

8/14/2023

ESDE CA2 Report

Bee Design Website Integrated With AWS

Name: Tan Yong Kit, Eden

StudentID: P2243605

Class: DIT/FT/2A/06

Table of Contents

Part 1 – Set up EC2 Instance on AWS	11
1.1 Creating an EC2 Instance	11
Step 1:.....	11
1.2 Configure EC2 Instance Settings.....	11
Step 1:.....	11
Step 2:.....	12
Step 3:.....	12
Step 4:.....	13
Step 5:.....	13
1.3 Creating an Elastic IP Address for your Web Server	14
Step 1:.....	14
Step 2:.....	14
Step 3:.....	15
Step 4:.....	15
Step 5:.....	16
Step 6:.....	16
1.4 Setting the Inbound Rules for our Ports 3001 and 5000.....	17
Step 1:.....	17
Step 2:.....	17
Step 3:.....	18
Step 4:.....	18
1.5 Download the following applications	19
Step 1:.....	19
1.6 Using Putty and WinSCP for our EC2 Web Server.....	19
Step 1:.....	19
Step 2:.....	20

Step 3:.....	21
Step 4:.....	21
Step 5:.....	22
Part 2 – Install NodeJS and copy Bee Design app to EC2	23
2.1 Creating the folder directories for Bee Design	23
Step 1:.....	23
Step 2:.....	23
2.2 Bringing respective Backend and Frontend Bee Design Folders to ESDE Folder	24
Step 1:.....	24
Step 2:.....	24
2.3 Set up and install node modules and NodeJS with Putty.....	25
Step 1:.....	25
Step 2:.....	25
Step 3:.....	26
Step 4:.....	26
Step 5:.....	27
Step 6:.....	28
Step 7 :.....	28
Step 8:.....	29
Step 9:.....	29
Part 3 – Set up RDS	30
3.1 Creating an AWS RDS	30
Step 1:.....	30
Step 2:.....	31
Step 3:.....	32
Step 4:.....	33
Step 5:.....	34

Step 6:.....	35
3.2 Configure Inbound rules for database.....	35
Step 1:.....	35
Step 2:.....	36
Step 3:.....	36
3.3 Setting up MySQL Workbench for seeding of data.....	37
Step 1:.....	37
Step 2:.....	37
Step 3:.....	38
Step 4:.....	39
3.4 Running the SQL Script in the connection to create Database etc.....	39
Step 1:.....	39
Step 2:.....	40
Step 3:.....	40
3.5 Using seadata.js to seed the database	41
Step 1:.....	41
Step 2:.....	41
Step 3:.....	42
Step 4:.....	42
Step 5:.....	43
3.6 Changing necessary codes in Frontend	43
Step 1:.....	43
Step 2:.....	44
Part 4 – DynamoDB.....	45
4.1 Exporting the table data from MySQL Workbench.....	45
Step 1:.....	45
Step 2:.....	45

Step 3:.....	46
Step 4:.....	47
Step 5:.....	47
Step 6:.....	48
Step 7:.....	49
Step 8:.....	49
Step 9:.....	50
4.2 Create a DynamoDB table for each table in MySQL Workbench	51
Step 1:.....	51
Step 2:.....	51
Step 3:.....	52
Step 4:.....	52
Step 5:.....	53
Step 6:.....	54
Step 7:.....	55
4.3 Adding the MySQL Workbench data to DynamoDB using Cloud9	55
Step 1:.....	55
Step 2:.....	56
Step 3:.....	57
Step 4:.....	57
Step 5:.....	58
Step 6:.....	59
Step 7:.....	59
Step 8:.....	60
Step 9:.....	60
Step 10:.....	62
Step 11:.....	64

Step 12:.....	66
Step 13:.....	67
Step 14:.....	67
Step 15:.....	68
Step 16:.....	68
Step 17:.....	69
Part 5 – Set up Lambda	70
5.1 Creating Lamba Function getDesigns	70
Step 1:.....	70
Step 2:.....	70
Step 3:.....	71
5.2 Creating files in Cloud9 for Lamba Function getDesigns.....	71
Step 1:.....	71
Step 2:.....	74
Step 3:.....	75
Step 4:.....	75
5.3 Configure Test Event for Lambda function getDesigns.....	76
Step 1:.....	76
Step 2:.....	77
Step 3:.....	78
Part 6 – API Gateway	79
6.1 Creating an API Gateway for Lambda Function getDesigns	79
Step 1:.....	79
Step 2:.....	80
Step 3:.....	80
Step 4:.....	81
Step 5:.....	81

Step 6:.....	82
Step 7:.....	82
Step 8:.....	83
Step 9:.....	83
Step 10:.....	84
Step 11:.....	84
Step 12:.....	85
Step 13:.....	85
Step 14:.....	86
Step 15:.....	87
Step 16:.....	87
6.2 Setting up the update design file in WinSCP	88
Step 1:.....	88
Step 2:.....	88
Step 3:.....	89
7. Additional Features.....	90
7.1 Creating Update Design Lambda Function	90
Step 1:.....	90
Step 2:.....	91
Step 3:.....	91
Step 4:.....	92
Step 5:.....	92
Step 6:.....	96
7.2 Creating API Gateway for updateDesign Lambda Function	96
Step 1:.....	96
Step 2:.....	97
Step 3:.....	97

Step 4:.....	98
Step 5:.....	98
Step 6:.....	99
Step 7:.....	99
Step 8:.....	100
Step 9:.....	100
Step 10:.....	101
Step 11:.....	101
Step 12:.....	102
Step 13:.....	102
Step 14:.....	104
Step 15:.....	104
7.3 Adding the updateDesign URL to WinSCP	105
Step 1:.....	105
Step 2:.....	106
Step 3:.....	106
Step 4:.....	107
Step 5:.....	107
Step 6:.....	108
Step 7:.....	108
7.4 Creating manageSubmissions Lambda Function and Index Name for File Table.....	109
Step 1:.....	109
Step 2:.....	110
Step 3:.....	110
Step 4:.....	111
Step 5:.....	111
Step 6:.....	115

Step 7:.....	115
Step 8:.....	116
Step 9:.....	116
7.5 Creating manageSubmissions API Gateway	117
Step 1:.....	117
Step 2:.....	117
Step 3:.....	118
Step 4:.....	118
Step 5:.....	119
Step 6:.....	119
Step 7:.....	120
Step 8:.....	120
Step 9:.....	121
Step 10:.....	121
Step 11:.....	122
Step 12:.....	123
7.6 Adding manageSubmissions API to WinSCP	123
Step 1:.....	123
Step 2:.....	124
Step 3:.....	124
7.8 Creating getProfile Lambda Function	125
Step 1:.....	125
Step 2:.....	125
Step 3:.....	126
Step 4:.....	126
Step 5:.....	129
7.9 Creating the getProfile API Gateway	129

Step 1:.....	129
Step 2:.....	130
Step 3:.....	131
Step 4:.....	131
Step 5:.....	132
Step 6:.....	133
Step 7:.....	133
Step 8:.....	134
Step 9:.....	134
Step 10:.....	135
Step 11:.....	135
Step 12:.....	137
Step 13:.....	137
Step 14:.....	138
7.10 Adding the getProfile API Gateway to WinSCP	138
Step 1:.....	138
Step 2:.....	139
Step 3:.....	139
7.11 Giving Least Privilege for Database credentials	140
Step 1:.....	140
Step 2:.....	140
Step 3:.....	141
Step 4:.....	141
7.12 MySQL RDS Database Security, Encryption and Logging	142
Step 1:.....	142
Step 2:.....	142
Step 3:.....	143

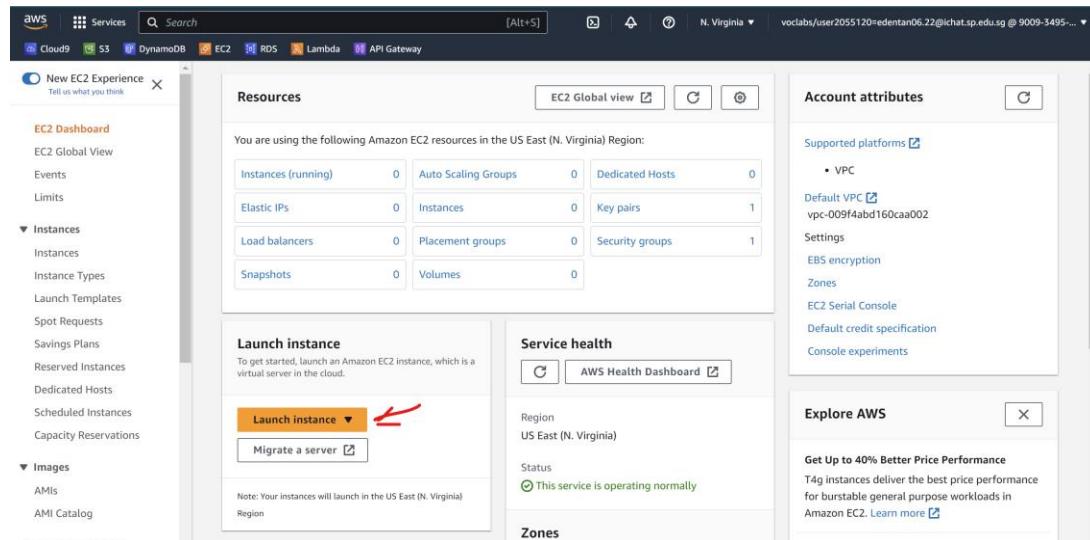
Step 4:	143
Step 5:	144
Step 6:	144
Step 7:	145
Step 8:	146
Step 9:	146
Step 9:	149
Step 10:	150
Step 11:	150
Step 12:	151
Step 13:	151
Step 14:	152
Step 15:	152
Step 16:	153
Step 17:	153
References	154

Part 1 – Set up EC2 Instance on AWS

1.1 Creating an EC2 Instance

Step 1:

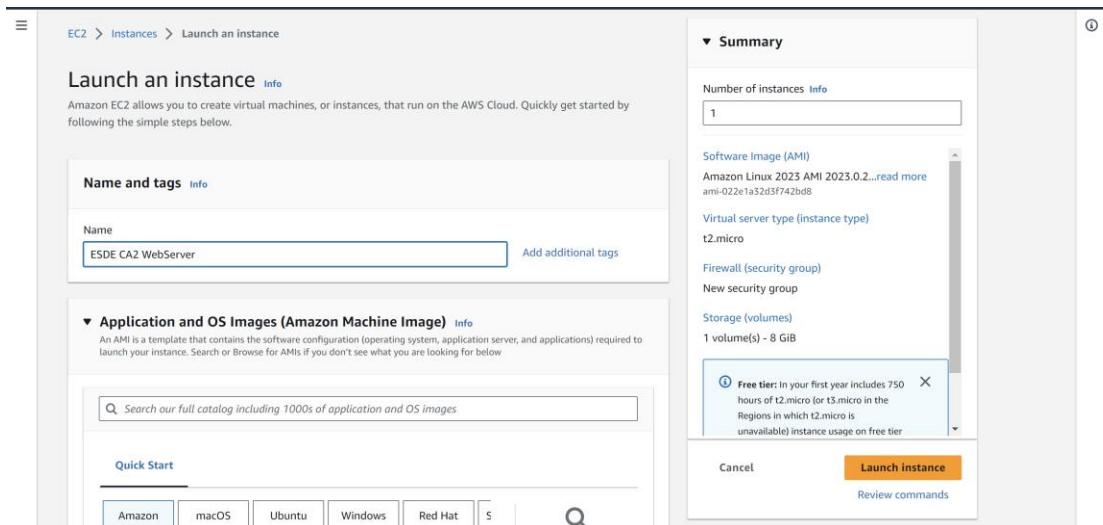
Go into the EC2 Instance Dashboard and click on “Launch Instance” to start a new EC2 Instance.



1.2 Configure EC2 Instance Settings

Step 1:

Add a name for your EC2 Instance.



Step 2:

Scroll down and make sure that you have the following default configurations selected, as they are one of the cheapest cost that AWS offers.

The screenshot shows the AWS Quick Start interface. At the top, there's a grid of logos for various AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE. A red arrow points to the 'Amazon Linux' logo. Below this is a search bar and a link to 'Browse more AMIs'. The main section displays the 'Amazon Linux 2023 AMI' details: 'ami-022e1a32d3f742bd8 (64-bit (x86)) / ami-0b54418bdd76353ce (64-bit (Arm))', 'Virtualization: hvm', 'ENI enabled: true', and 'Root device type: ebs'. It also indicates 'Free tier eligible'. Under 'Description', it says 'Amazon Linux 2023 AMI 2023.0.20230614.0 x86_64 HVM kernel-6.1'. The 'Architecture' dropdown is set to '64-bit (x86)' and shows 'ami-022e1a32d3f742bd8' with a 'Verified provider' badge. The second part of the screenshot shows the 'Instance type' section with a dropdown menu open for 't2.micro'. A red arrow points to the 't2.micro' option. The dropdown lists 'Family: t2', '1 vCPU', '1 GiB Memory', 'Current generation: true', and pricing for On-Demand Windows, SUSE, RHEL, and Linux. It also includes 'All generations' and 'Compare instance types' options.

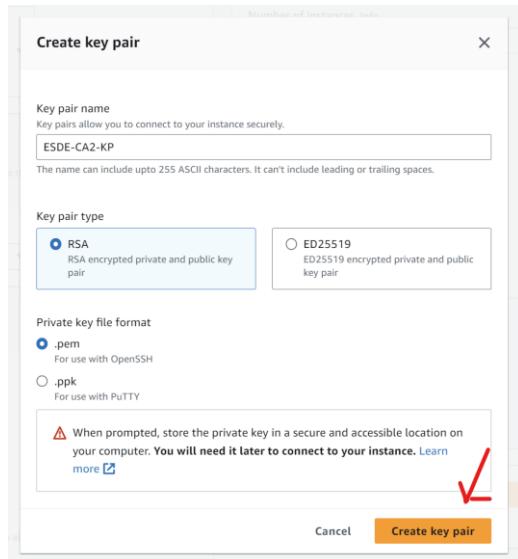
Step 3:

Create a new Key-Pair for your EC2 Instance, click on “Create new key pair”.

The screenshot shows the AWS EC2 instance creation wizard. The first section is 'Instance type', which shows 't2.micro' selected. A red arrow points to the 't2.micro' option. The dropdown also lists 'Family: t2', '1 vCPU', '1 GiB Memory', 'Current generation: true', and pricing for On-Demand Windows, SUSE, RHEL, and Linux. It includes 'All generations' and 'Compare instance types' options. The second section is 'Key pair (login)', which has a note: 'You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.' Below this is a 'Key pair name - required' field with a dropdown menu showing 'Select' and a 'Create new key pair' button with a red arrow pointing to it. The third section is 'Network settings', which has an 'Edit' button.

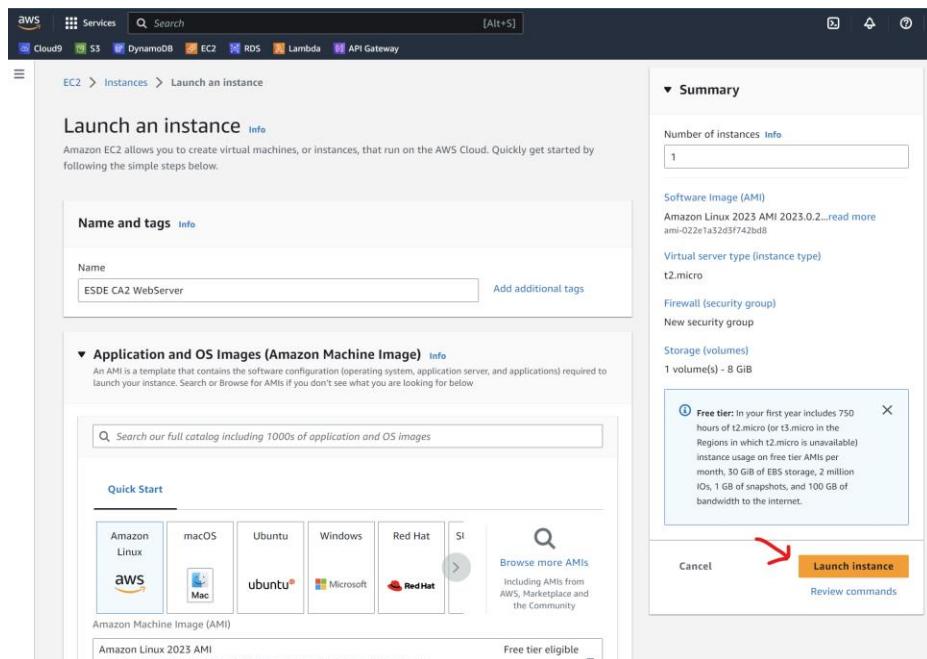
Step 4:

Click on “Create Key Pair” after adding your Key-Pair Name below and save the .pem file at a directory of your choice.



Step 5:

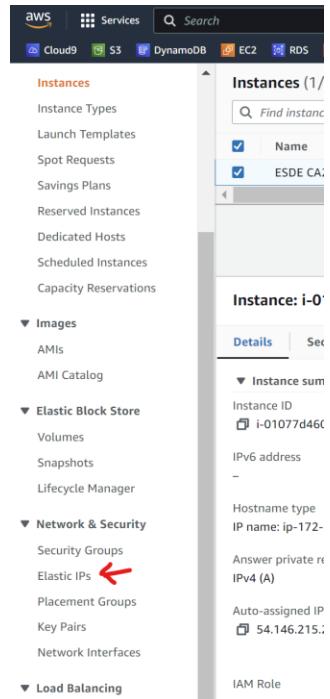
Finally click on the “Launch Instance” on the right side to finish creating your EC2 Instance.



1.3 Creating an Elastic IP Address for your Web Server

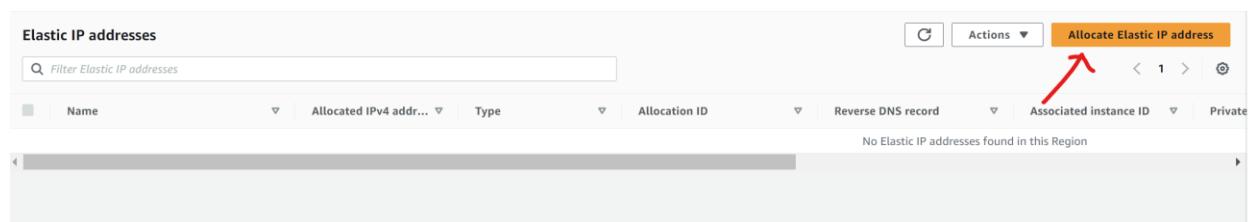
Step 1:

Click on the Elastic IPs on the left side.



Step 2:

Click on “Allocate Elastic IP address” to create a Elastic IP address.



Step 3:

Create a new tag name as shown below and click on the “Allocate” button to create an Elastic IP address.

Allocate Elastic IP address [Info](#)

Elastic IP address settings [Info](#)

Network Border Group [Info](#)
us-east-1

Public IPv4 address pool
 Amazon's pool of IPv4 addresses
 Public IPv4 address that you bring to your AWS account (option disabled because no pools found) [Learn more](#)
 Customer owned pool of IPv4 addresses (option disabled because no customer owned pools found) [Learn more](#)

Global static IP addresses
AWS Global Accelerator can provide global static IP addresses that are announced worldwide using anycast from AWS edge locations. This can help improve the availability and latency for your user traffic by using the Amazon global network. [Learn more](#)

[Create accelerator](#)

Tags - optional
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="Elastic IP For ESDE CA2"/>

Add new tag [Remove](#)

You can add up to 49 more tag

Cancel **Allocate**

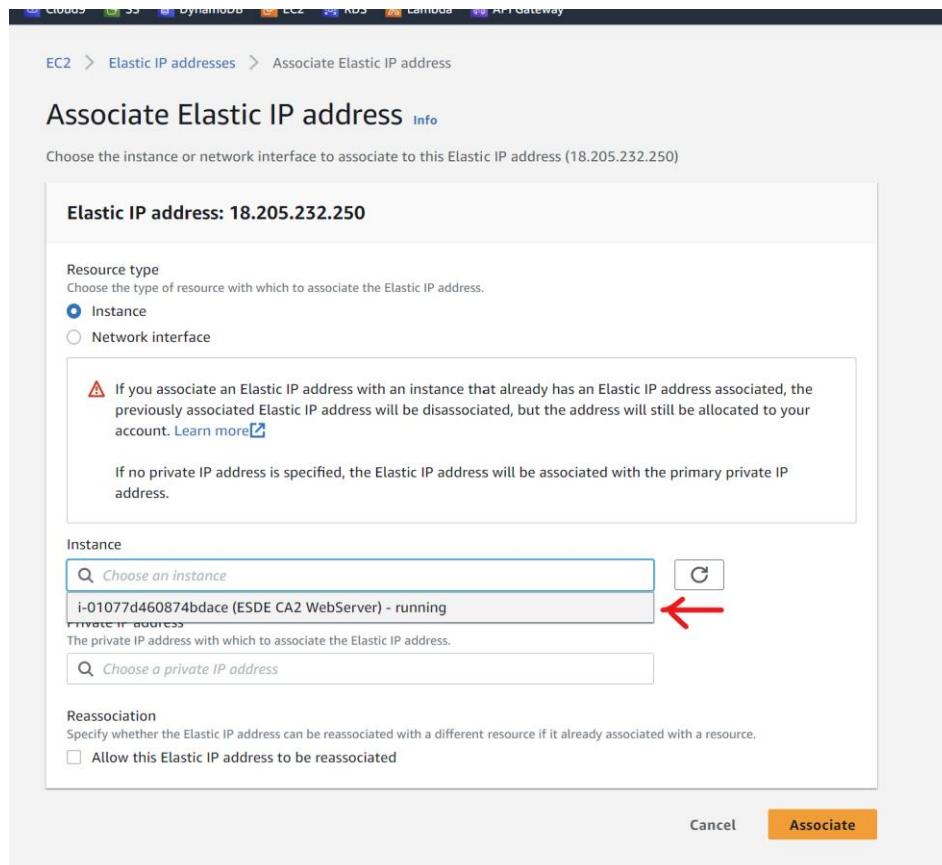
Step 4:

Next, click on the “Associate this Elastic IP Address” on the top right to associate this Elastic IP Address to your EC2 Instance.

Elastic IP addresses (1/1)								
Actions Allocate Elastic IP address								
Filter Elastic IP addresses Clear filters								
Name	Allocated IPv4 addr...	Type	Allocation ID	Reverse DNS record	Associated instance ID	Private		
Elastic IP For ESDE CA2	18.205.232.250	Public IP	eipalloc-06d7d377ba60f88e2	-	-	-		

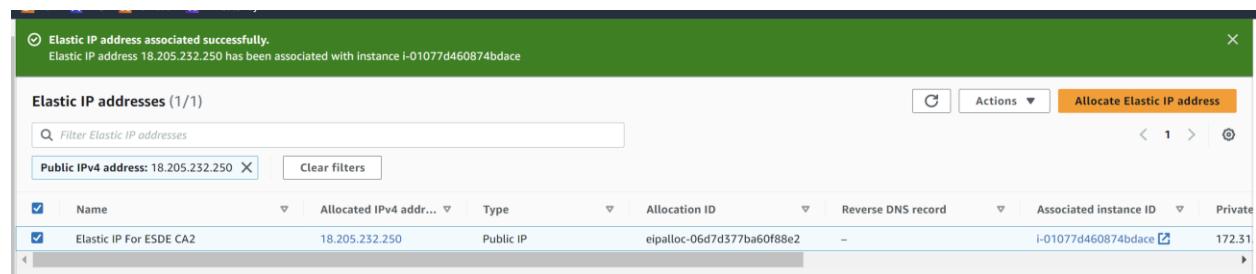
Step 5:

Next, Choose the current running instance and associate the Elastic IP Address we created to it and then click on the “Associate” button to associate it.



Step 6:

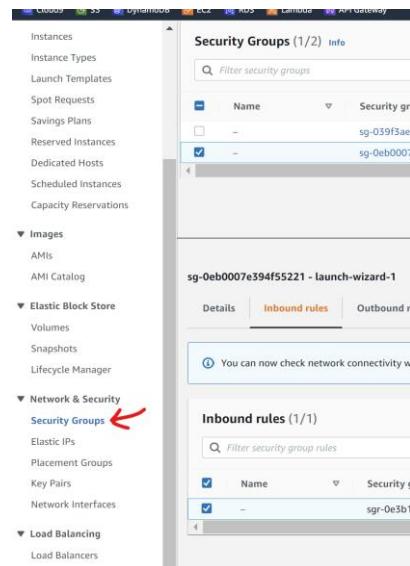
You should see the following Successful Message indicating that the Elastic IP Address has been associated with our EC2 Instance.



1.4 Setting the Inbound Rules for our Ports 3001 and 5000

Step 1:

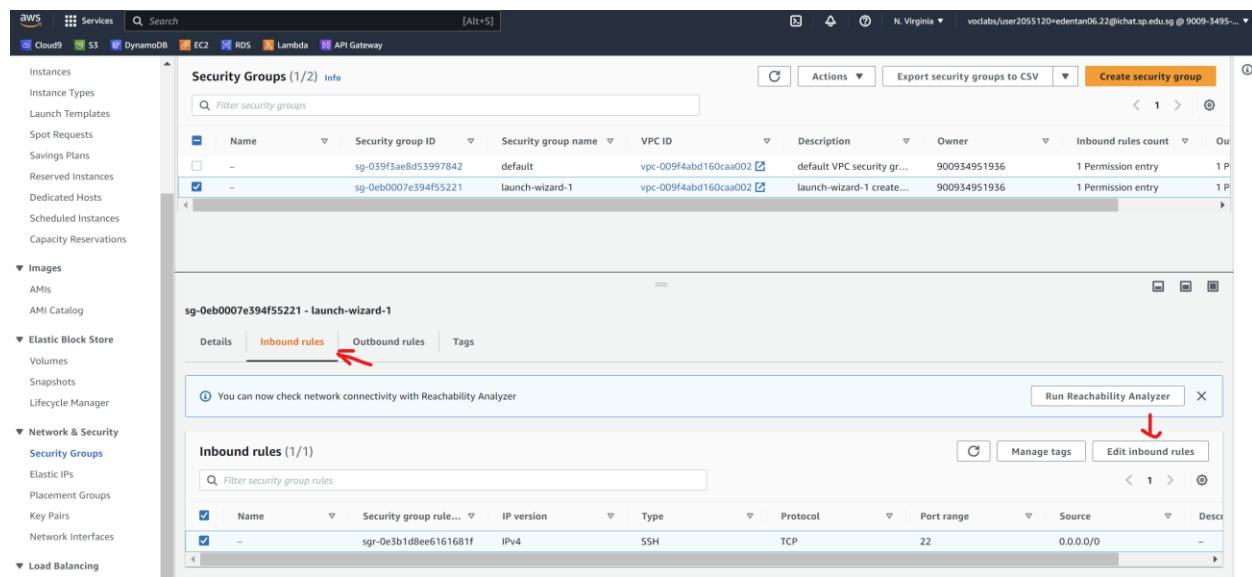
Go over to Security Groups under Network and Security



The screenshot shows the AWS Cloud9 interface. On the left, there's a navigation sidebar with various services like Cloud9, S3, DynamoDB, EC2, RDS, Lambda, and API Gateway. Under 'Network & Security', the 'Security Groups' link is highlighted with a red arrow. The main panel displays the 'Security Groups (1/2) Info' page, showing two security groups: 'sg-039f3ae8d53997842' and 'sg-0eb0007e394f55221'. Below this, the 'sg-0eb0007e394f55221 - launch-wizard-1' details are shown, with the 'Inbound rules' tab selected. It lists one inbound rule: 'sgr-0e3b1d8ee6161681f' (Protocol: TCP, Port range: 22, Source: 0.0.0.0/0).

Step 2:

Select Inbound Rules and click on the “Edit inbound rules” button.



This screenshot continues from the previous one, showing the 'Inbound rules (1/1)' section for the 'sg-0eb0007e394f55221 - launch-wizard-1' security group. The 'Inbound rules' tab is selected, and a red arrow points to it. The table shows one rule: 'sgr-0e3b1d8ee6161681f' (Protocol: TCP, Port range: 22, Source: 0.0.0.0/0). At the bottom right of this section, there are buttons for 'Manage tags' and 'Edit inbound rules', with a red arrow pointing to the 'Edit inbound rules' button.

Step 3:

Add the following 2 new rules, Ports 3001 and 5000 as shown below. And click on the “Save rules” button.

The screenshot shows the AWS Management Console interface for managing security groups. The user is in the 'Edit inbound rules' section of a specific security group. There are three existing rules listed, and two new rules are being added. The new rules have 'Custom TCP' type, 'TCP' protocol, and '0.0.0.0/0' source. Red arrows point to the 'Port range' fields for these new rules, which are set to 3001 and 5000 respectively. The 'Save rules' button is visible at the bottom right.

Step 4:

You should be able to see the added rules under the Inbound Rules as shown below.

The screenshot shows the 'Inbound rules' tab for the same security group. The newly added rules are now listed in the table. Red arrows point to the 'Port range' column for the two new rules, highlighting the values 3001 and 5000.

1.5 Download the following applications

Step 1:

Install Putty and WinSCP using the given links below.

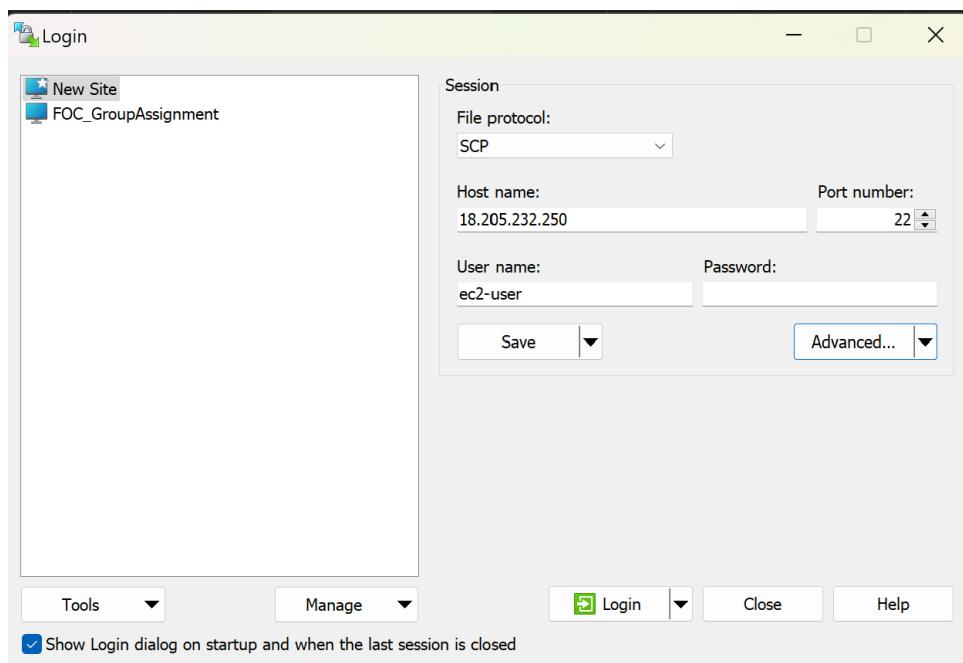
Putty: <https://the.earth.li/~sgtatham/putty/latest/w64/putty-64bit-0.78-installer.msi>

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

1.6 Using Putty and WinSCP for our EC2 Web Server

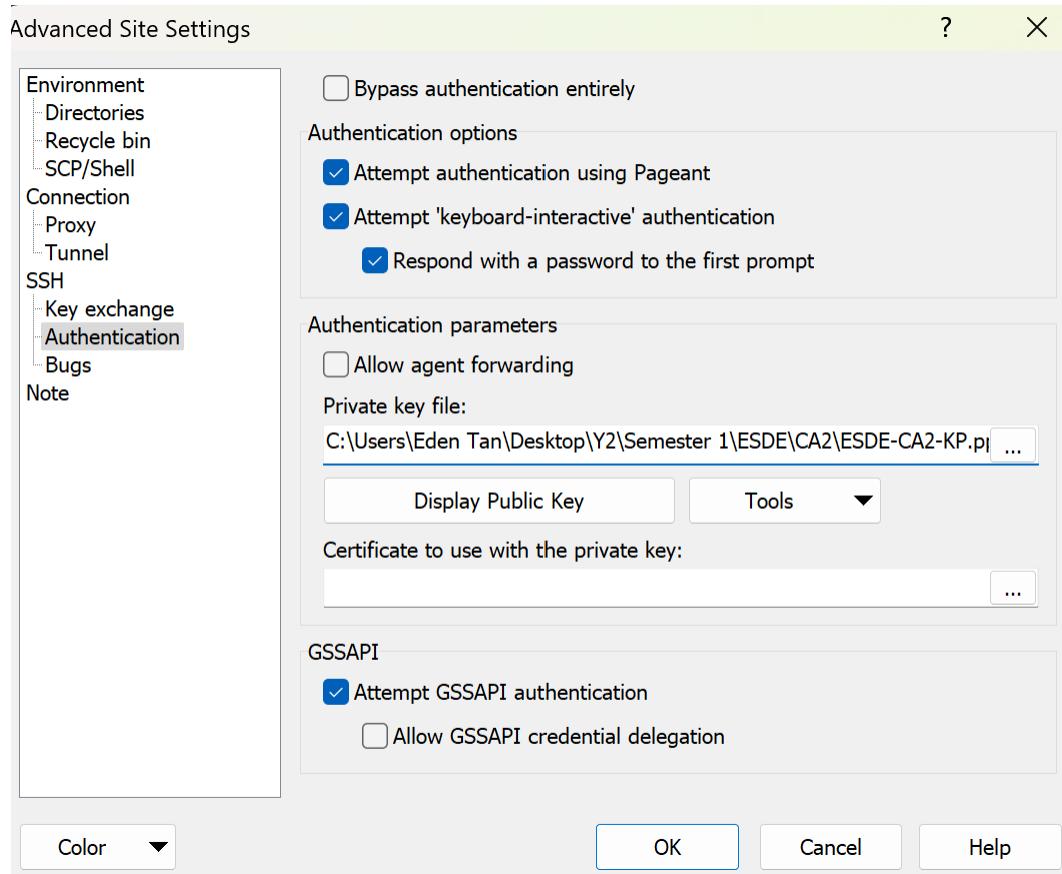
Step 1:

Open up WinSCP and you should see the following below, add in the following details as shown.

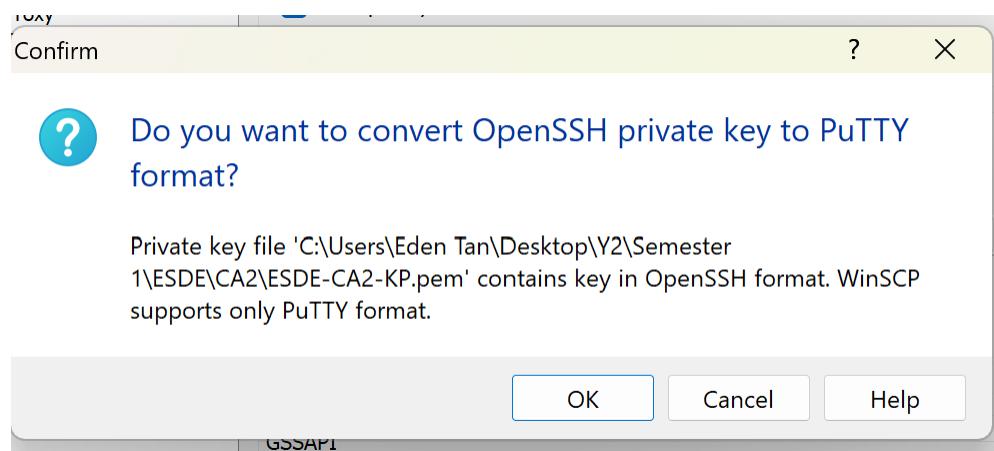


Step 2:

Click on the “Advanced...” button to add in your .pem Key-Pair that you created previously under SSH > Authentication.

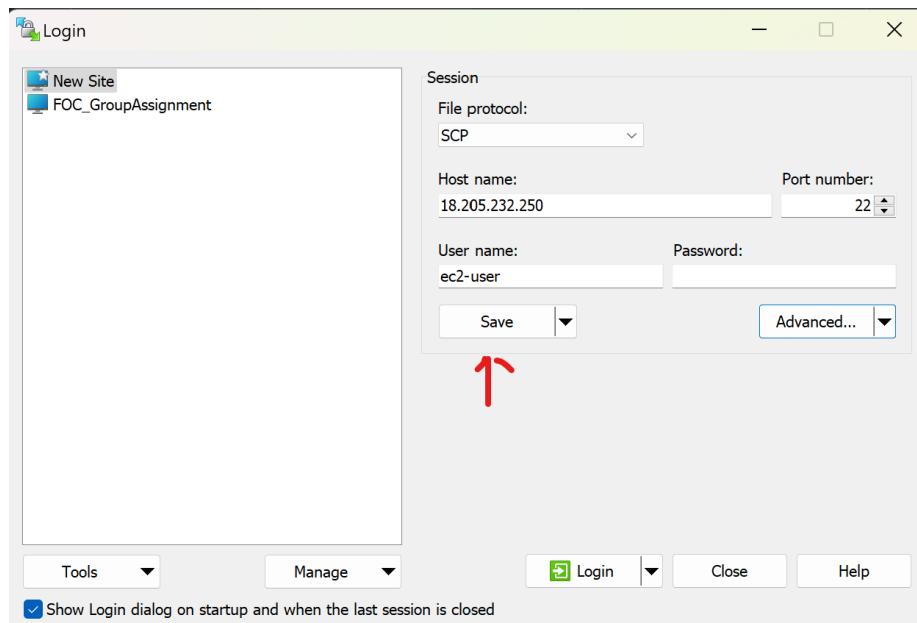


*Note: You will be prompted to convert your .pem key to Putty Format, click on “ok” to proceed.



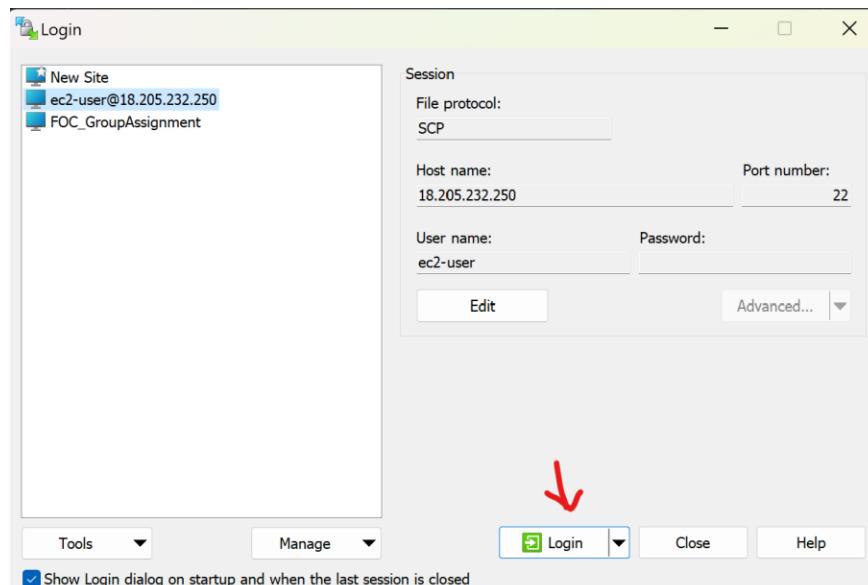
Step 3:

Once done, click “save” and “ok” in the pop up.

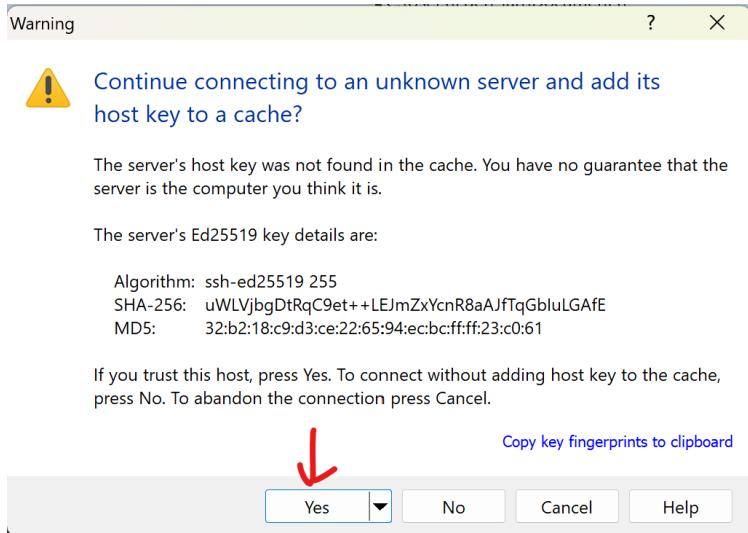


Step 4:

Click on “Login” to see if you can access the EC2 Server.

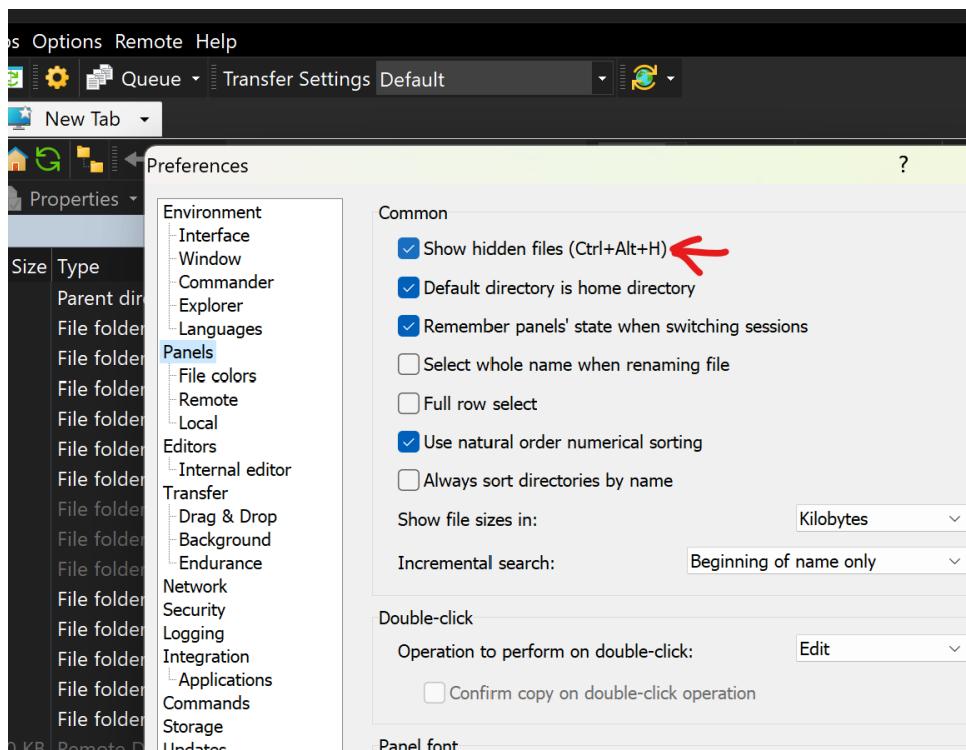


*Note: You will see the following warning, click on “Yes” to continue



Step 5:

Finally, at the top tabs go to Options > Preferences > Panels and check “Show hidden files”

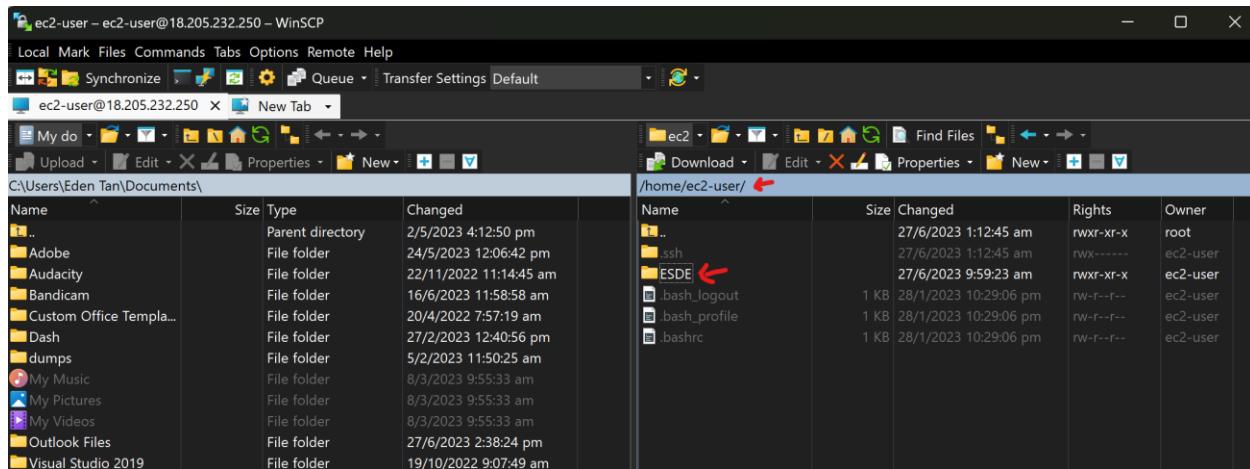


Part 2 – Install NodeJS and copy Bee Design app to EC2

2.1 Creating the folder directories for Bee Design

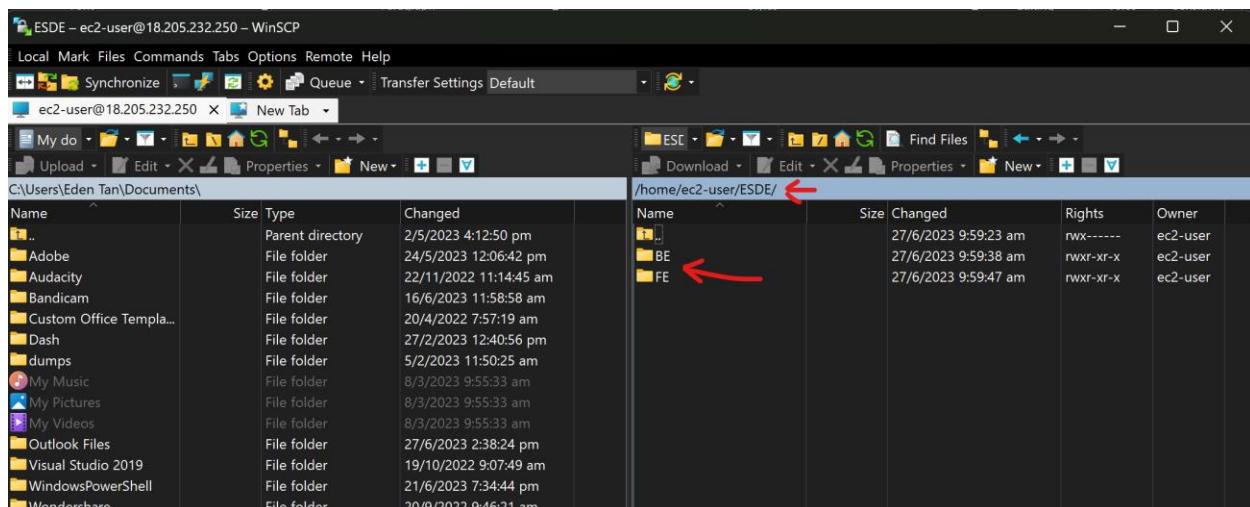
Step 1:

Go to /home/ec2-user/ and create a directory called ESDE.



Step 2:

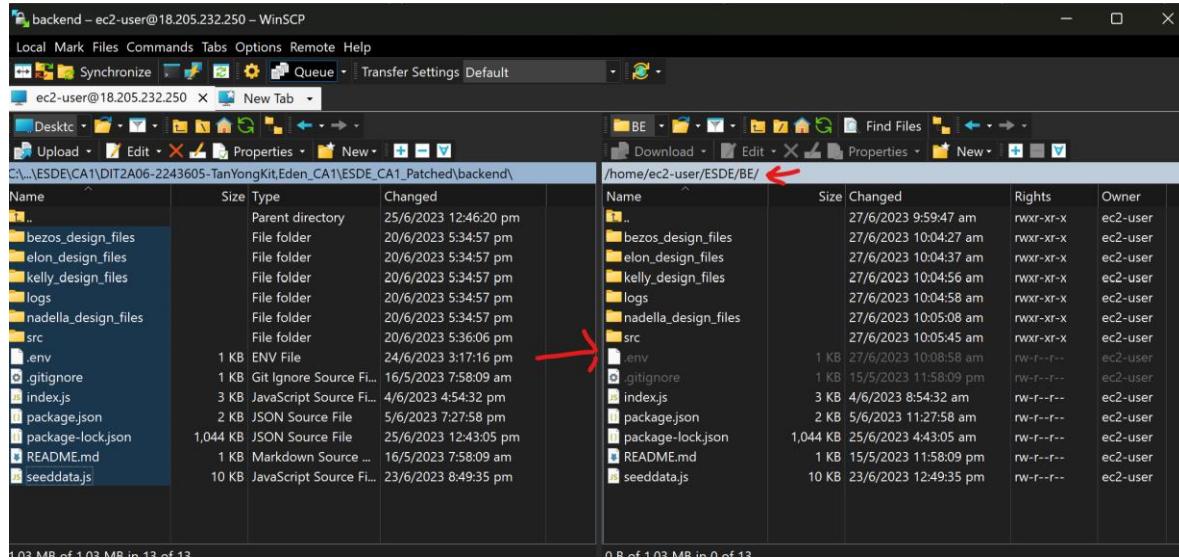
Next, go into the ESDE Directory and create 2 more directories BE and FE respectively.



2.2 Bringing respective Backend and Frontend Bee Design Folders to ESDE Folder

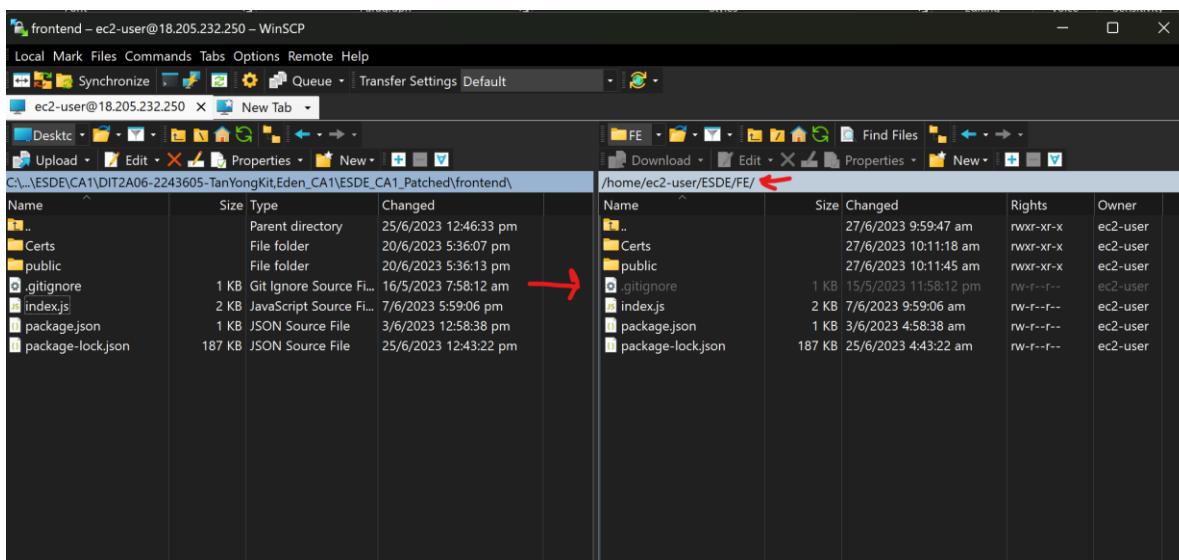
Step 1:

Go to the directory of your CA1 Bee Design Folder and drag the files in the ESDE_CA1_Patched\backend\ into the ESDE/BE Folder.



Step 2:

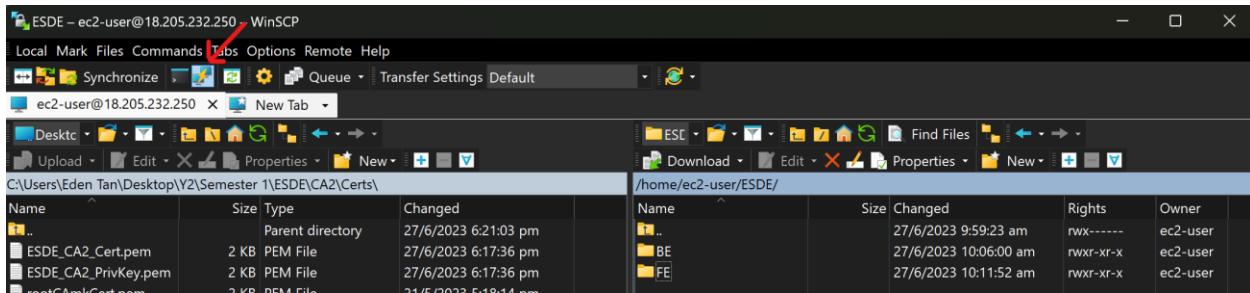
Go to the directory of your CA1 Bee Design Folder and drag the files in the ESDE_CA1_Patched\frontend\ into the ESDE/FE Folder.



2.3 Set up and install node modules and NodeJS with Putty

Step 1:

Click on the Open session with Putty Icon at the top to open a session terminal



Step 2:

Next, type in the following in order into the terminal to install NodeJS.

1. curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
2. . ~/nvm/nvm.sh
3. nvm install --lts

Reference: <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html>

```
[ec2-user@ip-172-31-89-155 ~]$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload   Total   Spent    Left  Speed
100 15916  100 15916    0     0  311k      0  --:--:--  --:--:--   317k
=> Downloading nvm as script to '/home/ec2-user/.nvm'

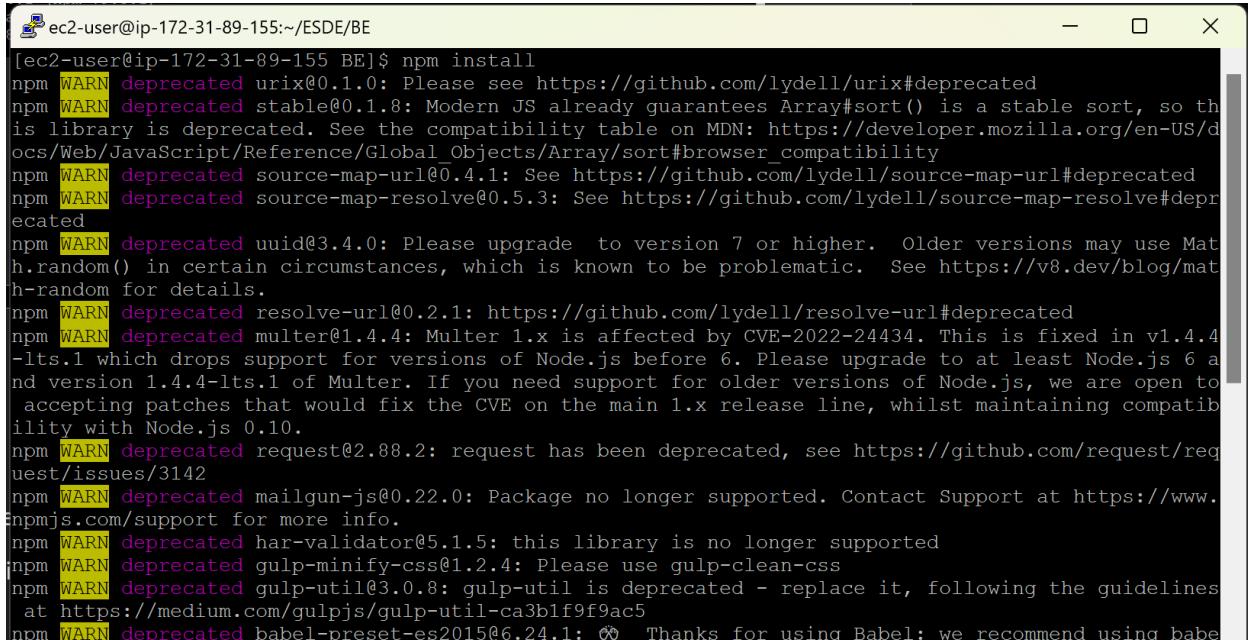
=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use
it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads
nvm bash_completion
[ec2-user@ip-172-31-89-155 ~]$ . ~/nvm/nvm.sh
[ec2-user@ip-172-31-89-155 ~]$ nvm install --lts
Installing latest LTS version.
Downloading and installing node v18.16.1...
Downloaded https://nodejs.org/dist/v18.16.1/node-v18.16.1-linux-x64.tar.xz...
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v18.16.1 (npm v9.5.1)
Creating default alias: default -> lts/* (-> v18.16.1)
[ec2-user@ip-172-31-89-155 ~]$
```

Step 3:

Cd into the backend directory ESDE/BE and run the following command.

Command: npm install

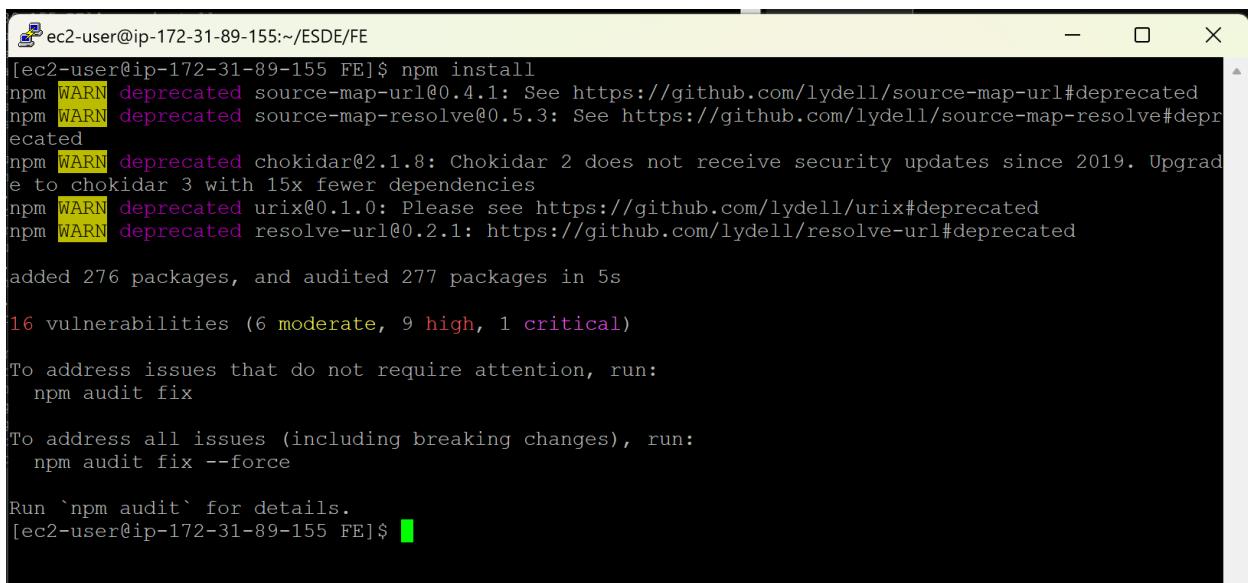


```
[ec2-user@ip-172-31-89-155 BE]$ npm install
npm [WARN] deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm [WARN] deprecated stable@0.1.8: Modern JS already guarantees Array#sort() is a stable sort, so this library is deprecated. See the compatibility table on MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#browser_compatibility
npm [WARN] deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm [WARN] deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm [WARN] deprecated uid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm [WARN] deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm [WARN] deprecated multer@1.4.4: Multer 1.x is affected by CVE-2022-24434. This is fixed in v1.4.4-lts.1 which drops support for versions of Node.js before 6. Please upgrade to at least Node.js 6 and version 1.4.4-lts.1 of Multer. If you need support for older versions of Node.js, we are open to accepting patches that would fix the CVE on the main 1.x release line, whilst maintaining compatibility with Node.js 0.10.
npm [WARN] deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm [WARN] deprecated mailgun-js@0.22.0: Package no longer supported. Contact Support at https://www.npmjs.com/support for more info.
npm [WARN] deprecated har-validator@5.1.5: this library is no longer supported
npm [WARN] deprecated gulp-minify-css@1.2.4: Please use gulp-clean-css
npm [WARN] deprecated gulp-util@3.0.8: gulp-util is deprecated - replace it, following the guidelines at https://medium.com/gulpjs/gulp-util-ca3b1f9f9ac5
npm [WARN] deprecated babel-preset-es2015@6.24.1: ☀️ Thanks for using Babel: we recommend using babel
```

Step 4:

Cd into the frontend directory ESDE/FE and run the following command.

Command: npm install



```
[ec2-user@ip-172-31-89-155 FE]$ npm install
npm [WARN] deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm [WARN] deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm [WARN] deprecated chokidar@2.1.8: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15x fewer dependencies
npm [WARN] deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm [WARN] deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated

added 276 packages, and audited 277 packages in 5s

16 vulnerabilities (6 moderate, 9 high, 1 critical)

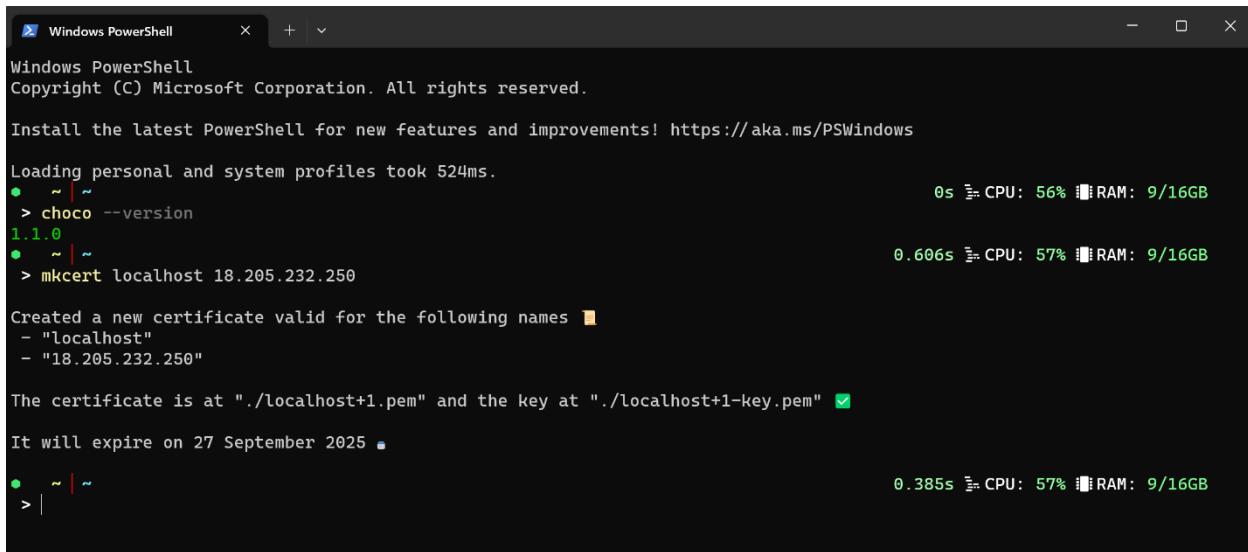
To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
[ec2-user@ip-172-31-89-155 FE]$
```

Step 5:

Create a new SSL Cert using Mkcrt in your command prompt and bring it into the respective Backend and Frontend folders in the WinSCP with the elastic ip address created in EC2.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Loading personal and system profiles took 524ms.
• ~ | ~
> choco --version
1.1.0
• ~ | ~
> mkcert localhost 18.205.232.250

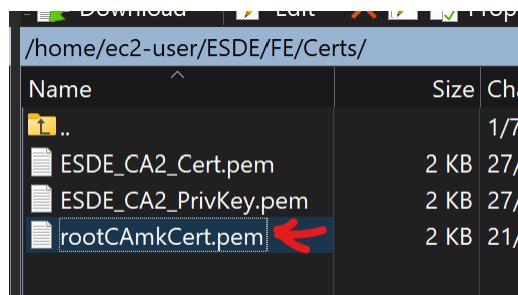
Created a new certificate valid for the following names
- "localhost"
- "18.205.232.250"

The certificate is at "./localhost+1.pem" and the key at "./localhost+1-key.pem" ✅
It will expire on 27 September 2025 ✅

• ~ | ~
> |
```

***Note: Remember to trust mkcrt in your trusted SSL certificates in your browser.**

Import the rootCAmcert.pem into the trusted SSL Certificates.



Step 6:

In the frontend folder under index.js, add in the credentials to read the key and cert file created and change the .listen function by **removing the hostname** to the following below as we are no longer using localhost and save it.

```
//Read private key and certificate for HTTPS ~ Sensitive Data Exposure Prevention
const credentials = {
  key: fs.readFileSync("./Certs/ESDE_CA2_PrivKey.pem"),
  cert: fs.readFileSync("./Certs/ESDE_CA2_Cert.pem"),
  requestCert: false,
  rejectUnauthorized: false
};
//End of Sensitive Data Exposure Prevention
```

```
// Create an HTTPS server with the 'credentials' for a secure connection
const httpsServer = https.createServer(credentials, app);
httpsServer.listen(port, function () {
  console.log(`Server hosted at https://18.205.232.250:${port}`);
});
```



***Note:** Do the same for the backend folder index.js and change the localhost to your Elastic IP address.

Step 7 :

CD in the FE folder in the terminal to test if your site is currently working.

```
ec2-user@ip-172-31-89-155:~/ESDE/FE
[ec2-user@ip-172-31-89-155 ~]$ cd ESDE/FE
[ec2-user@ip-172-31-89-155 FE]$ █
```

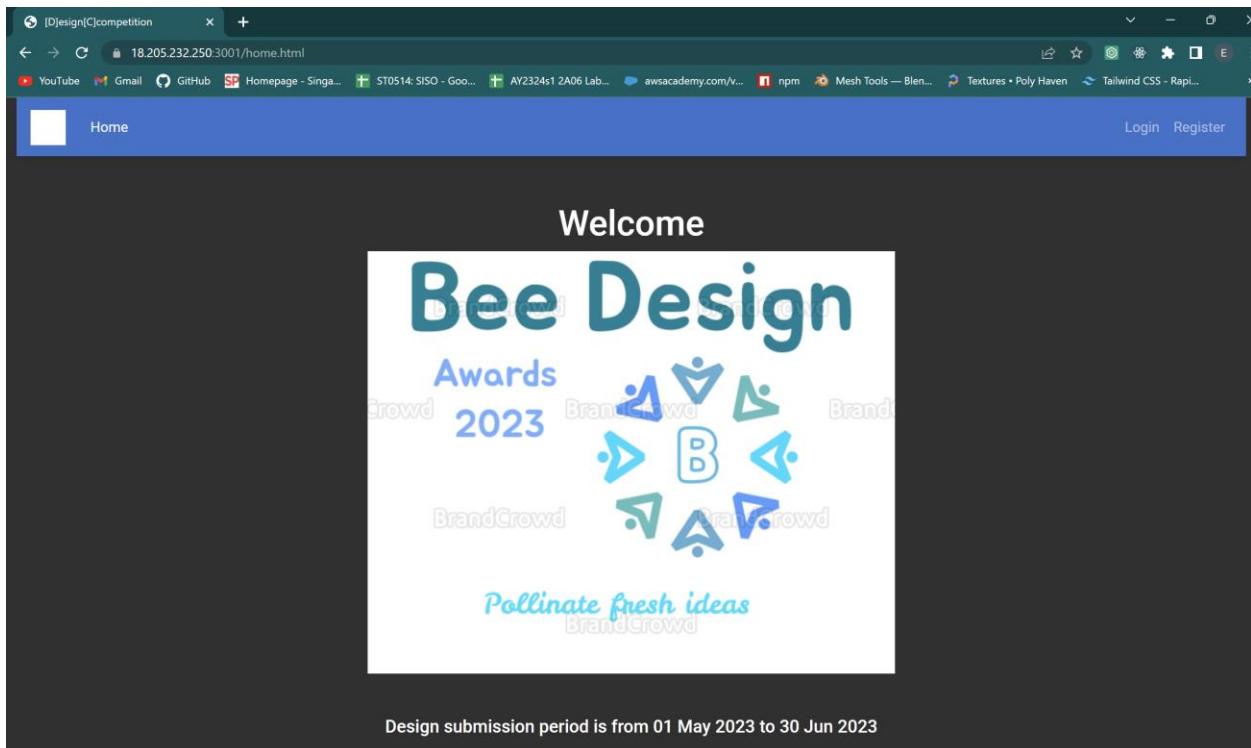
Step 8:

Run node index.js and go to the link to see if your front end site is working.

```
ec2-user@ip-172-31-89-155:~/ESDE/FE
[ec2-user@ip-172-31-89-155 ~]$ cd ESDE/FE
[ec2-user@ip-172-31-89-155 FE]$ node index.js
Server hosted at https://18.205.232.250:3001
```

Step 9:

If all is set up correctly, you should be able to see your Front end site in your browser. When you type <https://18.205.232.250:3001/home.html>.



Part 3 – Set up RDS

3.1 Creating an AWS RDS

Step 1:

Select the RDS Service by AWS and go to the dashboard as shown below.

The screenshot shows the AWS RDS Dashboard for the US East (N. Virginia) region. On the left, a sidebar lists various RDS management options like Dashboard, Databases, Query Editor, etc. The main content area features a prominent callout for the 'Multi-AZ deployment option' for MySQL and PostgreSQL, which promises faster failover and better scalability. Below this, there's a 'Resources' section displaying current usage statistics for DB Instances, DB Clusters, Reserved Instances, and Snapshots across different categories (Manual, Automated). To the right, there are 'Recommended for you' links to topics such as migrating SSRS to RDS for SQL Server, performing AWS Backup, and operational tasks. A 'Time-Series Tables in PostgreSQL' guide is also listed. At the bottom, there's an 'Additional information' section.

Step 2:

Click on the orange button “Create database” to create a new database you should see the following as shown below.

The screenshot shows the 'Create database' page in the AWS RDS console. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, a search bar, and a keyboard shortcut '[Alt+S]'. Below the navigation is a breadcrumb trail: 'RDS > Create database'. The main title is 'Create database'. Underneath, a section titled 'Choose a database creation method' offers two options: 'Standard create' (selected) and 'Easy create'. The 'Standard create' option allows users to set all configuration options, including availability, security, backups, and maintenance. The 'Easy create' option uses recommended best-practice configurations, with some options being changeable after creation. The next section, 'Engine options', lists various database engines with their icons and names. The 'PostgreSQL' engine is selected, indicated by a blue outline around its radio button and icon. Other engines listed include Aurora (MySQL Compatible), Aurora (PostgreSQL Compatible), MySQL, MariaDB, Oracle, and Microsoft SQL Server.

Engine Type	Icon	Status
Aurora (MySQL Compatible)		<input type="radio"/>
Aurora (PostgreSQL Compatible)		<input type="radio"/>
MySQL		<input type="radio"/>
MariaDB		<input type="radio"/>
PostgreSQL		<input checked="" type="radio"/>
Oracle		<input type="radio"/>
Microsoft SQL Server		<input type="radio"/>

Step 3:

Make sure that you choose Standard create, under Engine options choose MySQL, under Templates choose Free tier.

Create database

Choose a database creation method [Info](#)

Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type [Info](#)

Aurora (MySQL Compatible)

Aurora (PostgreSQL Compatible)

MySQL

MariaDB

PostgreSQL

Oracle

Microsoft SQL Server

Edition

MySQL Community

Known issues/limitations
Review the [Known issues/limitations](#) to learn about potential compatibility issues with specific database versions.

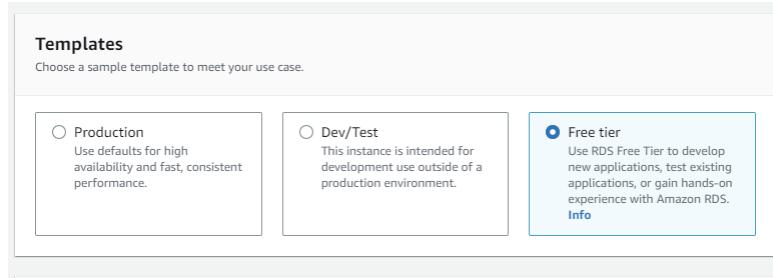
▼ Hide filters

Show versions that support the Multi-AZ DB cluster [Info](#)
Create a A Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

Show versions that support the Amazon RDS Optimized Writes [Info](#)
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Engine Version

MySQL 8.0.32



Step 4:

For the settings follow the image shown below.

Password: Edentan2003!

*Note: For the password we will use it when we go to MySQL Workbench later.

The screenshot shows the 'Settings' page for creating a new DB instance. It includes fields for the DB instance identifier ('database-1'), master username ('admin'), and master password ('Edentan2003'). It also shows options for managing credentials in AWS Secrets Manager and generating a password automatically.

DB instance identifier [Info](#)
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Credentials Settings

Master username [Info](#)
Type a login ID for the master user of your DB instance.

1 to 16 alphanumeric characters. First character must be a letter.

Manage master credentials in AWS Secrets Manager
Manage master user credentials in Secrets Manager. RDS can generate a password for you and manage it throughout its lifecycle.

Info If you manage the master user credentials in Secrets Manager, some RDS features aren't supported.
[Learn more](#)

Auto generate a password
Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).

Confirm master password [Info](#)

Step 5:

Scroll down to Connectivity, select “Yes” for public access as shown below and click on “Create database” at the bottom.

Connectivity [Info](#)

Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

Virtual private cloud (VPC) [Info](#)
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-009f4abd160caa002)
6 Subnets, 6 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

DB subnet group [Info](#)
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

default

Public access [Info](#)

Yes Red arrow pointing here
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

No
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) [Info](#)
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

Choose existing
Choose existing VPC security groups

Create new
Create new VPC security group

Existing VPC security groups

Choose one or more options

Estimated monthly costs

The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:

- 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance.
- 20 GB of General Purpose Storage (SSD).
- 20 GB for automated backup storage and any user-initiated DB Snapshots.

[Learn more about AWS Free Tier.](#)

When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the [Amazon RDS Pricing page](#).

④ You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services. Red arrow pointing here

[Cancel](#) **Create database**

Step 6:

Under Databases in Amazon RDS, you should be able to see the newly created database.

The screenshot shows the AWS RDS Databases page. On the left, there is a sidebar with various options like Dashboard, Databases (which is selected and highlighted with a red arrow), Query Editor, etc. The main content area shows a message about creating a database and a table titled 'Databases (1)'. The table has columns for DB identifier, Status, Role, Engine, Region & AZ, Size, and Actions. A single row is listed: 'database-1' (Status: Creating, Instance: MySQL Community, Region & AZ: us-east-1b, Size: db.t3.micro). A red arrow points to the 'DB identifier' column header.

3.2 Configure Inbound rules for database

Step 1:

Click on the database-1, and then go to VPC security groups and click the link as shown below.

The screenshot shows the 'database-1' details page under the 'Connectivity & security' tab. It displays basic information: DB identifier (database-1), CPU (-), Status (Creating), Class (db.t3.micro), Role (Instance), Engine (MySQL Community), Region & AZ (us-east-1b), and Port (-). Below this, the 'Connectivity & security' section is expanded, showing three tabs: Endpoint & port, Networking, and Security. The Security tab shows the VPC security group 'default (sg-039f3ae8d53997842)' with a red arrow pointing to it. The Networking tab shows the Availability Zone (us-east-1b) and VPC (publicly accessible).

Step 2:

Select Inbound rules and click on the “Edit inbound rules” button.

The screenshot shows the AWS Security Groups interface. At the top, there's a search bar with the value "sg-039f3ae8d53997842" and a "Create security group" button. Below the search bar are buttons for "Actions", "Export security groups to CSV", and a "Create security group" button. A message box at the top right says "You can now check network connectivity with Reachability Analyzer" with a "Run Reachability Analyzer" button. The main table has columns: Name, Security group ID, Security group name, VPC ID, Description, and Owner. One row is selected, showing "sg-039f3ae8d53997842" under Name and "default" under Security group name. The "Inbound rules" tab is highlighted with a red arrow pointing to it. At the bottom right of the table area, there's a "Edit inbound rules" button with a red arrow pointing to it.

Step 3:

Add a new rule, for MYSQL/Aurora as shown below and click on “Save rules” when done.

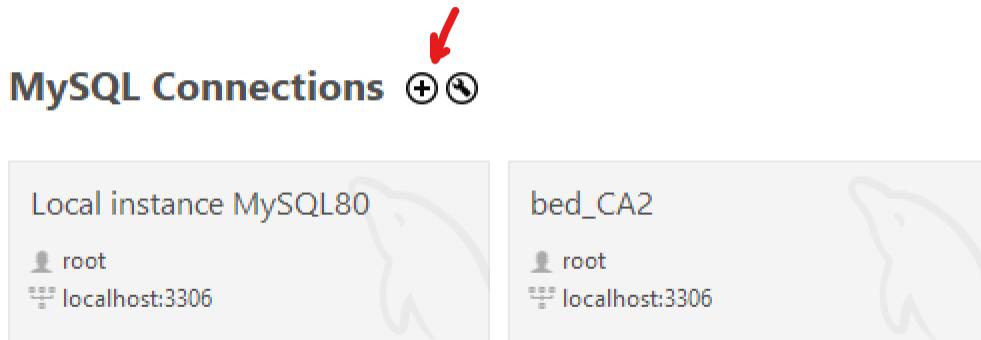
The screenshot shows the "Edit inbound rules" page. It has a header "Edit inbound rules" and a sub-header "Inbound rules". A note below the header says "Inbound rules control the incoming traffic that's allowed to reach the instance." The main table has columns: Security group rule ID, Type, Protocol, Port range, Source, and Description - optional. There are two rows. The first row has "sgr-09b856d14fe719b4f" under Security group rule ID, "All traffic" under Type, "All" under Protocol, "All" under Port range, and a dropdown menu showing "sg-039f3ae8d53997842" under Source. The second row has a dropdown menu showing "MySQL/Aurora" under Type, "TCP" under Protocol, "3306" under Port range, and a dropdown menu showing "Anywh..." under Source. A red arrow points from the "Add rule" button to the "MySQL/Aurora" dropdown. Another red arrow points from the "Source" dropdown to the "0.0.0.0/0" entry. At the bottom right, there are "Cancel", "Preview changes", and a prominent "Save rules" button with a red arrow pointing to it.

3.3 Setting up MySQL Workbench for seeding of data

Step 1:

Open MySQL Workbench and create a new connection.

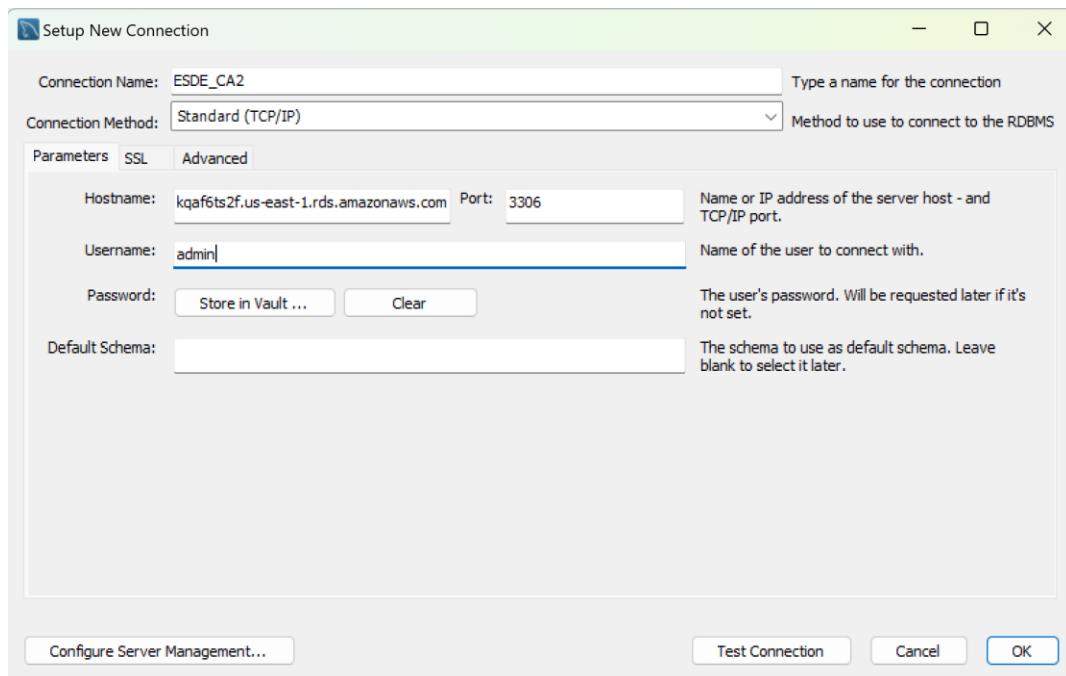
[Browse Documentation >](#)



Step 2:

Add a connection name. Next, for your Hostname copy the database endpoint you created just now and paste it in Hostname as shown below.

A screenshot of the AWS RDS 'Summary' and 'Connectivity & security' tabs. The 'Summary' tab shows basic metrics like CPU usage and status. The 'Connectivity & security' tab is active, showing the database endpoint 'database-1.czbkqaf6ts2f.us-east-1.rds.amazonaws.com'. A red arrow points to this endpoint. The 'Networking' and 'Security' sections show the VPC configuration, including the availability zone 'us-east-1b', VPC ID 'vpc-009f4abd160caa002', and security group 'default (sg-039f3ae8d53997842)'. The 'Security' section also indicates 'Active' status.

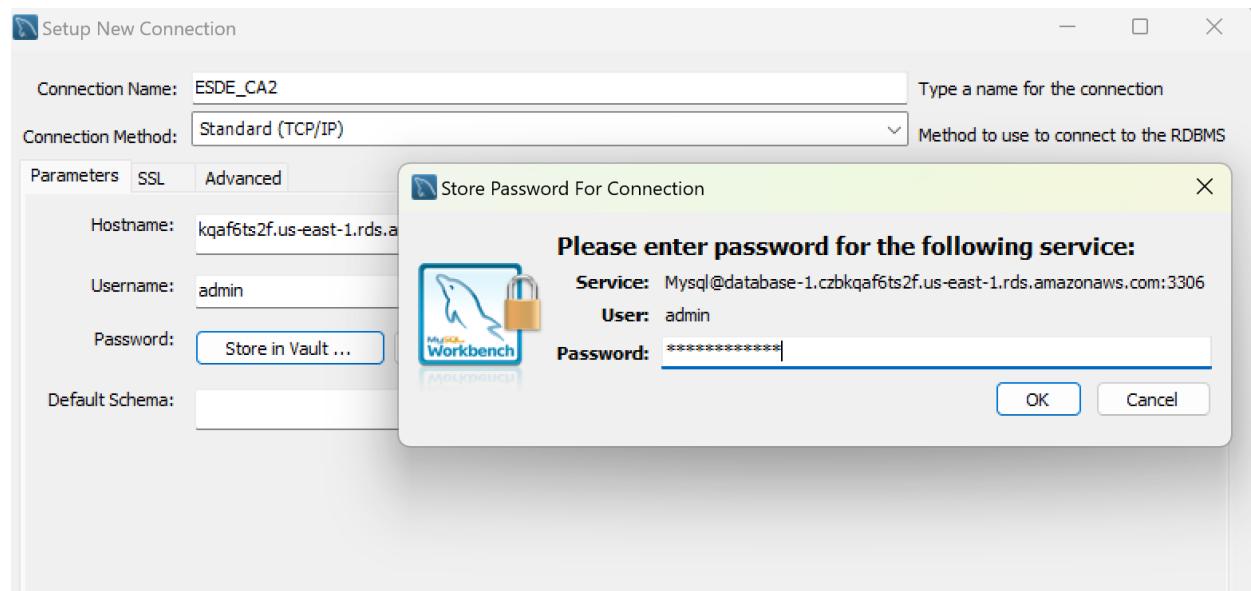


Step 3:

For the Username and Password, we will use the following which we used just now when creating the RDS.

Username: admin

Password: Edentan2003!



Step 4:

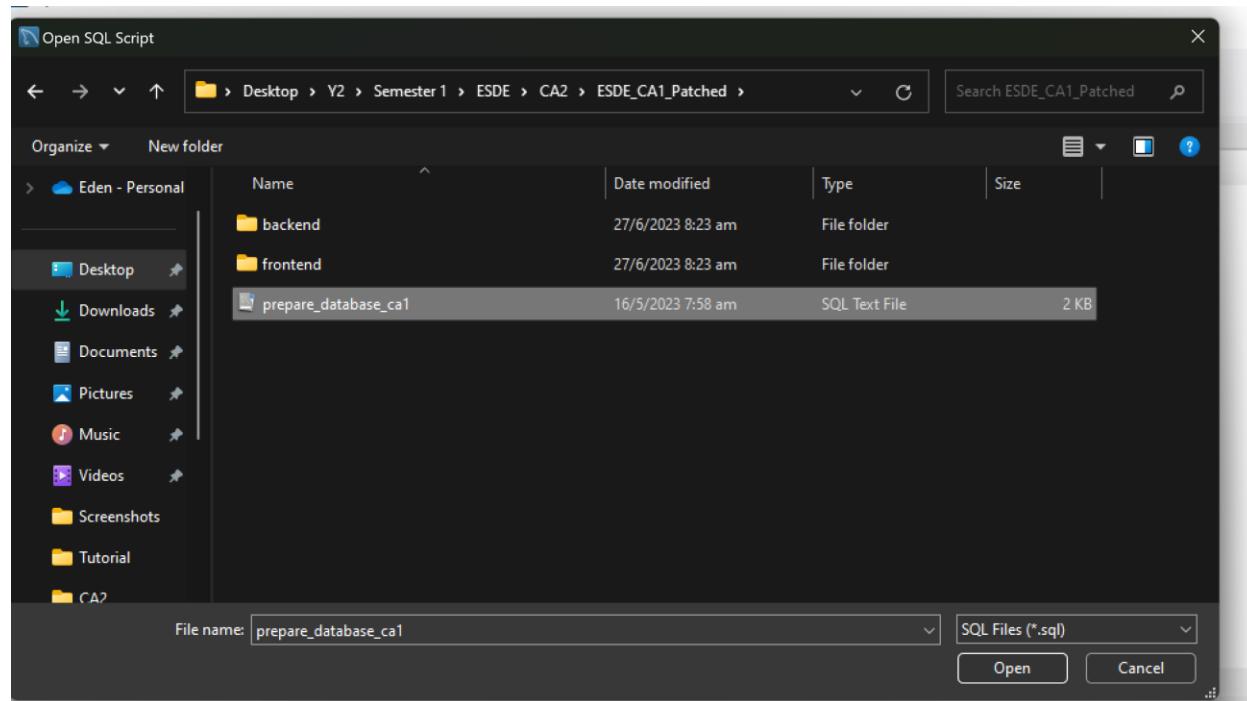
Finally click on Ok and you should see your ESDE_CA2 connection being created as shown below.



3.4 Running the SQL Script in the connection to create Database etc.

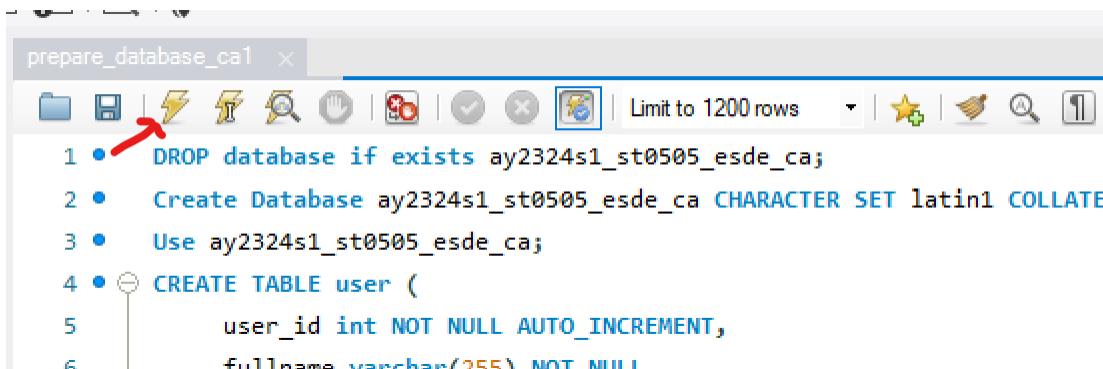
Step 1:

Go into the ESDE_CA2 connection and open the SQL Script from your CA1 as shown below.



Step 2:

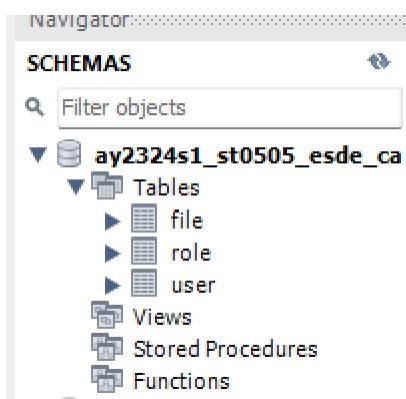
Run the script by clicking on the Lightning Icon as shown below.



```
prepare_database_ca1.x
1 • DROP database if exists ay2324s1_st0505_esde_ca;
2 • Create Database ay2324s1_st0505_esde_ca CHARACTER SET latin1 COLLATE
3 • Use ay2324s1_st0505_esde_ca;
4 • CREATE TABLE user (
    user_id int NOT NULL AUTO_INCREMENT,
    fullname varchar(50) NOT NULL
```

Step 3:

You should be able to see the tables being created in your schema.

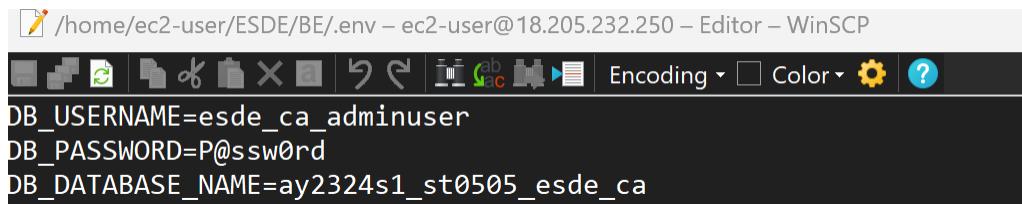


3.5 Using seedbackup.js to seed the database

Step 1:

Go into WinSCP and go to the Backend Directory and look for the .env file under ESDE/BE/.env.

*Note: Make sure that the DB_USERNAME, DB_PASSWORD and DB_DATABASE_NAME matches the one in your MySQL Workbench.

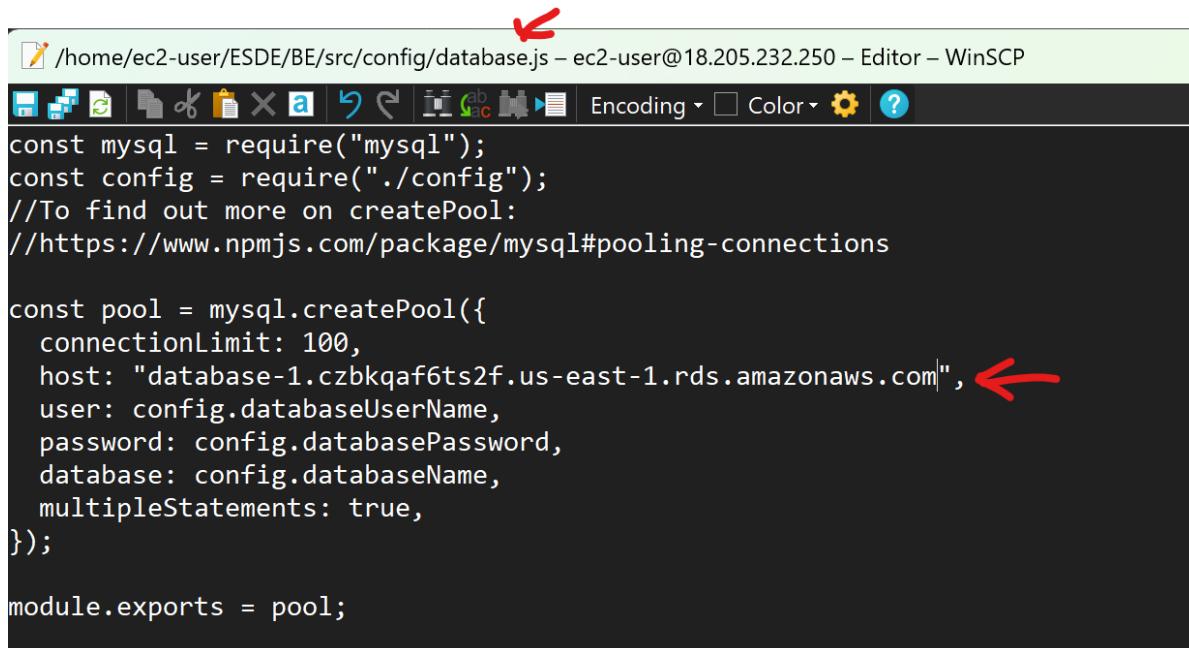


```
/home/ec2-user/ESDE/BE/.env – ec2-user@18.205.232.250 – Editor – WinSCP
DB_USERNAME=esde_ca_adminuser
DB_PASSWORD=P@ssw0rd
DB_DATABASE_NAME=ay2324s1_st0505_esde_ca
```

```
DROP USER IF EXISTS 'esde_ca_adminuser'@'%';
CREATE USER 'esde_ca_adminuser'@'%' IDENTIFIED WITH mysql_native_password BY 'P@ssw0rd';
GRANT ALL PRIVILEGES ON ay2324s1_st0505_esde_ca.* TO 'esde_ca_adminuser'@'%';
```

Step 2:

Go into database.js under ESDE/BE/src/config/database.js and change the host to the database endpoint from AWS RDS.



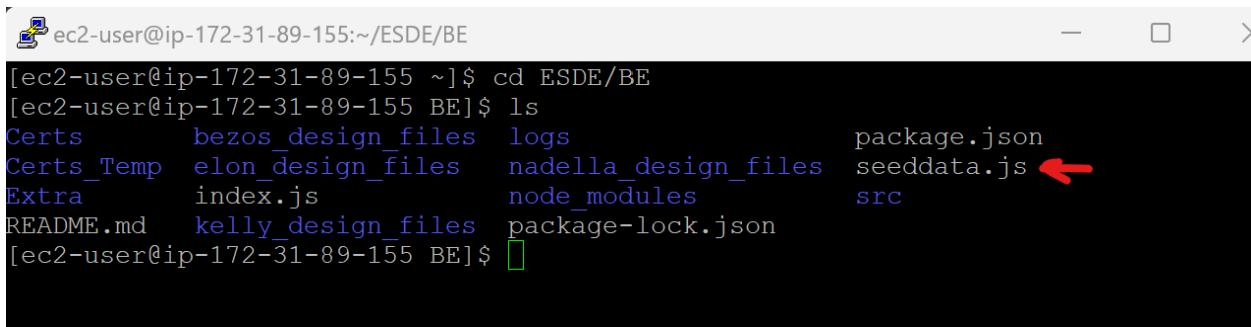
```
/home/ec2-user/ESDE/BE/src/config/database.js – ec2-user@18.205.232.250 – Editor – WinSCP
const mysql = require("mysql");
const config = require("./config");
//To find out more on createPool:
//https://www.npmjs.com/package/mysql#pooling-connections

const pool = mysql.createPool({
  connectionLimit: 100,
  host: "database-1.czbkqaf6ts2f.us-east-1.rds.amazonaws.com", ←
  user: config.databaseUserName,
  password: config.databasePassword,
  database: config.databaseName,
  multipleStatements: true,
});

module.exports = pool;
```

Step 3:

Open the terminal and go to the Backend Directory and make sure that you have seeddata.js in that BE Directory as shown below.



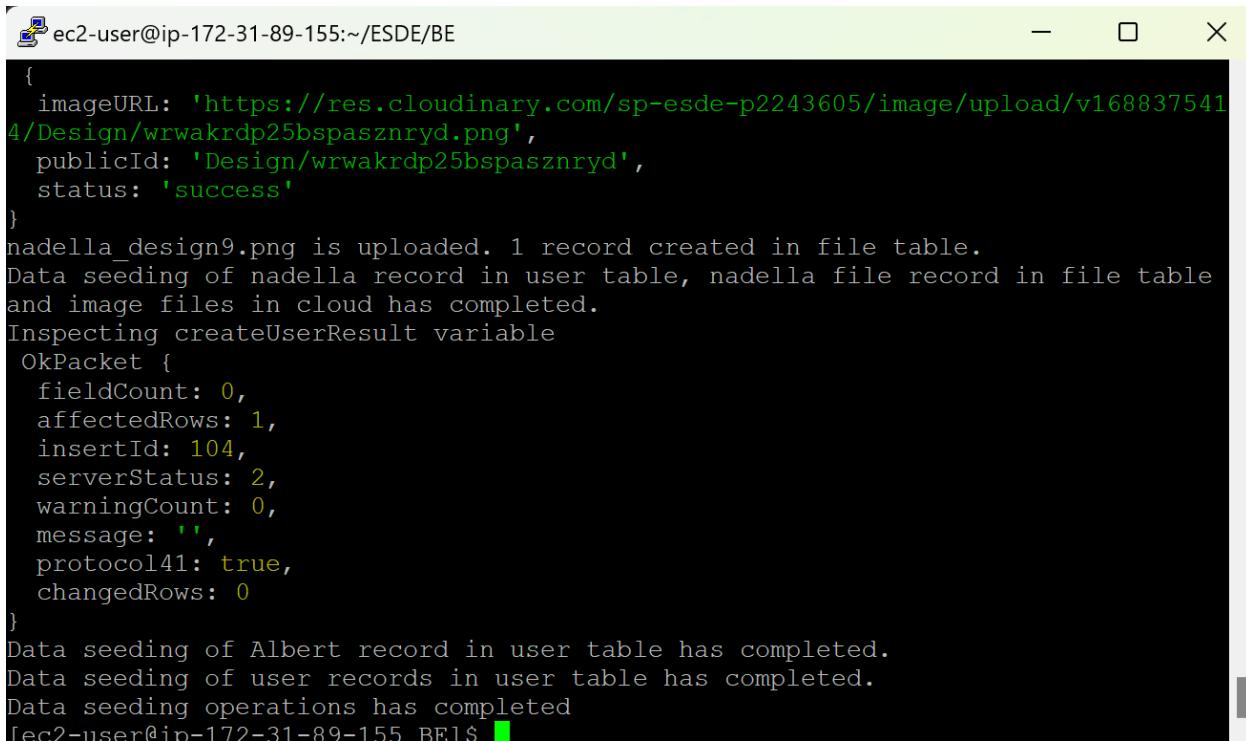
```
ec2-user@ip-172-31-89-155:~/ESDE/BE
[ec2-user@ip-172-31-89-155 ~]$ cd ESDE/BE
[ec2-user@ip-172-31-89-155 BE]$ ls
Certs      bezos_design_files  logs          package.json
Certs_Temp  elon_design_files  nadella_design_files  seeddata.js
Extra       index.js           node_modules    src
README.md   kelly_design_files package-lock.json
[ec2-user@ip-172-31-89-155 BE]$
```

Step 4:

Use seeddata.js to populate the database in the ESDE/BE directory.

Run the following command: **node seeddata.js**

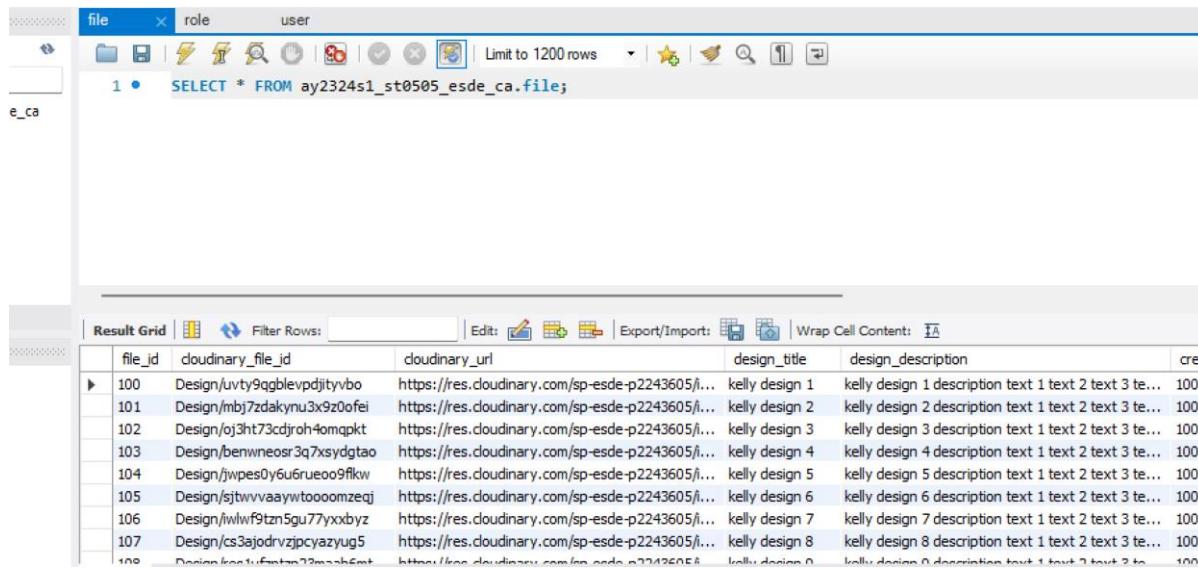
You should be able to successfully run the command without any errors.



```
ec2-user@ip-172-31-89-155:~/ESDE/BE
{
  imageURL: 'https://res.cloudinary.com/sp-esde-p2243605/image/upload/v1688375414/Design/wrwakrdp25bspasznryd.png',
  publicId: 'Design/wrwakrdp25bspasznryd',
  status: 'success'
}
nadella_design9.png is uploaded. 1 record created in file table.
Data seeding of nadella record in user table, nadella file record in file table
and image files in cloud has completed.
Inspecting createUserResult variable
OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 104,
  serverStatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
Data seeding of Albert record in user table has completed.
Data seeding of user records in user table has completed.
Data seeding operations has completed
[ec2-user@ip-172-31-89-155 BE]$
```

Step 5:

Go into MySQL Workbench and make sure that all the tables have been seeded correctly.



The screenshot shows the MySQL Workbench interface. In the top bar, there are tabs for 'file', 'role', and 'user'. Below the tabs is a toolbar with various icons. A query editor window is open with the following SQL command:

```
1 •  SELECT * FROM ay2324s1_st0505_esde_ca.file;
```

The results are displayed in a 'Result Grid' table. The columns are: file_id, clouddinary_file_id, clouddinary_url, design_title, design_description, and crea. There are 109 rows of data, each representing a file entry with its details.

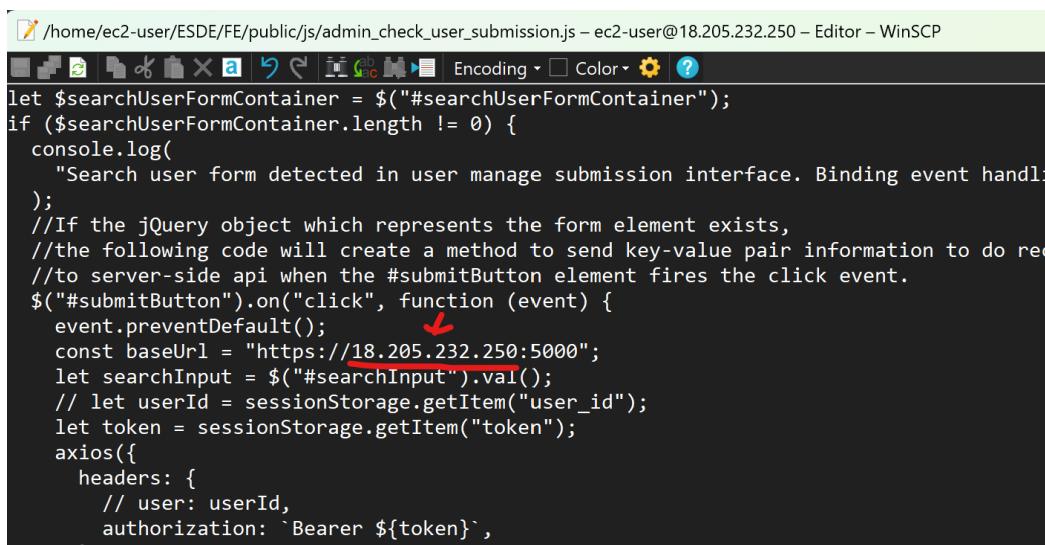
file_id	clouddinary_file_id	clouddinary_url	design_title	design_description	crea
100	Design/uvty9qgblevpdjityvbo	https://res.cloudinary.com/sp-esde-p2243605/i...	kelly design 1	kelly design 1 description text 1 text 2 text 3 te...	100
101	Design/nb7zdakynu3x920fei	https://res.cloudinary.com/sp-esde-p2243605/i...	kelly design 2	kelly design 2 description text 1 text 2 text 3 te...	100
102	Design/oj3ht73cdjr0h4ompkt	https://res.cloudinary.com/sp-esde-p2243605/i...	kelly design 3	kelly design 3 description text 1 text 2 text 3 te...	100
103	Design/bewinveosr3q7xsydgtao	https://res.cloudinary.com/sp-esde-p2243605/i...	kelly design 4	kelly design 4 description text 1 text 2 text 3 te...	100
104	Design/jwipes0y6ufrueo9flkw	https://res.cloudinary.com/sp-esde-p2243605/i...	kelly design 5	kelly design 5 description text 1 text 2 text 3 te...	100
105	Design/sjtwwvvaaywtooomzeqj	https://res.cloudinary.com/sp-esde-p2243605/i...	kelly design 6	kelly design 6 description text 1 text 2 text 3 te...	100
106	Design/iwlw9tzn5gu77xxbyz	https://res.cloudinary.com/sp-esde-p2243605/i...	kelly design 7	kelly design 7 description text 1 text 2 text 3 te...	100
107	Design/cs3sjodrvzjpacyayug5	https://res.cloudinary.com/sp-esde-p2243605/i...	kelly design 8	kelly design 8 description text 1 text 2 text 3 te...	100
108	Design/kon1ufmtnn?mnzhfnt	https://res.cloudinary.com/sp-esde-p2243605/i...	kelly design 9	kelly design 9 description text 1 text 2 text 3 te...	100

3.6 Changing necessary codes in Frontend

Step 1:

Go into ESDE/FE/public/js in WinSCP and change all the following .js files that uses “localhost” to your Elastic IP Address which in my case is “18.205.232.250”.

An example as shown below.



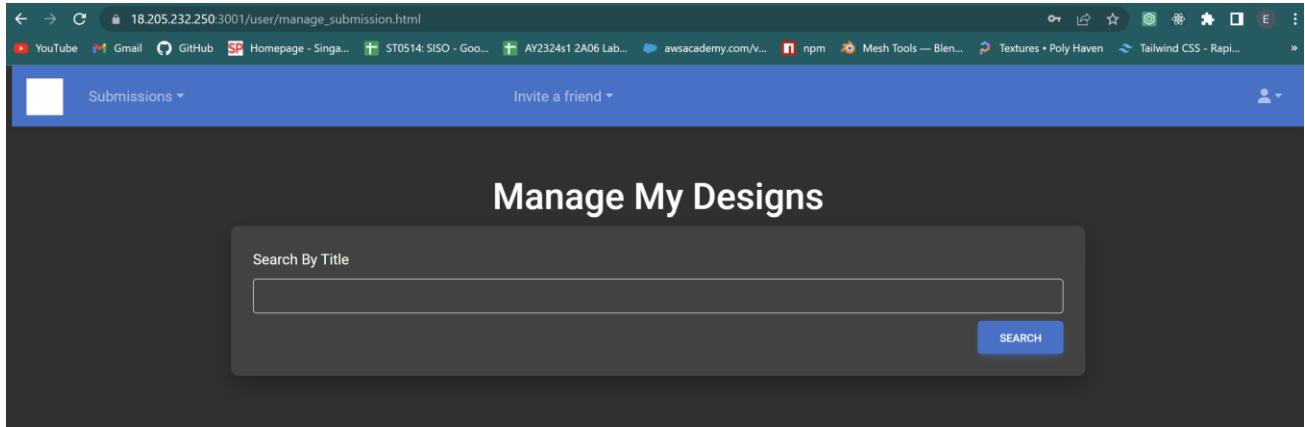
The screenshot shows the WinSCP Editor interface with a file path of '/home/ec2-user/ESDE/FE/public/js/admin_check_user_submission.js'. The code is written in JavaScript and contains a section for handling a form submission:

```
let $searchUserFormContainer = $("#searchUserFormContainer");
if ($searchUserFormContainer.length != 0) {
    console.log(
        "Search user form detected in user manage submission interface. Binding event handling"
    );
    //If the jQuery object which represents the form element exists,
    //the following code will create a method to send key-value pair information to do record
    //to server-side api when the #submitButton element fires the click event.
    $("#submitButton").on("click", function (event) {
        event.preventDefault();
        const baseUrl = "https://18.205.232.250:5000";
        let searchInput = $("#searchInput").val();
        // let userId = sessionStorage.getItem("user_id");
        let token = sessionStorage.getItem("token");
        axios({
            headers: {
                // user: userId,
                authorization: `Bearer ${token}`
            }
        })
```

A red arrow points to the line 'const baseUrl = "https://18.205.232.250:5000";' indicating where the localhost address should be changed.

Step 2:

Finally, after changing all localhost to your Elastic IP, run the respective index.js files and see if you can login as shown below.

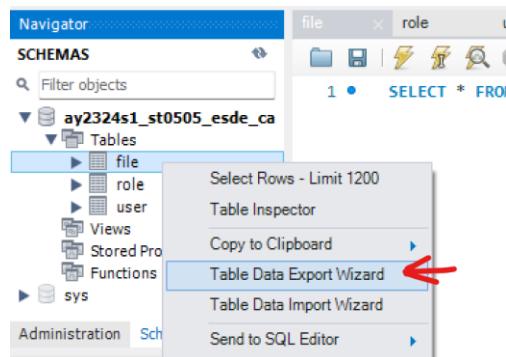


Part 4 – DynamoDB

4.1 Exporting the table data from MySQL Workbench

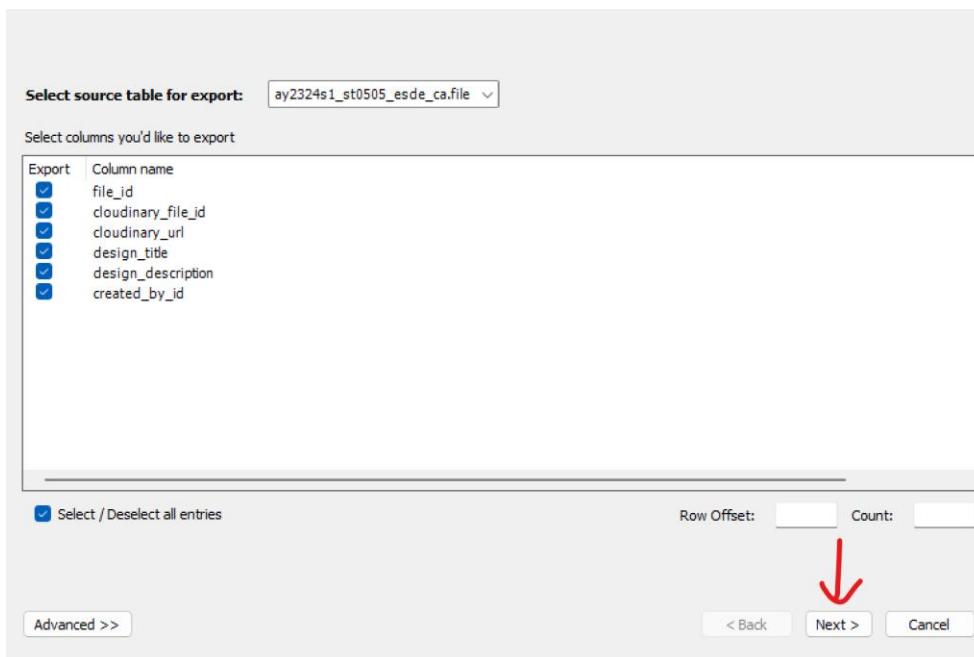
Step 1:

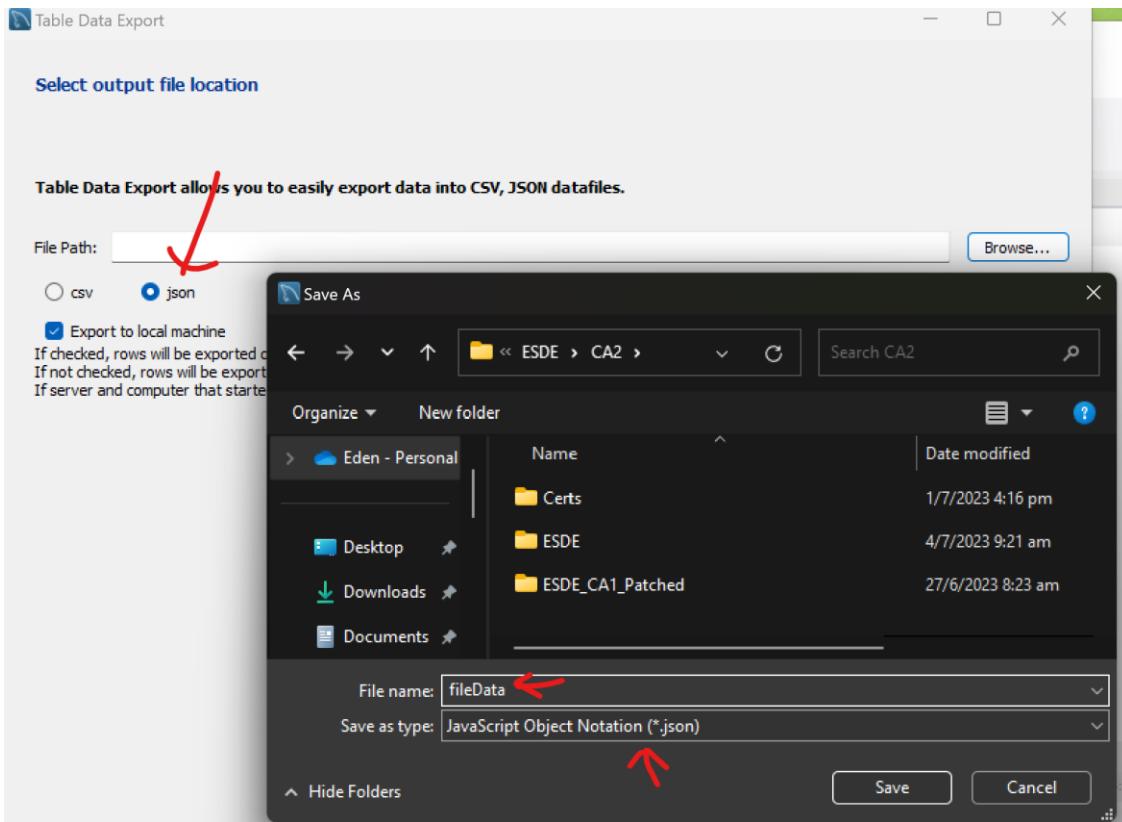
Go into MySQL Workbench and select the file table, right click, and select the “Table Data Export Wizard”.



Step 2:

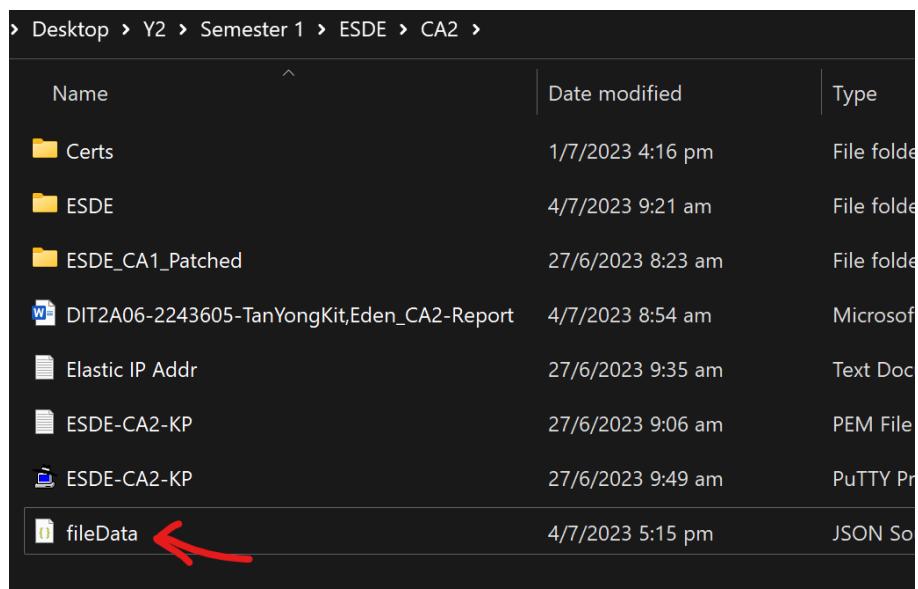
Click on Next and make sure that the file type is JSON. For the file name you can name it as fileData and save it.





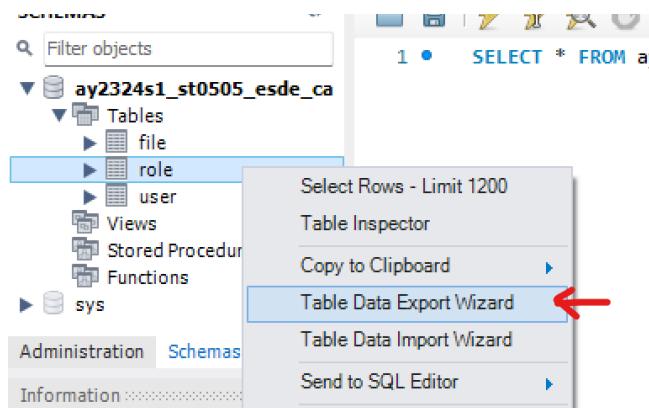
Step 3:

Afterwards, click Next and Next and Next and Finish and you should see the fileData in the directory you saved at.



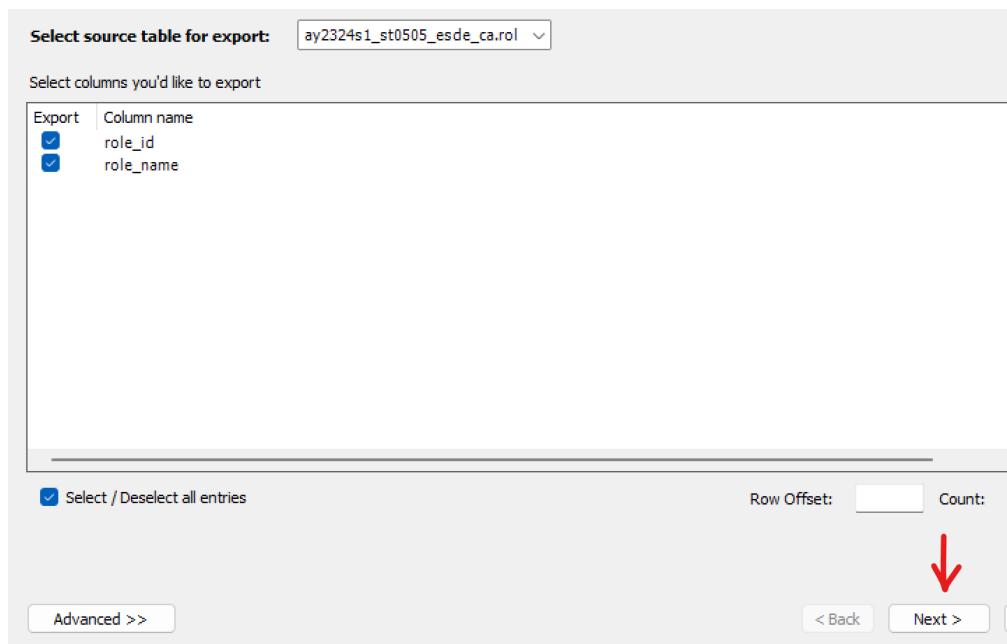
Step 4:

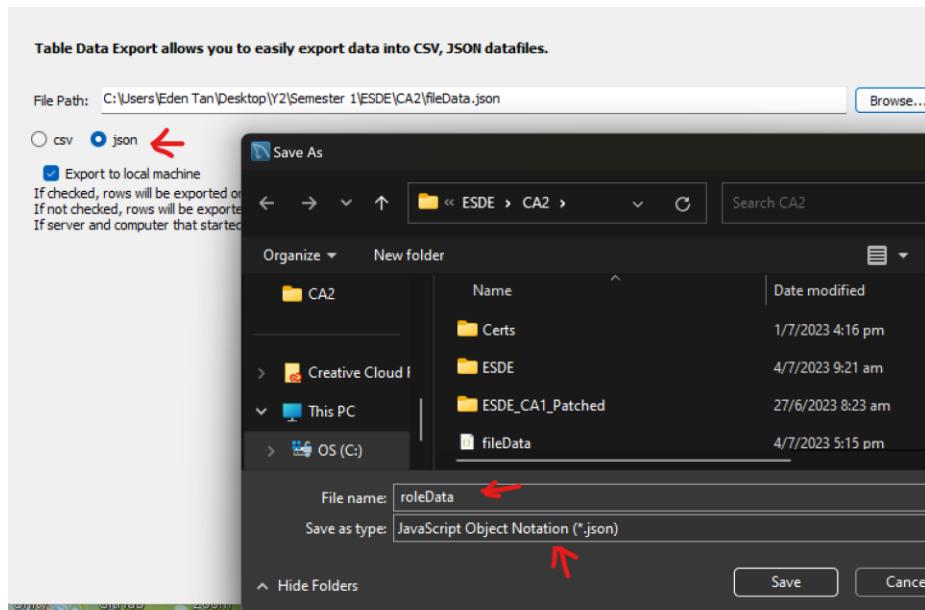
Now, let's do the same thing for the role table. Go to MySQL Workbench and right click and choose the "Table Data Export Wizard".



Step 5:

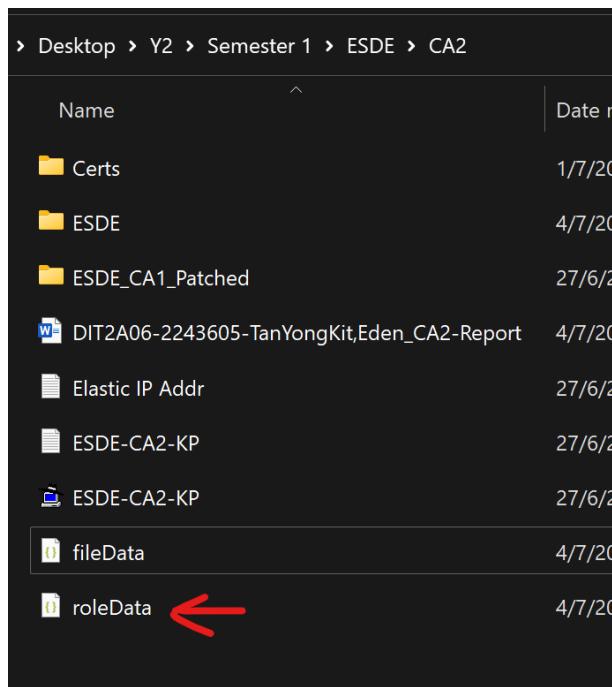
Click Next, and make sure that the file type JSON and you can name the file as roleData.





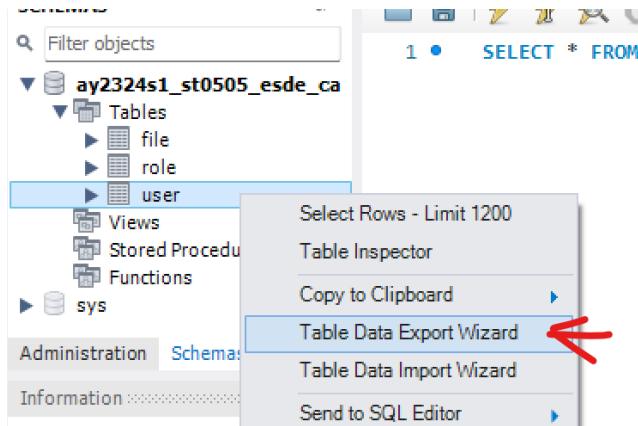
Step 6:

Afterwards, click Next and Next and Next and Finish and you should see the fileData in the directory you saved at.



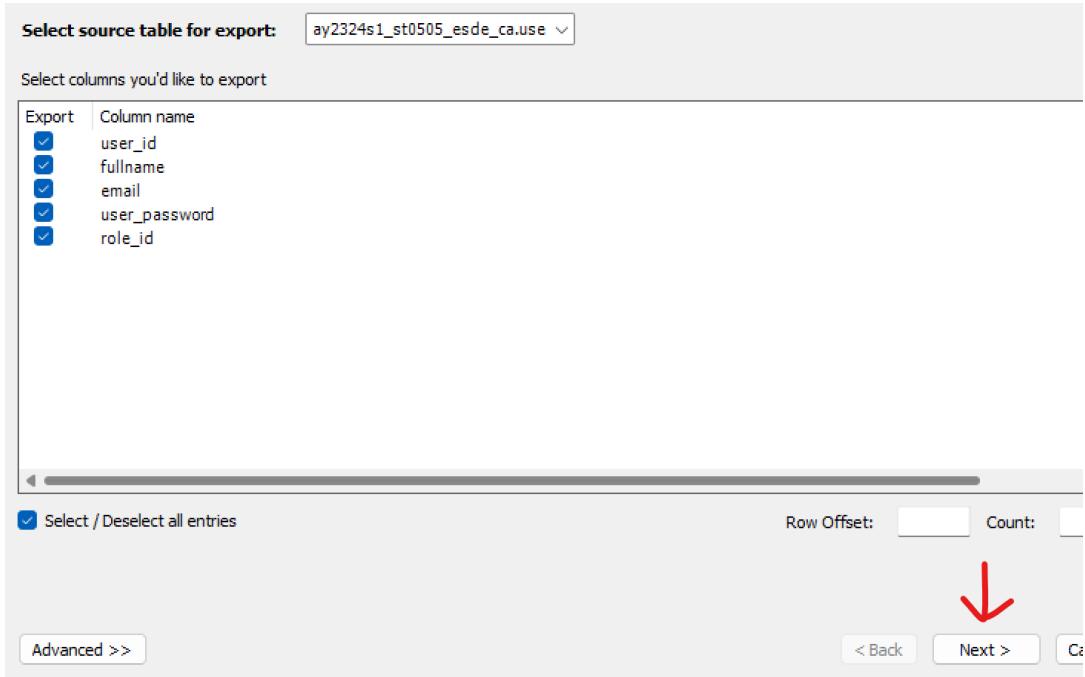
Step 7:

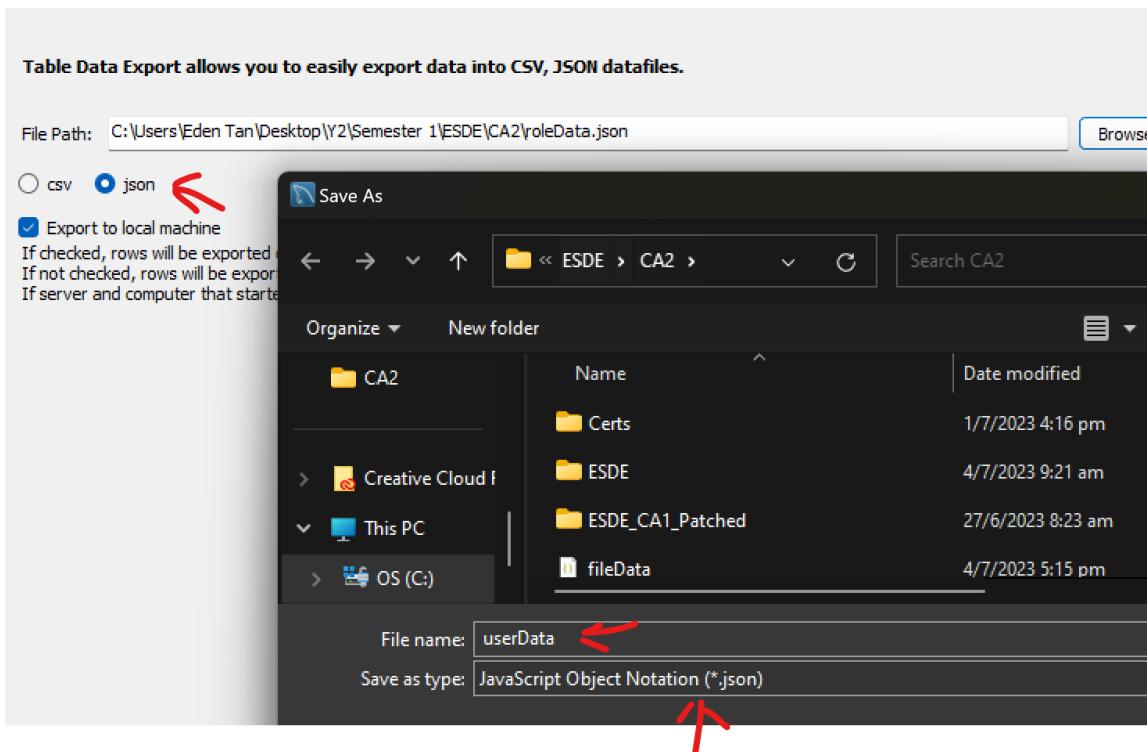
Finally, let's do the same thing for the user table. Go to MySQL Workbench and right click and choose the “Table Data Export Wizard”.



Step 8:

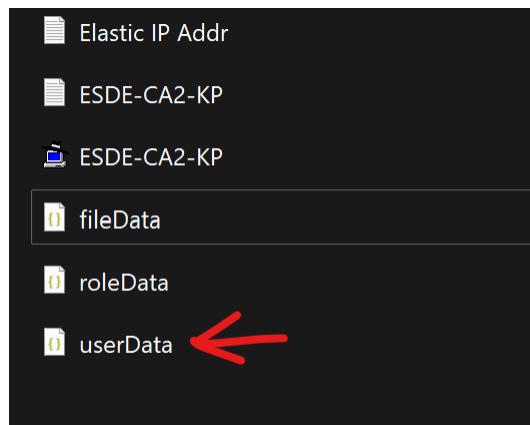
Click Next, and make sure that the file type JSON and you can name the file as userData.





Step 9:

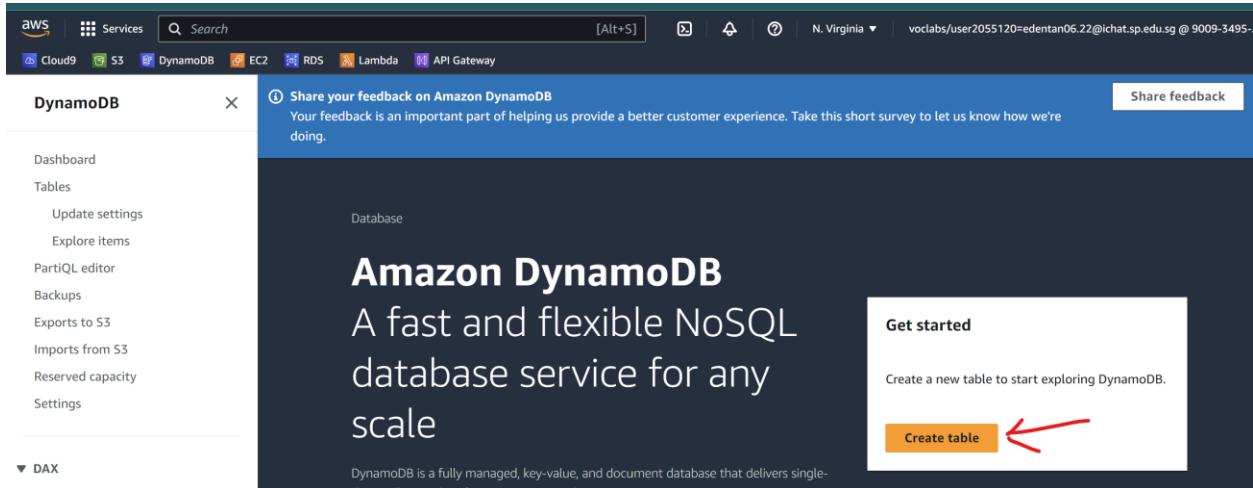
Afterwards, click Next and Next and Next and Finish and you should see the fileData in the directory you saved at.



4.2 Create a DynamoDB table for each table in MySQL Workbench

Step 1:

Go to DynamoDB in AWS and click on the “Create table” button.



Step 2:

We will create a DynamoDB table called file and a Partition Key called file_id with Number selected and click on “Create table” button as shown below.

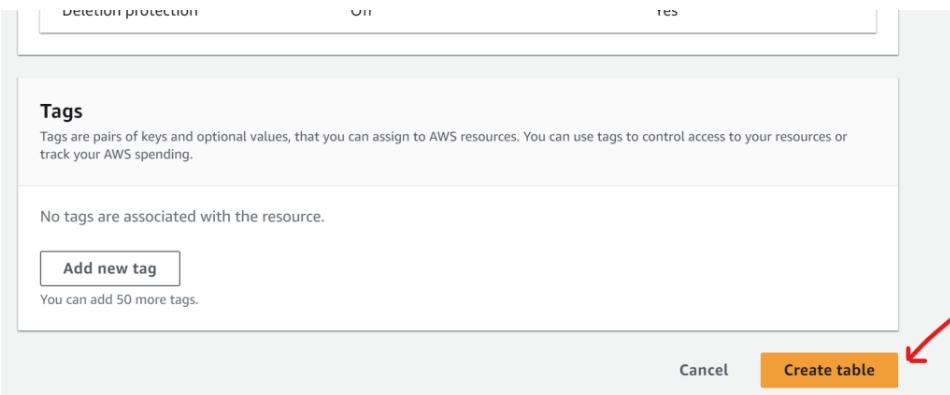
Create table

Table details Info
DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
 Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate hosts for scalability and availability.
 1 to 255 characters and case sensitive.

Sort key - optional



Step 3:

You should be able to see a file table created.

Tables (1) Info								
<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode
<input type="checkbox"/>	file	Active	file_id (N)	-	0	Off	Provisioned with auto scaling (5)	Provided by AWS

Step 4:

We will create another DynamoDB table called role and a Partition Key called role_id with Number selected and click on “Create table” button as shown below.

Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

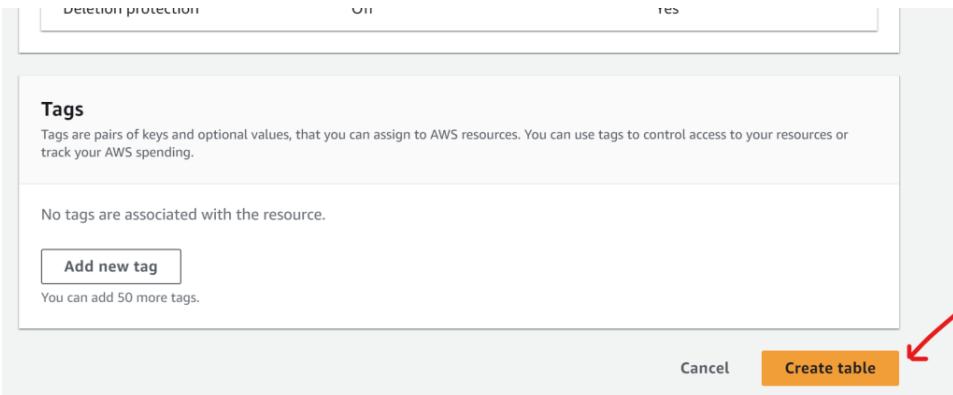
Table name
This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data hosts for scalability and availability.

1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.



Step 5:

You should be able to see a role table created.

Tables (2) Info						
	Name	Status	Partition key	Sort key	Indexes	Deletion protection
<input type="checkbox"/>	file	<input checked="" type="checkbox"/> Active	file_id (N)	-	0	<input type="checkbox"/> Off
<input type="checkbox"/>	role	<input checked="" type="checkbox"/> Active	role_id (N)	-	0	<input type="checkbox"/> Off

Step 6:

Finally, we will create another DynamoDB table called user and a Partition Key called user_id with Number selected and click on “Create table” button as shown below.

Create table

Table details Info
DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
 ▾
1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
 ▾
Delete protection On Off

Tags
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.
No tags are associated with the resource.

You can add 50 more tags.

↗

Step 7:

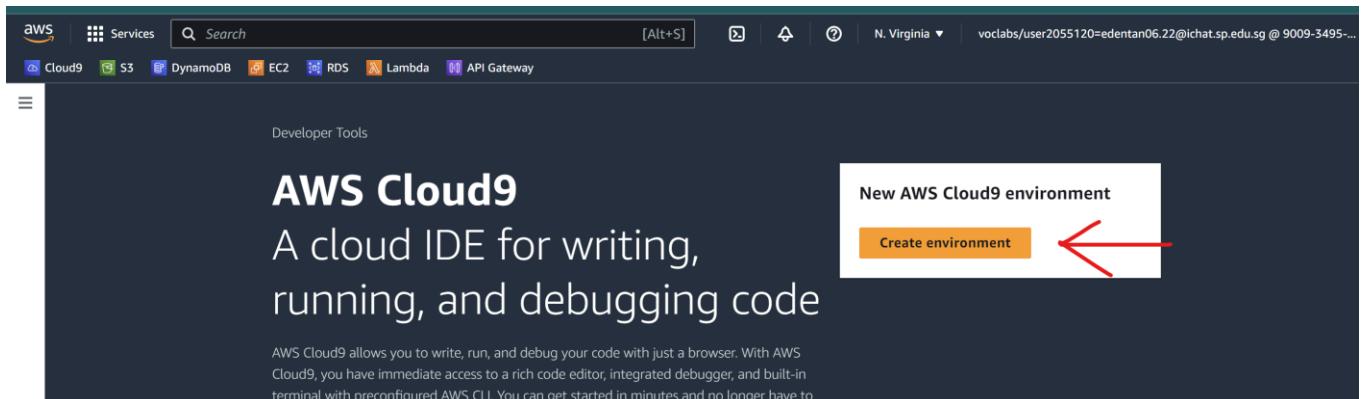
You should be able to see a user table created.

Tables (3) Info							Actions
	Name	Status	Partition key	Sort key	Indexes	Deletion protection	R
<input type="checkbox"/>	file	<input checked="" type="checkbox"/> Active	file_id (N)	-	0	<input type="checkbox"/> Off	P
<input type="checkbox"/>	role	<input checked="" type="checkbox"/> Active	role_id (N)	-	0	<input type="checkbox"/> Off	P
<input type="checkbox"/>	user	<input checked="" type="checkbox"/> Active	user_id (N)	-	0	<input type="checkbox"/> Off	P

4.3 Adding the MySQL Workbench data to DynamoDB using Cloud9

Step 1:

Go to Cloud9 in AWS and click on the “Create environment” button to create a new Cloud9 Environment.



Step 2:

Give a name for your Environment and make sure that the Environment and Instance type, Platform is the default as shown below.

Create environment [Info](#)

Details

Name 

Limit of 60 characters, alphanumeric, and unique per user.

Description - *optional*

Limit 200 characters.

Environment type [Info](#)
Determines what the Cloud9 IDE will run on.

New EC2 instance
Cloud9 creates an EC2 Instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.

Existing compute
You have an existing instance or server that you'd like to use.

New EC2 instance

Instance type [Info](#)
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.

t2.micro (1 GiB RAM + 1 vCPU)
Free-tier eligible. Ideal for educational users and exploration.

t3.small (2 GiB RAM + 2 vCPU)
Recommended for small web projects.

m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and most general-purpose development.

Additional instance types
Explore additional instances to fit your need.

Platform [Info](#)
This will be installed on your EC2 instance. We recommend Amazon Linux 2.

Amazon Linux 2

Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.

30 minutes

Step 3:

Select Secure Shell (SSH) under Network Settings and click on Create.

Network settings [Info](#)

Connection
How your environment is accessed.

AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).

Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.

▶ VPC settings [Info](#)

▶ Tags - optional [Info](#)
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

The following IAM resources will be created in your account

- AWSServiceRoleForAWSCloud9 - AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. [Learn more](#)

[Cancel](#) [Create](#)

Step 4:

You should be able to see an Environment being created as shown below, click on Open to go to the Cloud9 IDE.

[Success](#) Successfully created ESDE_CA2. To get the most out of your environment, see [Best practices for using AWS Cloud9](#)

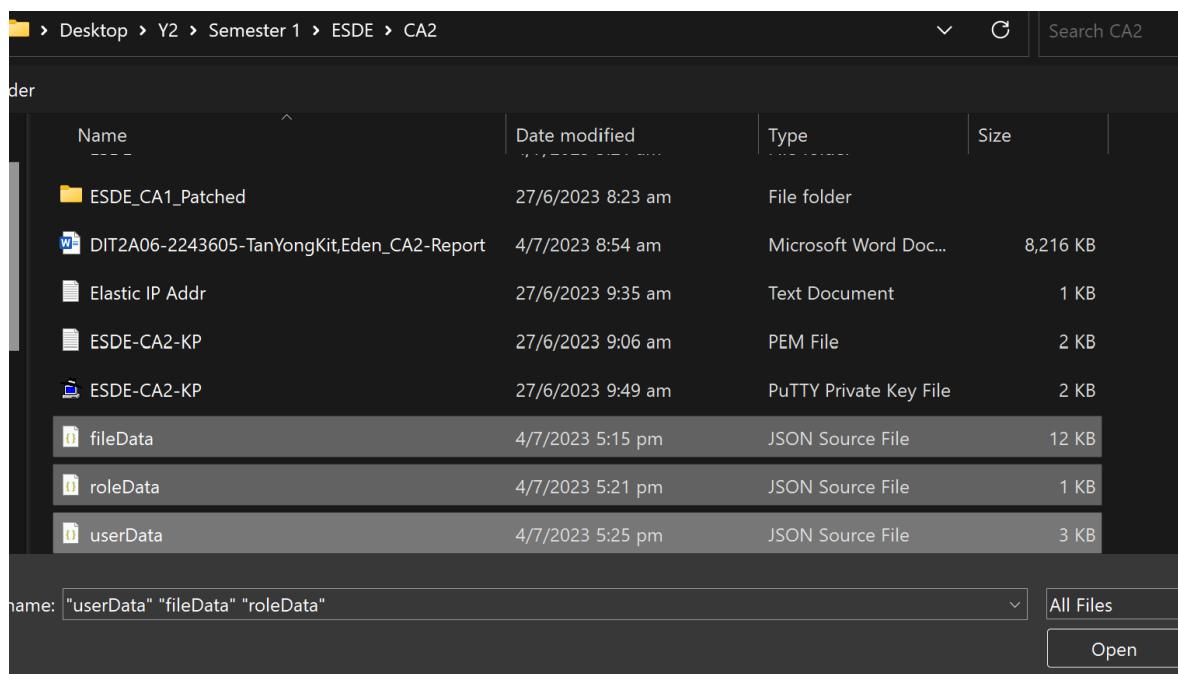
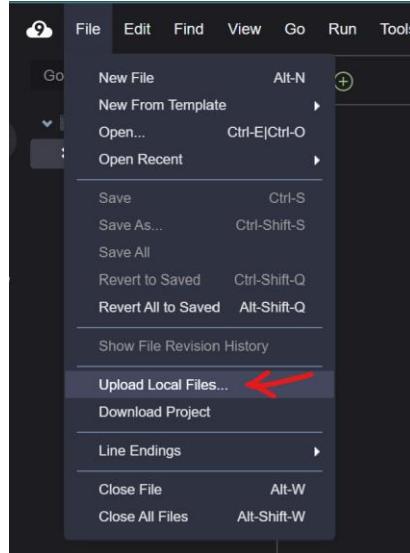
AWS Cloud9 > Environments

Environments (1)					
	Name	Cloud9 IDE	Environment type	Connection	Permission
<input type="radio"/>	ESDE_CA2	Open	EC2 instance	Secure Shell (SSH)	Owner

[Delete](#) [View details](#) [Open in Cloud9](#)

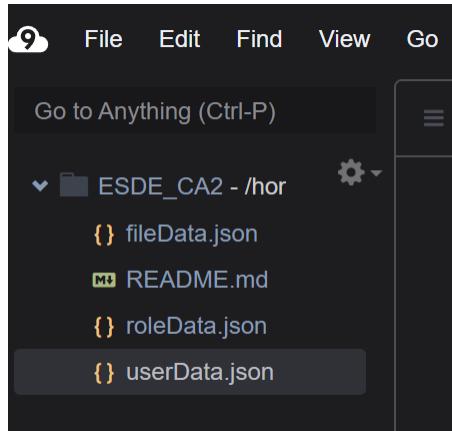
Step 5:

Click on File > Upload Local Files... to import the 3 JSON files that we got from just now.



Step 6:

You should be able to see the 3 files in your Cloud9 Environment as shown below.



Step 7:

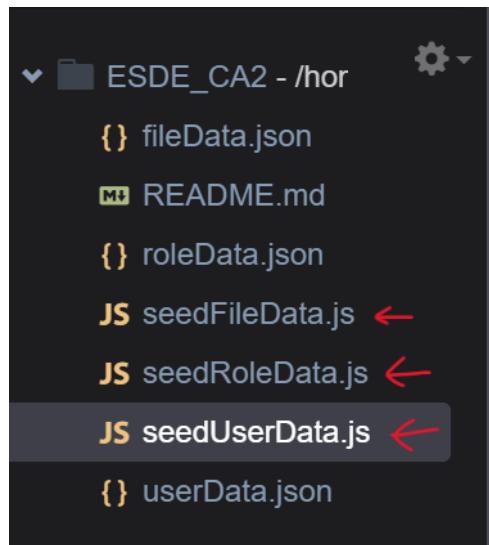
In the terminal run the following command.

Command: `npm install aws-sdk`

A screenshot of a terminal window. The title bar says "bash - 'ip-172-31-68-86.e.x'" and the tab bar says "Immediate". The main area of the terminal shows the command `voclabs:~/environment $ npm install aws-sdk` being typed. The cursor is at the end of the command.

Step 8:

Create a new file in your environment called seedFileData.js, seedRoleData.js and seedUserData.js as shown below.



Step 9:

Under seedFileData.js input the following code to seed the File DynamoDB Table.

```
var
  AWS = require("aws-sdk"),
  DDB = new AWS.DynamoDB({
    apiVersion: "2012-08-10",
    region: "us-east-1"
  },
  file_DATA_ARR = require("./fileData.json");

function addNewItemFromJSON(){
  console.log("All items now removed, re-seeding now");
  var
    file = {},
    num_items_left = file_DATA_ARR.length,
    offset_index = 0;
  console.log('num_items_left initial', num_items_left)

  while(num_items_left > 0){
```

```

var
    file_formatted_arr = [],
    params = {},
    item_added_count = 0;
    for(var i_int = offset_index; i_int < offset_index + Math.min(25,
num_items_left); 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);
    item_added_count = item_added_count + 1
}

```

```

params = {
    RequestItems: {
        "file": file_formatted_arr.reverse()
    }
};

DDB.batchWriteItem(params, function(err, data){
    if(err){
        throw err;
    }
    console.log("OK");
});

num_items_left = num_items_left-item_added_count;
offset_index = offset_index+item_added_count;
console.log('num_items_left', num_items_left)
}

(function init(){
    addNewItemFromJSON();
})();

```

Step 10:

Under seedRoleData.js input the following code to seed the Role DynamoDB Table.

```

var
    AWS = require("aws-sdk"),
    DDB = new AWS.DynamoDB({
        apiVersion: "2012-08-10",
        region: "us-east-1"
    }),
    file_DATA_ARR = require("./roleData.json");

function addNewItemFromJSON(){
    console.log("All items now removed, re-seeding now");
    var
        file = {},

```

```

        num_items_left = file_DATA_ARR.length,
        offset_index = 0;
    console.log('num_items_left initial', num_items_left)

    while(num_items_left > 0){
        var
            file_formatted_arr = [],
            params = {},
            item_added_count = 0;
        for(var i_int = offset_index; i_int < offset_index+Math.min(25,
num_items_left); 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);
            item_added_count = item_added_count + 1
        }
        params = {
            RequestItems: {
                "role": file_formatted_arr.reverse()
            }
        };
    DDB.batchWriteItem(params, function(err, data){
        if(err){
            throw err;
        }
        console.log("OK");
    });
}

```

```

        num_items_left = num_items_left-item_added_count;
        offset_index = offset_index+item_added_count;
        console.log('num_items_left', num_items_left)
    }
}

(function init(){
    addNewItemsFromJSON();
})();

```

Step 11:

Under seedUserData.js input the following code to seed the User DynamoDB Table.

```

var
AWS = require("aws-sdk"),
DDB = new AWS.DynamoDB({
    apiVersion: "2012-08-10",
    region: "us-east-1"
}),
file_DATA_ARR = require("./userData.json");

function addNewItemsFromJSON(){
    console.log("All items now removed, re-seeding now");
    var
        file = {},
        num_items_left = file_DATA_ARR.length,
        offset_index = 0;
    console.log('num_items_left initial', num_items_left)

    while(num_items_left > 0){
        var
            file_formatted_arr = [],
            params = {},
            item_added_count = 0;
            for(var i_int = offset_index; i_int < offset_index+Math.min(25,
num_items_left); 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);
        item_added_count = item_added_count + 1
    }
    params = {
        RequestItems: {
            "user": file_formatted_arr.reverse()
        }
    };
    DDB.batchWriteItem(params, function(err, data){
        if(err){
            throw err;
        }
        console.log("OK");
    });
    num_items_left = num_items_left-item_added_count;
    offset_index = offset_index+item_added_count;
}

```

```
        console.log('num_items_left', num_items_left)
    }
}

(function init(){
    addNewItemsFromJSON();
})();
```

Step 12:

In the bash terminal run the following command to seed the file table in DynamoDB.

Command: node seedFileData.js

```
vocabs:~/environment $ node seedFileData.js ↗
All items now removed, re-seeding now
num_items_left initial 36
num_items_left 11
num_items_left 0
(node:23005) NOTE: We are formalizing our plans to enter AWS SDK for JavaScript (v2) into maintenance mode in 2023.

Please migrate your code to use AWS SDK for JavaScript (v3).
For more information, check the migration guide at https://a.co/7PzMCCy
(Use `node --trace-warnings ...` to show where the warning was created)
OK
```

Step 13:

You should be able to see the file JSON data being populated into the DynamoDB file table.

Items returned (36)						
	file_id	▲	cloudinary_file_id	▼	cloudinary_url	▼
	created_by_id	▼	design_description	▼	design_title	▼
□	100		Design/uvty9qgblevpd...	https://res.cloudi...	100	kelly design 1 descripti...
□	101		Design/mbj7zdakynu3...	https://res.cloudi...	100	kelly design 2 descripti...
□	102		Design/oj3ht73cdjroh...	https://res.cloudi...	100	kelly design 3 descripti...
□	103		Design/benwneosr3q7...	https://res.cloudi...	100	kelly design 4 descripti...
□	104		Design/jwpes0y6u6ru...	https://res.cloudi...	100	kelly design 5 descripti...
□	105		Design/sjtwvvaaywtoo...	https://res.cloudi...	100	kelly design 6 descripti...
□	106		Design/iwlwf9tzn5gu7...	https://res.cloudi...	100	kelly design 7 descripti...
□	107		Design/cs3ajodrvzjpcy...	https://res.cloudi...	100	kelly design 8 descripti...
□	108		Design/res1ufzntzp23...	https://res.cloudi...	100	kelly design 9 descripti...
□	109		Design/gx7qpmsfxazjl...	https://res.cloudi...	101	bezos design 1 descrip...

Step 14:

Next, run the following command in the bash terminal to seed the DynamoDB role table.

```
vocabs:~/environment $ node seedRoleData.js ←
All items now removed, re-seeding now
num_items_left initial 2
num_items_left 0
(node:24171) NOTE: We are formalizing our plans to enter AWS SDK for JavaScript (v2) into maintenance mode in 2023.
Please migrate your code to use AWS SDK for JavaScript (v3).
For more information, check the migration guide at https://a.co/7PzMCCy
(Use `node --trace-warnings ...` to show where the warning was created)
OK
```

Step 15:

You should be able to see the role JSON data being populated into the DynamoDB role table.

Items returned (2)		
	role_id	role_name
<input type="checkbox"/>	2	user
<input type="checkbox"/>	1	admin

Step 16:

Finally, run the following command in the bash terminal to seed the DynamoDB user table.

```
voclabs:~/environment $ node seedUserData.js ←
All items now removed, re-seeding now
num_items_left initial 16
num_items_left 0
(node:24884) NOTE: We are formalizing our plans to enter AWS SDK for JavaScript (v2) into maintenance mode in 2023.

Please migrate your code to use AWS SDK for JavaScript (v3).
For more information, check the migration guide at https://a.co/7PzMCCy
(Use `node --trace-warnings ...` to show where the warning was created)
OK
```

Step 17:

You should be able to see the user JSON data being populated into the DynamoDB user table.

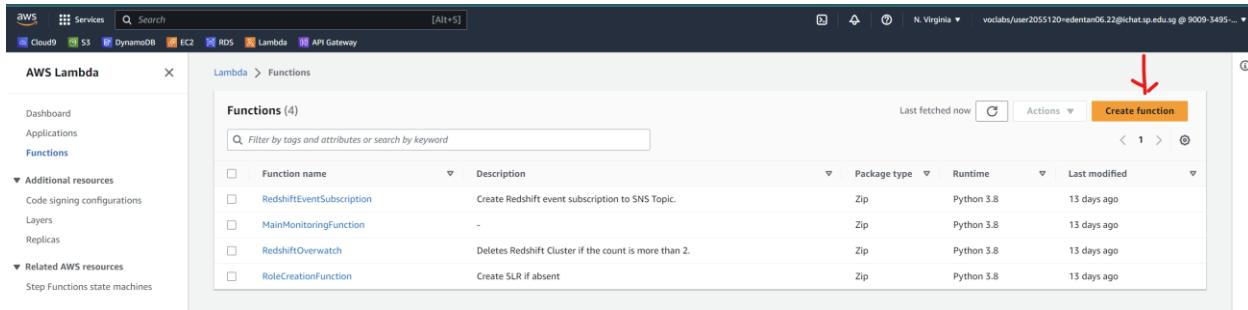
Items returned (16)						
	user_id	email	fullname	role_id	user_password	
<input type="checkbox"/>	115	hushien@d...	hushien	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	110	becky@desi...	becky	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	113	elsie@desig...	elsie	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	104	Albert@ad...	Albert	1	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	100	kelly@desi...	kelly	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	108	jacob@desi...	jacob	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	111	dylan@desi...	dylan	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	102	elon@desig...	elon	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	103	nadella@de...	nadella	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	106	christine@d...	christine	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	114	manfred@d...	manfred	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeg33.WrxJx/FeC9.gCy...	
<input type="checkbox"/>	105	alan@desiq...	alan	2	\$2b\$10\$K.0HwpsoPDGaB/atFBmmXOGTw4ceeq33.WrxJx/FeC9.qCy...	

Part 5 – Set up Lambda

5.1 Creating Lambda Function getDesigns

Step 1:

Go into AWS Lambda and click on “Create function” to create a new Lambda Function getDesigns.



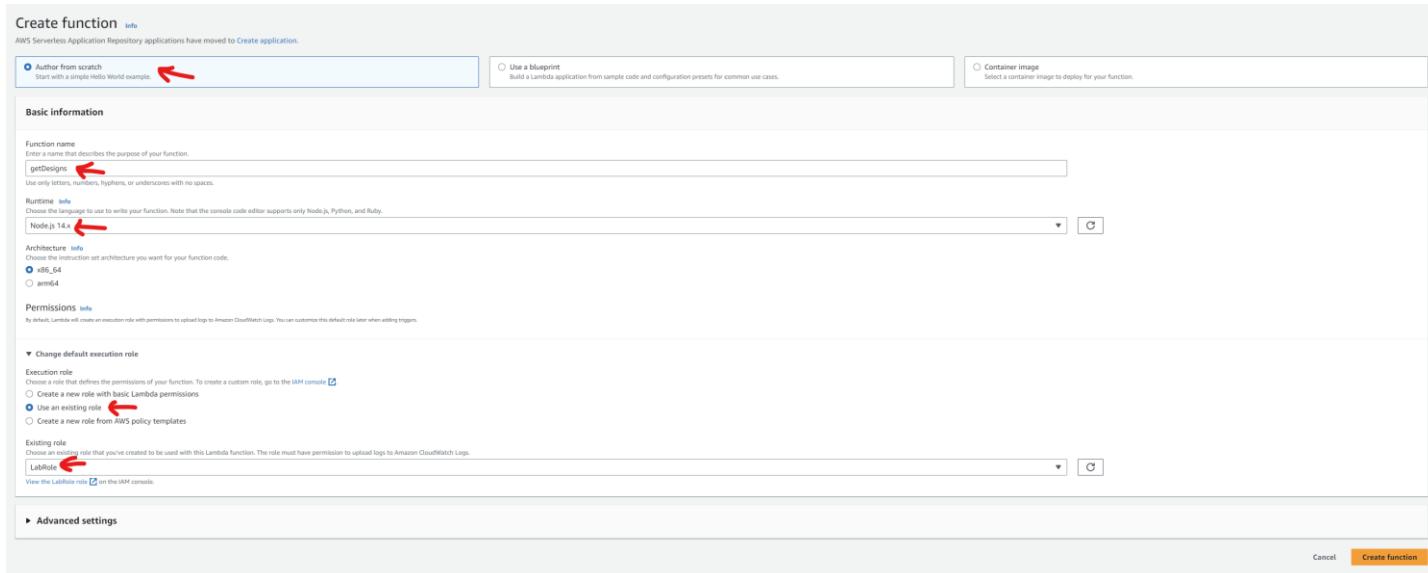
The screenshot shows the AWS Lambda console interface. In the top navigation bar, there are icons for Cloud9, S3, DynamoDB, EC2, RDS, Lambda, and API Gateway. Below the navigation bar, the main title is "AWS Lambda" and the sub-section is "Lambda > Functions". The main area is titled "Functions (4)" and contains a table with the following data:

Function name	Description	Package type	Runtime	Last modified
RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.	Zip	Python 3.8	13 days ago
MainMonitoringFunction	-	Zip	Python 3.8	13 days ago
RedshiftOverwatch	Deletes Redshift Cluster if the count is more than 2.	Zip	Python 3.8	13 days ago
RoleCreationFunction	Create SLR if absent	Zip	Python 3.8	13 days ago

At the top right of the table, there is a "Create function" button. A red arrow points to this button.

Step 2:

Give the function a name of “getDesigns”, Select “Node.js 14.x” as your runtime, Select “Use an existing role” and choose “LabRole” under Execution role and click on “Create function”.



The screenshot shows the "Create function" wizard. At the top, it says "Create function" and "AWS Serverless Application Repository applications have moved to Create application." There are three radio button options: "Author from scratch" (selected), "Use a blueprint" (Build a Lambda application from sample code and configuration presets for common use cases), and "Container image" (Select a container image to deploy for your function).

The "Basic information" section includes:

- "Function name": "getDesigns" (highlighted with a red arrow)
- "Runtime": "Node.js 14.x" (highlighted with a red arrow)
- "Architecture": "x86_64" (highlighted with a red arrow)

The "Permissions" section includes:

- "Execution role": "Use an existing role" (highlighted with a red arrow)
- "LabRole": "LabRole" (highlighted with a red arrow)

At the bottom right of the form, there are "Cancel" and "Create function" buttons.

Step 3:

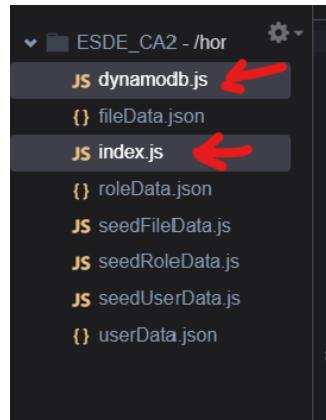
You should be able to see that the function has been successfully created as shown below.

The screenshot shows the AWS Lambda console and the Cloud9 IDE. The Lambda function 'getDesigns' is displayed with its basic configuration and triggers. Below it, the Cloud9 interface shows the 'Code source' tab with the 'index.js' file open. The file contains a simple Node.js script with an export named 'handler' that returns a JSON response with status code 200 and body 'Hello from Lambda!'. Red arrows point to the 'dynamodb.js' and 'index.js' files in the Cloud9 file list.

5.2 Creating files in Cloud9 for Lambda Function getDesigns

Step 1:

Go into Cloud9 and bring in 2 files called dynamodb.js and index.js as shown below.



dynamodb.js codes:

```
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
```

index.js codes:

```
var dynamodbQuery = require("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 region = "us-east-1";
    var table_name = "file";
    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 = { filedatal: 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
};
```

Step 2:

Run the following command as shown below to zip up the index.js and dynamodb.js into a function.zip.

Command: **zip function.zip index.js dynamodb.js**

```
voclabs:~/environment $ zip function.zip index.js dynamodb.js ↗
  adding: index.js (deflated 62%)
  adding: dynamodb.js (deflated 56%)
voclabs:~/environment $
```

Step 3:

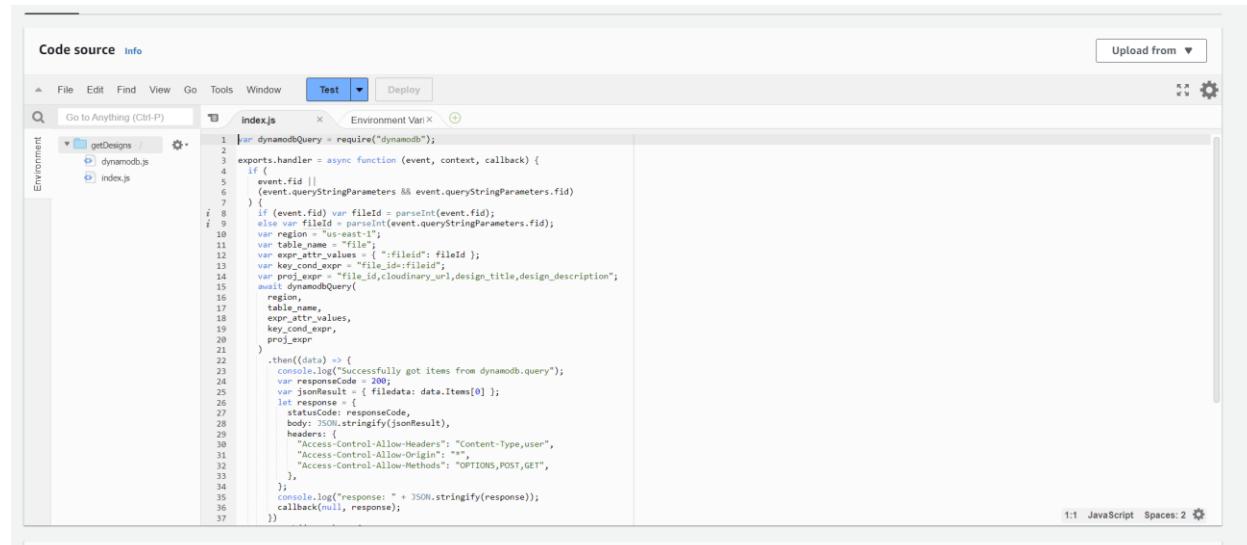
Run the following command to bring your Cloud9 dynamodb.js and index.js files into the Lambda function called getDesigns.

Command: `aws lambda update-function-code --function-name getDesigns --zip-file fileb://function.zip`

```
vocabs:~/environment $ aws lambda update-function-code --function-name getDesigns --zip-file fileb://function.zip { "LastUpdateStatus": "InProgress", "FunctionName": "getDesigns", "LastModified": "2023-07-10T06:26:13.000+0000", "RevisionId": "d9e4610d-d4f6-48b6-9dc9-e09fb75b9deb", "LastUpdateStatusReason": "The function is being created.", "MemorySize": 128, "State": "Active", "Version": "$LATEST", }
```

Step 4:

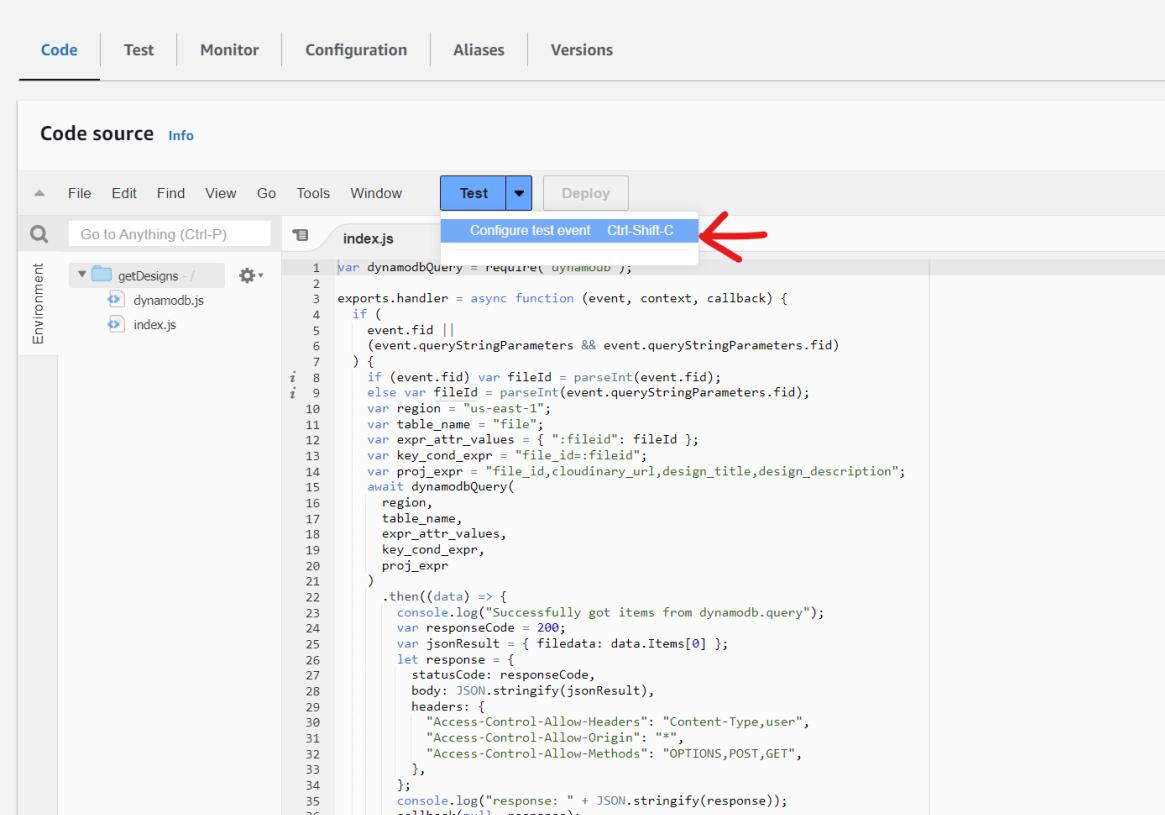
When you go back to your getDesigns Lambda function you should see the dynamodb.js and index.js files filled with the respective codes.



5.3 Configure Test Event for Lambda function getDesigns

Step 1:

Under Test, click on “Configure test event” to create a new event to test the Lambda function getDesigns.



The screenshot shows the AWS Lambda console interface. The top navigation bar has tabs for 'Code', 'Test' (which is currently selected), 'Monitor', 'Configuration', 'Aliases', and 'Versions'. Below the tabs is a toolbar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', and a search bar labeled 'Go to Anything (Ctrl-P)'. The main area is titled 'Code source' with 'Info' sub-tab selected. On the left, there's a sidebar for 'Environment' showing a folder named 'getDesigns - /' containing 'dynamodb.js' and 'index.js'. The right side shows the code editor with 'index.js' open. The code is a Node.js script for a Lambda function named 'getDesigns'. It uses the 'dynamodb' package to query a DynamoDB table named 'file'. The 'Test' tab is highlighted in blue, and a red arrow points to the 'Configure test event' button in the toolbar above the code editor.

```
1 var dynamodbQuery = require('dynamodb');
2
3 exports.handler = async function (event, context, callback) {
4     if (
5         event.fid ||
6         (event.queryStringParameters && event.queryStringParameters.fid)
7     ) {
8         if (event.fid) var fileId = parseInt(event.fid);
9         else var fileId = parseInt(event.queryStringParameters.fid);
10        var region = "us-east-1";
11        var table_name = "file";
12        var expr_attr_values = { ":fileid": fileId };
13        var key_cond_expr = "file_id=:fileid";
14        var proj_expr = "file_id,cloudinary_url,design_title,design_description";
15        await dynamodbQuery(
16            region,
17            table_name,
18            expr_attr_values,
19            key_cond_expr,
20            proj_expr
21        )
22            .then((data) => {
23                console.log("Successfully got items from dynamodb.query");
24                var responseCode = 200;
25                var jsonResult = { filedatal: data.Items[0] };
26                let response = {
27                    statusCode: responseCode,
28                    body: JSON.stringify(jsonResult),
29                    headers: {
30                        "Access-Control-Allow-Headers": "Content-Type,user",
31                        "Access-Control-Allow-Origin": "*",
32                        "Access-Control-Allow-Methods": "OPTIONS,POST,GET",
33                    },
34                };
35                console.log("response: " + JSON.stringify(response));
36                callback(null, response);
37            });
38        }
39    }
40}
```

Step 2:

Give the test event a name of “Test” and replace the Event JSON with the following as shown below.

Note: The following is to retrieve the Kelly Design 1.

Configure test event X

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

Create new event Edit saved event

Event name

Test

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

Event JSON Format JSON

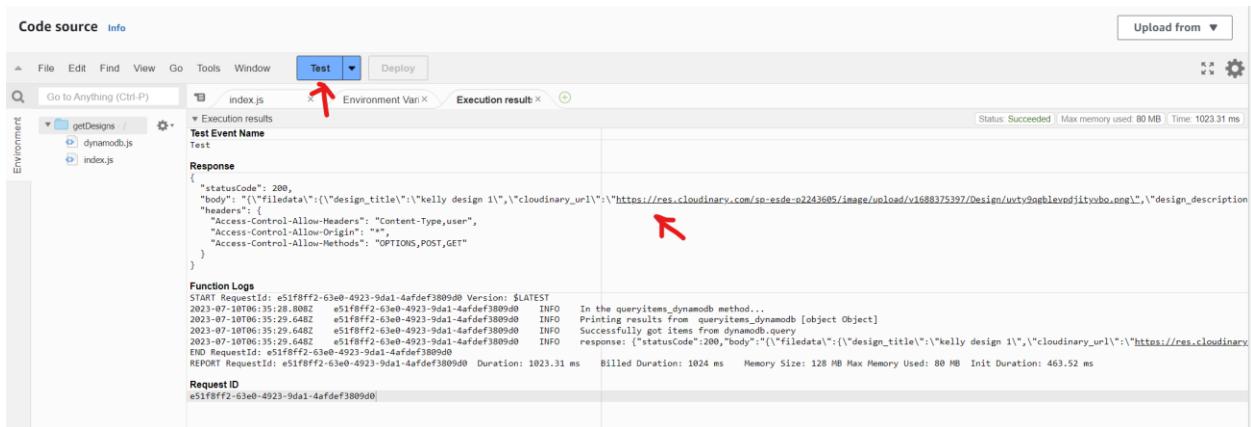
```
1+ {  
2   "fid": 100  
3 }
```




Cancel Save

Step 3:

Finally, run the Test and you should see that the response shows Kelly's Design 1.



The screenshot shows the AWS Lambda function configuration interface. At the top, there are tabs for 'Code source' and 'Info'. Below the tabs is a toolbar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', a 'Test' button (which has a red arrow pointing to it), and a 'Deploy' button. The 'Test' button is highlighted with a blue background. To the right of the toolbar is a dropdown menu labeled 'Upload from' and a gear icon. The main area contains a search bar ('Go to Anything (Ctrl-P)'), a file tree ('Environment', 'getDesigns', 'dynamodb.js', 'index.js'), and two tabs: 'Execution results' and 'Execution result'. The 'Execution result' tab is selected and shows the following output:

```
Response
{
  "statusCode": 200,
  "body": "{\"filedata\": \"\\\"kelly design 1\\\", \"cloudinary_url\": \"https://res.cloudinary.com/sp-esde-e2243605/image/upload/v1688375397/Design/unvty9agblevndjtyxbo.png\", \"design_description\": \"\", \"Access-Control-Allow-Headers\": \"Content-Type, user\", \"Access-Control-Allow-Origin\": \"\", \"Access-Control-Allow-Methods\": \"OPTIONS,POST,GET\"}"
}

Function Logs
START RequestId: e51ff0ff2-63e0-4923-9da1-4afdef3809d0 Version: $LATEST
2023-07-10T06:35:29.648Z e51ff0ff2-63e0-4923-9da1-4afdef3809d0 INFO In the queryItems_dynamodb method...
2023-07-10T06:35:29.648Z e51ff0ff2-63e0-4923-9da1-4afdef3809d0 INFO Printing results from queryItems_dynamodb [object Object]
2023-07-10T06:35:29.648Z e51ff0ff2-63e0-4923-9da1-4afdef3809d0 INFO Successfully got items from dynamodb.query
2023-07-10T06:35:29.648Z e51ff0ff2-63e0-4923-9da1-4afdef3809d0 INFO response: {"statusCode":200,"body": "{\"filedata\": \"\\\"kelly design 1\\\", \"cloudinary_url\": \"https://res.cloudinary.com/sp-esde-e2243605/image/upload/v1688375397/Design/unvty9agblevndjtyxbo.png\", \"design_description\": \"\", \"Access-Control-Allow-Headers\": \"Content-Type, user\", \"Access-Control-Allow-Origin\": \"\", \"Access-Control-Allow-Methods\": \"OPTIONS,POST,GET\"}"}
END RequestId: e51ff0ff2-63e0-4923-9da1-4afdef3809d0
REPORT RequestId: e51ff0ff2-63e0-4923-9da1-4afdef3809d0 Duration: 1023.31 ms Billed Duration: 1024 ms Memory Size: 128 MB Max Memory Used: 80 MB Init Duration: 463.52 ms
RequestID
e51ff0ff2-63e0-4923-9da1-4afdef3809d0
```

Part 6 – API Gateway

6.1 Creating an API Gateway for Lambda Function getDesigns

Step 1:

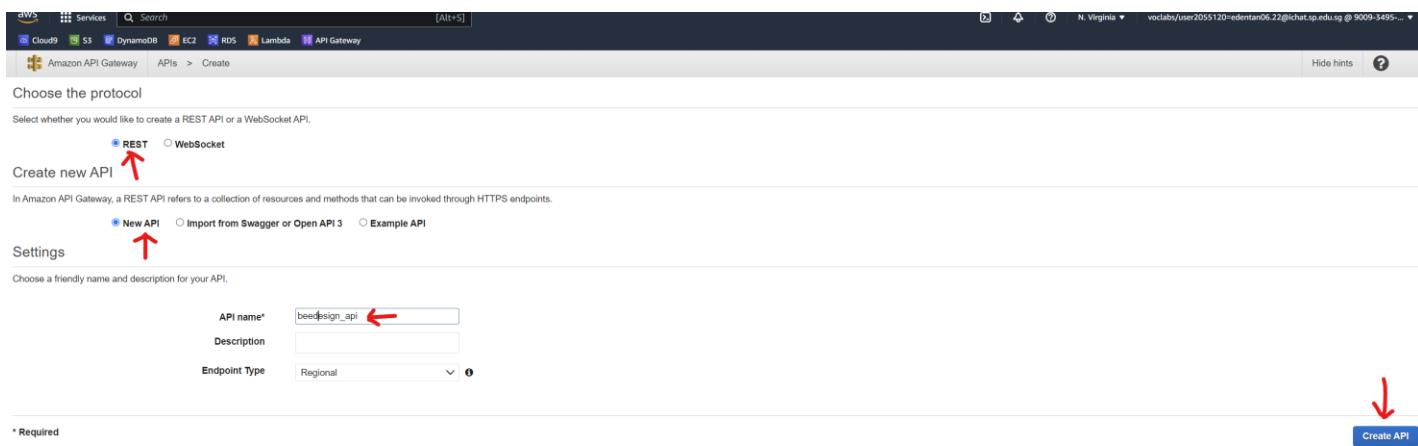
Go into AWS API Gateway Dashboard and click on the “Build” under REST API.

The screenshot shows the AWS API Gateway dashboard with the 'APIs' tab selected. A modal window titled 'Choose an API type' is open. It lists four options: 'HTTP API', 'WebSocket API', 'REST API', and 'REST API Private'. Each option has a brief description and a 'Build' button. The 'REST API' option is highlighted with a red arrow pointing to it. Another red arrow points to the 'Build' button for the REST API section.

API Type	Description	Action Buttons
HTTP API	Build low-latency and cost-effective REST APIs with built-in features such as OAuth2, and native CORS support. Works with the following: Lambda, HTTP backends	Import, Build
WebSocket API	Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards. Works with the following: Lambda, HTTP, AWS Services	Build
REST API	Develop a REST API where you gain complete control over the request and response along with API management capabilities. Works with the following: Lambda, HTTP, AWS Services	Import, Build
REST API Private	Create a REST API that is only accessible from within a VPC. Works with the following: Lambda, HTTP, AWS Services	Import

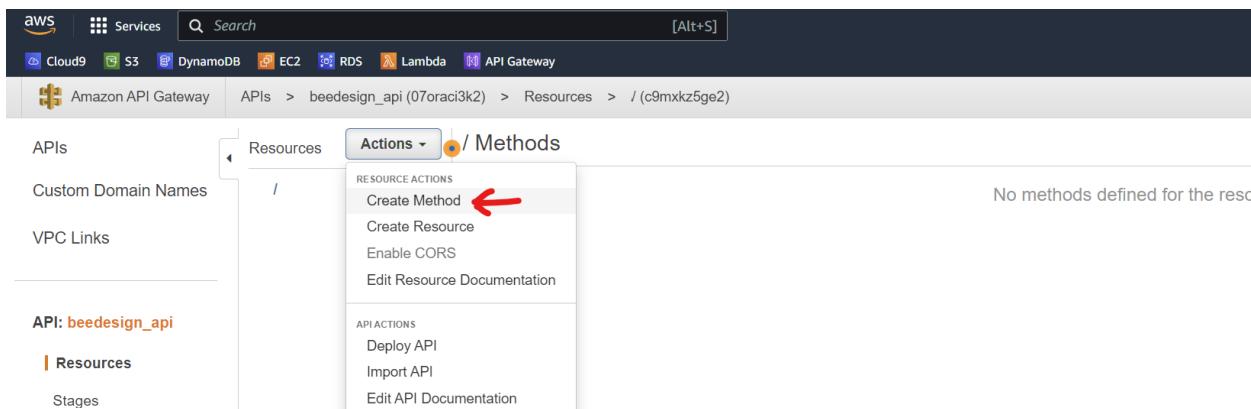
Step 2:

Make sure that you selected REST, and New API. For the API Name, name it as beedesign_api, and click on “Create API”.



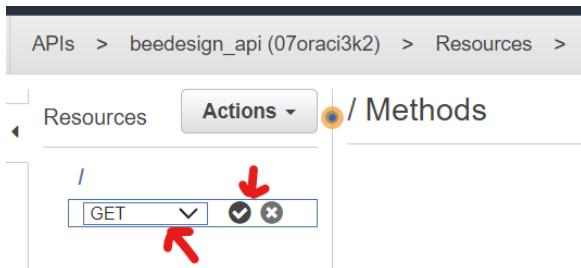
Step 3:

Click on the / (Root) and Under the Actions click on Create Method.



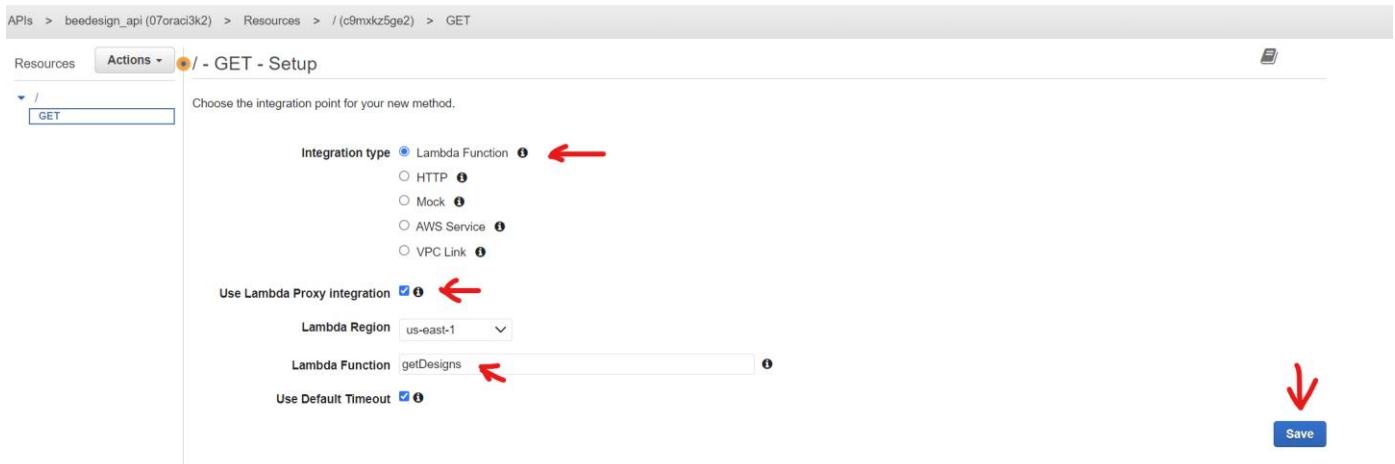
Step 4:

Under the Dropdown list select “GET” Method and click on the tick icon as shown below.



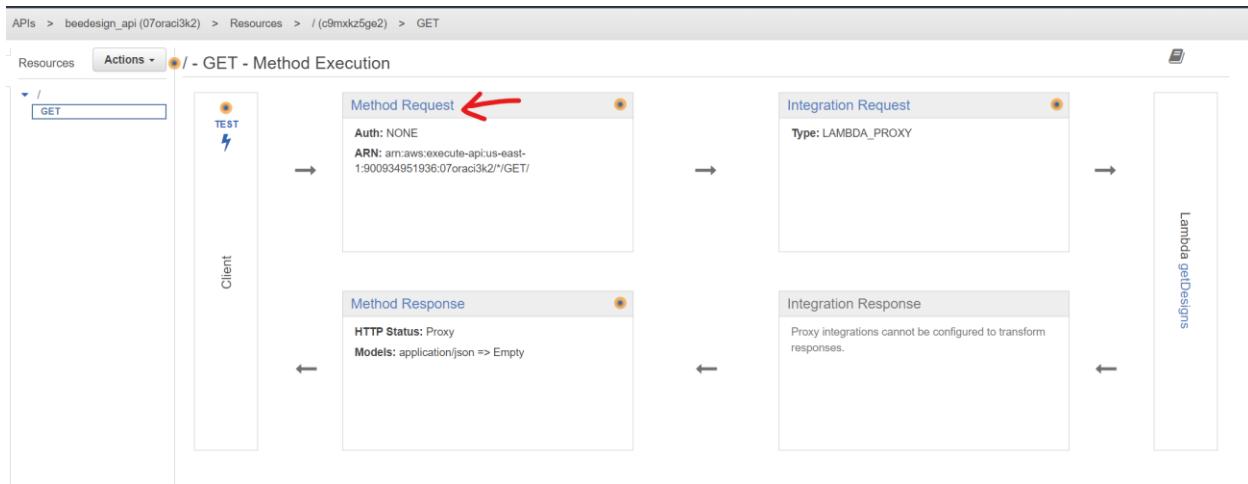
Step 5:

Make sure that you have selected the following as shown below for the GET Method. Select Lambda Function, Check “Use Lambda Proxy Integration” and for the Lambda Function name it as “getDesigns” and then click on the “Save” button.



Step 6:

You will see the same as shown below, Click on the Method Request.



Step 7:

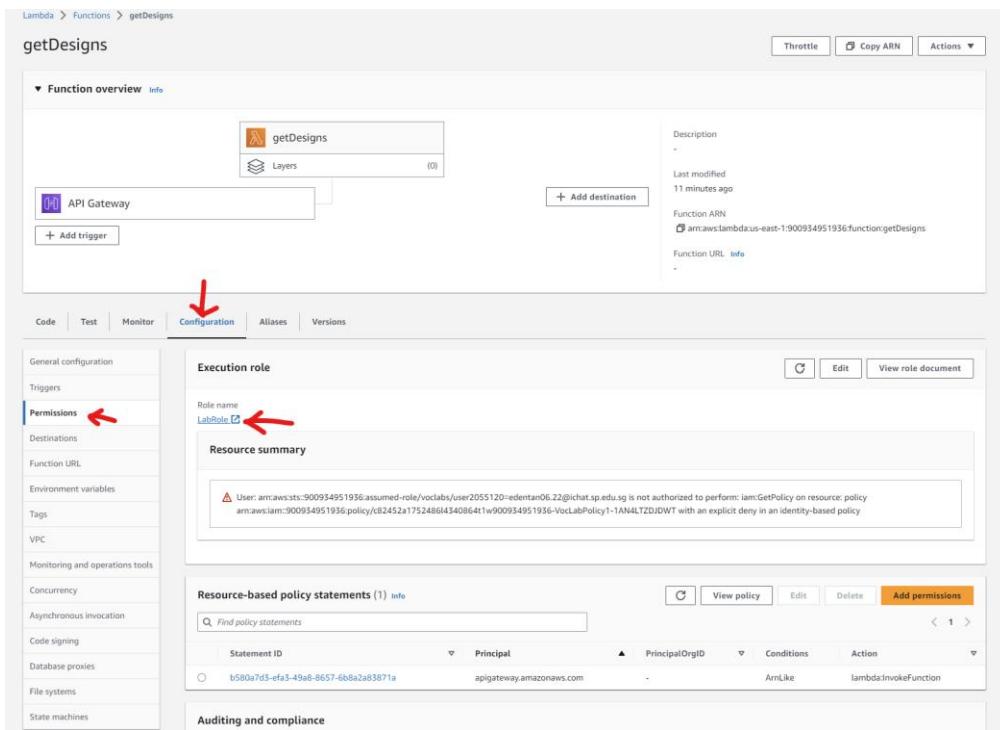
Under URL Query String Parameters add a new query string called “fid” and click on the tick icon to save it.

The screenshot shows the 'Method Request' settings for a GET method. Under 'URL Query String Parameters', a new parameter named 'fid' is added. A red arrow points to the 'fid' input field. Another red arrow points to the save icon (a checkmark) at the top right of the table.

Name	Required	Caching
fid		

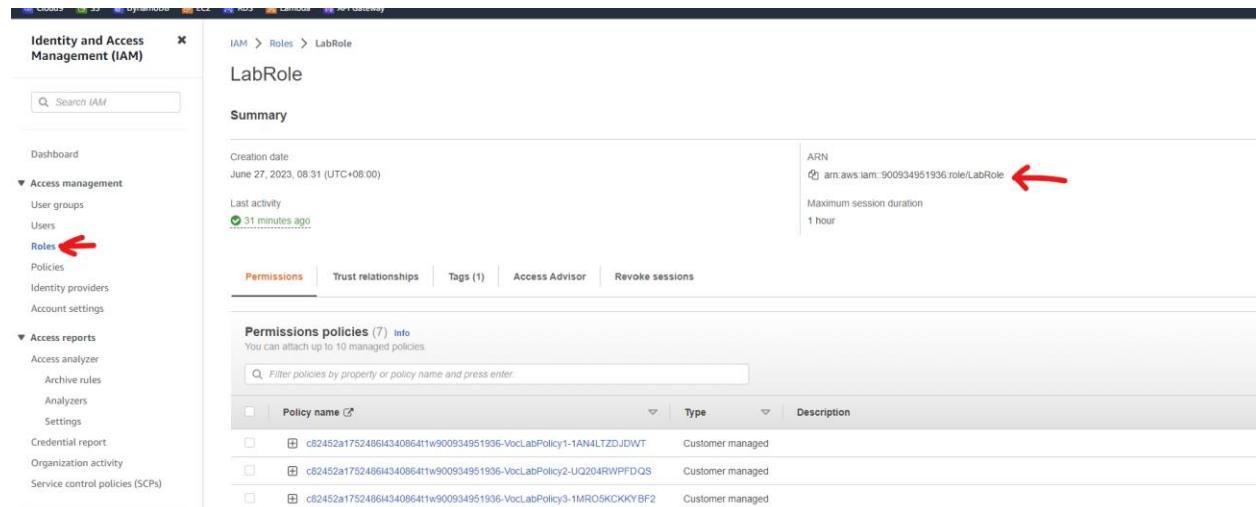
Step 8:

Next, go to your getDesigns Lambda function and click on “Configuration” and then “Permissions” as shown below and click on “LabRole” as shown below.



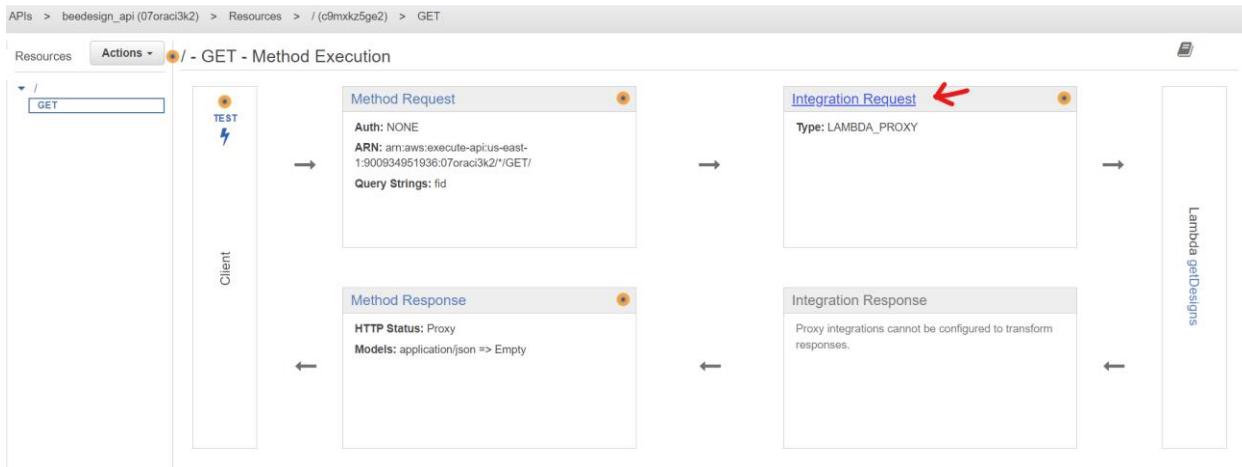
Step 9:

Under the IAM > Roles, copy the ARN and keep it somewhere where we can retrieve it later.



Step 10:

Next, go back to the GET Method in your API Gateway and click on Integration Request.



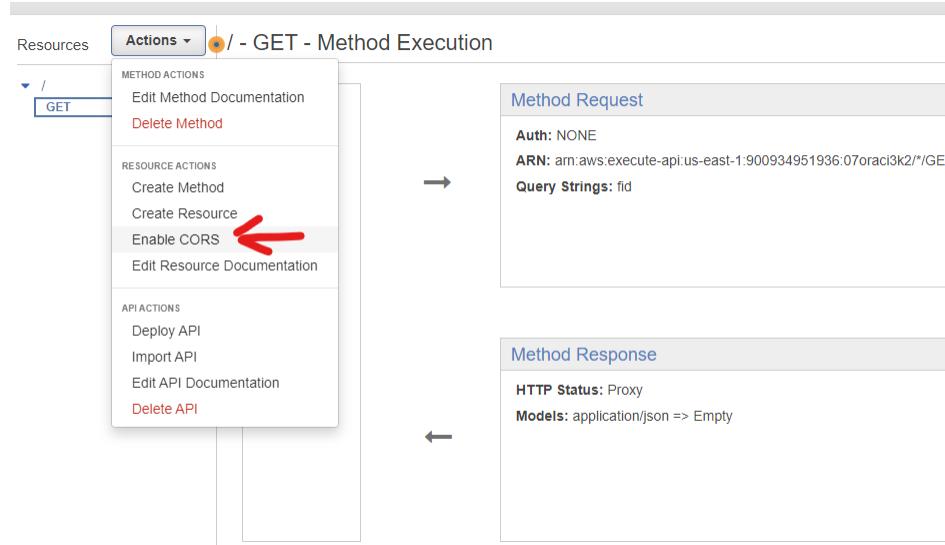
Step 11:

Paste the ARN Role in the Execution role as shown below and click on the Tick Icon to save it.

The screenshot shows the configuration for the GET method's integration request. The 'Integration type' is set to 'Lambda Function'. The 'Execution role' field contains the ARN: arn:aws:lambda:900934951936:role/LabRole, with a red arrow pointing to it. A red arrow also points to the 'Save' icon (a checkmark) in the top right corner. Other settings include 'Lambda Region' (us-east-1), 'Lambda Function' (getDesigns), and 'Invoke with caller credentials' (unchecked).

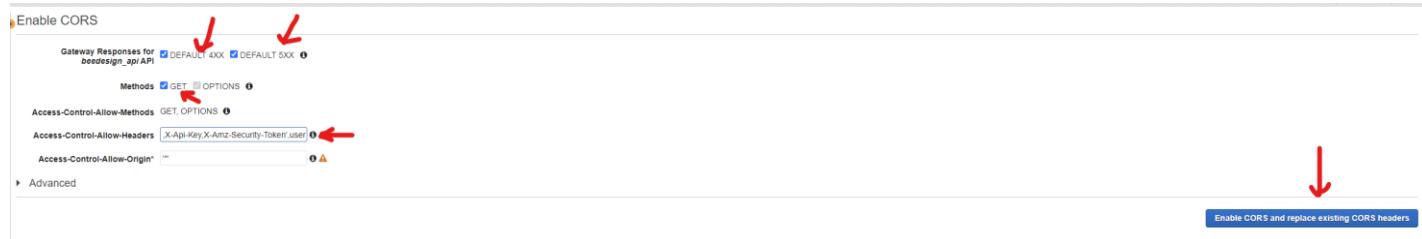
Step 12:

Under Actions, click on Enable CORS.

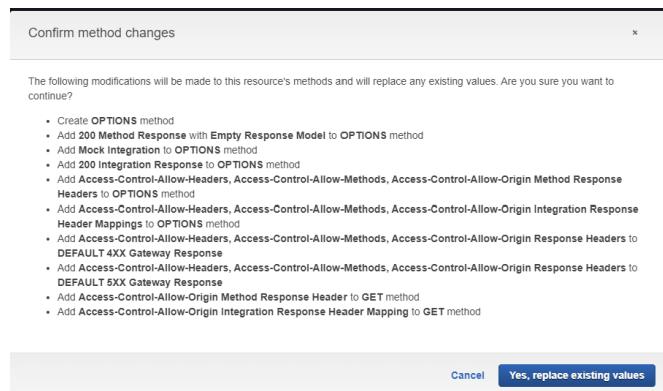


Step 13:

Select DEFAULT 4XX and 5XX, and add a “,user” at the end of Access-Control-Allow-Headers before the closing quotations and click on “Enable CORS” button as shown below.



Click on “Yes, replace existing values”.

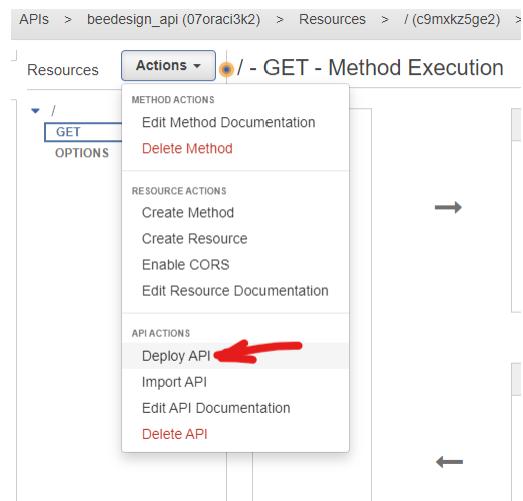


Enable CORS

- ✓ Create **OPTIONS** method
- ✓ Add 200 Method Response with **Empty Response Model** to **OPTIONS** method
- ✓ Add Mock Integration to **OPTIONS** method
- ✓ Add 200 Integration Response to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin** Method Response Headers to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin** Integration Response Header Mappings to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin** Response Headers to **DEFAULT 4XX Gateway Response**
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin** Response Headers to **DEFAULT 5XX Gateway Response**
- ✓ Add **Access-Control-Allow-Origin** Method Response Header to **GET** method
- ✓ Add **Access-Control-Allow-Origin** Integration Response Header Mapping to **GET** method

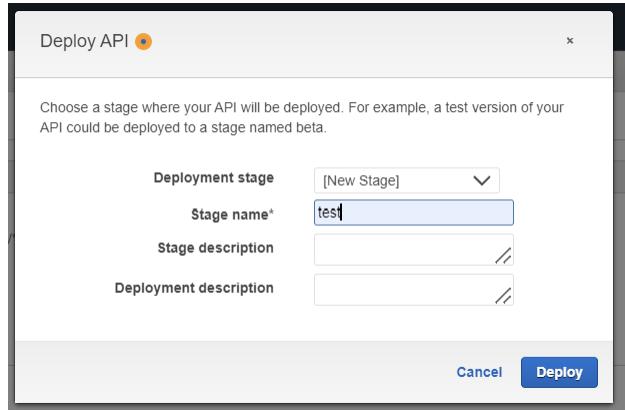
Step 14:

Click on “Deploy API” under Actions.



Step 15:

Select “New Stage” and Enter “test” as the Stage name and click on “Deploy” button.



Step 16:

You should be able to see the Invoke URL. Copy it and save it somewhere for us to use later.

6.2 Setting up the update design file in WinSCP

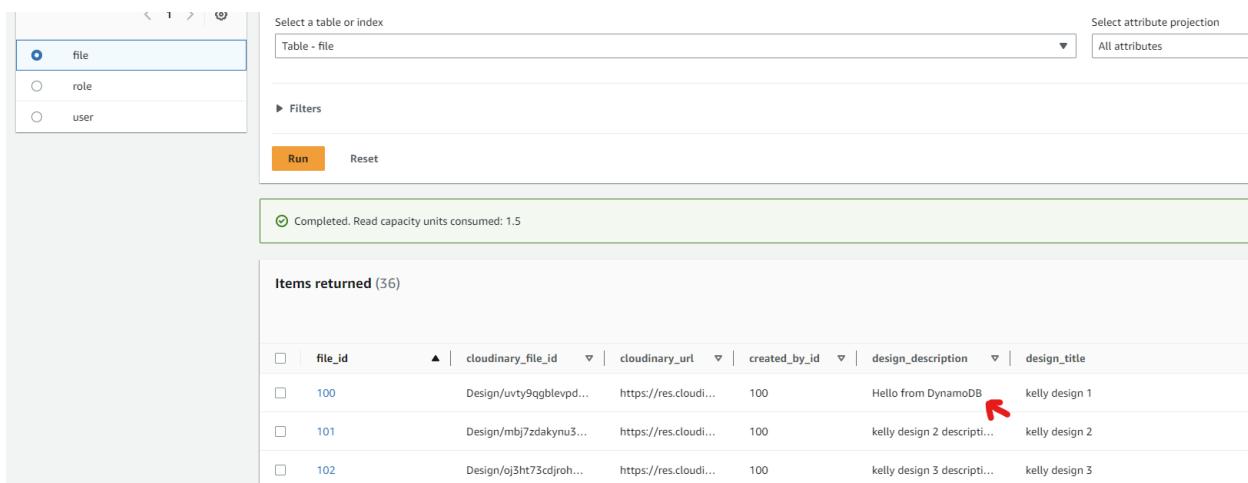
Step 1:

Go to the update_design.js under ESDE/FE/public/js/update_design.js under WinSCP and change the getOneData() function url to the following below.

```
function getOneData() {
  const baseUrl = "https://18.205.232.250:5000";
  //Get the fileId information from the web browser URL textbox
  let query = window.location.search.substring(1);
  let arrayData = query.split("=");
  let fileId = arrayData[1];
  let token = sessionStorage.getItem("token");
  console.dir("Obtained file id from URL : ", fileId);
  let userId = sessionStorage.getItem("user_id");
  axios({
    headers: {
      user: userId,
      authorization: `Bearer ${token}`,
    },
    method: "get",
    url: 'https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test?fid=' + fileId, ↖
  })
  .then(function (response) {
```

Step 2:

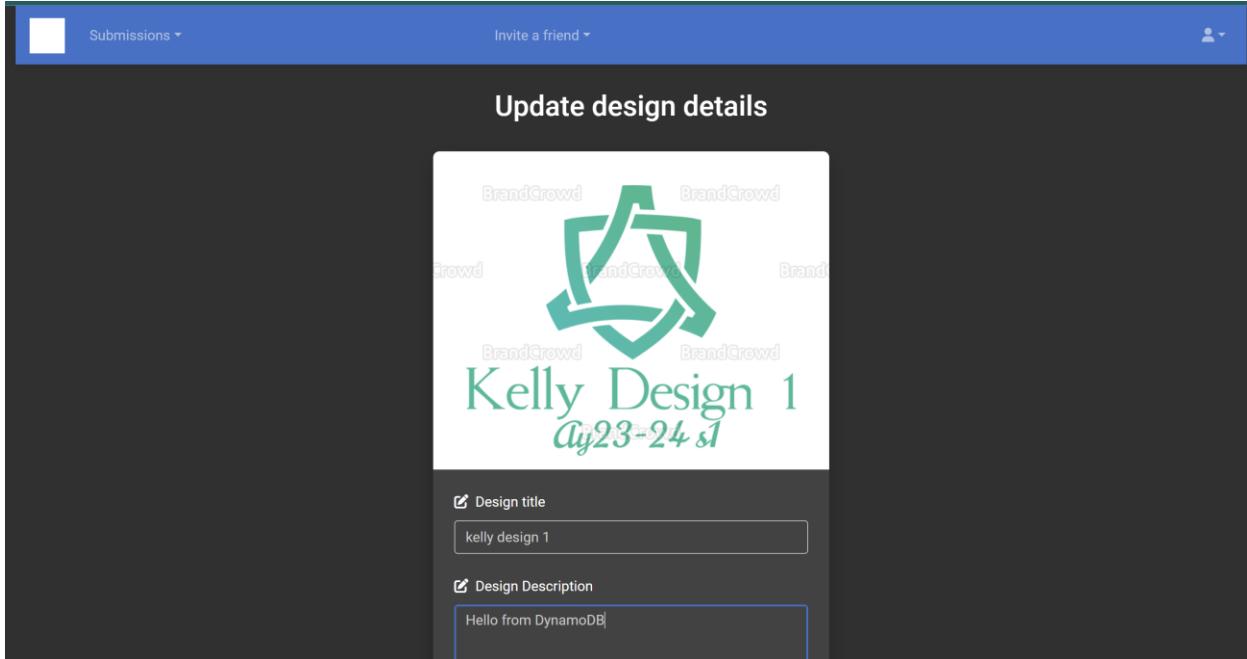
Go into DynamoDB and change the file_id 100 design description to “Hello from DynamoDB” to check if it is reflected in our website.



file_id	cloudinary_file_id	cloudinary_url	created_by_id	design_description	design_title
100	Design/vtvy9qgblevpd...	https://res.cloudi...	100	Hello from DynamoDB	kelly design 1
101	Design/mbj7zdakynu3...	https://res.cloudi...	100	kelly design 2 descripti...	kelly design 2
102	Design/oj3ht73cdjroh...	https://res.cloudi...	100	kelly design 3 descripti...	kelly design 3

Step 3:

Go into your website and login as Kelly and view the 1st design. Click on Update and you should see that you are able to view the “Hello from DynamoDB”.

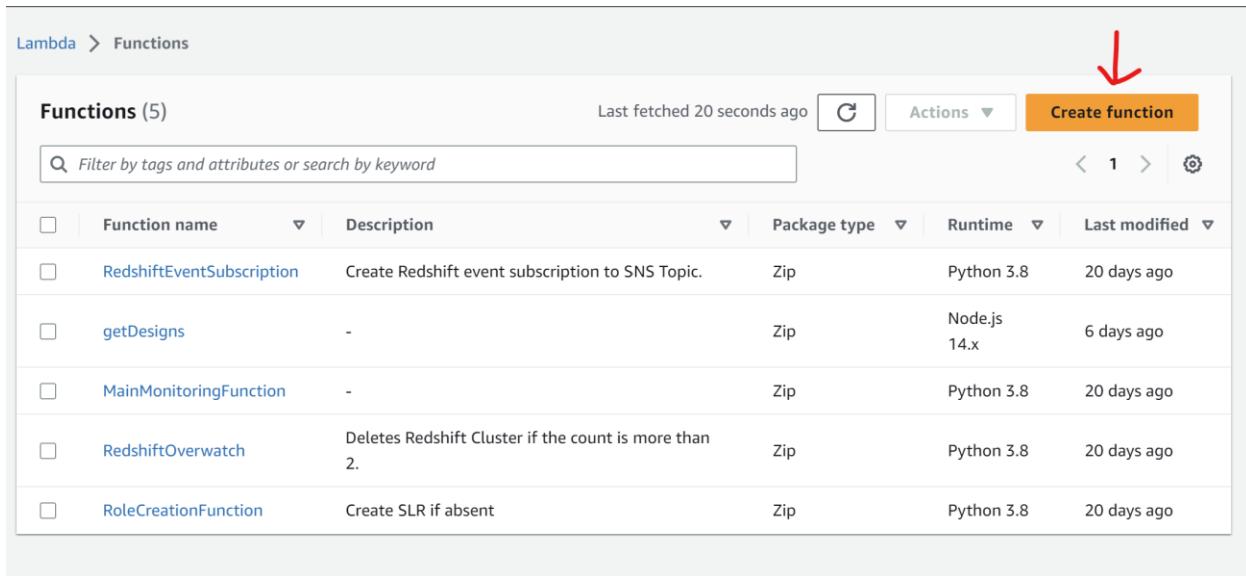


7. Additional Features

7.1 Creating Update Design Lambda Function

Step 1:

Go into AWS Lambda and click on “Create function” to create a new Lambda function.



The screenshot shows the AWS Lambda Functions page. At the top, there is a breadcrumb navigation: Lambda > Functions. Below the header, a search bar contains the placeholder text "Filter by tags and attributes or search by keyword". To the right of the search bar are buttons for "Actions" and "Create function". A red arrow points to the "Create function" button. The main area displays a table titled "Functions (5)" with the following data:

Function name	Description	Package type	Runtime	Last modified
RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.	Zip	Python 3.8	20 days ago
getDesigns	-	Zip	Node.js 14.x	6 days ago
MainMonitoringFunction	-	Zip	Python 3.8	20 days ago
RedshiftOverwatch	Deletes Redshift Cluster if the count is more than 2.	Zip	Python 3.8	20 days ago
RoleCreationFunction	Create SLR if absent	Zip	Python 3.8	20 days ago

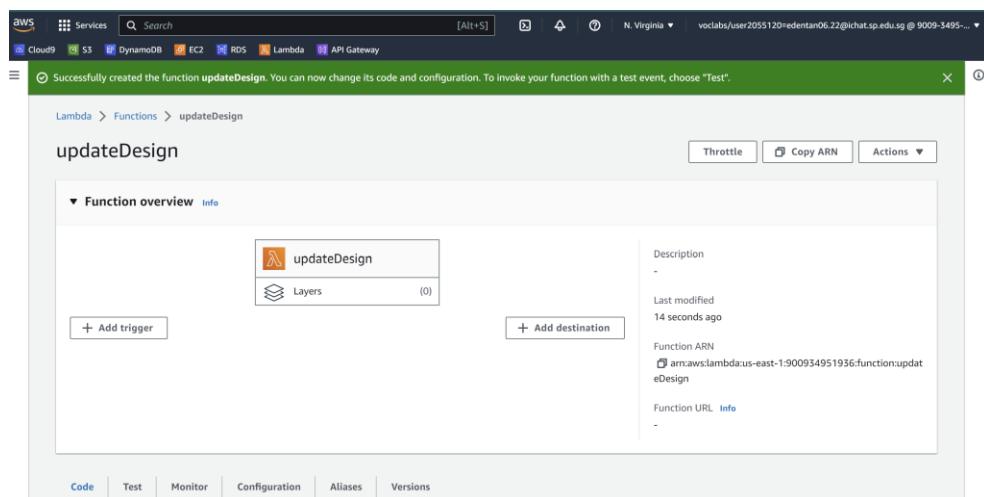
Step 2:

Select “Author from scratch”, and type “updateDesign” as the function name, and for the runtime make sure that you select “Node.js 14.x” and for default execution role select “Use an existing role” and select LabRole and click on the “Create function” button as shown below.

The screenshot shows the 'Basic information' step of the AWS Lambda 'Create Function' wizard. The 'Function name' field is set to 'updateDesign'. The 'Runtime' dropdown is set to 'Node.js 14.x'. The 'Execution role' dropdown is set to 'Use an existing role'. The 'Permissions' section shows 'LabRole' selected. The 'Create function' button is visible at the bottom right.

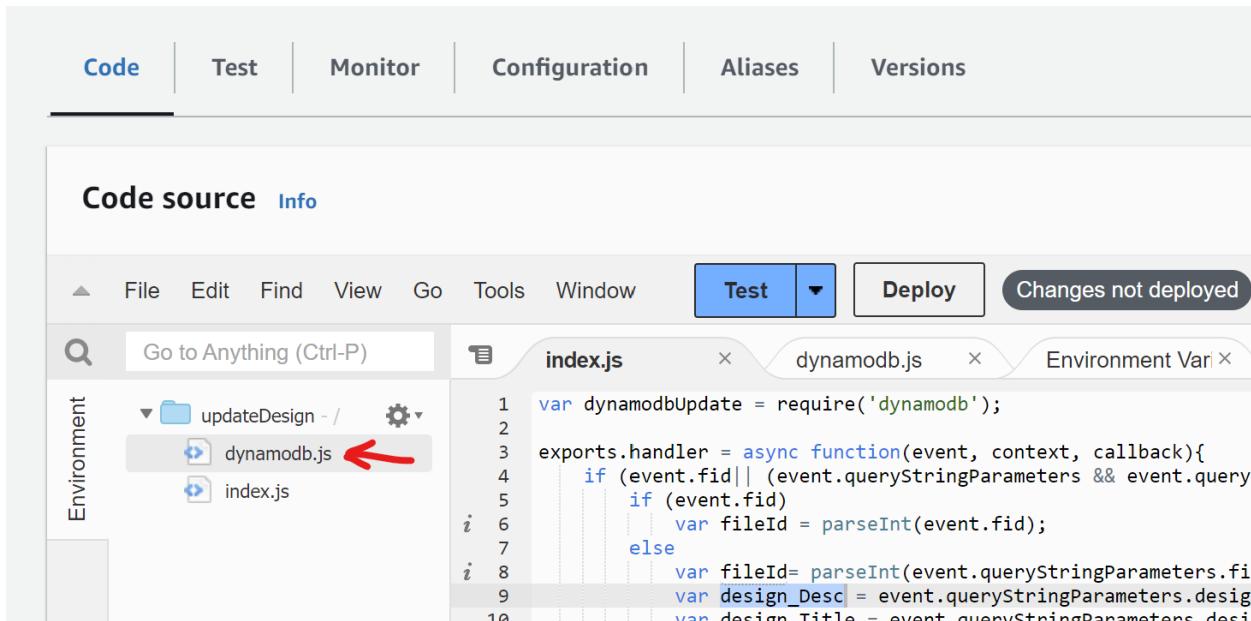
Step 3:

You should be able to see a successfully created message as shown below.



Step 4:

At the bottom under Code Source create a file called dynamodb.js.



The screenshot shows the AWS Lambda Code Source interface. At the top, there are tabs: Code (which is selected), Test, Monitor, Configuration, Aliases, and Versions. Below the tabs, there's a toolbar with File, Edit, Find, View, Go, Tools, Window, Test (highlighted in blue), Deploy, and Changes not deployed. On the left, there's a sidebar with Environment and a search bar labeled 'Go to Anything (Ctrl-P)'. The main area shows a file tree with a folder 'updateDesign - /' containing 'dynamodb.js' (which has a red arrow pointing to it) and 'index.js'. To the right of the file tree is the code editor for 'index.js', which contains some JavaScript code. Below the code editor is another tab for 'dynamodb.js' with its code content.

Step 5:

Type in the following codes for index.js file and dynamodb.js file as shown below.

dynamodb.js codes:

```
var AWS = require('aws-sdk');

async function updateitems_dynamodb(region,
table_name,expr_attr_values,key_cond_expr,proj_expr,design_Desc,design_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:{
```

```

":dD": {
  S: design_Desc
},
":dT": {
  S: design_Title
},
},
TableName: table_name,
UpdateExpression: `SET design_description=:dD, design_title=:dT`  

};

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;

```

index.js codes

```

var dynamodbUpdate = require('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);

    if (event.designTitle)
      var design_Title = event.designTitle;

```

```

else
    var design_Title = event.queryStringParameters.designTitle;

if (event.designDescription)
    var design_Desc = event.designDescription;
else
    var design_Desc = event.queryStringParameters.designDescription;

var region = "us-east-1"
var table_name = "file"
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,design_Desc,design_Title)
.then((data) => {
    console.log(data)
    if (data) {
        var responseCode = 200;
        let response = {
            statusCode: responseCode,
            body: JSON.stringify("Successfully Updated Design!"),
            headers: {
                "Access-Control-Allow-Headers" : "Content-Type,user",
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Methods": "OPTIONS,POST,GET,PUT"
            },
            "isBase64Encoded": true
        }
        console.log("response: " + JSON.stringify(response))
        callback(null, response);
    }
    else {
        var responseCode = 500;
        let response = {
            statusCode: responseCode,

```

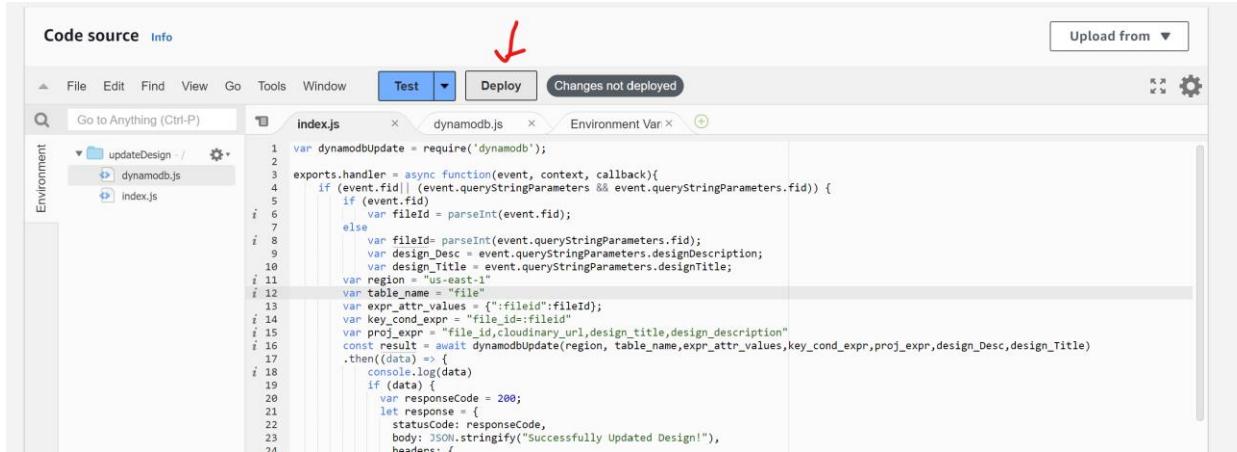
```
body: JSON.stringify("Failed to Update Design."),
headers: {
    "Access-Control-Allow-Headers" : "Content-Type,user",
    "Access-Control-Allow-Origin": "*",
    "Access-Control-Allow-Methods": "OPTIONS,POST,GET,PUT"
},
"isBase64Encoded": true
}
console.log("response: " + JSON.stringify(response))
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
}
```

Step 6:

After adding the codes to the respective files click on Deploy as shown below.



The screenshot shows the AWS Lambda function editor interface. The top navigation bar includes File, Edit, Find, View, Go, Tools, Window, Test, Deploy (which is highlighted with a red arrow), and Changes not deployed. The left sidebar shows the project structure with files dynamodb.js and index.js. The main code editor window contains the following JavaScript code:

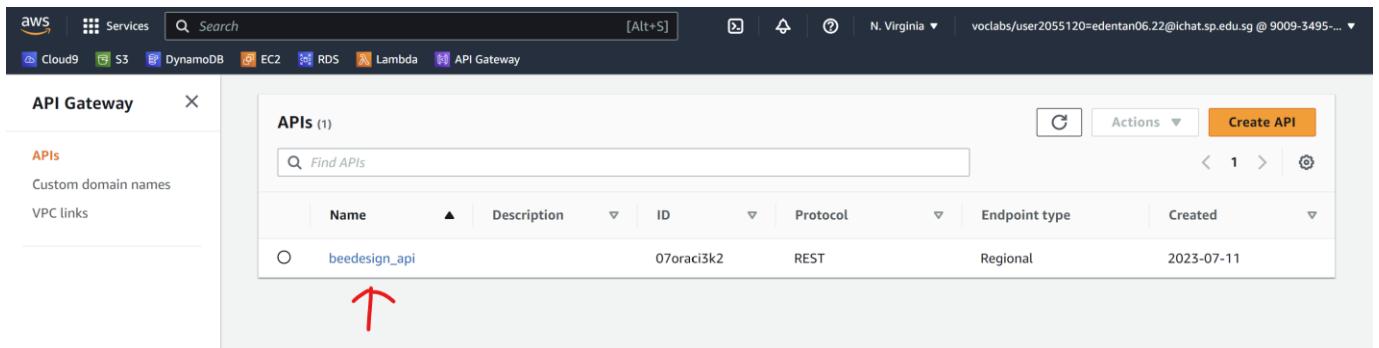
```
var dynamodbUpdate = require('dynamodb');

exports.handler = async function(event, context, callback){
    if (event.fid) {
        if (event.queryStringParameters && event.queryStringParameters.fid) {
            var fileId = parseInt(event.fid);
        } else {
            var fileId = parseInt(event.queryStringParameters.fid);
            var design_Desc = event.queryStringParameters.designDescription;
            var design_Title = event.queryStringParameters.designTitle;
            var region = "us-east-1";
            var tableName = "file";
            var expr_attr_values = {"fileId":fileId};
            var key_cond_expr = "file_id=:fileId"
            var proj_expr = "file_id,cloudinary_url,design_title,design_description"
            const result = await dynamodbUpdate(region, tableName,expr_attr_values,key_cond_expr,proj_expr,design_Desc,design_Title)
            .then((data) => {
                console.log(data)
                if (data) {
                    var responseCode = 200;
                    let response = {
                        statusCode: responseCode,
                        body: JSON.stringify("Successfully Updated Design!"),
                        headers: {
                            "Content-Type": "application/json"
                        }
                    }
                    callback(null, response);
                } else {
                    var responseCode = 400;
                    let response = {
                        statusCode: responseCode,
                        body: JSON.stringify("Design Not Found!"),
                        headers: {
                            "Content-Type": "application/json"
                        }
                    }
                    callback(null, response);
                }
            })
        }
    }
}
```

7.2 Creating API Gateway for updateDesign Lambda Function

Step 1:

Go to AWS API Gateway and click on beedesign_api that we created above under Part 6 API Gateway.



The screenshot shows the AWS API Gateway console. The top navigation bar includes AWS, Services, Search, Cloud9, S3, DynamoDB, EC2, RDS, Lambda, and API Gateway. The left sidebar shows APIs (1) and lists beedesign_api. The main table displays the following information:

Name	Description	ID	Protocol	Endpoint type	Created
beedesign_api		07oraci3k2	REST	Regional	2023-07-11

Step 2:

Go to Resources and Click on the / as shown below.

The screenshot shows the AWS API Gateway interface. The top navigation bar includes services like Cloud9, S3, DynamoDB, EC2, RDS, Lambda, and API Gateway. Below the navigation, the path is shown as APIs > beedesign_api (07oraci3k2) > Resources. On the left sidebar, there are links for APIs, Custom Domain Names, VPC Links, and a section for the API 'beedesign_api' which includes Resources, Stages, and Authorizers. The main content area displays a single resource entry: ' / ' with methods GET and OPTIONS. An 'Actions' dropdown menu is open above the resource, with a red arrow pointing to the slash character in the path ' / '.

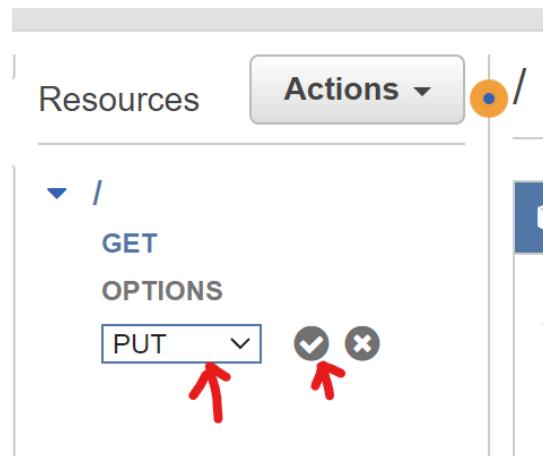
Step 3:

Under Actions, click on “Create Method”.

The screenshot shows the 'Actions' dropdown menu for the resource ' / ' in the AWS API Gateway. The menu is divided into two sections: 'RESOURCE ACTIONS' and 'API ACTIONS'. The 'RESOURCE ACTIONS' section contains 'Create Method', 'Create Resource', 'Enable CORS', and 'Edit Resource Documentation'. The 'API ACTIONS' section contains 'Deploy API', 'Import API', and 'Edit API Documentation'. A red arrow points to the 'Create Method' option in the 'RESOURCE ACTIONS' section.

Step 4:

Select PUT from the dropdown list and click on the tick icon to confirm.



Step 5:

Next, after clicking on the tick icon a setup pane will appear. Select “Lambda Function” for the integration type, and check the use Lambda Proxy integration and for Lambda Function type in “updateDesign” and click on the “Save” button.

A screenshot of the 'PUT - Setup' configuration pane. The left sidebar shows the methods: 'GET', 'OPTIONS', and 'PUT' (which is selected). The main area starts with a note: 'Choose the integration point for your new method.' Below this, the 'Integration type' section has a radio button selected for 'Lambda Function'. Other options like 'HTTP', 'Mock', 'AWS Service', and 'VPC Link' are available but not selected. Underneath, the 'Use Lambda Proxy integration' checkbox is checked. The 'Lambda Region' dropdown is set to 'us-east-1'. The 'Lambda Function' input field contains 'updateDesign'. At the bottom right is a large blue 'Save' button.

Step 6:

Go to “Method Request” as shown below.



Step 7:

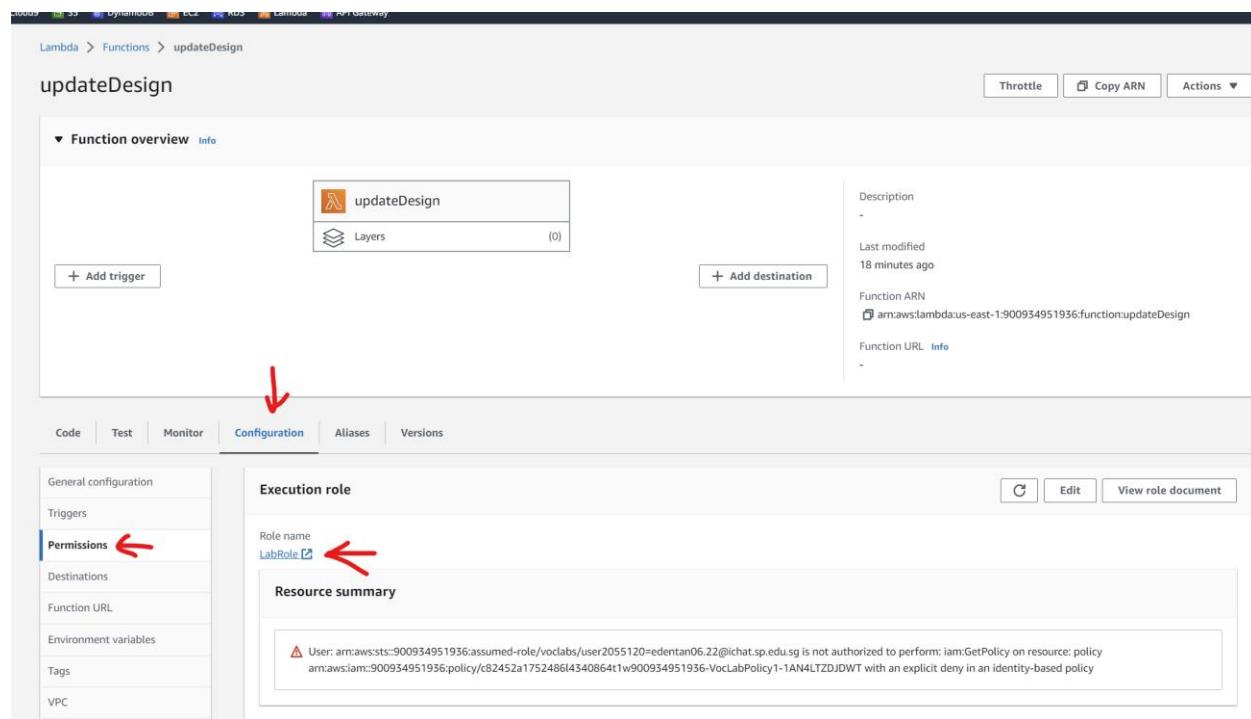
Under URL Query String Parameters add the following as shown below.

This screenshot shows the 'Settings' section of an API resource. It includes fields for 'Authorization' (NONE), 'Request Validator' (NONE), and 'API Key Required' (false). Below this, the 'URL Query String Parameters' section is expanded, showing three parameters: 'designDescription', 'designTitle', and 'fid'. Each parameter has a 'Required' checkbox (unchecked) and a 'Caching' checkbox (unchecked). At the bottom, there's a button to 'Add query string'.

Name	Required	Caching
designDescription	<input type="checkbox"/>	<input type="checkbox"/>
designTitle	<input type="checkbox"/>	<input type="checkbox"/>
fid	<input type="checkbox"/>	<input type="checkbox"/>

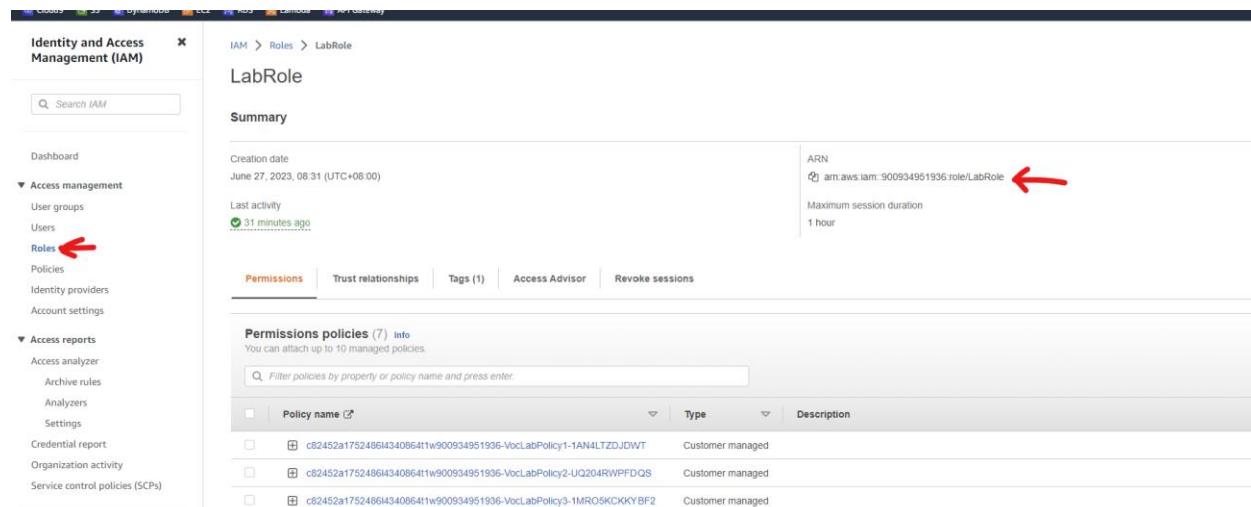
Step 8:

Next, go to your updateDesign Lambda function and click on “Configuration” and then “Permissions” as shown below and click on “LabRole” as shown below.



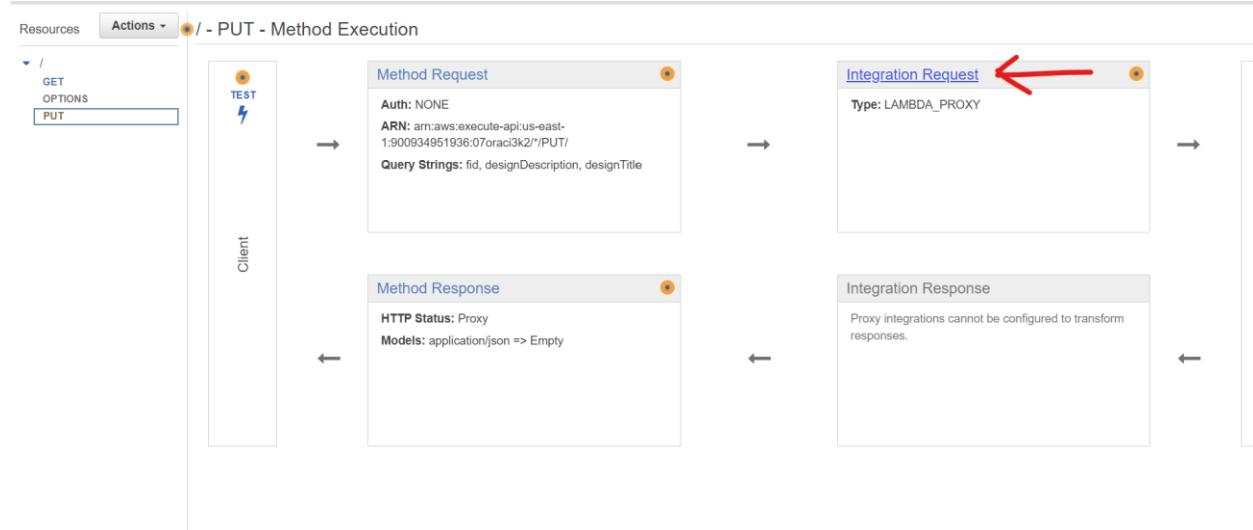
Step 9:

Under the IAM > Roles, copy the ARN and keep it somewhere where we can retrieve it later.



Step 10:

Next, go back to the GET Method in your API Gateway and click on Integration Request.



Step 11:

Paste the ARN Role in the Execution role as shown below and click on the Tick Icon to save it.

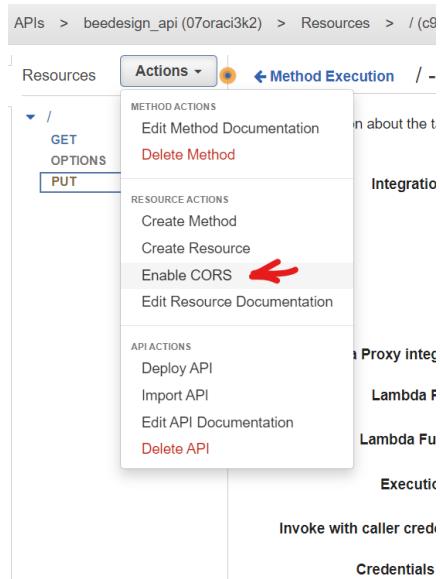
The screenshot shows the configuration for the / - PUT - Integration Request:

- Integration type:** Lambda Function (radio button selected).
- Use Lambda Proxy Integration:** Checked.
- Lambda Region:** us-east-1.
- Lambda Function:** updateDesign.
- Execution role:** arn:aws:iam::900934951936:role/LabRole (highlighted with a red arrow).
- Invoke with caller credentials:** Unchecked.
- Credentials cache:** Do not add caller credentials to cache key.
- Use Default Timeout:** Checked.

A red arrow also points to the checkmark icon at the bottom right of the Execution role input field.

Step 12:

Next, under Actions click on Enable CORS.



Step 13:

Check on the Default 4XX and 5XX and add a “,user” at the end right before the closing quotation and then click on “Enable CORS” button.

The screenshot shows the 'Enable CORS' configuration dialog. It includes fields for 'Gateway Responses for beedesign_api API', 'Methods' (GET, OPTIONS, PUT), and 'Access-Control-Allow-Methods' (GET, OPTIONS, PUT). The 'Access-Control-Allow-Headers' field contains 'X-Api-Key,X-Amz-Security-Token,user' with a red arrow pointing to the 'user' part. A note below says 'Access-Control-Allow-Origin '*''. At the bottom is a blue button labeled 'Enable CORS and replace existing CORS headers'.

Click on Yes, replace existing values.

Confirm method changes

x

The following modifications will be made to this resource's methods and will replace any existing values. Are you sure you want to continue?

- Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Method Response Headers** to **OPTIONS** method
- Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Integration Response Header Mappings** to **OPTIONS** method
- Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Response Headers** to **DEFAULT 4XX Gateway Response**
- Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Response Headers** to **DEFAULT 5XX Gateway Response**
- Add **Access-Control-Allow-Origin Method Response Header** to **GET** method
- Add **Access-Control-Allow-Origin Integration Response Header Mapping** to **GET** method
- Add **Access-Control-Allow-Origin Method Response Header** to **PUT** method
- Add **Access-Control-Allow-Origin Integration Response Header Mapping** to **PUT** method

Cancel

Yes, replace existing values

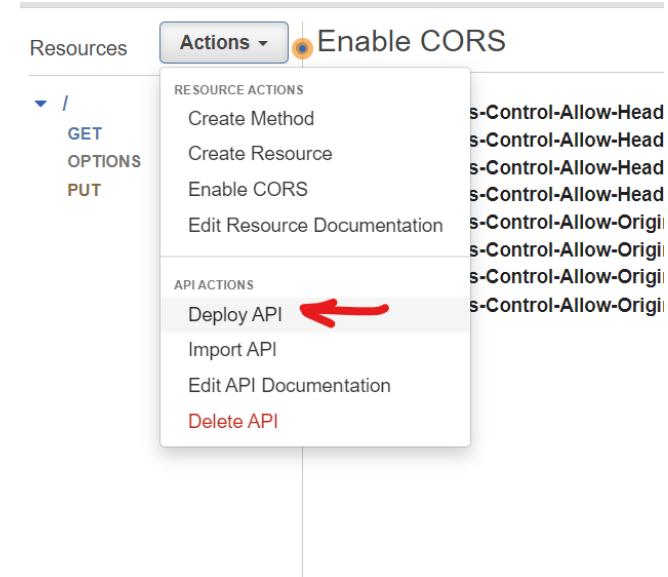
You should see the following.

Enable CORS

- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Method Response Headers** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Integration Response Header Mappings** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Response Headers** to **DEFAULT 4XX Gateway Response**
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Response Headers** to **DEFAULT 5XX Gateway Response**
- ✓ Add **Access-Control-Allow-Origin Method Response Header** to **GET** method
- ✓ Add **Access-Control-Allow-Origin Integration Response Header Mapping** to **GET** method
- ✓ Add **Access-Control-Allow-Origin Method Response Header** to **PUT** method
- ✓ Add **Access-Control-Allow-Origin Integration Response Header Mapping** to **PUT** method

Step 14:

Under Actions, click on Deploy API.



Step 15:

Under Deployment stage select “test” and click on Deploy.

A screenshot of the 'Deploy API' dialog box. It asks to choose a deployment stage (set to 'test') and provides a deployment description field. At the bottom are 'Cancel' and 'Deploy' buttons.

You should be able to see the following.

test Stage Editor

Invoke URL: <https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test>

Cache Settings

Enable API cache

Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. [Read more about API Gateway throttling](#)

Enable throttling ⓘ

Rate: 10000 requests per second

Burst: 5000 requests

Web Application Firewall (WAF) [Learn more.](#)

Select the Web ACL to be applied to this stage.

Web ACL: None [Create Web ACL](#)

Client Certificate

Select the client certificate that API Gateway will use to call your integration endpoints in this stage.

Certificate: None [None](#)

Save Changes

7.3 Adding the updateDesign URL to WinSCP

Step 1:

Go into WinSCP and go to the directory “/home/ec2-user/ESDE/FE/public/js”.

Name	Size	Changed	Rights	Owner
..		27/6/2023 10:11:45 am	rwxx-r-x	ec2-user
admin_check_user_sub...	8 KB	1/7/2023 9:04:18 am	rw-r--r--	ec2-user
admin_manage_user.js	9 KB	1/7/2023 9:04:39 am	rw-r--r--	ec2-user
admin_update_user.js	4 KB	1/7/2023 9:05:02 am	rw-r--r--	ec2-user
axios.js	14 KB	15/5/2023 11:58:12 pm	rw-r--r--	ec2-user
checkAdminRole.js	2 KB	4/7/2023 12:39:41 am	rw-r--r--	ec2-user
checkUserRole.js	2 KB	4/7/2023 12:39:50 am	rw-r--r--	ec2-user
global.js	1 KB	26/5/2023 8:14:18 am	rw-r--r--	ec2-user
login.js	3 KB	1/7/2023 9:03:56 am	rw-r--r--	ec2-user
manage_invite.js	2 KB	1/7/2023 9:05:45 am	rw-r--r--	ec2-user
profile.js	2 KB	1/7/2023 9:05:55 am	rw-r--r--	ec2-user
register.js	2 KB	1/7/2023 9:06:16 am	rw-r--r--	ec2-user
submit_design.js	3 KB	1/7/2023 9:06:29 am	rw-r--r--	ec2-user
update_design.js	4 KB	11/7/2023 2:29:50 am	rw-r--r--	ec2-user
user_manage_submissi...	9 KB	1/7/2023 9:07:06 am	rw-r--r--	ec2-user

Step 2:

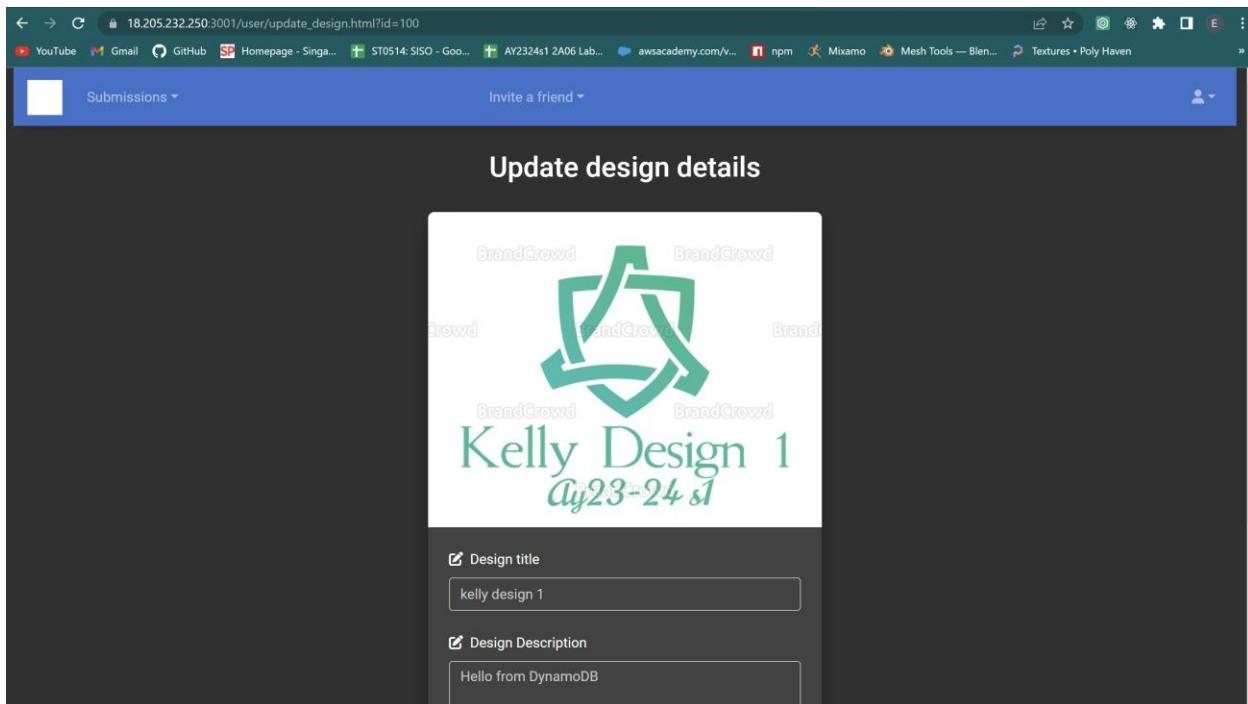
Click on update_design.js file and paste the following into the URL and save the file.

```
'https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test?fid='+fileId+'&designTitle='+designTitle+'&designDescription='+designDescription
```

```
//making a PUT HTTP request.
let webFormData = new FormData();
webFormData.append("designTitle", designTitle);
webFormData.append("designDescription", designDescription);
webFormData.append("fileId", fileId);
axios({
  method: "put",
  url: 'https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test?fid='+fileId+'&designTitle='+designTitle+'&designDescription='+designDescription,
  data: webFormData,
  headers: {
    "Content-Type": "multipart/form-data",
  }
})
```

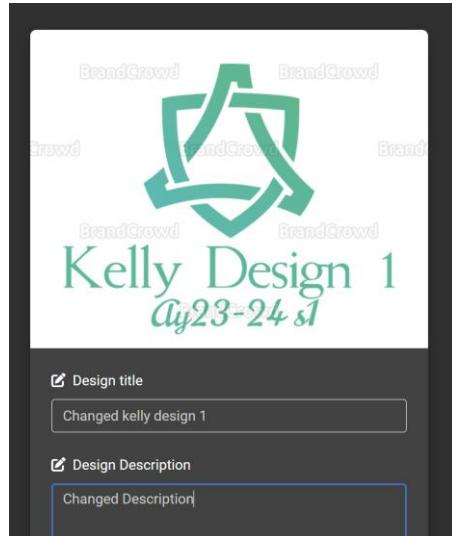
Step 3:

Run the frontend and backend server and go to Kelly's Account and go to her designs page and click on Kelly Design 1.



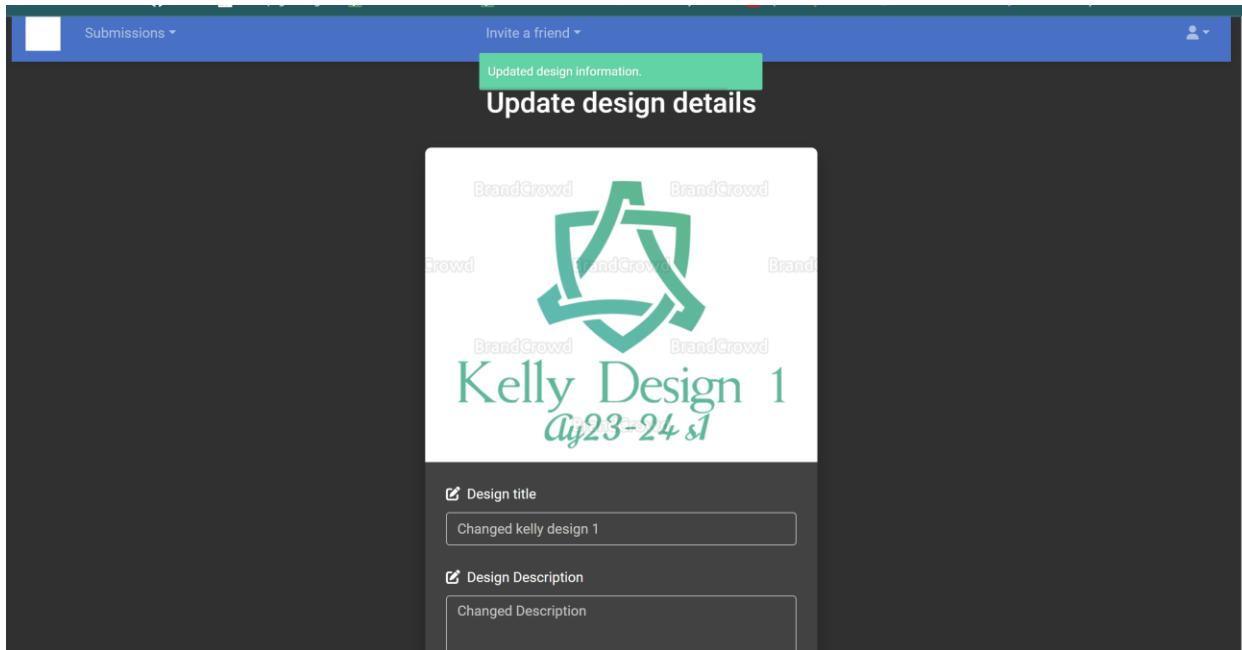
Step 4:

Change the Design Title to “Changed Kelly design 1” and Design Description to “Changed Description” and click on the submit button.



Step 5:

You should be able to see the following Successful Update Message.



Step 6:

Go to AWS dynamoDB and click on file table as shown below.

The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with various navigation options like Dashboard, Tables, Update settings, Explore items, etc. Below that is a section for DAX Clusters. The main area is titled 'DynamoDB > Explore items > file'. A sub-menu titled 'Tables (3)' is open, showing three tables: 'file' (selected), 'role', and 'user'. To the right of the table list, there's a section for 'Scan or query items' with a 'Scan' button highlighted, and a 'Run' button at the bottom.

Step 7:

You should be able to see the changes reflected in the dynamoDB file table.

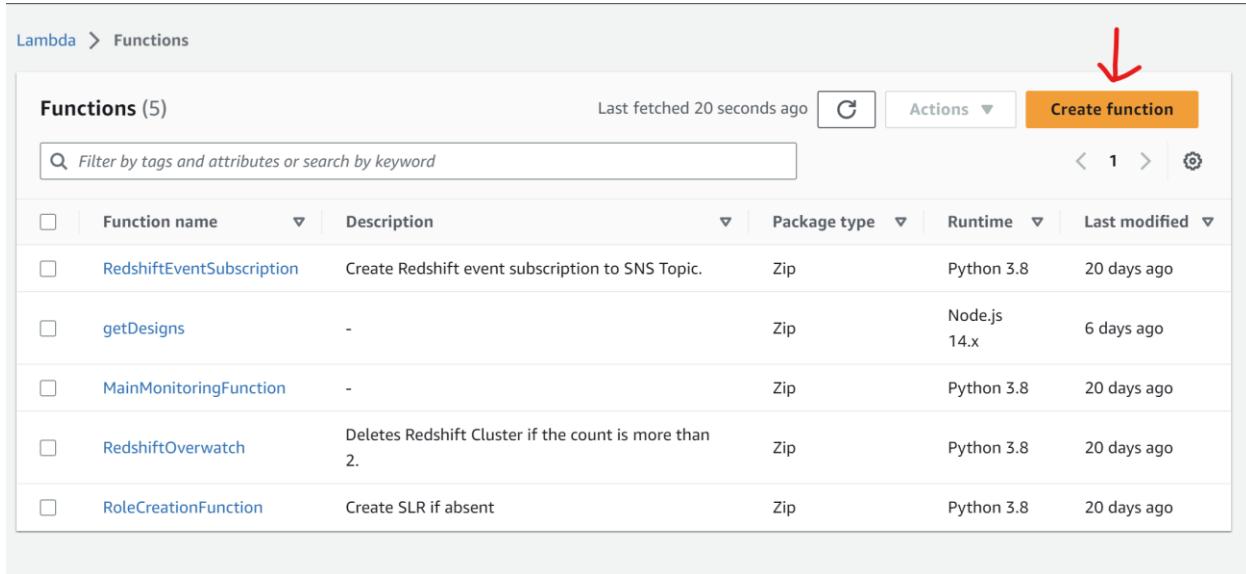
This screenshot shows the results of a scan operation on the 'file' table. At the top, it says 'Completed. Read capacity units consumed: 1.5'. Below that is a table titled 'Items returned (36)' with columns: file_id, clouddinary_file_id, clouddinary_url, created_by_id, design_description, and design_title. One row in the table is highlighted with a red arrow pointing to the 'design_title' column, which contains the value 'Changed kelly design 1'. The table has standard AWS pagination controls at the bottom.

file_id	clouddinary_file_id	clouddinary_url	created_by_id	design_description	design_title
100	Design/uvty9qgblevpd...	https://res.cloud...	100	Changed Description	Changed kelly design 1
101	Design/mbj7zdakynu3...	https://res.cloud...	100	kelly design 2 descripti...	kelly design 2
102	Design/oj3ht73cdjroh...	https://res.cloud...	100	kelly design 3 descripti...	kelly design 3

7.4 Creating manageSubmissions Lambda Function and Index Name for File Table

Step 1:

Go into AWS Lambda and click on “Create function” to create a new Lambda function.



The screenshot shows the AWS Lambda Functions page. At the top, there is a breadcrumb navigation: Lambda > Functions. Below the header, a search bar contains the placeholder text "Filter by tags and attributes or search by keyword". To the right of the search bar are buttons for "Actions" and "Create function". A red arrow points to the "Create function" button. The main area displays a table titled "Functions (5)" with the following data:

	Function name	Description	Package type	Runtime	Last modified
<input type="checkbox"/>	RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.	Zip	Python 3.8	20 days ago
<input type="checkbox"/>	getDesigns	-	Zip	Node.js 14.x	6 days ago
<input type="checkbox"/>	MainMonitoringFunction	-	Zip	Python 3.8	20 days ago
<input type="checkbox"/>	RedshiftOverwatch	Deletes Redshift Cluster if the count is more than 2.	Zip	Python 3.8	20 days ago
<input type="checkbox"/>	RoleCreationFunction	Create SLR if absent	Zip	Python 3.8	20 days ago

Step 2:

Select “Author from scratch”, and type “manageSubmissions” as the function name, and for the runtime make sure that you select “Node.js 14.x” and for default execution role select “Use an existing role” and select LabRole and click on the “Create function” button as shown below.

Create function [Info](#)
AWS Serverless Application Repository applications have moved to [Create application](#).

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Container image
Select a container image to deploy for you

Basic information

Function name
Enter a name that describes the purpose of your function.
ManageSubmissions

A function with that name already exists.
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 14.x

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
 x86_64

arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console

Create a new role with basic Lambda permissions

Use an existing role

Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LabRole

[View the LabRole role](#) on the IAM console.

Step 3:

You should be able to see the following below.

Lambda > Functions > manageSubmissions

manageSubmissions

[Throttle](#) [Copy ARN](#) [Actions ▾](#)

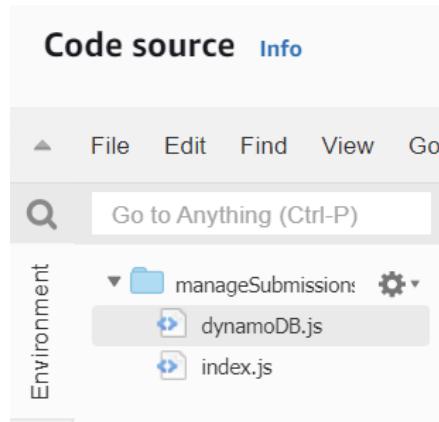
▼ Function overview [Info](#)

manageSubmissions	Description -
Layers (0)	Last modified 21 hours ago
+ Add trigger	Function ARN arn:aws:lambda:us-east-1:900934951936:function:manageSubmissions
	Function URL Info

[Code](#) [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

Step 4:

Scroll down and make sure that you have a dynamoDB.js and index.js file created as shown below.



Step 5:

Add in the following codes to your index.js and dynamoDB.js files and click on the deploy button to save the codes.

dynamoDB.js

```
var AWS = require('aws-sdk');

async function queryitems_dynamodb(region, table_name, expr_attr_values,
key_cond_expr, proj_expr, creator, pageNumber, searchInput) {
    console.log("In the queryitems_dynamodb method...");
    var dynamodb = new AWS.DynamoDB.DocumentClient({ region: region });
    var index = 0;
    var count = 0;

    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, index) {
```

```

        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: "file",
        IndexName: "created_by_id-file_id-index",
        KeyConditionExpression: "#createdBy = :creator",
        ExpressionAttributeNames: {
            "#createdBy": "created_by_id"
        },
        ExpressionAttributeValues: {
            ":creator": creator,
        },
        ScanIndexForward: true,
    };

    // Conditionally add the FilterExpression if searchInput is not empty
    if (searchInput && searchInput.trim().length > 0) {
        params.FilterExpression = "contains(design_title, :searchInput)";
        params.ExpressionAttributeValues[":searchInput"] = searchInput;
    }

    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;
```

index.js

```
var dynamodbQuery = require('dynamoDB');

exports.handler = async function(event, context, callback){
    if (event.userId || event.queryStringParameters.userId) {
        if (event.userId)
            var creator = parseInt(event.userId)
        else
            var creator = parseInt(event.queryStringParameters.userId)

        var region = "us-east-1";
        var table_name = "file";
        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);

        var searchInput; // Initialize searchInput with an empty string
        if (event.queryStringParameters &&
            event.queryStringParameters.searchInput) {
            searchInput = event.queryStringParameters.searchInput || null;
        }
        else {
            searchInput = event.searchInput || null;
        }

        await dynamodbQuery(region,
            table_name,expr_attr_values,key_cond_expr,proj_expr,creator,pageNum,
            searchInput)
            .then(data => {
                console.log("Successfully got items from dynamodb.query");
            })
    }
}
```

```

        console.log(data)
        var responseCode = 200;
        var totalNumberOfRecords = data.Count;
        var jsonResult = {
            'filedata': data.Items,
            'total_number_of_records': totalNumberOfRecords
        };
        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
}

```

Step 6:

Next, go to dynamoDB and look for the file table and click on the Indexes Tab.

The screenshot shows the AWS DynamoDB console. On the left, there's a sidebar with various options like Dashboard, Tables, Update settings, etc. The 'Tables' option is selected. The main area shows a 'Tables (3)' list with three entries: 'Any tag key', 'Any tag value', and a search bar. Below this is a list of tables: 'file' (selected), 'role', and 'user'. A red arrow points to the 'file' entry. To the right of the table list is the 'file' table detail page. At the top of this page, there are tabs: Overview (selected), Indexes, Monitor, Global tables, Backups, and Exports and streams. A red arrow points to the 'Indexes' tab. Below the tabs, there's a note about protecting the table from accidental writes and deletes, mentioning PITR and backup charges. Under the 'General information' section, it shows the Partition key as 'file_id (Number)' and the Sort key as '-'. At the bottom right of the table detail page, there are buttons for Actions (with a dropdown menu) and Explore table items (highlighted with a red box).

Step 7:

Click on Create Index to create a new index for us to use in the index.js.

This screenshot shows the 'Indexes' tab for the 'file' table. The top navigation bar includes tabs for Overview, Indexes (selected), Monitor, Global tables, Backups, Exports and streams, and Additional settings. Below the tabs, it says 'Global secondary indexes (1) Info'. There is a search bar and a 'Create index' button highlighted with a red box. At the bottom right, there are buttons for Delete and Actions (with a dropdown menu). A red arrow points to the 'Create index' button.

Step 8:

Input “created_by_id” as the Partition Key and “file_id” as the Sort Key. Ignore the Index Name and leave it as it is since it is auto generated and click on “Create Index” at the bottom.

Create global secondary index [Info](#)

Global secondary indexes allow you to perform queries on attributes that are not part of a table's primary key. Note that global secondary index read and write capacity settings are separate from those of the table, and they will incur additional costs.

Index details Info	
Partition key	Data type
created_by_id 	Number 
1 to 255 characters.	
Sort key - optional	Data type
file_id 	Number 
1 to 255 characters.	
Index name	
created_by_id-file_id-index	
Between 3 and 255 characters. Only A-Z, a-z, 0-9, underscore characters, hyphens, and periods allowed.	

Step 9:

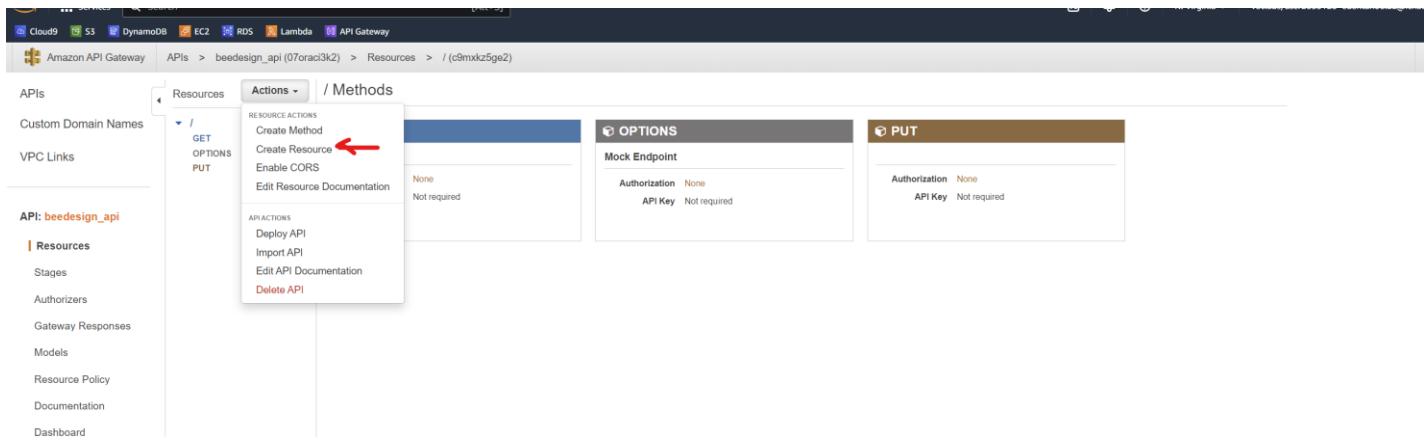
Upon successful creation, you should be able to see the successfully created index name as shown below.

Global secondary indexes (1) Info										Delete	Create index				
<input type="text"/> Find indexes												<	1	>	②
Name	Status	Partition key	Sort key	Read capacity	Write capacity	Projected attributes	Size								
○ created_by_id-file_id-index	 Active	created_by_id (Number)	file_id (Number)	Range: 1 - 10 Auto scaling at 70% Current provisioned units: 1	Range: 1 - 10 Auto scaling at 70% Current provisioned units: 1	All	10.3 kilobytes								

7.5 Creating manageSubmissions API Gateway

Step 1:

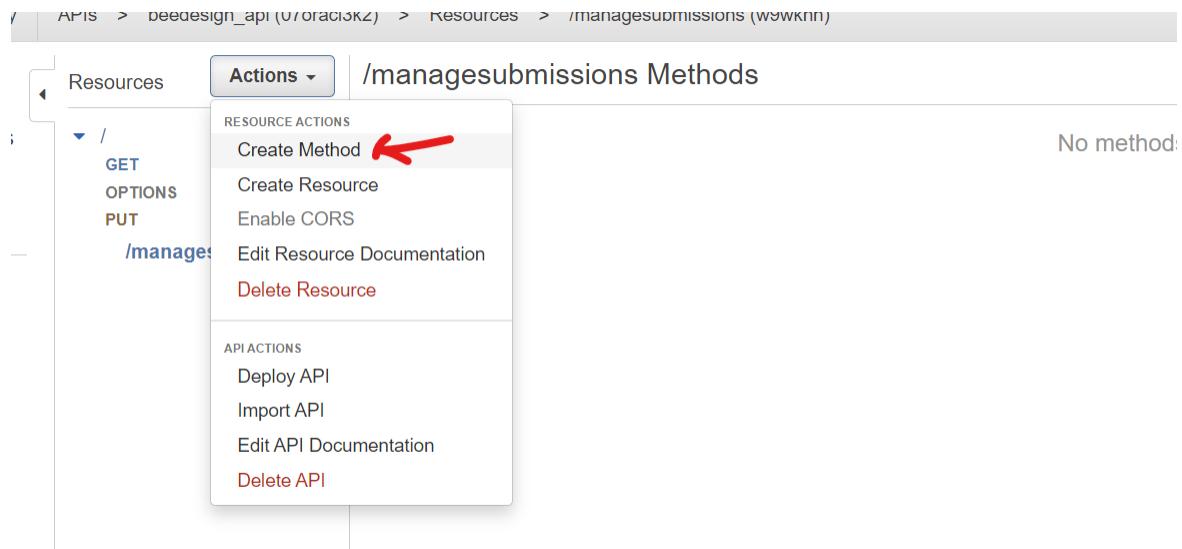
Go to AWS API Gateway and click on Actions and click on Create Resource and name the resource as manageSubmissions.



The screenshot shows the AWS API Gateway interface. On the left, there's a sidebar with options like APIs, Custom Domain Names, VPC Links, and others. Under 'API: beedesign_api', there's a 'Resources' section. In the main area, a 'Actions' dropdown is open over a resource path '/'. The 'Actions' dropdown menu has several options: 'Create Method', 'Create Resource' (which is highlighted with a red arrow), 'Enable CORS', and 'Edit Resource Documentation'. Below this, under 'API ACTIONS', are 'Deploy API', 'Import API', 'Edit API Documentation', and 'Delete API'.

Step 2:

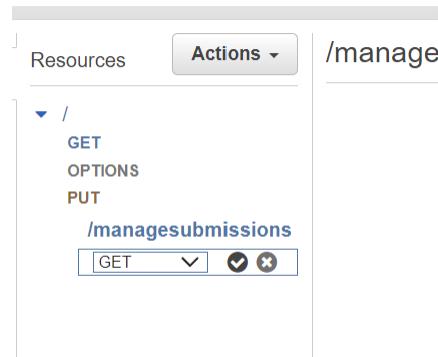
Click on the newly created resource and click on Actions followed by Create Method.



This screenshot shows the AWS API Gateway interface again, but now focusing on a specific resource. The path in the top navigation bar is '/ APIs > beedesign_api (U/oracikZ) > Resources > /managesubmissions (W9WKNH)'. The 'Actions' dropdown is open over the '/managesubmissions' resource. The 'Create Method' option is highlighted with a red arrow. Other options in the 'RESOURCE ACTIONS' section include 'Create Resource', 'Enable CORS', 'Edit Resource Documentation', and 'Delete Resource'. The 'API ACTIONS' section includes 'Deploy API', 'Import API', 'Edit API Documentation', and 'Delete API'. To the right of the actions, it says 'No method'.

Step 3:

Next select the GET Method and click on the tick icon to confirm.



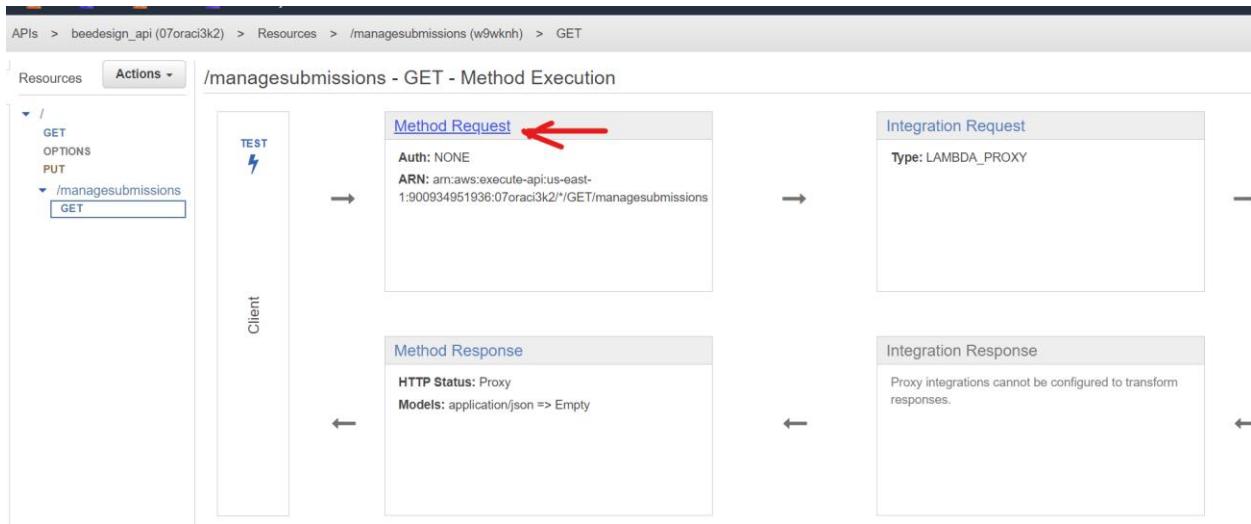
Step 4:

A Setup pane will appear, make sure that you follow the following setup as shown below where Lambda Function is manageSubmissions, click on save and ok.

A screenshot of the 'Setup' pane for the '/managesubmissions - GET' method. At the top, it says 'Choose the integration point for your new method.' Below that, 'Integration type' is set to 'Lambda Function' (radio button selected). Other options include 'HTTP', 'Mock', 'AWS Service', and 'VPC Link'. A checkbox 'Use Lambda Proxy integration' is checked. Under 'Lambda Region', a dropdown menu shows 'us-east-1'. In the 'Lambda Function' field, the text 'manageSubmissions' is entered. A checkbox 'Use Default Timeout' is checked. At the bottom right is a blue 'Save' button.

Step 5:

Click on Method Request.



Step 6:

Under URL Query String Parameters make sure that you have userId, pageNumber and searchInput being added and for request headers make sure you have user added.

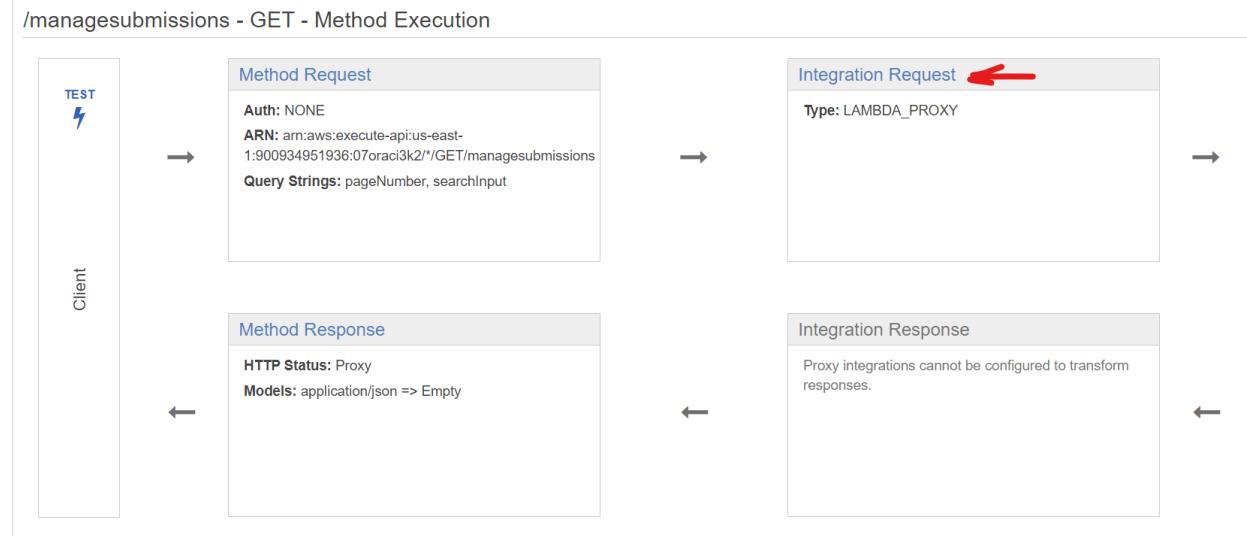
The screenshot shows the AWS API Gateway Method Request configuration page for the /managesubmissions - GET - Method Request. It includes sections for Settings, Authorization (NONE), Request Validator (NONE), API Key Required (false), and URL Query String Parameters. The URL Query String Parameters table is as follows:

Name	Required	Caching	Actions
pageNumber	<input type="checkbox"/>	<input type="checkbox"/>	
searchInput	<input type="checkbox"/>	<input type="checkbox"/>	
userId	<input type="checkbox"/>	<input type="checkbox"/>	

At the bottom, there's a link to Add query string.

Step 7:

Go back and click on Integration request.



Step 8:

We will use the same execution role that we used from [7.2 Step 9](#).

◀ Method Execution /managesubmissions - GET - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function [?](#)
 HTTP [?](#)
 Mock [?](#)
 AWS Service [?](#)
 VPC Link [?](#)

Use Lambda Proxy integration [?](#)

Lambda Region us-east-1 [✎](#)

Lambda Function manageSubmissions [✎](#)

Execution role arn:aws:iam::900934951936:role/LabRole [✎](#) ←

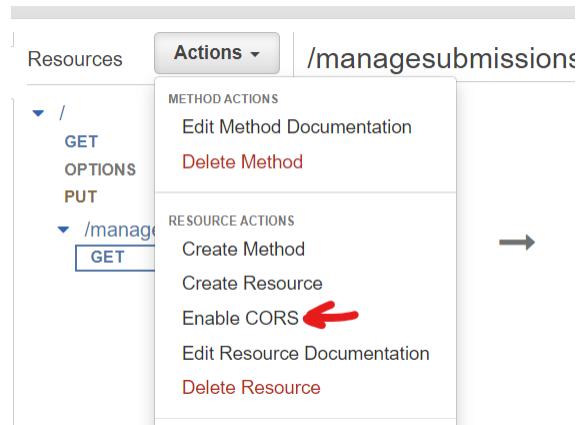
Invoke with caller credentials [?](#)

Credentials cache Do not add caller credentials to cache key [✎](#)

Use Default Timeout [?](#)

Step 9:

Go back and click on Actions and click on Enable CORS.



Step 10:

Make sure that you have the following configurations as shown below, where at the end of Headers add a “,user” and check the necessary things. Once done, click on Enable CORS.

A screenshot of the 'Enable CORS' configuration page. It shows settings for gateway responses, methods (GET, OPTIONS), access control allow methods (GET, OPTIONS), access control allow headers (X-Api-Key, X-Amz-Security-Token, user), and access control allow origin (*). An 'Advanced' section is collapsed. A blue button at the bottom right says 'Enable CORS and replace existing'.

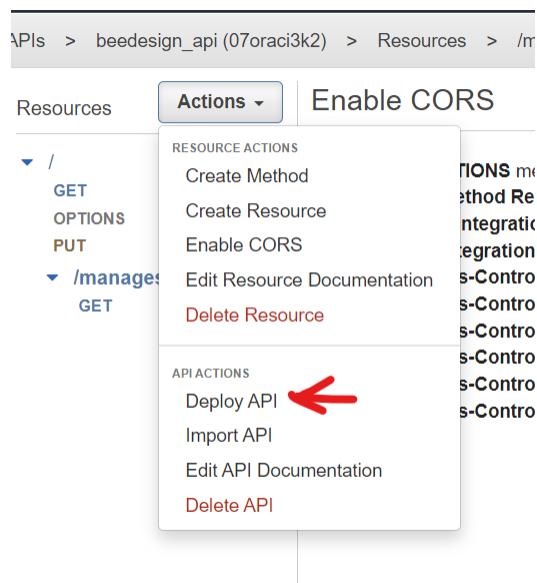
You should see the following.

Enable CORS

- ✓ Create **OPTIONS** method
- ✓ Add 200 Method Response with Empty Response Model to **OPTIONS** method
- ✓ Add Mock Integration to **OPTIONS** method
- ✓ Add 200 Integration Response to **OPTIONS** method
- ✓ Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Method Response Headers to **OPTIONS** method
- ✓ Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Integration Response Header Mappings to **OPTIONS** method
- ✓ Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Response Headers to **DEFAULT 4XX Gateway Response**
- ✓ Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Response Headers to **DEFAULT 5XX Gateway Response**
- ✓ Add Access-Control-Allow-Origin Method Response Header to **GET** method
- ✓ Add Access-Control-Allow-Origin Integration Response Header Mapping to **GET** method

Step 11:

Click on Actions and click on Deploy API.



Make sure to use test for deployment stage and click on Deploy.

A screenshot of a 'Deploy API' dialog box. At the top, it says 'Deploy API' and has a close button 'x'. Below that, a message says: 'Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.' Underneath, there are two fields: 'Deployment stage' with a dropdown menu showing 'test' selected, and 'Deployment description' with an empty input field. At the bottom right are two buttons: 'Cancel' and a blue 'Deploy' button.

Step 12:

You should be able to see the following. Showing that you have successfully created the API Gateway.

The screenshot shows the AWS Lambda test configuration interface. At the top, it says "test - GET - /managesubmissions". Below that is a blue bar with the text "Invoke URL: https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test/managesubmissions". Underneath, there's a note: "Use this page to override the test stage settings for the GET to /managesubmissions method." There are two radio button options under "Settings": "Inherit from stage" (selected) and "Override for this method". At the bottom right is a "Save Changes" button.

7.6 Adding manageSubmissions API to WinSCP

Step 1:

Go into WinSCP and go to the directory “/home/ec2-user/ESDE/FE/public/js”.

/home/ec2-user/ESDE/FE/public/js/					
Name	^	Size	Changed	Rights	Owner
..			27/6/2023 10:11:45 am	rwxxr-xr-x	ec2-user
admin_check_user_sub...	js	8 KB	1/7/2023 9:04:18 am	rw-r--r--	ec2-user
admin_manage_user.js	js	9 KB	1/7/2023 9:04:39 am	rw-r--r--	ec2-user
admin_update_user.js	js	4 KB	1/7/2023 9:05:02 am	rw-r--r--	ec2-user
axios.js	js	14 KB	15/5/2023 11:58:12 pm	rw-r--r--	ec2-user
checkAdminRole.js	js	2 KB	4/7/2023 12:39:41 am	rw-r--r--	ec2-user
checkUserRole.js	js	2 KB	4/7/2023 12:39:50 am	rw-r--r--	ec2-user
global.js	js	1 KB	26/5/2023 8:14:18 am	rw-r--r--	ec2-user
login.js	js	3 KB	1/7/2023 9:03:56 am	rw-r--r--	ec2-user
manage_invite.js	js	2 KB	1/7/2023 9:05:45 am	rw-r--r--	ec2-user
profile.js	js	2 KB	1/7/2023 9:05:55 am	rw-r--r--	ec2-user
register.js	js	2 KB	1/7/2023 9:06:16 am	rw-r--r--	ec2-user
submit_design.js	js	3 KB	1/7/2023 9:06:29 am	rw-r--r--	ec2-user
update_design.js	js	4 KB	11/7/2023 2:29:50 am	rw-r--r--	ec2-user
user_manage_submissi...	js	9 KB	1/7/2023 9:07:06 am	rw-r--r--	ec2-user

Step 2:

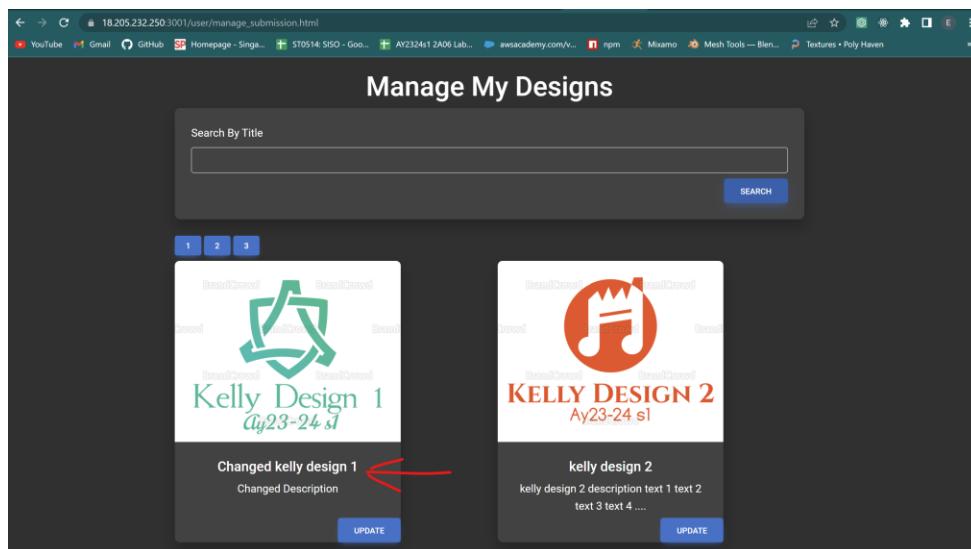
Click on user_manage_submission.js and add in the following url for the first endpoint and save it.

```
"https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test/managesubmissions?pageNumber=' + pageNumber + '&searchInput=' + searchInput"
```

```
let $searchDesignFormContainer = $("#searchDesignFormContainer");
if ($searchDesignFormContainer.length != 0) {
    console.log(
        "Search design form detected in user manage submission interface. Binding event handling logic to form elements."
    );
    //If the jquery object which represents the form element exists,
    //the following code will create a method to send key-value pair information to do record searching
    //to server-side api when the #submitButton element fires the click event.
    $("#submitButton").on("click", function (event) {
        event.preventDefault();
        const baseUrl = "https://18.205.232.250:5000";
        let searchInput = $("#searchInput").val();
        let userId = sessionStorage.getItem("user_id");
        let token = sessionStorage.getItem("token");
        axios({
            headers: {
                user: userId,
                authorization: `Bearer ${token}`
            },
            method: "get",
            url: 'https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test/managesubmissions?userId=' + userId + '&pageNumber=1&searchInput=' + searchInput,
        })
        .then(function (response) {
            //console.log(response)
            //Using the following to inspect the response.data data structure
        })
    });
}
```

Step 3:

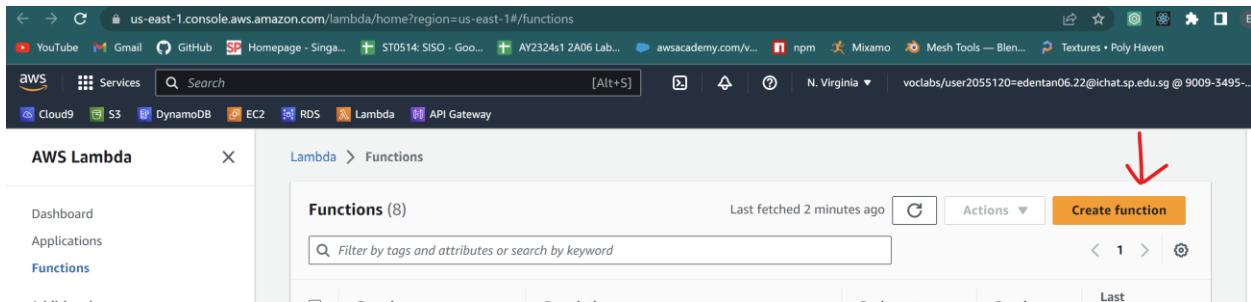
Now, run the frontend and backend server and see if you can view the updated Kelly Design 1 without going to the update page.



7.8 Creating getProfile Lambda Function

Step 1:

Go to AWS Lambda and create a new function by clicking on the “Create function” button.



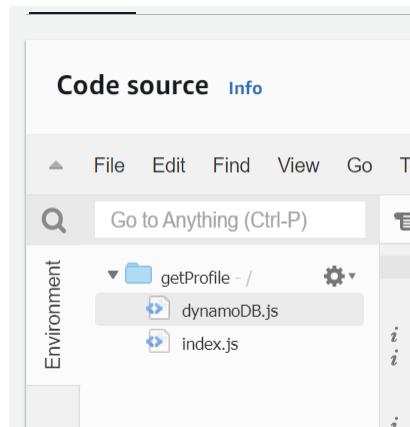
Step 2:

Fill in the function name as getProfile and choose Node.js 14.x for the Runtime and choose LabRole for the Role and click on “Create function” below.

A screenshot of the AWS Lambda "Create function" wizard. The "Basic information" step is active. The "Function name" field contains "getProfile", which is highlighted with a red border and has a warning message: "⚠ A function with that name already exists." Below the function name, it says "Use only letters, numbers, hyphens, or underscores with no spaces." The "Runtime" dropdown is set to "Node.js 14.x". Under "Architecture", "x86_64" is selected. In the "Permissions" section, "Change default execution role" is expanded, showing options for "Execution role" (with "Use an existing role" selected), "Existing role" (with "LabRole" chosen), and "View the LabRole role on the IAM console." Other sections like "Container image" and "Blueprints" are also visible.

Step 3:

Next, go into the getProfile Lambda function and go to the Code source at the bottom and make sure that you have a index.js and dynamoDB.js file created as shown below.



Step 4:

Input the following codes into eh index.js and dynamoDB.js files accordingly.

index.js

```
var dynamodbQuery = require('dynamodb');

exports.handler = async (event, context, callback) => {
  if (event.headers['user']){
    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);
});
}
};
```

dynamoDB.js

```
var AWS = require('aws-sdk');

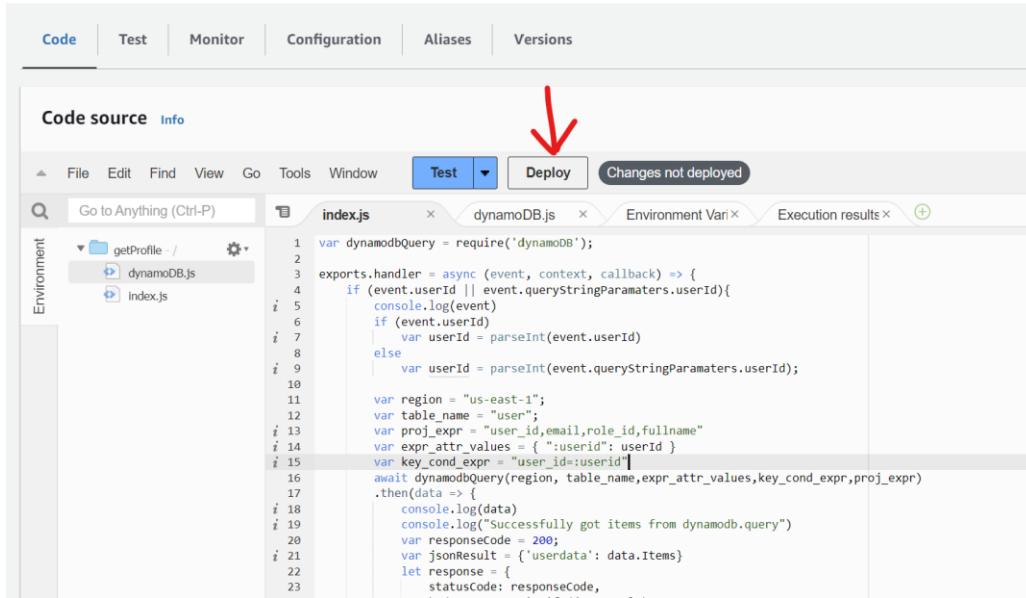
async function queryitems_dynamodb(region,
table_name,expr_attr_values,key_cond_expr,proj_expr) {
  console.log("In the queryitems_dynamodb method...")
  var dynamodb = new AWS.DynamoDB.DocumentClient({region: region});

  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
```

Step 5:

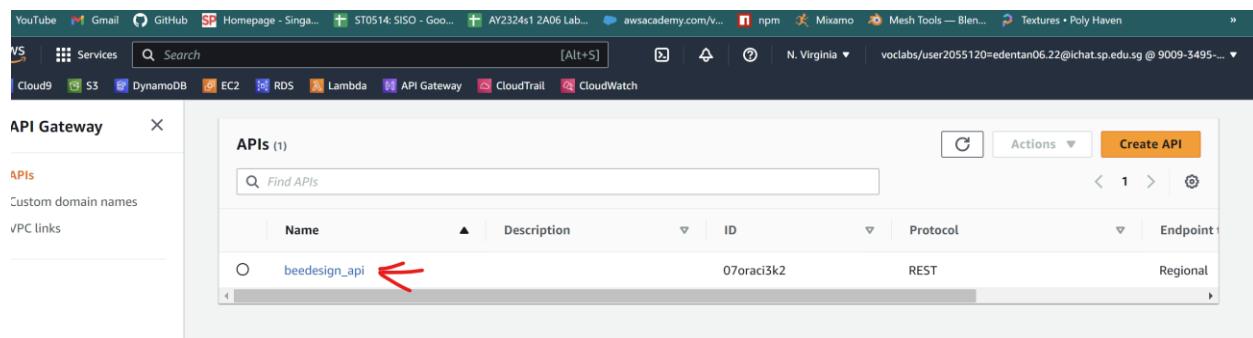
Click on the deploy button as shown below to save the codes that you have added.



7.9 Creating the getProfile API Gateway

Step 1:

Go into AWS API Gateway and click on the beedesign_api.



Step 2:

Under Actions, click on Create Resource and name it as getProfile as shown below and click on create resource.

The screenshot shows the API management interface. In the top navigation bar, the path is: APIs > beedesign_api (0/oracl3K2) > Resources. On the left, there's a tree view of resources: a root node with a dropdown arrow, followed by /, GET, OPTIONS, PUT, and a expanded node /manager with GET and OPTIONS methods. On the right, under the 'Actions' dropdown, a modal window titled '/ Methods' is open. The 'RESOURCE ACTIONS' section contains: Create Method, Create Resource (with a red arrow pointing to it), Enable CORS, and Edit Resource Documentation. Below that, in the 'API ACTIONS' section, is Deploy API. At the bottom of the modal, there's a 'New Child Resource' section with fields: 'Configure as proxy resource' (unchecked), 'Resource Name*' (set to 'getProfile'), 'Resource Path*' (set to '/ getprofile'), and 'Enable API Gateway CORS' (unchecked). At the very bottom of the modal are 'Cancel' and 'Create Resource' buttons.

New Child Resource

Use this page to create a new child resource for your resource.

Configure as proxy resource ⓘ

Resource Name*

Resource Path*

You can add path parameters using brackets. For example, the resource path `{username}` represents a path parameter called 'username'. Configuring `/{{proxy+}}` as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to `/foo`. To handle requests to `/`, add a new ANY method on the `/` resource.

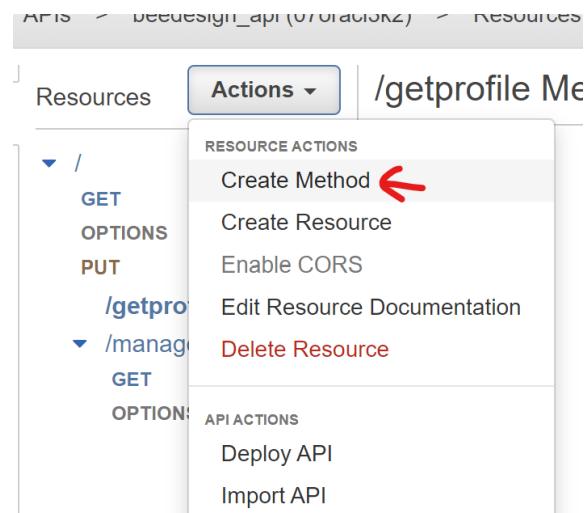
Enable API Gateway CORS ⓘ

* Required

Cancel Create Resource

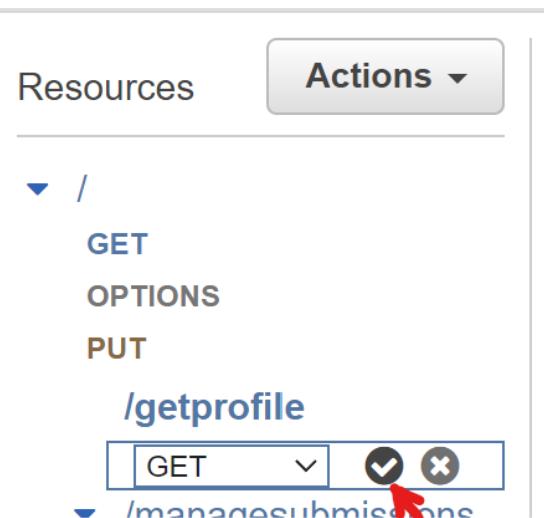
Step 3:

Next, click on Actions and select Created Method.



Step 4:

Select the GET request and click on the tick icon.



Step 5:

For the setup, follow the set up as shown below and click on the Save button.

/getprofile - GET - Setup 

Choose the integration point for your new method.

Integration type Lambda Function 
 HTTP 
 Mock 
 AWS Service 
 VPC Link 

Use Lambda Proxy integration 

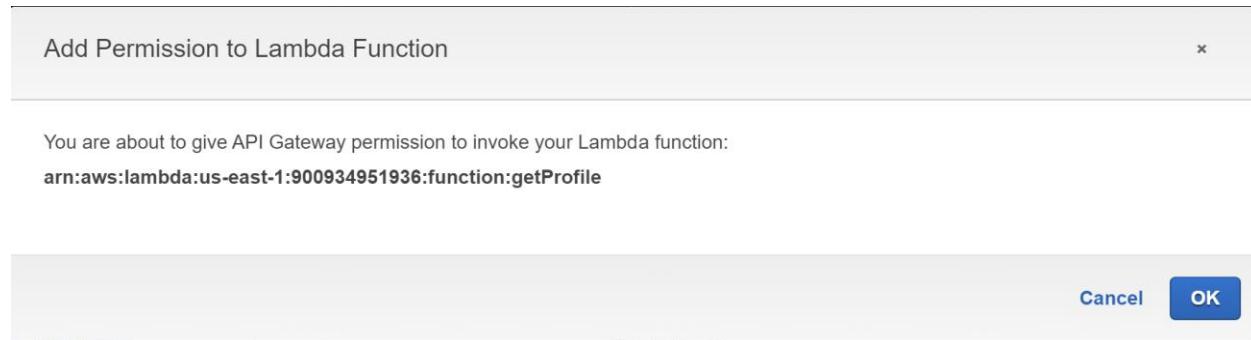
Lambda Region 

Lambda Function 

Use Default Timeout 

Save

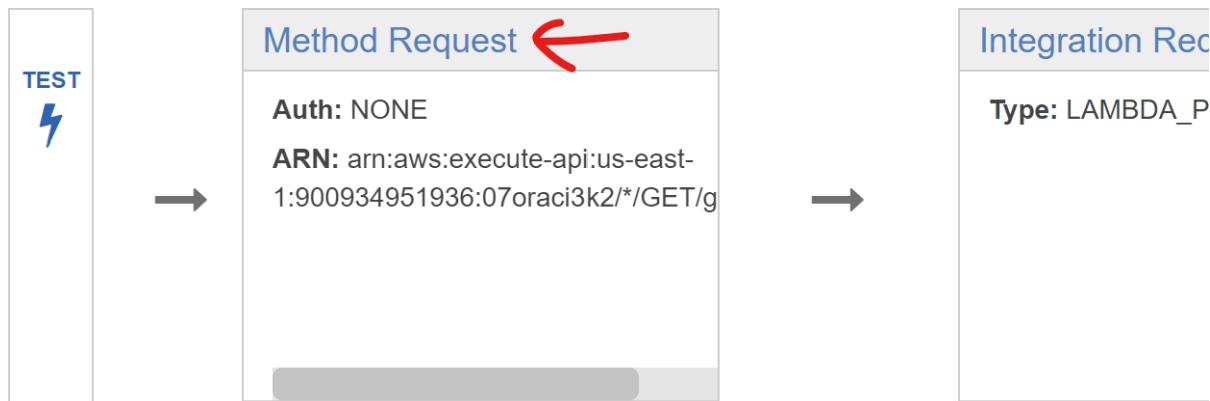
Click on the OK button.



Step 6:

Click on the Method Request as shown below.

/getprofile - GET - Method Execution



Step 7:

For the HTTP Request Headers add the following as shown below.

The screenshot shows the 'HTTP Request Headers' section of the AWS API Gateway Method Request configuration. It lists a single header: 'Access-Control-Allow-Headers'. The 'Required' and 'Caching' columns are present but have no visible values. At the bottom, there is a button labeled '+ Add header'.

Name	Required	Caching
Access-Control-Allow-Headers	<input type="checkbox"/>	<input type="checkbox"/>

Step 8:

Next, go back and click on the Integration Request.

/getprofile - GET - Method Execution



Step 9:

Fill in the Execution Role with LabRole and click on the tick icon, we will use the same execution role that we used from [7.2 Step 9](#).

2) > Resources > /getprofile (8ttohg) > GET Show hints

◀ Method Execution /getprofile - GET - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function ⓘ
 HTTP ⓘ
 Mock ⓘ
 AWS Service ⓘ
 VPC Link ⓘ

Use Lambda Proxy integration ⓘ

Lambda Region us-east-1 ✎

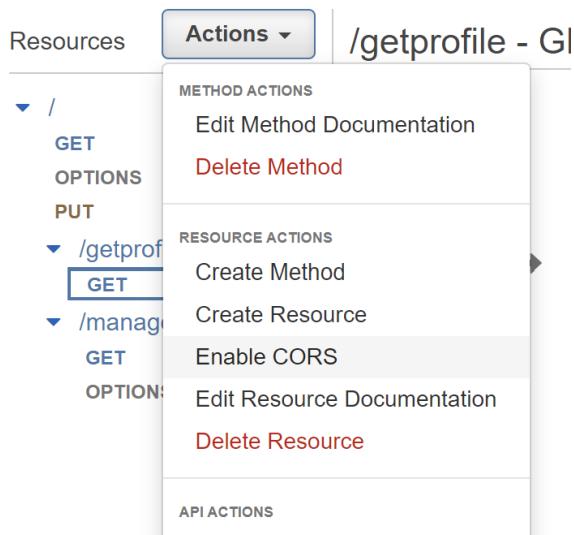
Lambda Function getProfile ✎

Execution role arn:aws:iam::900934951936:role/LabRole ✓ ✎

Invoke with caller credentials ⓘ

Step 10:

Click on Actions and click on Enable CORS.



Step 11:

Remember to check the following as shown below and add a “,user” before the closing quote for Access-Control_Allow_Headers and click on Enable CORS.

Enable CORS

Gateway Responses for DEFAULT 4XX DEFAULT 5XX ⓘ
beedesign_api API

Methods GET OPTIONS ⓘ

Access-Control-Allow-Methods GET, OPTIONS ⓘ

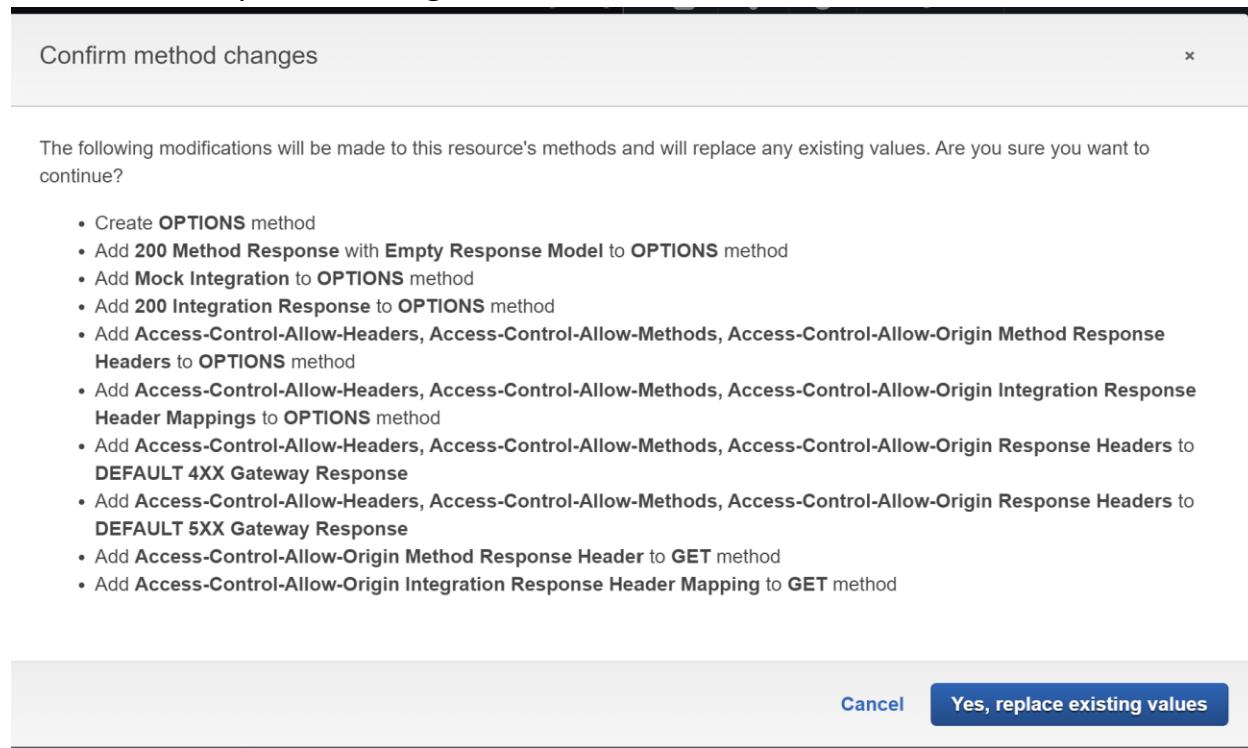
Access-Control-Allow-Headers ⓘ

Access-Control-Allow-Origin* ⓘ ⚠

▶ Advanced

Enable CORS and replace existing CORS headers

Click on Yes, replace existing values.



You should see the following.

Enable CORS

- ✓ Create **OPTIONS** method
- ✓ Add **200 Method Response with Empty Response Model** to **OPTIONS** method
- ✓ Add **Mock Integration** to **OPTIONS** method
- ✓ Add **200 Integration Response** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Method Response Headers** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Integration Response Header Mappings** to **OPTIONS** method
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Response Headers** to **DEFAULT 4XX Gateway Response**
- ✓ Add **Access-Control-Allow-Headers**, **Access-Control-Allow-Methods**, **Access-Control-Allow-Origin Response Headers** to **DEFAULT 5XX Gateway Response**
- ✓ Add **Access-Control-Allow-Origin Method Response Header** to **GET** method
- ✓ Add **Access-Control-Allow-Origin Integration Response Header Mapping** to **GET** method

Step 12:

Click on Actions and then click on Deploy API.

The screenshot shows the Oracle API Platform interface. At the top, there is a breadcrumb navigation: APIs > beedesign_api (07oraci3k2) > Resources. Below this, there is a main header with tabs: Resources, Actions (which is currently selected), and Enable CORS. The main content area displays a tree view of API resources and their methods. A context menu is open over the 'Actions' tab, listing several options under 'RESOURCE ACTIONS' and 'API ACTIONS'. The 'Deploy API' option under 'API ACTIONS' is highlighted with a light gray background.

Action	Description
Create Method	Method
Create Resource	Resource
Enable CORS	CORS
Edit Resource Documentation	Documentation
Delete Resource	Resource
Deploy API	API
Import API	API
Edit API Documentation	Documentation

Step 13:

Select test as deployment stage and click on Deploy button.

The screenshot shows a modal dialog box titled 'Deploy API'. Inside the dialog, there is a message: 'Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.' Below this message, there are two input fields: 'Deployment stage' with a dropdown menu set to 'test', and 'Deployment description' with an empty text area. At the bottom right of the dialog are two buttons: 'Cancel' and a blue 'Deploy' button.

Step 14:

You should be able to see the following.

The screenshot shows the AWS API Gateway interface. On the left, under 'Stages', the 'test' stage is selected. In the main area, the path '/getprofile' is chosen, and the 'GET' method is highlighted. At the top right, there is a 'Create' button. Below it, the title 'test - GET - /getprofile' is displayed. A blue bar at the top contains the 'Invoke URL: https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test/getprofile'. Below this, a note says 'Use this page to override the test stage settings for the GET to /getprofile method.' There are two radio buttons for 'Settings': one for 'Inherit from stage' (selected) and one for 'Override for this method'. At the bottom right is a 'Save Changes' button.

7.10 Adding the getProfile API Gateway to WinSCP

Step 1:

Go into WinSCP and go to the directory “/home/ec2-user/ESDE/FE/public/js”.

/home/ec2-user/ESDE/FE/public/js/					
Name	^	Size	Changed	Rights	Owner
..			27/6/2023 10:11:45 am	rwxr-xr-x	ec2-user
admin_check_user_sub...	js	8 KB	1/7/2023 9:04:18 am	rw-r--r--	ec2-user
admin_manage_user.js	js	9 KB	1/7/2023 9:04:39 am	rw-r--r--	ec2-user
admin_update_user.js	js	4 KB	1/7/2023 9:05:02 am	rw-r--r--	ec2-user
axios.js	js	14 KB	15/5/2023 11:58:12 pm	rw-r--r--	ec2-user
checkAdminRole.js	js	2 KB	4/7/2023 12:39:41 am	rw-r--r--	ec2-user
checkUserRole.js	js	2 KB	4/7/2023 12:39:50 am	rw-r--r--	ec2-user
global.js	js	1 KB	26/5/2023 8:14:18 am	rw-r--r--	ec2-user
login.js	js	3 KB	1/7/2023 9:03:56 am	rw-r--r--	ec2-user
manage_invite.js	js	2 KB	1/7/2023 9:05:45 am	rw-r--r--	ec2-user
profile.js	js	2 KB	1/7/2023 9:05:55 am	rw-r--r--	ec2-user
register.js	js	2 KB	1/7/2023 9:06:16 am	rw-r--r--	ec2-user
submit_design.js	js	3 KB	1/7/2023 9:06:29 am	rw-r--r--	ec2-user
update_design.js	js	4 KB	11/7/2023 2:29:50 am	rw-r--r--	ec2-user
user_manage_submissi...	js	9 KB	1/7/2023 9:07:06 am	rw-r--r--	ec2-user

Step 2:

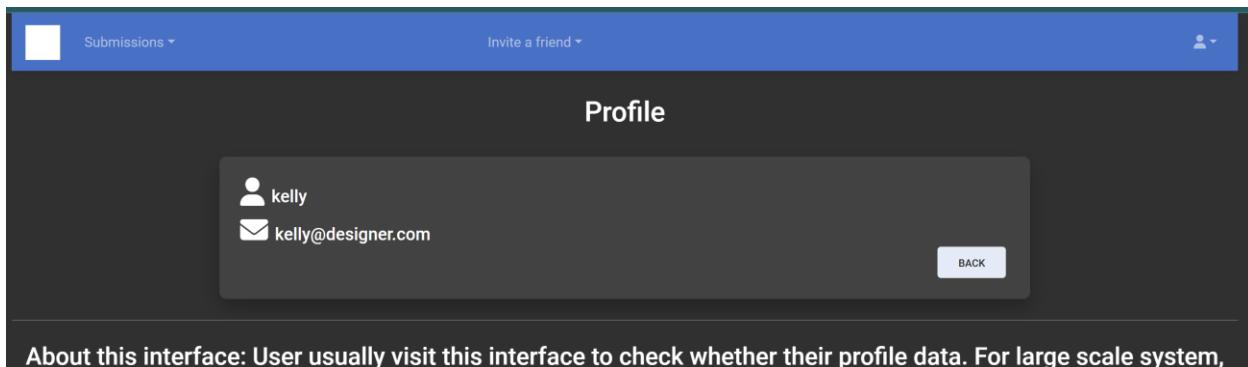
Click on the profile.js file and add the API Gateway Invoke URL to the URL as shown below and save the file.

URL Link: 'https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test/getprofile?userId=' + userId

```
function getOneUser() {
  const baseUrl = "https://18.205.232.250:5000";
  let userId = sessionStorage.getItem("user_id");
  let token = sessionStorage.getItem("token");
  axios({
    headers: {
      user: userId,
      authorization: `Bearer ${token}`,
    },
    method: "get",
    url: 'https://07oraci3k2.execute-api.us-east-1.amazonaws.com/test/getprofile', ←
  })
  .then(function (response) {
    //Using the following to inspect the response.data data structure
    //before deciding the code which dynamically populate the elements with data.
    console.dir(response.data.userdata); ←
    const record = response.data.userdata; ←
    $("#fullNameOutput").append(record[0].fullname); //text(record.fullname); ←
    $("#emailOutput").append(record[0].email); //text(record.email); ←
  })
  .catch(function (response) {
    //Handle error
    console.dir(response);
  });
}
```

Step 3:

Now, run the frontend and backend server and see if you can view the current logged in user's profile.

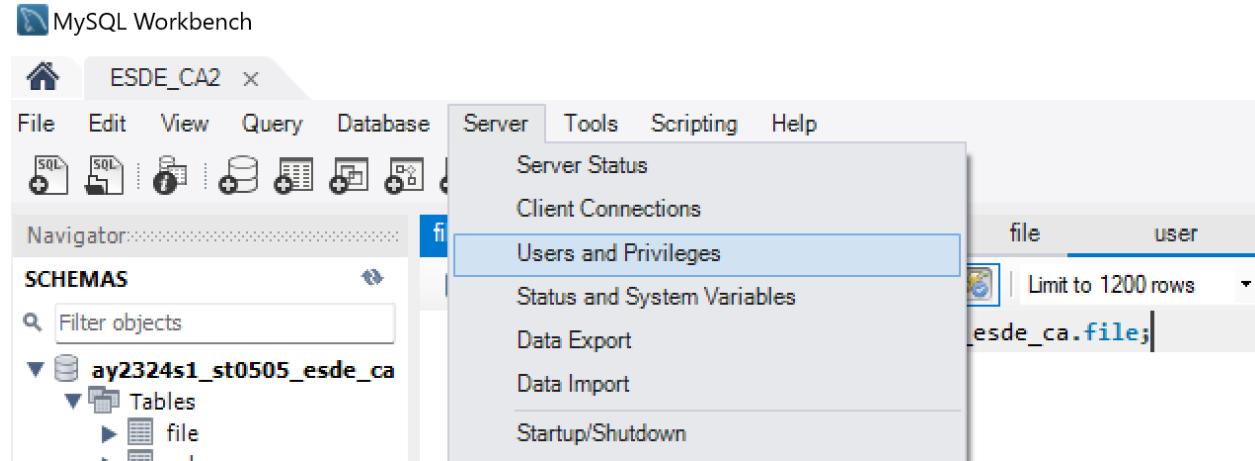


About this interface: User usually visit this interface to check whether their profile data. For large scale system,

7.11 Giving Least Privilege for Database credentials

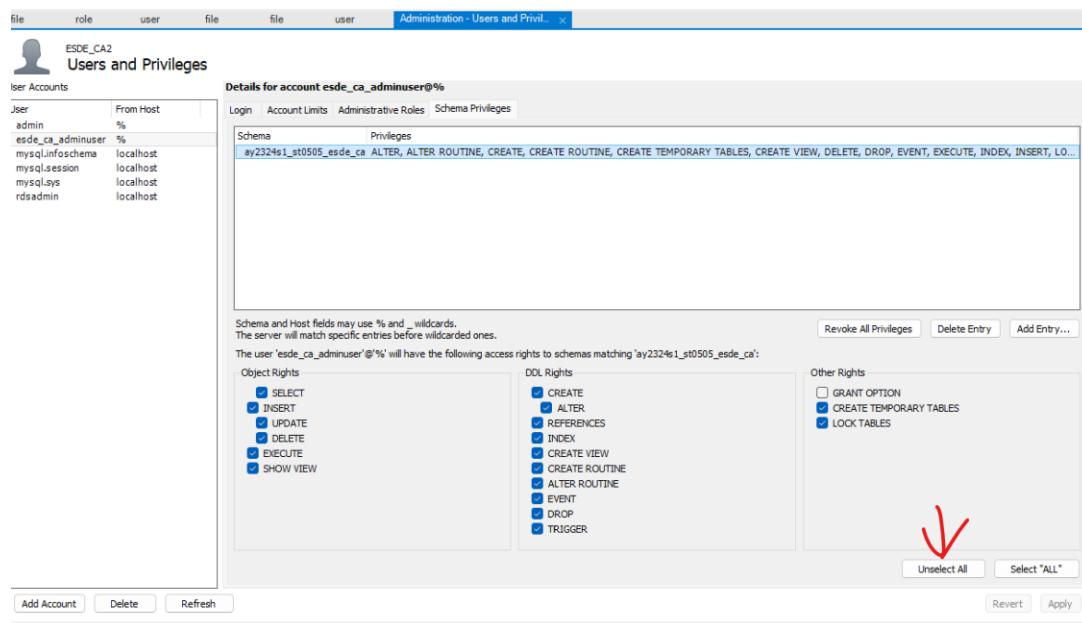
Step 1:

Go to MySQL Workbench and go to users and Privileges.



Step 2:

Go to Schema Privileges and click on unselect all.



Step 3:

Select all under Object Rights and click on Apply.

Details for account esde_ca_adminuser@%

Schema Privileges

ay2324s1_st0505_esde_ca DELETE, EXECUTE, INSERT, SELECT, SHOW VIEW, UPDATE

Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.

The user 'esde_ca_adminuser'@'%' will have the following access rights to schemas matching 'ay2324s1_st0505_esde_ca':

Object Rights

SELECT
INSERT
UPDATE
DELETE
EXECUTE
SHOW VIEW

DDL Rights

CREATE
ALTER
REFERENCES
INDEX
CREATE VIEW
CREATE ROUTINE
ALTER ROUTINE
EVENT
DROP
TRIGGER

Other Rights

GRANT OPTION
CREATE TEMPORARY TABLE
LOCK TABLES

Unselect

Step 4:

You should be able to see all the recent selected privileges for the User.

User Accounts

User	From Host
admin	%
esde_ca_adminuser	%
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
rdsadmin	localhost

Details for account esde_ca_adminuser@%

Schema Privileges

ay2324s1_st0505_esde_ca DELETE, EXECUTE, INSERT, SELECT, SHOW VIEW, UPDATE

7.12 MySQL RDS Database Security, Encryption and Logging

Step 1:

Go to AWS EC2 and go to Security Groups and select the default Security group name and click on Edit Inbound Rules as shown below.

The screenshot shows the AWS Management Console interface for Security Groups. On the left, the navigation pane is visible with various services like Cloud9, S3, DynamoDB, EC2, RDS, Lambda, API Gateway, CloudTrail, CloudWatch, and IAM. Under Network & Security, the 'Security Groups' option is selected, indicated by a red arrow. The main content area displays a table of security groups. One row is selected, and its details are shown in a modal window. The modal window has tabs for Details, Inbound rules, Outbound rules, and Tags. The Inbound rules tab is active, as indicated by a red arrow. It contains a table of inbound rules, with two entries visible: one for MySQL/Aurora (TCP port 3306) and another for All traffic (All port range). A red arrow points to the 'Edit inbound rules' button at the top right of this table.

Step 2:

Open a separate tab and go to AWS EC2 Instances and choose the EC2 Instance using the RDS and copy the Private IP Address.

The screenshot shows the AWS Management Console interface for Instances. The navigation pane on the left includes Cloud9, S3, DynamoDB, EC2, RDS, Lambda, API Gateway, CloudTrail, CloudWatch, and IAM. The main content area shows a table of instances. One instance, 'ESDE CA2 Web...', is selected, indicated by a red arrow. Below the table, a detailed view for the selected instance is displayed. The instance summary section shows the Public IPv4 address (18.205.232.250), Instance state (Running), and Private IPv4 address (172.31.89.155, indicated by a red arrow). Other details include the Public IPv4 DNS (ec2-18-205-232-250.compute-1.amazonaws.com) and the Private IP DNS name (IPv4 only).

Step 3:

Next, paste the Private IP Address into the text box beside the Source as 172.31.89.155/32 and save it.

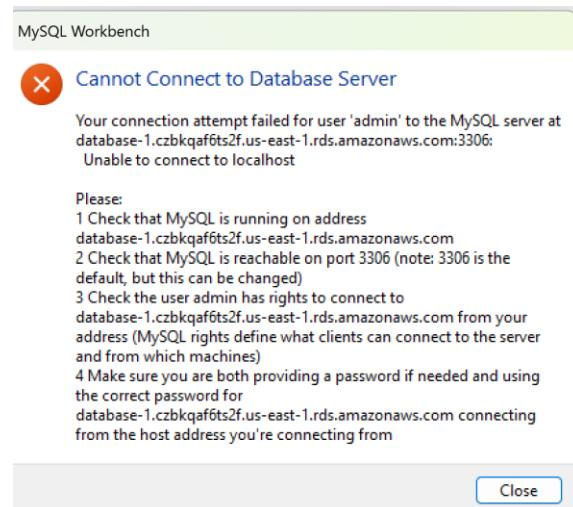
The screenshot shows the AWS Management Console with the URL [vocabs/user2055120=edentan06.22@ichat.sp.edu.sg @ 9009-3495-...](#). The navigation bar includes Cloud9, S3, DynamoDB, EC2, Lambda, API Gateway, CloudTrail, CloudWatch, and IAM. The main content is titled "Edit inbound rules" with a "Info" link. It says, "Inbound rules control the incoming traffic that's allowed to reach the instance." A table lists two security group rules:

Inbound rules	Type	Protocol	Port range	Source	Description - optional
sgr-0278d7491260c1620	MySQL/Aurora	TCP	3306	Custom	172.31.89.155/32
sgr-09b856d14fe719b4f	All traffic	All	All	Custom	sg-039f3ae8d53997842

Buttons at the bottom include "Add rule", "Cancel", "Preview changes", and a highlighted "Save rules" button.

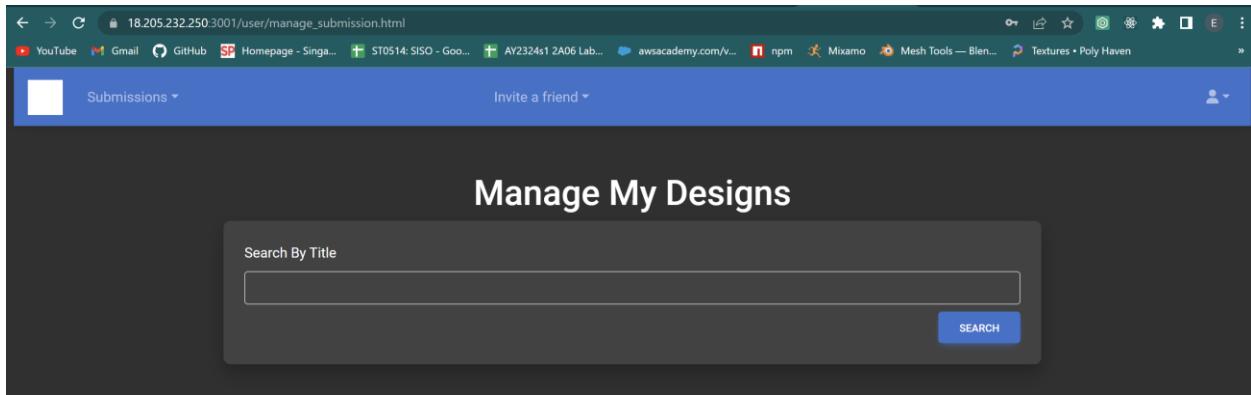
Step 4:

Now go to your MySQL Workbench and you should see that you are not able to view your database.



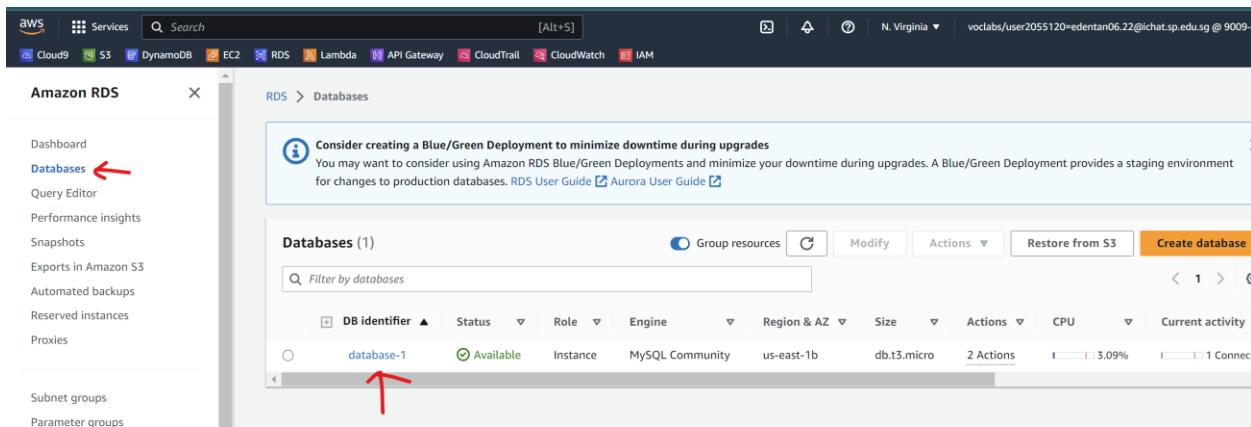
Step 5:

Check that you are still able to login to the Bee Design Website by logging in as Kelly.



Step 6:

Now, go to AWS RDS and click on Databases and click on database-1.



Step 7:

Under Actions, click on Take snapshot.

The screenshot shows the AWS RDS console for a database named 'database-1'. The 'Summary' tab is selected. On the right, a context menu is open under the 'Actions' button. The 'Take snapshot' option is highlighted with a red arrow. Other options in the menu include 'Modify', 'Quick Actions - New', 'Convert to Multi-AZ deployment', 'Stop temporarily', 'Reboot', 'Delete', 'Set up EC2 connection', 'Create read replica', 'Create Aurora read replica', 'Create Blue/Green Deployment - new', 'Promote', 'Restore to point in time', 'Migrate snapshot', and 'Create ElastiCache cluster - new'.

Give it a name of encryptedSnapshot and click on the Take snapshot button.

The screenshot shows the 'Take DB Snapshot' dialog box. It has a 'Preferences' section with a note: 'To take a DB Snapshot, choose a DB Instance and name your DB Snapshot.' Below is a 'DB Instance' dropdown set to 'database-1'. The 'Snapshot Name' field is filled with 'encryptedSnapshot'. A note below says: 'Snapshot identifier is case insensitive, but stored as all lower-case, as in "mysnapshot". Cannot be null, empty, or blank. Must contain from 1 to 255 alphanumeric characters or hyphens. First character must be a letter. Cannot end with a hyphen or contain two consecutive hyphens.' At the bottom are 'Cancel' and 'Take snapshot' buttons, with 'Take snapshot' highlighted by a red arrow.

Step 8:

Make sure that your snapshot status is complete. Next, select the snapshot and click on Actions and select “Restore snapshot”.

RDS > Snapshots

Snapshots

Manual System Shared with me Public Backup service Exports in Amazon S3

Manual snapshots (1)

Filter by manual snapshots

Snapshot name	DB instance or cluster	Snapshot creation time
encryptedsnapshot	database-1	July 22, 2023, 17:56 (UTC+08:00)

Actions ▾ Take snapshot

- Restore snapshot
- Copy snapshot
- Share snapshot
- Migrate snapshot
- Export to Amazon S3
- Delete snapshot

Step 9:

For the DB Instance identifier name, it as encrypted-rds-database as shown below.

Settings

DB snapshot ID
The identifier for the DB snapshot.
encryptedsnapshot

DB instance identifier [Info](#)
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.
encrypted-rds-database

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

For Availability and durability select Single DB instance.

Availability and durability

Deployment options [Info](#)

The deployment options below are limited to those supported by the engine you selected above.

Multi-AZ DB Cluster - new

Creates a DB cluster with a primary DB instance and two readable standby DB instances, with each DB instance in a different Availability Zone (AZ). Provides high availability, data redundancy and increases capacity to serve read workloads.

Multi-AZ DB instance

Creates a primary DB instance and a standby DB instance in a different AZ. Provides high availability and data redundancy, but the standby DB instance doesn't support connections for read workloads.

Single DB instance

Creates a single DB instance with no standby DB instances.

And for Instance configuration select Burstable classes and then db.t3.micro.

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class [Info](#)

Standard classes (includes m classes)

Memory optimized classes (includes r and x classes)

Burstable classes (includes t classes)

db.t3.micro

2 vCPUs 1 GiB RAM Network: 2,085 Mbps



Include previous generation classes

For Connectivity set public access to Yes.

Connectivity [Info](#)

Virtual private cloud (VPC) [Info](#)
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-009f4abd160caa002) ▾
6 Subnets, 6 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

DB subnet group [Info](#)
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

default-vpc-009f4abd160caa002 ▾
6 Subnets, 6 Availability Zones

Public access [Info](#)

Yes
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

No
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) [Info](#)
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the

For Additional configuration select all the Log exports as shown below.

Copy tags to snapshots

Log exports
Select the log types to publish to Amazon CloudWatch Logs

Audit log
 Error log
 General log
 Slow query log

IAM role
The following service-linked role is used for publishing logs to CloudWatch Logs.

RDS service-linked role

Maintenance
Auto minor version upgrade [Info](#)

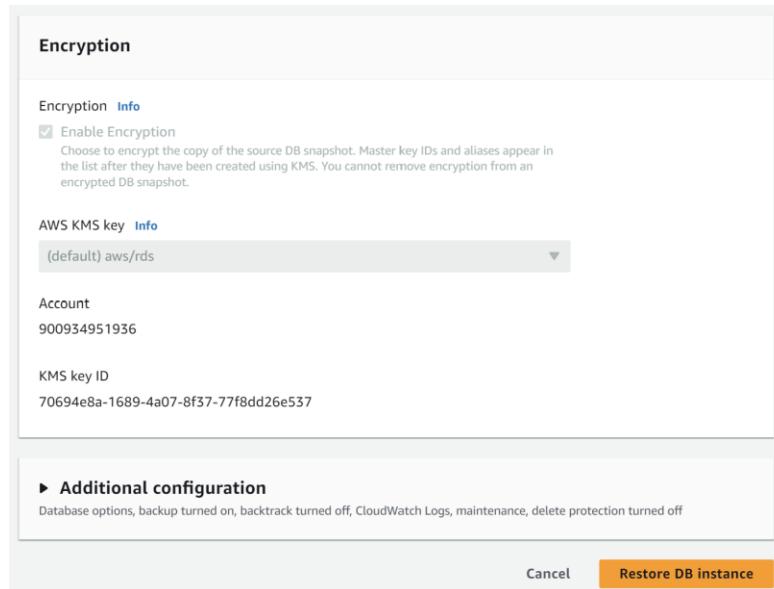
Enable auto minor version upgrade
Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.

Deletion protection

Enable deletion protection
Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

[Cancel](#) [Restore DB instance](#)

And for Encryption make sure that Enable Encryption is checked and then click on “Restore DB instance”.



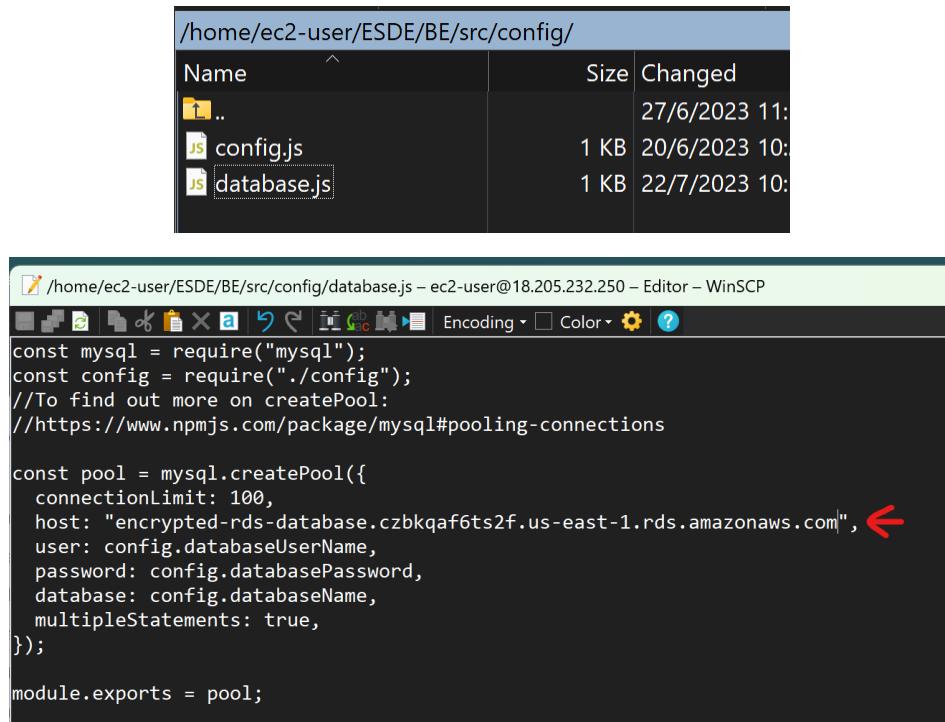
Step 9:

Go into the encrypted database and copy the Endpoint.

The screenshot shows the 'Amazon RDS' dashboard with the 'Databases' tab selected. The main view displays the details for a database named 'encrypted-rds-database'. In the 'Connectivity & security' section, the 'Endpoint' field is highlighted with a red arrow pointing to it. The endpoint value is 'encrypted-rds-database.czbkqaf6ts2f.us-east-1.rds.amazonaws.com'. Other fields in this section include 'Port' (3306), 'Networking' (Availability Zone: us-east-1d, VPC: vpc-009f4abd160caa002), and 'Security' (VPC security groups: default (sg-039f3ae8d53997842), Active).

Step 10:

Go to WinSCP and go to the directory as shown below and add the encrypted rds database endpoint to the host.



The screenshot shows two windows from the WinSCP interface. The top window is a file browser titled '/home/ec2-user/ESDE/BE/src/config/'. It lists three files: 'config.js' (1 KB, 20/6/2023 10:), 'database.js' (1 KB, 22/7/2023 10:), and a folder named '..'. The bottom window is a code editor for 'database.js'. The code is as follows:

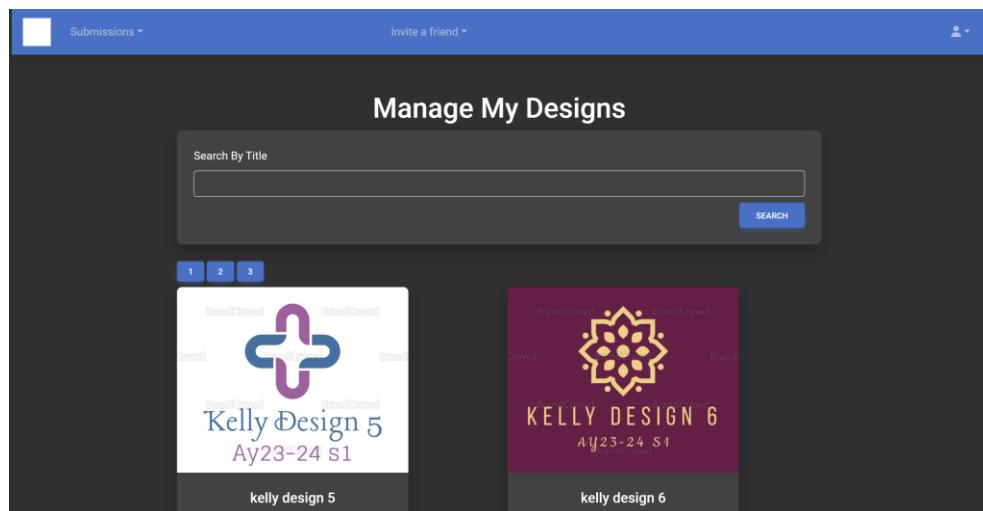
```
const mysql = require("mysql");
const config = require("./config");
//To find out more on createPool:
//https://www.npmjs.com/package/mysql#pooling-connections

const pool = mysql.createPool({
  connectionLimit: 100,
  host: "encrypted-rds-database.czbkqaf6ts2f.us-east-1.rds.amazonaws.com", ←
  user: config.databaseUserName,
  password: config.databasePassword,
  database: config.databaseName,
  multipleStatements: true,
});

module.exports = pool;
```

Step 11:

Now go to the Bee Design Website and try logging in if you are able to login then you have successfully encrypted the rds database.



Step 12:

Go to AWS RDS and click on Parameter groups and click on Create parameter group.

The screenshot shows the AWS RDS Parameter groups page. On the left, there's a sidebar with various RDS-related options like Dashboard, Databases, and Subnet groups. The main area shows a table titled 'Parameter groups (1)' with one entry: 'default.mysql8.0'. The table has columns for Name, Family, Type, and Description. A large orange button at the top right labeled 'Create parameter group' is highlighted with a red arrow pointing to it.

Step 13:

For the group name and description name it as shown below.

The screenshot shows the 'Create parameter group' wizard. The first step, 'Parameter group details', is displayed. It asks for a 'Parameter group family' (set to 'aurora-mysql5.7'), 'Type' (set to 'DB Parameter Group'), 'Group name' (set to 'saveme'), and 'Description' (set to 'save'). At the bottom, there are 'Cancel' and 'Create' buttons, with 'Create' being highlighted.

Step 14:

Go into the created Parameter group and make sure that log_output value is FILE.

RDS > Parameter groups > saveme

Parameters

Name	Values	Allowed values	Modifiable	Source	Apply type	Data type	Description
log_output	FILE	TABLE, FILE, NONE	true	system	dynamic	string	Controls where to store query logs

Edit parameters

Step 15:

Head over to CloudWatch and under Log groups see if there is a /aws/rds/instance/encrypted-rds-database/error

CloudWatch

Favorites and recentes

CloudWatch > Log groups

Log groups (9)

By default, we only load up to 10000 log groups.

Log group	Data protection	Sensitive data c...	Retention
/aws/apigateway/welcome	-	-	Never expire
/aws/lambda/RedshiftEventSubscription	-	-	Never expire
/aws/lambda/RedshiftOverwatch	-	-	Never expire
/aws/lambda/RoleCreationFunction	-	-	Never expire
/aws/lambda/getDesigns	-	-	Never expire
/aws/lambda/getProfile	-	-	Never expire
/aws/lambda/manageSubmissions	-	-	Never expire
/aws/lambda/updateDesign	-	-	Never expire
/aws/rds/instance/encrypted-rds-database/error	-	-	Never expire

Step 16:

Click on the Log group and click on the encrypted-rds-database Log stream.

The screenshot shows the AWS CloudWatch Log Groups interface. At the top, there is a breadcrumb navigation: /aws/rds/instance/encrypted-rds-database/error. Below it is a header with actions: Actions, View in Logs Insights, Start tailing, and Search log group. A sub-header Log group details is expanded, showing the ARN (arn:aws:logs:us-east-1:900934951936:log-group:/aws/rds/instance/encrypted-rds-database/error:*) and various metrics like stored bytes, metric filters, and subscription filters. Below this is a navigation bar with tabs: Log streams (selected), Tags, Metric filters, Subscription filters, Contributor Insights, and Data protection. The main content area shows a table of log streams. The first row is a header with columns: Log stream, Last event time, and a delete button. The second row contains the log stream name "encrypted-rds-database" and its last event time "2023-07-10 01:38:55 (UTC+08:00)". A red arrow points to the "encrypted-rds-database" link.

Step 17:

You should be able to see all the encrypted-rds-database logs as shown below.

The screenshot shows the AWS CloudWatch Log Events interface. The breadcrumb navigation is CloudWatch > Log groups > /aws/rds/instance/encrypted-rds-database/error > encrypted-rds-database. The header includes actions: Actions, Start tailing, Create metric filter, and a filter bar. Below the header is a toolbar with filter, clear, and display options. The main content area is titled "Log events" and contains a message: "There are older events to load. Load more.". A table lists log events with columns: Timestamp and Message. The table shows several warning messages from MySQL servers about deprecated syntax and configuration parameters. A red arrow points to the timestamp of the last event listed.

Timestamp	Message
2023-07-23T20:56:53.533+08:00	2023-07-23T12:56:53.533358Z 0 [Warning] [MY-011068] [Server] The syntax 'log_slave_updates' is deprecated and will be removed in a futu...
2023-07-23T20:56:53.533+08:00	2023-07-23T12:56:53.533378Z 0 [Warning] [MY-011069] [Server] The syntax '--master-info-repository' is deprecated and will be removed in...
2023-07-23T20:56:53.533+08:00	2023-07-23T12:56:53.533385Z 0 [Warning] [MY-011069] [Server] The syntax '--master-info-repository' is deprecated and will be removed in...
2023-07-23T20:56:53.533+08:00	2023-07-23T12:56:53.533408Z 0 [Warning] [MY-011069] [Server] The syntax '--relay-log-info-file' is deprecated and will be removed in a ...
2023-07-23T20:56:53.533+08:00	2023-07-23T12:56:53.533452Z 0 [Warning] [MY-011069] [Server] The syntax '--relay-log-info-repository' is deprecated and will be removed...
2023-07-23T20:56:53.533+08:00	2023-07-23T12:56:53.533478Z 0 [Warning] [MY-011068] [Server] The syntax 'skip_slave_start' is deprecated and will be removed in a futur...
2023-07-23T20:56:53.533+08:00	2023-07-23T12:56:53.533488Z 0 [Warning] [MY-011068] [Server] The syntax 'slave_exec_mode' is deprecated and will be removed in a future...
2023-07-23T20:56:53.533+08:00	2023-07-23T12:56:53.533494Z 0 [Warning] [MY-011068] [Server] The syntax 'slave_load_tmpdir' is deprecated and will be removed in a futu...
2023-07-23T20:56:53.536+08:00	2023-07-23T12:56:53.536749Z 0 [Warning] [MY-010918] [Server] 'default_authentication_plugin' is deprecated and will be removed in a futu...
2023-07-23T20:56:53.541+08:00	2023-07-23T12:56:53.541389Z 0 [System] [MY-010116] [Server] /rdsdbbin/mysql/bin/mysqld (mysqld 8.0.32) starting as process 795
2023-07-23T20:56:53.743+08:00	2023-07-23T12:56:53.743753Z 0 [Warning] [MY-010075] [Server] No existing UUID has been found, so we assume that this is the first time ...
2023-07-23T20:56:53.756+08:00	2023-07-23T12:56:53.756486Z 0 [Warning] [MY-013907] [InnoDB] Deprecated configuration parameters innodb_log_file_size and/or innodb_log...

References

- <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html>