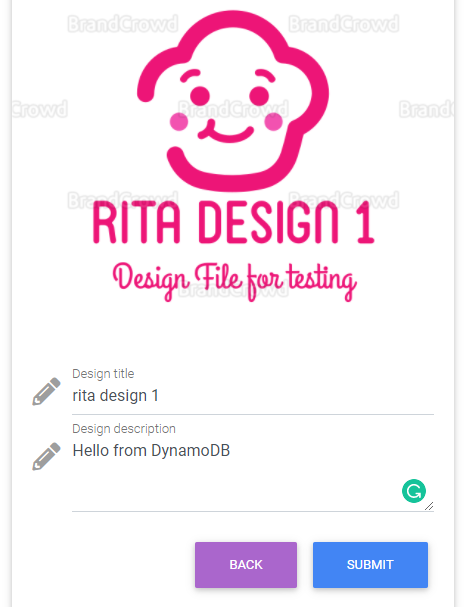
## 7.4 Verify that the API route is being used

After changing the route, go to dynamodb and change one of the file’s design\_description to

“Hello from DynamoDB”



Return to the website and refresh the page.



If you see the design description as shown, congratulations ! you have successfully configured your website to use the API Gateway with Lambda Function !

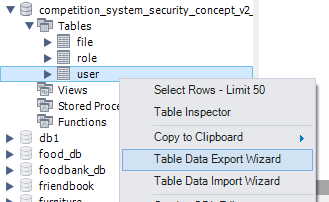
# 8. Creating DynamoDB database

## 8.1 Exporting user table data from MySQL

On MySQL Workbench, go to “competition\_system\_security\_concept\_v2\_db” schemas

Open up the Tables and right click the “user” table

Select “Table Data Export Wizard”

****

Leave the settings as default and click “Next”



On File Path, select on “Browse”

Save File name as “user-data” and File type as “.json” (You may save the file location wherever you want)

Click “Save”



Click “Next >”



Then “Finish”

On your AWS Dashboard, go to the DynamoDB services and click “Create Table”

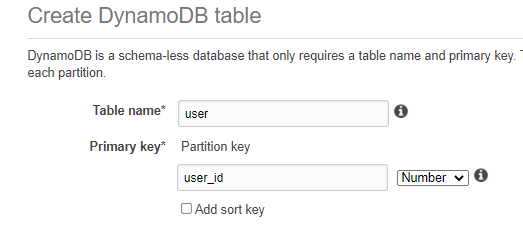


Insert “user” as table name

Insert user\_id as primary key

Select “Number” from the drop down box

This is how the table should look like.



Click on “Create” button



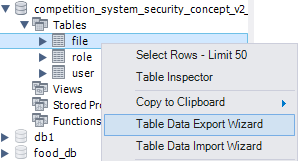
Congratulations, you have successfully created the user DynamoDB table.

## 8.2 Exporting file table data from MySQL

On MySQL Workbench, go to “competition\_system\_security\_concept\_v2\_db” schemas

Open up the Tables and right click the “file” table

Select “Table Data Export Wizard”

****

Leave the settings as default and click “Next”



On File Path, select on “Browse”

Save File name as “file-data” and File type as “.json” (You may save the file location wherever you want)

Click “Save”

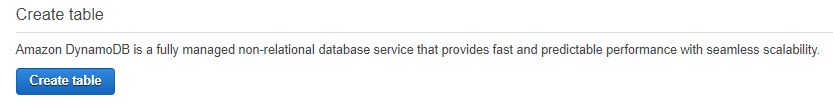


Click “Next >”



Then “Finish”

On DynamoDB Dashboard, click “Create Table”

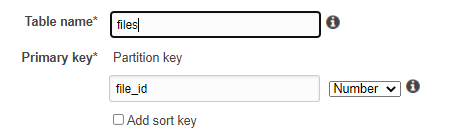


Insert “files” as table name

Insert file\_id as primary key

Select “Number” from the drop down box

This is how the table should look like.



Click on “Create” button



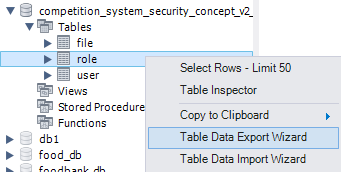
Congratulations, you have successfully created the files DynamoDB table.

## 8.3 Exporting role table data from MySQL

On MySQL Workbench, go to “competition\_system\_security\_concept\_v2\_db” schemas

Open up the Tables and right click the “role” table

Select “Table Data Export Wizard”

****

Leave the settings as default and click “Next”



On File Path, select on “Browse”

Save File name as “role-data” and File type as “.json” (You may save the file location wherever you want)

Click “Save”

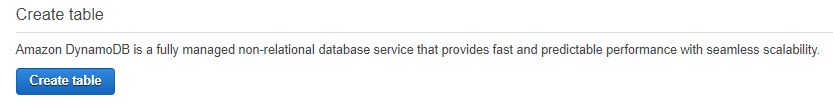


Click “Next >”



Then “Finish”

On your AWS Dashboard, go to the DynamoDB services and click “Create Table”

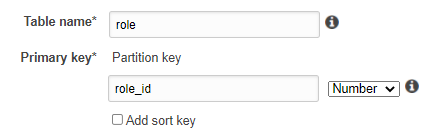


Insert “role” as table name

Insert role\_id as primary key

Select “Number” from the drop down box

This is how the table should look like.



Click on “Create” button



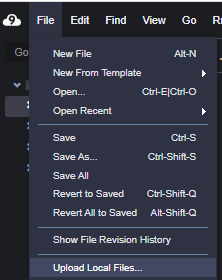
Congratulations, you have successfully created role DynamoDB table.

## 8.4 Uploading MySQL database to DynamoDB using Cloud9

### 8.4.1 Uploading user Table

Go to the Cloud9 Environment

On the File tab, Click “Upload Local Files”



Drag & drop the user-data.json file into the uploader

After uploading, ensure that the user-data.json has been uploaded.



Create a new file named user-data-seeder.js

Insert the following codes into the js folder.

var

AWS = require("aws-sdk"),

DDB = new AWS.DynamoDB({

    apiVersion: "2012-08-10",

    region: "us-east-1"

}),

file\_DATA\_ARR = require("./user-data.json");

function addNewItemsFromJSON(){

console.log("All items now removed, re-seeding now");

var

    file = {},

    file\_formatted\_arr = [],

    params = {};

for(var i\_int = 0; i\_int < file\_DATA\_ARR.length; i\_int += 1){

    file = {

        PutRequest: {

            Item: {

                user\_id: {

                    "N": file\_DATA\_ARR[i\_int].user\_id.toString()

                },

                fullname: {

                    "S": file\_DATA\_ARR[i\_int].fullname

                },

                email: {

                    "S": file\_DATA\_ARR[i\_int].email

                },

                user\_password: {

                    "S": file\_DATA\_ARR[i\_int].user\_password

                },

                role\_id: {

                    "N": file\_DATA\_ARR[i\_int].role\_id.toString()

                }

            }

        }

    };

    file\_formatted\_arr.push(file);

}

params = {

    RequestItems: {

        "user": file\_formatted\_arr.reverse()

    }

};

DDB.batchWriteItem(params, function(err, data){

    if(err){

        throw err;

    }

    console.log("OK");

});

}

(function init(){

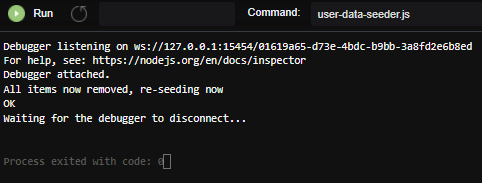
addNewItemsFromJSON();

})();

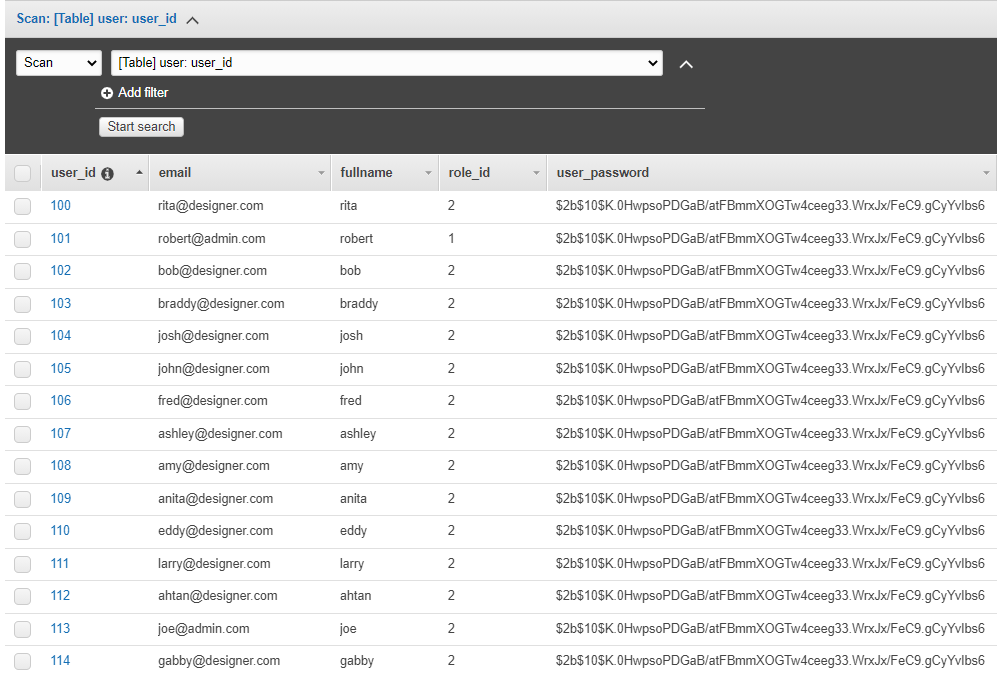
Click “Run”



If no errors has occurred, the results should look like this.



**Verify that the user data has been uploaded to DynamoDB.**

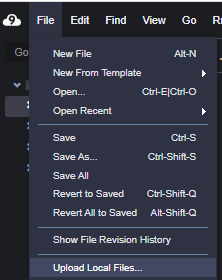


Congratulations ! You have successfully uploaded user data to DynamoDB.

### 8.4.2 Uploading files Table

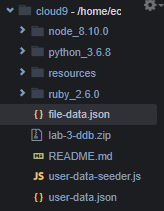
Go to the Cloud9 Environment

On the File tab, Click “Upload Local Files”



Drag & drop the file-data.json file into the uploader

After uploading, ensure that the file-data.json has been uploaded.



Create a new file named file-data-seeder.js

Insert the following codes into the js folder.

var

    AWS = require("aws-sdk"),

    DDB = new AWS.DynamoDB({

        apiVersion: "2012-08-10",

        region: "us-east-1"

    }),

    file\_DATA\_ARR = require("./file-data.json");

function addNewItemsFromJSON(){

    console.log("All items now removed, re-seeding now");

    var

        file = {},

        file\_formatted\_arr = [],

        params = {};

    for(var i\_int = 0; i\_int < file\_DATA\_ARR.length; i\_int += 1){

        file = {

            PutRequest: {

                Item: {

                    file\_id: {

                        "N": file\_DATA\_ARR[i\_int].file\_id.toString()

                    },

                    cloudinary\_file\_id: {

                        "S": file\_DATA\_ARR[i\_int].cloudinary\_file\_id

                    },

                    cloudinary\_url: {

                        "S": file\_DATA\_ARR[i\_int].cloudinary\_url

                    },

                    design\_title: {

                        "S": file\_DATA\_ARR[i\_int].design\_title

                    },

                    design\_description: {

                        "S": file\_DATA\_ARR[i\_int].design\_description

                    },

                    created\_by\_id: {

                        "N": file\_DATA\_ARR[i\_int].created\_by\_id.toString()

                    }

                }

            }

        };

        file\_formatted\_arr.push(file);

    }

    params = {

        RequestItems: {

            "files": file\_formatted\_arr.reverse()

        }

    };

    DDB.batchWriteItem(params, function(err, data){

        if(err){

            throw err;

        }

        console.log("OK");

    });

}

(function init(){

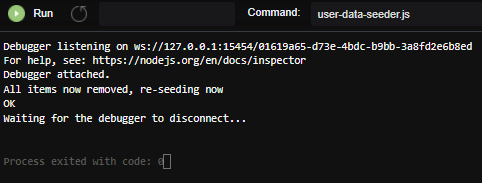
    addNewItemsFromJSON();

})();

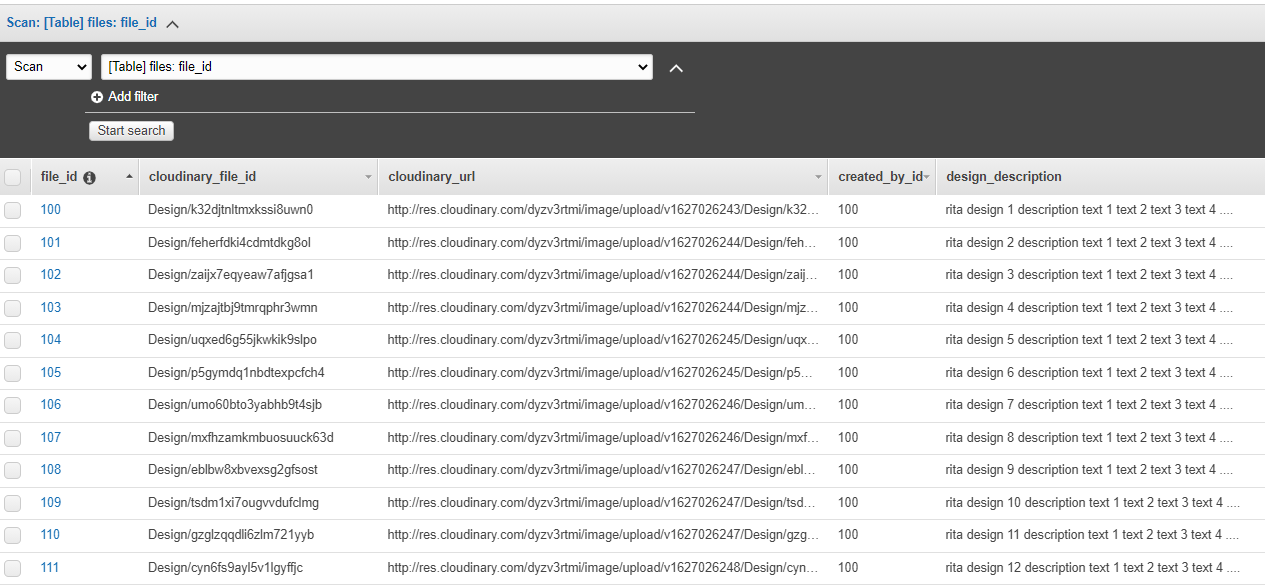
Click “Run”



If no errors has occurred, the results should look like this.



**Verify that the file data has been uploaded to DynamoDB.**

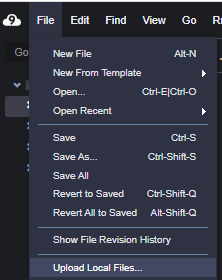


Congratulations ! You have successfully uploaded file data to DynamoDB.

### 8.4.3 Uploading role Table

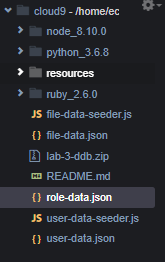
Go to the Cloud9 Environment

On the File tab, Click “Upload Local Files”



Drag & drop the role-data.json file into the uploader

After uploading, ensure that the role-data.json has been uploaded.



Create a new file named role-data-seeder.js

Insert the following codes into the js folder.

var

AWS = require("aws-sdk"),

DDB = new AWS.DynamoDB({

    apiVersion: "2012-08-10",

    region: "us-east-1"

}),

file\_DATA\_ARR = require("./role-data.json");

function addNewItemsFromJSON(){

console.log("All items now removed, re-seeding now");

var

    file = {},

    file\_formatted\_arr = [],

    params = {};

for(var i\_int = 0; i\_int < file\_DATA\_ARR.length; i\_int += 1){

    file = {

        PutRequest: {

            Item: {

                role\_id: {

                    "N": file\_DATA\_ARR[i\_int].role\_id.toString()

                },

                role\_name: {

                    "S": file\_DATA\_ARR[i\_int].role\_name

                }

            }

        }

    };

    file\_formatted\_arr.push(file);

}

params = {

    RequestItems: {

        "role": file\_formatted\_arr.reverse()

    }

};

DDB.batchWriteItem(params, function(err, data){

    if(err){

        throw err;

    }

    console.log("OK");

});

}

(function init(){

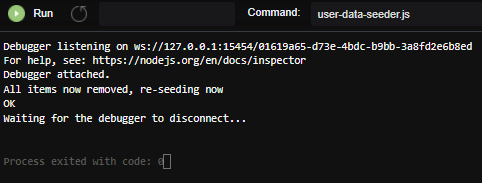
addNewItemsFromJSON();

})();

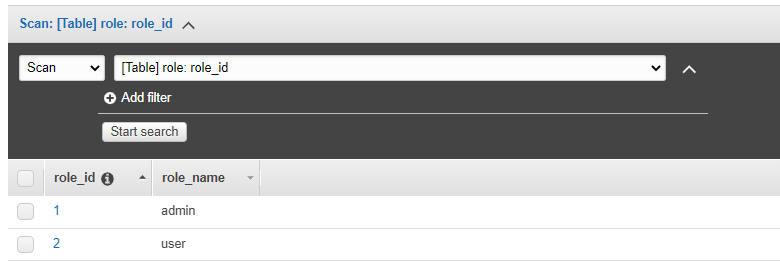
Click “Run”



If no errors has occurred, the results should look like this.



**Verify that the file data has been uploaded to DynamoDB.**



Congratulations ! You have successfully uploaded role data to DynamoDB.

# 9. AWS CloudWatch

|  |
| --- |
| **What is AWS CloudWatch ?**  Amazon CloudWatch is a monitoring and an observability service built for DevOps engineers, developers, site reliability engineers (SREs) and IT managers.  **What does it do ?**  It provides you with data and actionable insights to monitor your applications, responds to performance changes, optimizes resource utilization, and get unified view of operational health. It collects monitoring and operational data in the form of logs, metrics, and events, providing unified view of AWS resources.  **How can it be used ?**  It can be used to detect anomalous behaviour in your environments, set alarms, visualize logs and metrics side by side. |

## 9.1 Centralized logging for Web API Access

We will be setting up CloudWatch logging for REST API in API Gateway to help debug issues related to request execution or client access to your API.

### 9.1.1 Create an IAM role for logging to CloudWatch

In IAM console sidebar, select “**Roles**”

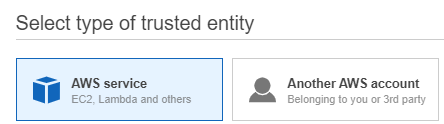


Select “**Create role**”

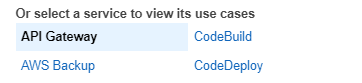


On the **Create role** page, do the following:

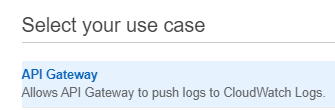
For **Select type of trusted entity**, choose **AWS Service**.



For **Choose a use case**, choose **API Gateway**.



For **Select your use case**, choose **API Gateway**.



Choose **Next: Permissions**.

Leave the permissions as it is and choose **Next: Tags**.

Choose **Next: Review**

Under Reviews, do the following:

For **Role name**, enter “API-Gateway-Logs”

Select “Create role”



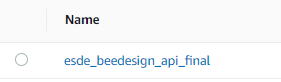
On the **Roles** pane, look for the newly created role “**API-Gateway-Logs**”.

On the **Summary** pane, copy the **Role ARN** on a notepad.

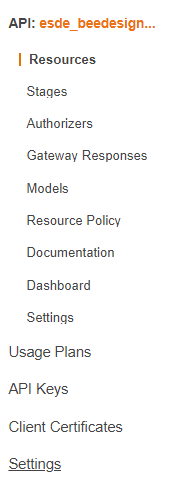


### 9.1.2 Adding the IAM role in the API Gateway console

In the API Gateway console, choose “esde\_beedesign\_api\_final”.

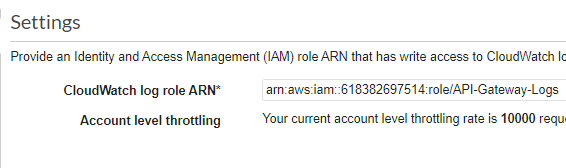


On the sidebar, choose **Settings** under **Client Certificates**



Under **Settings**, for **CloudWatch log role ARN**, paste the **IAM role ARN** that you copied to the notepad.

This has to be done to provide the API with IAM role ARN with write access to the CloudWatch logs.



Choose **Save**.



### 9.1.3 Turn on logging for API and stage

In the **API Gateway console**, go to **Stages**.



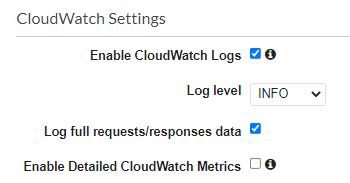
Choose the stage named “**test**”

A test **Stage Editor** pane will appear, choose the **Logs/Tracing** tab

Under **CloudWatch** **Settings**, check **Enable** **CloudWatch Logs.**

For **Log level**, choose “**INFO**”

Check **Log full requests/responses data**



Under **Custom Access Logging**:

Check **Enable Access Logging.**

For **Access Log Destination ARN**, input “arn:aws:logs:us-east-1:618382697514:log-group:API-Gateway-Execution-Logs\_y2d1tebz53/test”

This is the Log Format that will appear on the CloudWatch Log group.

**Log Format**:

{ "requestId":"$context.requestId", "ip": "$context.identity.sourceIp", "caller":"$context.identity.caller", "user":"$context.identity.user","requestTime":"$context.requestTime", "httpMethod":"$context.httpMethod","resourcePath":"$context.resourcePath", "status":"$context.status","protocol":"$context.protocol", "responseLength":"$context.responseLength" }

Choose “Save Changes”



### 9.1.4 Test the logging setup

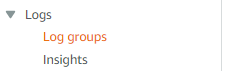
Go to the beedesign competition website,

<https://3.92.106.130:3001/user/manage_submission.html>

Click on “search”

When the API has been invoked, Go to **CloudWatch** console.

Under Logs, choose Log groups

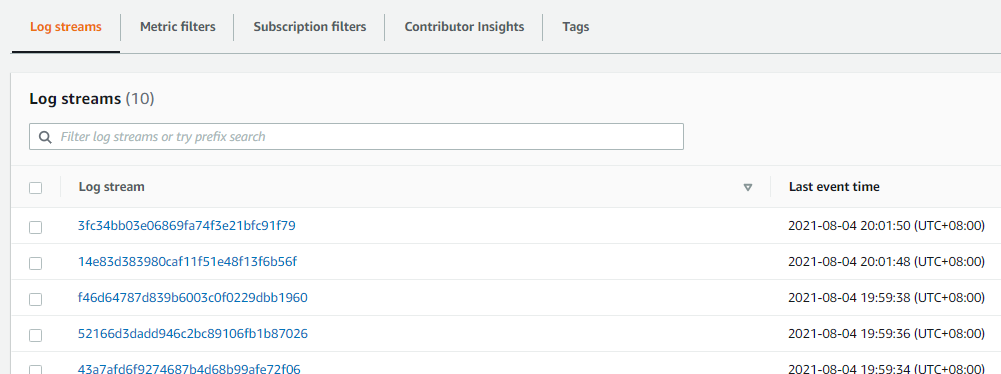


Choose the automatically created API Log group named “[API-Gateway-Execution-Logs\_y2d1tebz53/test](https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#logsV2:log-groups/log-group/API-Gateway-Execution-Logs_y2d1tebz53$252Ftest)”

**Verify that the API Gateway Logging is working**

Under Log streams, if there are logs streams generated.

Click into one of the log streams.



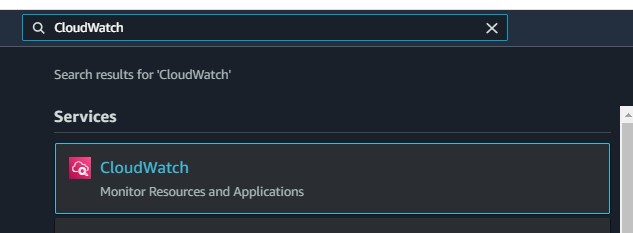
Text

Description automatically generated

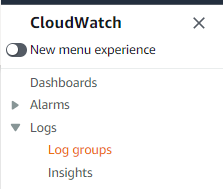
Congratulations ! You have successfully created logging for the API gateway.

## 9.2 Viewing Lambda Function CloudWatch logs

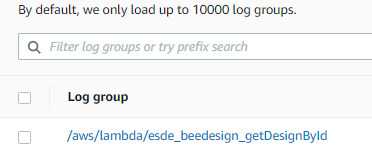
Go to the search bar and search for CloudWatch.



On the sidebar, Go to “Log groups”.



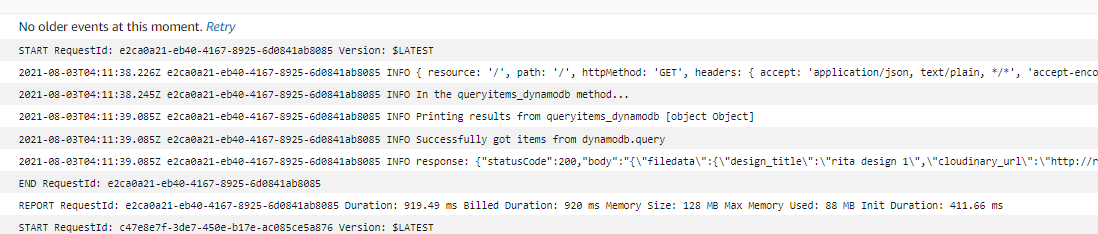
Select “/aws/lambda/esde\_beedesign\_getDesignById”



Under “Log streams”

Click on any of the logs.

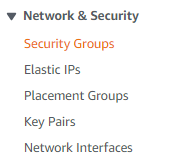
You would be able to see the logs of the Lambda Function “esde\_beedesign\_getDesignById”



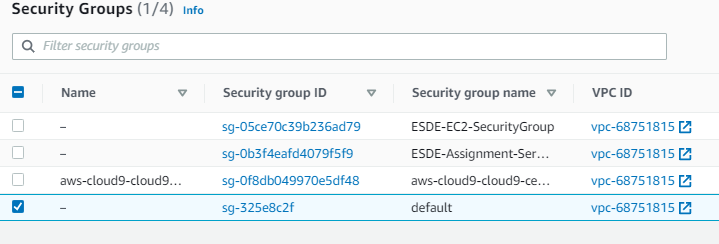
# 10. MySQL RDS Database security

## 10.1 Ensure only EC2 Instance running the web application can have access to RDS

On EC2 Dashboard sidebar, Under **Network & Security**, choose **Security Groups.**



Click on the **Security Group** with the **group name** as “**default**”.

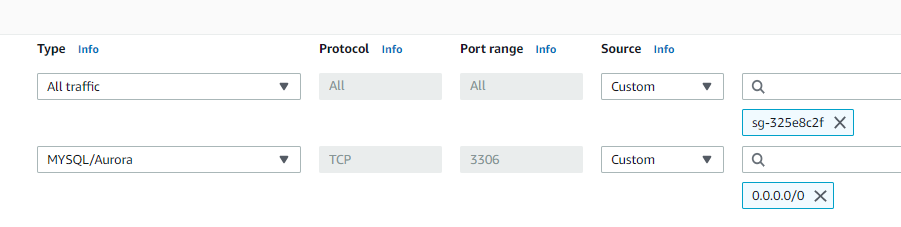


Under the **Inbound rules**, choose **Edit inbound rules.**



Under the inbound rule **type** MySQL/Aurora.

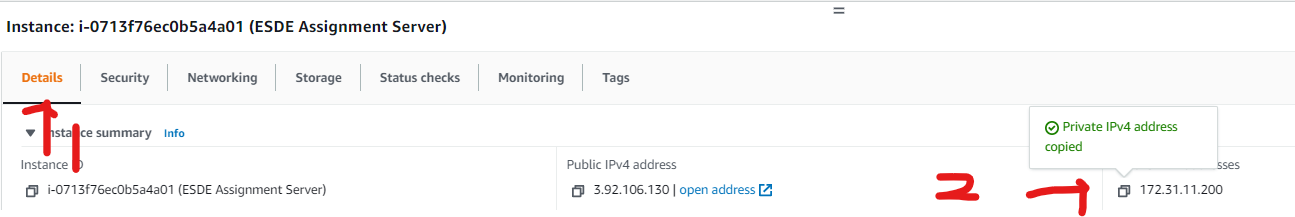
Change the source from 0.0.0.0/0 which is anywhere, to the **Private IP** of the EC2 Instance running the web application.



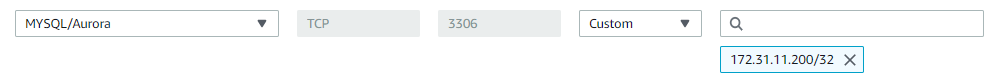
On the **EC2 Dashboard**, choose **Instances**.

Choose the instance that you have attached the Elastic IP to.

Under the **details** tab, copy the **Private IPv4 Address**.



Return back to the Security Group, remove the source “0.0.0.0/0” to “172.31.11.200/32”



Choose **Save rules**



### 10.1.1 Verify that only the EC2 Instance can connect to the database.

Open up MySQL Workbench.

Log into the “AWS-EC2-Encrypted-Database”.

Graphical user interface, text, application

Description automatically generated

You should be getting an error as shown.

Graphical user interface, text, application

Description automatically generated

**Verify that only the EC2 Instance can be connected to the Database.**

Open up the website.

Go to <https://3.92.106.130:3001/login.html>

Attempt to login as [rita@designer.com](mailto:rita@designer.com)

With the password as password.

Graphical user interface, application, Teams

Description automatically generated

If you managed to login, Congratulations ! You have successfully secured the RDS database !

## 10.2 Encrypting RDS Database Snapshot using KMS keys

**What is RDS Database Encryption ?**

|  |
| --- |
| Amazon RDS encrypted DB instances provide an additional layer of data protection by securing your data from unauthorized access to the underlying storage. |

On the Amazon RDS console sidebar, choose **Databases**.

Under the **databases**, select “database-1”.

Graphical user interface, text, application

Description automatically generated

Select **Actions**, then **Take Snapshot**.

Graphical user interface, application

Description automatically generated

On the **Snapshot name**, type in “encryptedsnapshot”. Graphical user interface, text, application, email

Description automatically generated

Select **Take snapshot**.



A snapshot would be created.

Return back to the AWS RDS console,

On the sidebar, choose **Snapshots**.

Graphical user interface, application

Description automatically generated

Under the **Manual snapshots**, select “**encryptedsnapshot**”.

Graphical user interface, text, application

Description automatically generated

Select **Actions**, then choose **Restore snapshot**.

Graphical user interface, text, application

Description automatically generated

Under **Settings**,

For **DB instance identifier**, enter “**encrypted-rds-database**”

Graphical user interface, text, application, email

Description automatically generated

Under **Connectivity**,

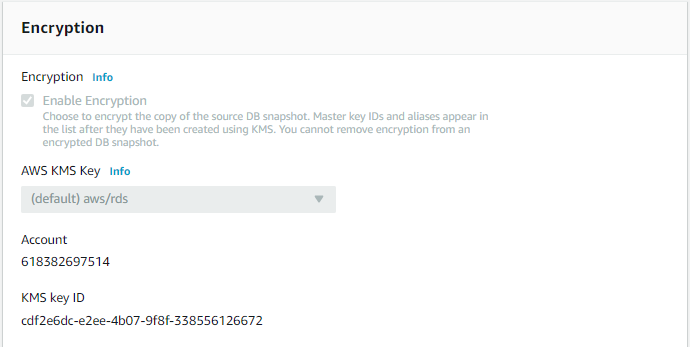
For **Public access**, select **Yes**.

Graphical user interface, text, application, email

Description automatically generated

Under **Encryption**,

Ensure that the **Enable Encryption** checkbox is selected.



Choose **Restore DB instance**.

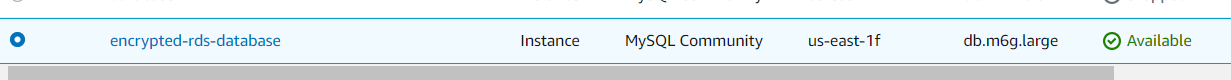


After restoring the snapshot, Go to **Databases**.

Graphical user interface, text, application, email

Description automatically generated

Verify that the snapshot has been restored into a Database.



Congratulations ! You have successfully encrypted the RDS Database !

## 10.3 Connecting to Encrypted RDS Database.

On **Amazon RDS** console sidebar, choose **Databases**.

Graphical user interface, text, application, email

Description automatically generated

Click on the “encrypted-rds-database”.

Under Connectivity & security,

Copy the Endpoint into a notepad.

Graphical user interface, text, application

Description automatically generated

In WinSCP, Go to “/home/ubuntu/esde/be/src/config”.

Double-click database.js

Change the host: endpoint to

“host:'encrypted-rds-database.cativmsladsx.us-east-1.rds.amazonaws.com'”

Graphical user interface, text, application

Description automatically generated

Save the file and refresh the website.

This has to be done to direct requests to the database.

Verify that the EC2 Instance is connected to the encrypted RDS database.

Go to <https://3.92.106.130:3001/login.html>

Login as [rita@designer.com](mailto:rita@designer.com)

Password: password

If the login is successful, Congratulations ! You have successfully connected to the encrypted RDS database !

## 10.4 Publishing MySQL logs to CloudWatch - Database logging

Configure your MySQL RDS Database to publish log data to log group in Amazon CloudWatch Logs.

On Amazon RDS console sidebar, go to Databases

Graphical user interface, text, application, email

Description automatically generated

Select “**encrypted-rds-database**” Instance.

Graphical user interface, application

Description automatically generated

Select **Modify**.



Under **Additional configuration**,

For **Log exports**, select **Audit log**, **Error log**, **General log**, and **slow query log** checkbox.

Graphical user interface, text, application, chat or text message

Description automatically generated

Select **Continue**.



If the modification is successful, this message below will be shown.



After enabling the Log exports, a Log group will be generated.

### 10.4.1 Change value of log\_output parameter

Go to RDS console sidebar, select **Parameter groups**.

Select **Create parameter group**.

Name the **Group name** as “**saveme**”

Enter “Save” for **Description**.

Select **Create**.



After creating the **Parameter groups**, click on the newly created **Parameter groups**.

Graphical user interface, application

Description automatically generated

Under **Parameter group** actions, select **Edit**.

Under the **Parameters**, search for “**log\_output**”.

Change the **Values** from **TABLE** to **FILE**.

Graphical user interface, text, application, email

Description automatically generated

Click on Save Changes



### 10.4.2 Verifying logs has been logged

Go to **CloudWatch** console, under **Logs** go to **Log Groups**.

Graphical user interface, text, application

Description automatically generated

You would see that a new Log group has been generated.



Click into the Log group.

You can now see the database access loggings.

Graphical user interface, text, application, email

Description automatically generated

Congratulations ! You have successfully created a database logging !

## 10.5 Lease Privilege for Database credentials

Open up MySQL Workbench and connect to the RDS database.

Under the Navigator, go to Administration.

Graphical user interface, text, application

Description automatically generated

Select **Users and Privileges**.

A list of User Accounts in the database will be shown.

Select “adminuser” (This is the user that our AWS EC2 Instance is using.)

Graphical user interface, text

Description automatically generated A picture containing text

Description automatically generated

Under **Details for account adminuser@%**.

Select **Schema Privileges**.

This is the Privileges that the adminuser has.

Graphical user interface, text

Description automatically generated

Now our objective is the give the user the least privileges.

Select **Unselect All**.

Select **everything** under **Object Rights**.

Graphical user interface, text, application

Description automatically generated

Select **Apply**.



After applying the permissions.

If it is successful, you will be able to see the privileges as show below.

Graphical user interface, text, application

Description automatically generated

Congratulations ! You have successfully given the least permission needed for the EC2 Instance to function !

# 11. CloudTrail

|  |
| --- |
| **What is AWS CloudTrail ?**  AWS Cloud Trail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account.  **What does it do ?**  CloudTrail can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. It provides event history of your AWS account’s activity, including actions taken through the AWS Management Console and many more. |

## 11.1 Creating CloudTrail

On **AWS CloudTrail**, select **Create trail**.



Under **Quick trail create**,

Enter “esde-management-events” as **Trail name**.

Graphical user interface, text, application, email

Description automatically generated

Select **Create trail**.



After creating the trail, click the newly created cloud trail.

Graphical user interface, text, application

Description automatically generated

## 11.2 Viewing AWS CloudTrail logs

Under **General details**.

Select **Trail log location** link.

<https://console.aws.amazon.com/s3/buckets/aws-cloudtrail-logs-618382697514-44c80091/AWSLogs/618382697514/?region=us-east-1>

Graphical user interface, text, application, Word

Description automatically generated

It would direct you to the S3 bucket for the cloudtrail logs.

Graphical user interface, text, application, email

Description automatically generated

Under Objects,

Go into AWSLogs/618382697514/CloudTrail/us-east-1/2021/08/05

Select the latest log file.

On the top right, select **Open**.



When you click **Open**, it will open up a page with all of the logs.

However, it is difficult to look at the logs as you can see all the logs are clamped together.

So, we would need to create CloudWatch log group to be able to look at the logs easily.

Text

Description automatically generated

## 11.3 Creating CloudWatch Logs for CloudTrail

After clicking on the newly created the cloud trail.

On the right side of CloudWatch Logs.

Select **Edit**.

Graphical user interface, text, application

Description automatically generated

Under **CloudWatch Logs**.

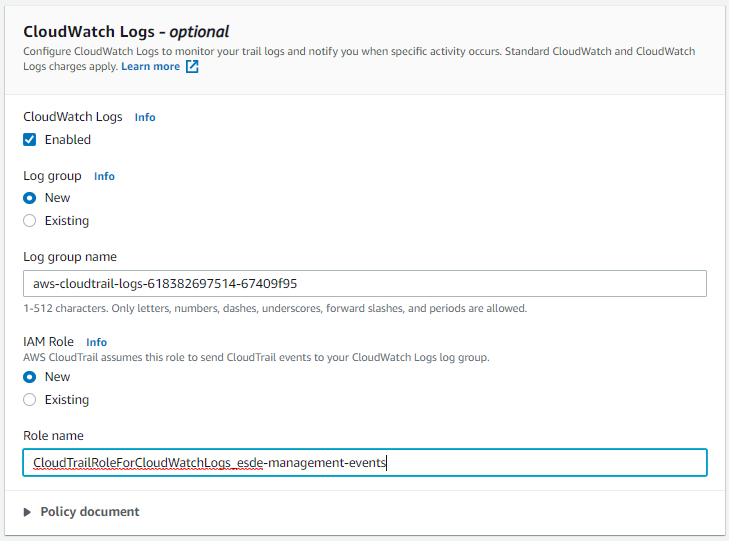
Check the **CloudWatch Logs** checkbox to **Enabled**.

For **Log group**, select **New**.

A Log group name would be generated for you.

For **IAM Role**, select **New**.

For **Role name**, enter “**CloudTrailRoleForCloudWatchLogs\_esde-management-events**”



Select **Save changes**.



### 11.3.1 Verify that CloudWatch logs has been created for CloudTrail

On **AWS CloudWatch** **console** sidebar, go to **Log groups**.

Graphical user interface, application

Description automatically generated

Under **Log groups**.

Look for the newly created cloudtrail log group.

Click on it.



You will be able to see that a log has been created under **Log streams**.

Click on the log stream.

Graphical user interface, text, application

Description automatically generated

You will be able to see the Log events that’s happening in the cloud trail.

Graphical user interface, application

Description automatically generated

Now you will be able to see the logs easily through the use of CloudWatch.

Congratulations ! You have successfully created a CloudWatch log for CloudTrail !

# 12. getProfile AWS Lambda Function

## 12.1 Create getProfile Lambda Function

The getProfile Lambda Functions gets the profile details from DynamoDB.

It queries the DynamoDB database to look for the requested userid that matches the ones in the database.

On AWS Lambda Dashboard, select **Create function**.



Under **Create function**.

Select **Author from scratch**.

Graphical user interface, text

Description automatically generated with medium confidence

Under **Basic information**.

For Function name, enter “esde\_beedesign\_getProfile”.

Select **Create function**.



After creating the function, click into it.



## 12.2 getProfile source code

**Index.js**

In index.js, insert the following codes.

var dynamodbQuery = require('dynamodb/query\_dynamodb');

exports.handler = async (event, context, callback) => {

    console.log(event)

        var userId = parseInt(event.headers['user'])

        var region = "us-east-1";

        var table\_name = "user";

        var proj\_expr = "user\_id,email,role\_id,fullname"

        var expr\_attr\_values = { ":userid": userId }

        var key\_cond\_expr = "user\_id=:userid"

        await dynamodbQuery(region, table\_name,expr\_attr\_values,key\_cond\_expr,proj\_expr)

        .then(data => {

            console.log(data)

                    console.log("Successfully got items from dynamodb.query")

                    var responseCode = 200;

                    var jsonResult = {'userdata': data.Items}

                    let response = {

                            statusCode: responseCode,

                            body: JSON.stringify(jsonResult),

                            headers: {

                                "Access-Control-Allow-Headers" : "Content-Type,user",

                                "Access-Control-Allow-Origin": "\*",

                                "Access-Control-Allow-Methods": "OPTIONS,POST,GET"

                            }

                    }

                    console.log("response: " + JSON.stringify(response))

                    callback(null, response);

            })

            .catch(error => {

                    console.log('There has been a problem with your fetch operation: ' + error.message);

                    var responseCode = 500;

                    let response = {

                        statusCode: responseCode,

                        body: JSON.stringify(error)

                    }

                    console.log("response: " + JSON.stringify(response))

                    callback(null, response);

             });

};

Under the main folder, create a folder named “dynamodb”.

Inside the dynamodb folder, create a file named “query\_dynamodb.js”.

Graphical user interface, text, application, Word

Description automatically generated

**query dynamodb.js**

Insert the following code into query\_dynamodb.js.

var AWS = require('aws-sdk');

async function queryitems\_dynamodb(region, table\_name,expr\_attr\_values,key\_cond\_expr,proj\_expr) {

    console.log("In the queryitems\_dynamodb method...")

    var dynamodb = new AWS.DynamoDB.DocumentClient({region: region});

    try{

        var params = {  TableName: table\_name,

                        ExpressionAttributeValues: expr\_attr\_values,

                        KeyConditionExpression: key\_cond\_expr ,

                        ProjectionExpression: proj\_expr}

        const results = await dynamodb.query(params).promise()

        console.log("Printing results from  queryitems\_dynamodb " + results)

        return results;

    }

    catch(tryerror) {

        console.log("Error occurred in dynamodb.query..")

        console.log(tryerror, tryerror.stack); // an error occurred

    }

} //end function

module.exports = queryitems\_dynamodb

## 12.3 Setting up getProfile API Gateway

On **AWS API Gateway console**,

On the sidebar, go to **Resources**.

Graphical user interface

Description automatically generated with medium confidence

Under **Actions**, select **Create Resource**.

Text

Description automatically generated

A **New Child Resource** pane will appear.

Under **New Child Resource** :

For **Resource Name\***, enter “**getprofile**”

Select **Enable API Gateway CORS** checkbox.

Graphical user interface, text, application, email

Description automatically generated

Select **Create Resource**.

A new resource named /getprofile will be created.

Click on the /getprofile resource.



Under **Actions**, select **Create Method**.

A picture containing graphical user interface

Description automatically generated

A dropdown menu will appear.

Select a **GET** method and click on the **tick**.



A Setup pane will appear.

For **Integration type**, select **Lambda Function**.

Select **Use Lambda Proxy integration** checkbox.

For **Lambda Function**, enter “**esde\_beedesign\_getProfile”**

Graphical user interface, text, application

Description automatically generated

Select Save.



A **Method Execution** pane will be created.

**Method Request**

On the **Method Execution** pane, Select **Method Request**.

Graphical user interface, text, application, email

Description automatically generated

Select on the **HTTP Request Headers** section.

Select **Add header**.

Enter “**Access-Control-Allow-Headers**” and click on the **Tick**.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Integration Request**

On **Method Execution** pane, select **Integration Request**.

For **Execution role**, enter “**arn:aws:iam::618382697514:role/service-role/esde\_beedesign\_getProfile-role-5xpr16vv**”

Select the **Tick**.

**Method Response**

Under **Method Execution**, select **Method Response**.

Graphical user interface, text, application, email

Description automatically generated

Under **Method Response**, click on the **arrow** sign.

Under **Response Headers for 200**, select **Add** **Header**.

Enter “**Access-Control-Allow-Origin**” click on the **Tick**.

Graphical user interface, text, application

Description automatically generated

Under **Actions**, select **Enable CORS**.

Graphical user interface, text, application

Description automatically generated

Select **DEFAULT 4xx** and **DEFAULT 5xx** checkbox.

For **Access-Control-Allow-Headers**, enter “**'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token,user**'”

Select **Enable CORS and replace existing headers**.



A picture containing calendar

Description automatically generated

**Deploy API**

Under **Actions**, select **Deploy API**.

Graphical user interface, application

Description automatically generated

Under **Deploy API**:

For **Deployment stage**, select **test** and click on **Deploy**.

Graphical user interface, text, application, email

Description automatically generated

Once deployed, you will be able to see the new resource under **Stages**.

Graphical user interface

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## 12.4 Setting IAM role permissions

We will need to grant permissions to the **IAM** **roles** for it to execute required tasks.

On **IAM Dashboard** sidebar, select **Roles**.

Select the “[**esde\_beedesign\_getProfile-role-5xpr16vv**](https://console.aws.amazon.com/iam/home#/roles/esde_beedesign_getProfile-role-5xpr16vv)” **IAM role**.



On the IAM role **Summary**, under **Permissions**.

Select **Attach policies**.

Under **Attach** **policies**, search for “**AWSLambdaDynamoDBExecutionRole**” policy.



And also search for “**AWSLambdaInvocation-DynamoDB**” policy.



Select **Attach policy**.



**Trust relationships**

Under **Summary**, select **Trust relationships**.

Select **Edit** **trust** **relationship**.



Copy and paste the codes highlighted in yellow below.

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": [

"lambda.amazonaws.com",

"apigateway.amazonaws.com",

"dynamodb.amazonaws.com"

]

},

"Action": "sts:AssumeRole"

}

]

}

Select **Update Trust Policy**.



## 12.5 Test getProfile Lambda Function on Postmen

Open up Postmen.

Set the request method to **GET**.

Set the url to <https://y2d1tebz53.execute-api.us-east-1.amazonaws.com/test/getprofile>

Go to Headers, Insert a Key “user” with a value of 100.

A screen shot of a computer

Description automatically generated with low confidence

Click **Send**.

A blue screen with white text

Description automatically generated with low confidence

Verify that the Lambda function works in Postmen.

A screenshot of a computer

Description automatically generated

Congratulations ! This Lambda Function successfully retrieved a record from DynamoDB !

## 12.6 Test that the Function works for the website.

Open **WinSCP**, go to **“/home/ubuntu/esde/fe/public/js**”.

Double-click **profile.js**.

Under the function getOneUser(), change the axios url to “'https://y2d1tebz53.execute-api.us-east-1.amazonaws.com/test/getprofile'”.

Text, letter

Description automatically generated

Save the folder.

Open up a web browser.

Log in as [rita@designer.com](mailto:rita@designer.com) with the password as password.

Graphical user interface, text, application, website

Description automatically generated

When logged in, on the top right.

Go to **My Profile**.

If the profile data is able to retrieve the user data.

Graphical user interface, text, application, chat or text message

Description automatically generated

Congratulations ! This Lambda function is working as intended !

# 13. updateDesigns AWS Lambda Function

## 13.1 Create updateDesigns Lambda Function

The updateDesigns Lambda Functions updates the submitted designs in DynamoDB.

It updates the DynamoDB database record.

On AWS Lambda Dashboard, select **Create function**.



Under **Create function**.

Select **Author from scratch**.

Graphical user interface, text

Description automatically generated with medium confidence

Under **Basic information**.

For Function name, enter “esde\_beedesign\_updateDesigns”.

Select **Create function**.



After creating the function, click into it.



## 13.2 updateDesigns source code

**Index.js**

In index.js, insert the following codes.

Insert the following codes:

var dynamodbUpdate = require('dynamodb/update\_dynamodb');

exports.handler = async function(event, context, callback){

    if (event.fid|| (event.queryStringParameters && event.queryStringParameters.fid)) {

        if (event.fid)

            var fileId = parseInt(event.fid);

        else

            var fileId= parseInt(event.queryStringParameters.fid);

            var desn\_description = event.queryStringParameters.designDescription;

            var desn\_title = event.queryStringParameters.designTitle;

        console.log(event)

        var region = "us-east-1"

        var table\_name = "files"

        var expr\_attr\_values = fileId

        var key\_cond\_expr = "file\_id=:fileid"

        var proj\_expr = "file\_id,cloudinary\_url,design\_title,design\_description"

        const result = await dynamodbUpdate(region, table\_name,expr\_attr\_values,key\_cond\_expr,proj\_expr,desn\_description,desn\_title)

        .then(data => {

            console.log(data)

            if (data) {

                    var responseCode = 200;

                    let response = {

                            statusCode: responseCode,

                            body: JSON.stringify("Successful Updated Design !"),

                            headers: {

                                "Access-Control-Allow-Headers" : "Content-Type,user",

                                "Access-Control-Allow-Origin": "\*",

                                "Access-Control-Allow-Methods": "OPTIONS,POST,GET"

                            },

                            "isBase64Encoded": true

                    }

                    console.log("response: " + JSON.stringify(response))

                    callback(null, response);

            }

                    var responseCode = 500;

                    let response = {

                            statusCode: responseCode,

                            body: JSON.stringify("Failed to Update Design"),

                            headers: {

                                "Access-Control-Allow-Headers" : "Content-Type,user",

                                "Access-Control-Allow-Origin": "\*",

                                "Access-Control-Allow-Methods": "OPTIONS,POST,GET"

                            },

                            "isBase64Encoded": true

                    }

                    console.log("response: " + JSON.stringify(response))

                    callback(null, response);

            })

            .catch(error => {

                    console.log(error)

                    console.log('Error: ' + error.message);

                    var responseCode = 500;

                    let response = {

                        statusCode: responseCode,

                        body: JSON.stringify(error)

                    }

                    console.log("response: " + JSON.stringify(response))

                    callback(null, response);

             });

    } //end if

}

Under the main folder, create a folder named “dynamodb”.

Inside the dynamodb folder, create a file named “update\_dynamodb.js”.

Graphical user interface, application, Word

Description automatically generated

**update\_dynamodb.js**

Insert the following code into update\_dynamodb.js.

var AWS = require('aws-sdk');

async function updateitems\_dynamodb(region, table\_name,expr\_attr\_values,key\_cond\_expr,proj\_expr,desn\_description,desn\_title) {

    console.log("In the queryitems\_dynamodb method...")

    var dynamodb = new AWS.DynamoDB({region: region});

    try{

    var params = {

    Key:{

        "file\_id": {

            N: String(expr\_attr\_values)

        }

    },

    ExpressionAttributeValues:{

        ":b": {

            "S": desn\_description

        },

        ":c": {

            "S": desn\_title

        },

    },

    ReturnValues:"UPDATED\_NEW",

    TableName: "files",

    UpdateExpression: "SET design\_description = :b, design\_title = :c"

};

    return dynamodb.updateItem(params, function(err, data) {

        console.log(err, data);

    });

    } catch(tryerror) {

        console.log("Error occurred!")

        console.log(tryerror, tryerror.stack); // an error occurred

    }

} //end function

module.exports = updateitems\_dynamodb

## 13.3 Setting up updateDesign API Gateway

On **AWS API Gateway console**,

On the sidebar, go to **Resources**.

Graphical user interface

Description automatically generated with medium confidence

Click on the / resource.



Under **Actions**, select **Create Method**.

A picture containing text

Description automatically generated

A dropdown menu will appear.

Select a **PUT** method and click on the **tick**.



A Setup pane will appear.

For **Integration type**, select **Lambda Function**.

Select **Use Lambda Proxy integration** checkbox.

For **Lambda Function**, enter “**esde\_beedesign\_updateDesigns”**

Graphical user interface, application

Description automatically generated

Select Save.



A **Method Execution** pane will be created.

**Method Request**

On the **Method Execution** pane, Select **Method Request**.

Graphical user interface, text, application, email

Description automatically generated

Select on the **HTTP Request Headers** section.

Select **Add header**.

Enter “**Access-Control-Allow-Origin**” and click on the **Tick**.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Integration Request**

On **Method Execution** pane, select **Integration Request**.

For **Execution role**, enter “**arn:aws:iam::618382697514:role/service-role/esde\_beedesign\_updateDesigns-role-z9rsn2ku**”

Select the **Tick**.

**Method Response**

Under **Method Execution**, select **Method Response**.

Graphical user interface, text, application, email

Description automatically generated

Under **Method Response**, click on the **arrow** sign.

Under **Response Headers for 200**, select **Add** **Header**.

Enter “**Access-Control-Allow-Origin**” click on the **Tick**.

Graphical user interface, text, application

Description automatically generated

Under **Actions**, select **Enable CORS**.

Graphical user interface, text, application

Description automatically generated

Select **DEFAULT 4xx** and **DEFAULT 5xx** checkbox.

For **Access-Control-Allow-Headers**, enter “**'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token,user**'”

Select **Enable CORS and replace existing headers**.



A picture containing calendar

Description automatically generated

**Deploy API**

Under **Actions**, select **Deploy API**.

Graphical user interface, application

Description automatically generated

Under **Deploy API**:

For **Deployment stage**, select **test** and click on **Deploy**.

Graphical user interface, text, application, email

Description automatically generated

Once deployed, you will be able to see the new resource under **Stages**.

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

## 13.4 Setting IAM role permissions

We will need to grant permissions to the **IAM** **roles** for it to execute required tasks.

On **IAM Dashboard** sidebar, select **Roles**.

Select the “[**esde\_beedesign\_updateDesigns-role-z9rsn2ku**](https://console.aws.amazon.com/iam/home#/roles/esde_beedesign_updateDesigns-role-z9rsn2ku)” **IAM role**.



On the IAM role **Summary**, under **Permissions**.

Select **Attach policies**.

Under **Attach** **policies**, search for “**AmazonDynamoDBFullAccess**” policy.



And also search for “**AWSLambda\_FullAccess**” policy.



Lastly, search for “**AmazonAPIGatewayInvokeFullAccess**” policy.



Select **Attach policy**.



**Trust relationships**

Under **Summary**, select **Trust relationships**.

Select **Edit** **trust** **relationship**.



Copy and paste the codes highlighted in yellow below.

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": [

"lambda.amazonaws.com",

"apigateway.amazonaws.com",

"dynamodb.amazonaws.com"

]

},

"Action": "sts:AssumeRole"

}

]

}

Select **Update Trust Policy**.



## 13.5 Verify that the Lambda Function works.

Open up WinSCP.

Go into “/home/ubuntu/esde/fe/public/js”.

Double-click on update\_design.js

Modify the url under axios to “'https://y2d1tebz53.execute-api.us-east-1.amazonaws.com/test?fid='+ fileId + '&designTitle=' + designTitle +'&designDescription=' + designDescription”

Save the folder.

Refresh the website.

This is the original design details for file id 101.

Graphical user interface, application

Description automatically generated

Change the Design title to “Title Changed”.

Change the Design description to “Description Changed”.

Select **Submit**.

If a green pop up as shown below appears.

Graphical user interface

Description automatically generated with low confidence

Congratulations ! esde\_beedesign\_updateDesigns Lambda function is working as intended !

Graphical user interface, application

Description automatically generated

# 14. manageSubmissions AWS Lambda Function

## 14.1 Create manageSubmissions Lambda Function

The manageSubmissions Lambda Functions gets all the record designs in DynamoDB created by the user.

On AWS Lambda Dashboard, select **Create function**.



Under **Create function**.

Select **Author from scratch**.

Graphical user interface, text

Description automatically generated with medium confidence

Under **Basic information**.

For Function name, enter “esde\_beedesign\_manageSubmissions”.

Select **Create function**.



After creating the function, click into it.



## 14.2 manageSubmissions source code

**Index.js**

In index.js, insert the following codes.

var dynamodbQuery = require('dynamodb/query\_dynamodb');

exports.handler = async function(event, context, callback){

    if (event.fid|| (event.queryStringParameters && event.queryStringParameters.fid)) {

        if (event.fid)

            var fileId = parseInt(event.fid)

        else

            var fileId= parseInt(event.queryStringParameters.fid);

            console.log(event)

            var creator = parseInt(event.headers['user'])

            var region = "us-east-1";

            var table\_name = "files";

            var expr\_attr\_values = { ":creator": creator };

            var key\_cond\_expr = "created\_by\_id=:creator";

            var proj\_expr = "file\_id,cloudinary\_url,design\_title,design\_description,created\_by\_id";

            var pageNum = parseInt(event.pageNumber)

            await dynamodbQuery(region, table\_name,expr\_attr\_values,key\_cond\_expr,proj\_expr,creator,pageNum)

            .then(data => {

                    console.log("Successfully got items from dynamodb.query");

                    console.log(data)

                    var responseCode = 200;

                    var jsonResult = {'filedata': data.Items};

                    let response = {

                            statusCode: responseCode,

                            body: JSON.stringify(jsonResult),

                            headers: {

                                "Access-Control-Allow-Headers" : "Content-Type,user",

                                "Access-Control-Allow-Origin": "\*",

                                "Access-Control-Allow-Methods": "OPTIONS,POST,GET"

                            }

                    };

                    console.log("response: " + JSON.stringify(response));

                    callback(null, response);

            })

            .catch(error => {

                    console.log('There has been a problem with your fetch operation: ' + error.message);

                    var responseCode = 500;

                    let response = {

                        statusCode: responseCode,

                        body: JSON.stringify(error)

                    }

                    console.log("response: " + JSON.stringify(response))

                    callback(null, response);

             });

    } //end if

}

Under the main folder, create a folder named “dynamodb”.

Inside the dynamodb folder, create a file named “query\_dynamodb.js”.

Graphical user interface, application

Description automatically generated

**query\_dynamodb.js**

Insert the following code into query\_dynamodb.js.

var AWS = require('aws-sdk');

async function queryitems\_dynamodb(region, table\_name,expr\_attr\_values,key\_cond\_expr,proj\_expr,creator, pageNumber) {

    console.log("In the queryitems\_dynamodb method...");

    var dynamodb = new AWS.DynamoDB.DocumentClient({region: region});

    var index = 0;

        var count = 0;

        var limit = 4;

            async function search(err,data) {

            console.log(data)

        if (err) {

            console.log("Search Error");

        } else {

            console.log("Successfully Searched");

            console.log(params)

            data.Items.forEach(function(itemData) {

            console.log("Item :", ++count,JSON.stringify(itemData));

            });

            if (data.LastEvaluatedKey) {

                console.log("Scanning for more...");

                params.ExclusiveStartKey = data.LastEvaluatedKey;

                return dynamodb.query(params, search);

            }

        }

    }

    try {

    var

    params = {

    TableName : "files",

    IndexName : "created\_by\_id-file\_id-index",

    KeyConditionExpression: "#createdBy = :creator",

    ExpressionAttributeNames:{

        "#createdBy": "created\_by\_id"

    },

    ExpressionAttributeValues: {

        ":creator": creator

    },

    ScanIndexForward: true

    }

    const results = await dynamodb.query(params, search).promise()

    return results

    }

    catch(tryerror) {

        console.log("Error occurred in dynamodb.query..");

        console.log(tryerror, tryerror.stack); // an error occurred

    }

} //end function

module.exports = queryitems\_dynamodb;

## 14.3 Setting up managesubmissions API Gateway

On **AWS API Gateway console**,

On the sidebar, go to **Resources**.

Graphical user interface

Description automatically generated with medium confidence

Under **Actions**, select **Create Resource**.

Text

Description automatically generated

A **New Child Resource** pane will appear.

Under **New Child Resource** :

For **Resource Name\***, enter “**getalldesign**”

Select **Enable API Gateway CORS** checkbox.

Graphical user interface, text, application

Description automatically generated

Select **Create Resource**.



A new resource named /getalldesign will be created.

Click on the /getprofile resource.



Under **Actions**, select **Create Method**.

Graphical user interface, text, chat or text message

Description automatically generated

A dropdown menu will appear.

Select a **GET** method and click on the **tick**.



A Setup pane will appear.

For **Integration type**, select **Lambda Function**.

Select **Use Lambda Proxy integration** checkbox.

For **Lambda Function**, enter “**esde\_beedesign\_managesubmissions”**

Graphical user interface, text, application, chat or text message

Description automatically generated

Select Save.



A **Method Execution** pane will be created.

**Method Request**

On the **Method Execution** pane, Select **Method Request**.

Graphical user interface, text, application, email

Description automatically generated

Select on the **URL Query String Parameters**.

Select **Add query string**.

Enter “**fid**” and “**pageNumber**”.

Graphical user interface, text, application, chat or text message

Description automatically generated

Select on the **HTTP Request Headers** section.

Select **Add header**.

Enter “**Access-Control-Allow-Headers**” and click on the **Tick**.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Integration Request**

On **Method Execution** pane, select **Integration Request**.

For **Execution role**, enter “**arn:aws:iam::618382697514:role/service-role/esde\_beedesign\_managesubmissions-role-j3tyzo66**”

Select the **Tick**.

**Method Response**

Under **Method Execution**, select **Method Response**.

Graphical user interface, text, application, email

Description automatically generated

Under **Method Response**, click on the **arrow** sign.

Under **Response Headers for 200**, select **Add** **Header**.

Enter “**Access-Control-Allow-Origin**” click on the **Tick**.

Graphical user interface, text, application

Description automatically generated

Under **Actions**, select **Enable CORS**.

Text, application

Description automatically generated

Select **DEFAULT 4xx** and **DEFAULT 5xx** checkbox.

For **Access-Control-Allow-Headers**, enter “**'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token,user**'”

Select **Enable CORS and replace existing headers**.



A picture containing calendar

Description automatically generated

**Deploy API**

Under **Actions**, select **Deploy API**.

Graphical user interface, application

Description automatically generated

Under **Deploy API**:

For **Deployment stage**, select **test** and click on **Deploy**.

Graphical user interface, text, application, email

Description automatically generated

Once deployed, you will be able to see the new resource under **Stages**.



Graphical user interface, text, application

Description automatically generated

## 14.4 Setting IAM role permissions

We will need to grant permissions to the **IAM** **roles** for it to execute required tasks.

On **IAM Dashboard** sidebar, select **Roles**.

Select the “[**esde\_beedesign\_managesubmissions-role-j3tyzo66**](https://console.aws.amazon.com/iam/home#/roles/esde_beedesign_managesubmissions-role-j3tyzo66)” **IAM role**.



On the IAM role **Summary**, under **Permissions**.

Select **Attach policies**.

Under **Attach** **policies**, search for “**AmazonDynamoDBFullAccess**” policy.



And also search for “**AWSLambda\_FullAccess**” policy.



Lastly, search for “**AmazonAPIGatewayInvokeFullAccess**” policy.



Select **Attach policy**.



## 14.5 Verify that managesubmissions Lambda Function works

Open **WinSCP**.

Go into “**/home/ubuntu/esde/fe/public/js**”.

Double-click “**user\_manage\_submission.js**”

Change the url: to “**https://y2d1tebz53.execute-api.us-east-1.amazonaws.com/test/getalldesign?fid=' + userId + '&pageNumber=1**”

Graphical user interface, application, Word

Description automatically generated

Save the file.

Go to <https://3.92.106.130:3001/user/manage_submission.html>

Select **Search**.

If the results are returned, it means that the function is working.

Graphical user interface, application, Teams

Description automatically generated