

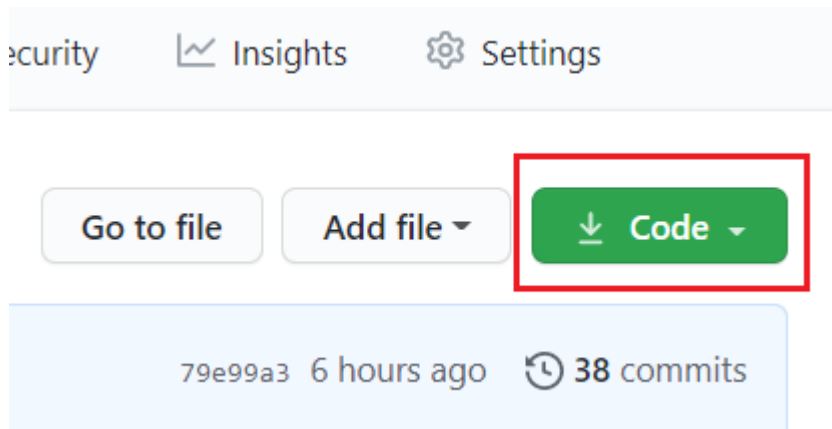
Table Of Content

1 Cloning Repository	2
1.1 Cloning a repository using the command line	2
2 Software Installation	4
2.1 Download and Install Visual Studio Code	4
2.2 Download and Install Postman	5
2.3 Install Node.js and NPM on Windows	7
2.3.1 Download Node.js Installer	7
2.3.2 Install Node.js and NPM from browser	8
2.3.3 Verify Installation	8
3 Setup Command	9
3.1 Install jsonschema package	9
4 Setup Database	Error! Bookmark not defined.
4.1 Setup Database on ElephantSQL	10
4.1.1 Create a new instance	10
4.1.2 Connect application to database	13
4.1.3 Create table on database created on ElephantSQL	16
4.1.4 Insert records into tables	18
5 Start the Application	20
5.1 node server.js	21
6 Testing the Project	22
6.1 Test using Postman	22

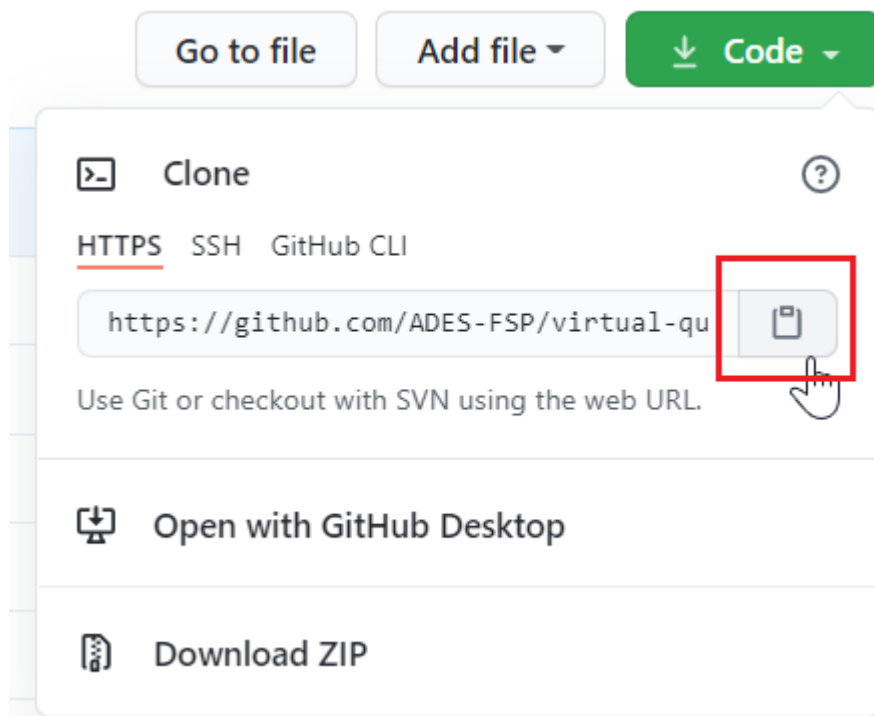
1 | Cloning Repository

1.1 | Cloning a repository using the command line

1. Navigate to the main page of the repository on GitHub.
2. Below the navigation bar, click [download Code](#).

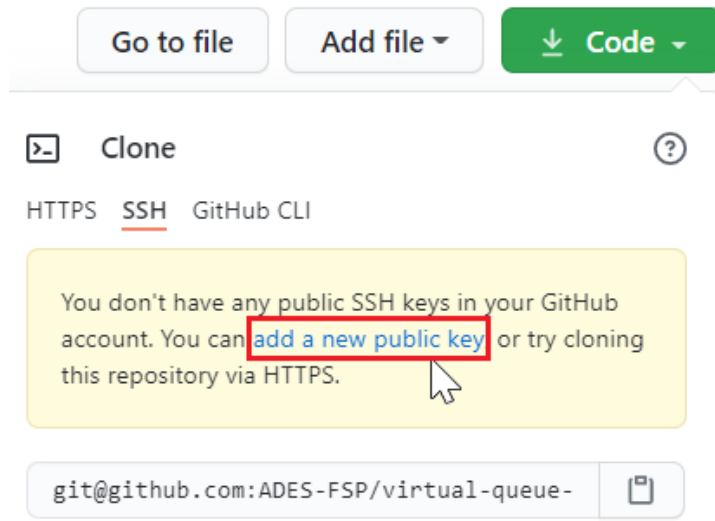



3. To clone the repository by using HTTPS, click  under **HTTPS**.

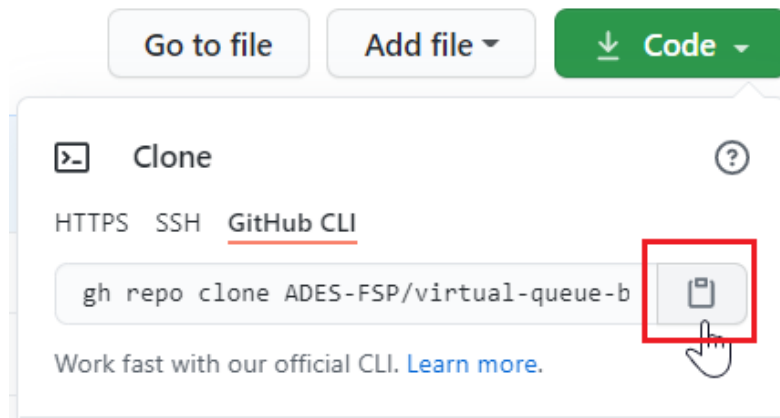


To clone the repository by using an SSH key, click Use SSH, then click .

If you do not have any public SSH keys in your GitHub account. You can add a new public key by clicking on [add a new public key](#).



To clone a repository by using GitHub CLI, click **GitHub CLI**, then click .



4. Open Git Bash.
5. On Git Bash, change the current directory to the location where you want to store the cloned directory.
6. Type `git clone`, then paste the URL that copied earlier.

`$ git clone (URL)`

An example is shown below:

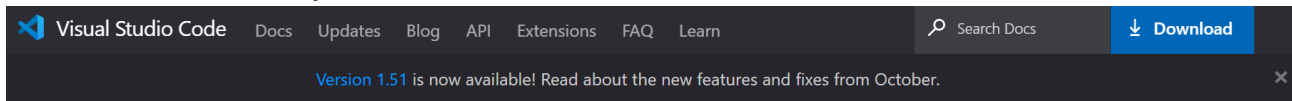
```
$ git clone gh repo clone ADES-FSP/virtual-queue-backend-2b03-adesca1
```

7. Hit Enter to create a clone on your local host.

2 | Software Installation

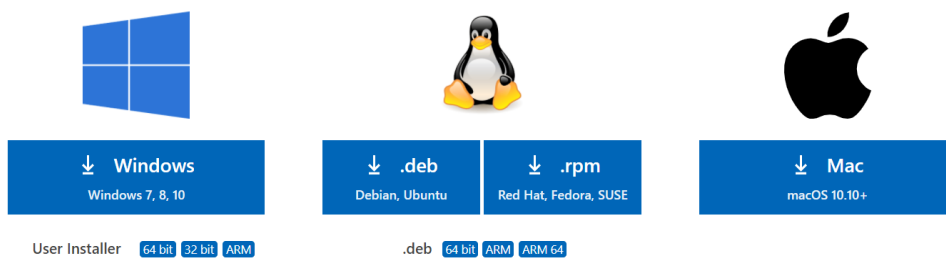
2.1 | Download and Install Visual Studio Code

1. Navigate to the website: <https://code.visualstudio.com/download> to download and install Visual Studio Code if you do not have it.



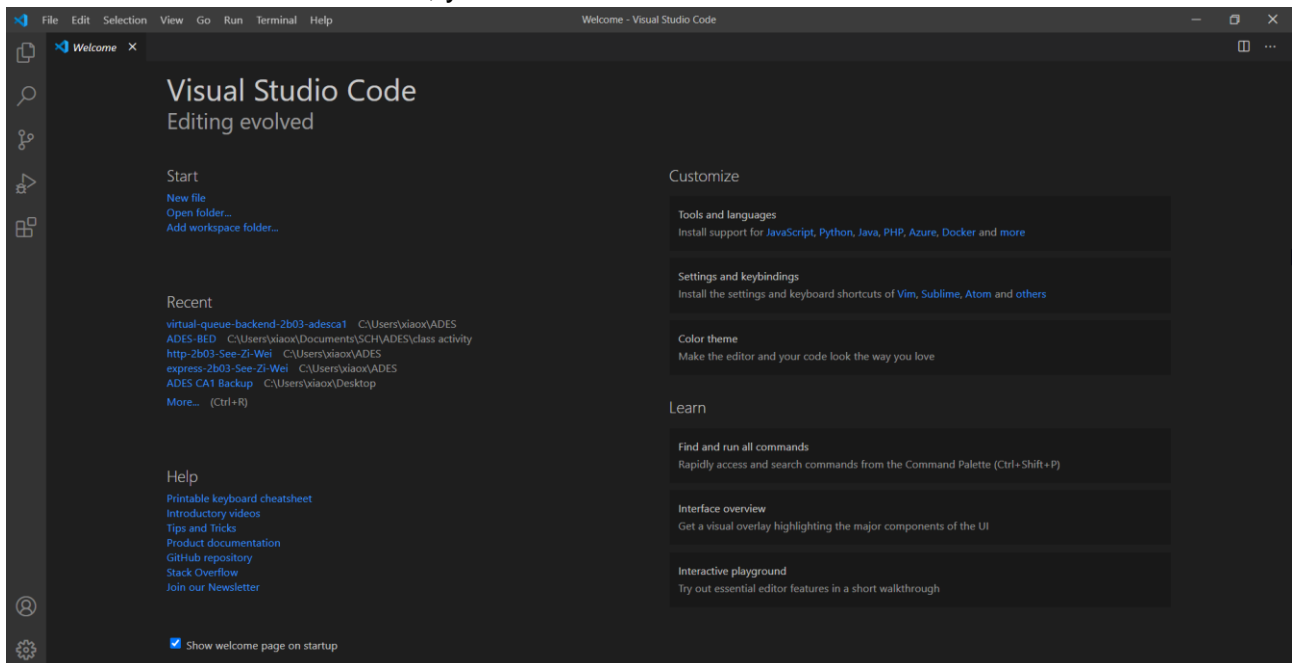
Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



2. Once the exe file is downloaded, open the exe file to install it on your system.

3. After successful installation, you will see the below Visual Studio Code toolkit.



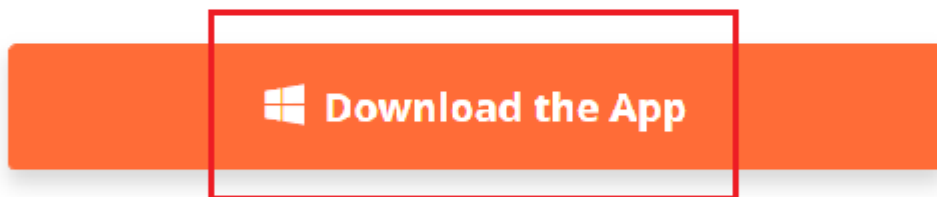
2.2 | Download and Install Postman

We will use Postman to test if you have successfully set up the application. Below are the guides on how to download Postman.

1. To install Postman, go to this link: <https://www.postman.com/downloads/> and click [Download the App](#).

The Postman app

The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience.

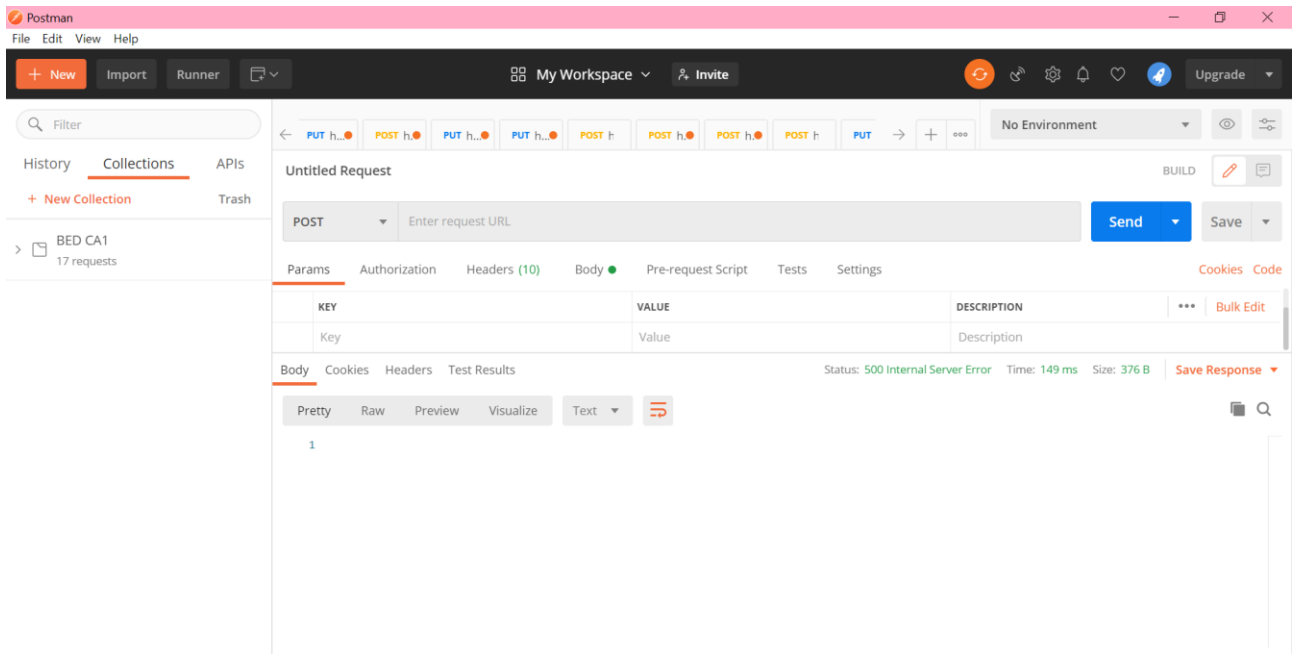


By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

Version 7.36.0 | [Release Notes](#) | [Product Roadmap](#)

***Not your OS?** Download for Mac ([macOS](#)) or Linux ([x64](#))*

2. Once the exe file is downloaded, open the exe file to install it on your system.
3. Once installation of Postman is completed, you will be asked to create an account. You can create an account on Postman now or create an account later by clicking on "Take me straight to the app, I'll create an account another time" link.
4. After successful installation and registration, you will see the below Postman toolkit.



2.3 | Install Node.js and NPM on Windows




2.3.1 | Download Node.js Installer

1. Navigate to <https://nodejs.org/en/download/>.
2. Click onto the Windows Installer button.
(Node.js installer includes the NPM package manager.)

Downloads

Latest LTS Version: **14.15.1** (includes npm 6.14.8)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features	
 Windows Installer node-v14.15.1-x64.msi	 macOS Installer node-v14.15.1.pkg	 Source Code node-v14.15.1.tar.gz

Windows Installer (.msi)
Windows Binary (.zip)
macOS Installer (.pkg)
macOS Binary (.tar.gz)
Linux Binaries (x64)
Linux Binaries (ARM)
Source Code

32-bit	64-bit
32-bit	64-bit
64-bit	
64-bit	
64-bit	
ARMv7	ARMv8
node-v14.15.1.tar.gz	

2.3.2 | Install Node.js and NPM from browser

1. After the exe file of the windows installer is downloaded, double-click onto the exe file to launch it.
2. Click **Run** when the system asks if you want to run the software.
3. Click **Next** when the system shows welcome to the Node.js Setup Wizard.
4. The next screen will be reviewing the license agreement. Click **Next** if you agree to the terms.
5. When the system prompts you for the installation location, you can leave the default location or change it to the location that you want to install it. Then, click **Next**.
6. After that, the system will ask you to select the components to include or remove from the installation. You can accept the defaults by clicking **Next** unless you have a specific need.
7. Click **Install** to run the installer.
8. Click **Finish** when it finishes installing.

2.3.3 | Verify Installation

1. Open a command prompt, type: `node -v` and hit Enter.
The system should display the Node.js version installed on your system.
2. Type: `npm -v` and hit Enter.
The system should display the npm version installed on your system.

3 | Setup Command

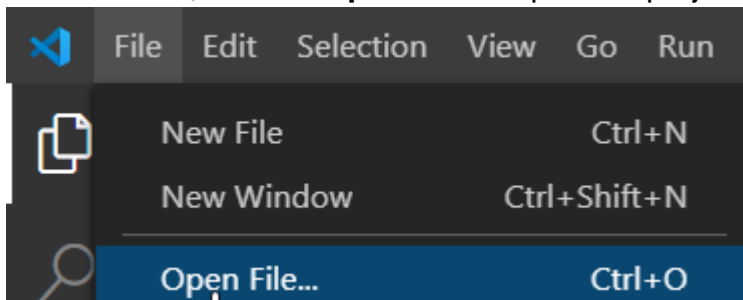
3.1 | Install jsonschema package

1. Open Visual Studio Code.

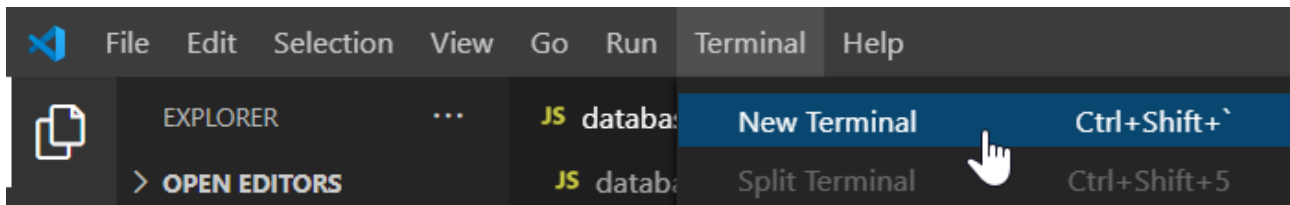


Visual Studio Code
App

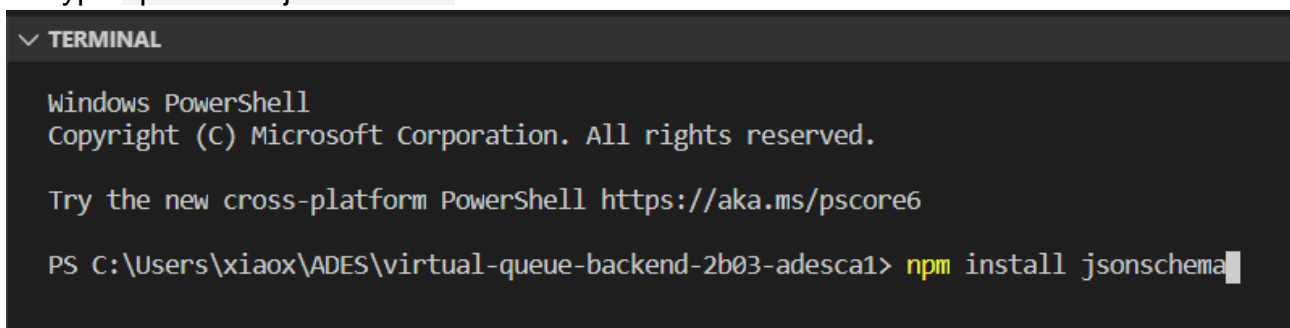
2. Under File, click on **Open File** to open the project you stored previously.



3. Under Terminal, click on **New Terminal**.



4. Type `npm install jsonschema` and hit Enter.



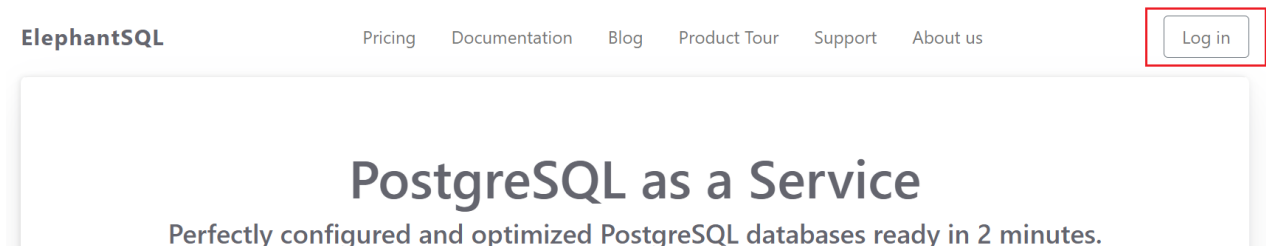
4 | Setup Database

4.1 | Setup Database on ElephantSQL

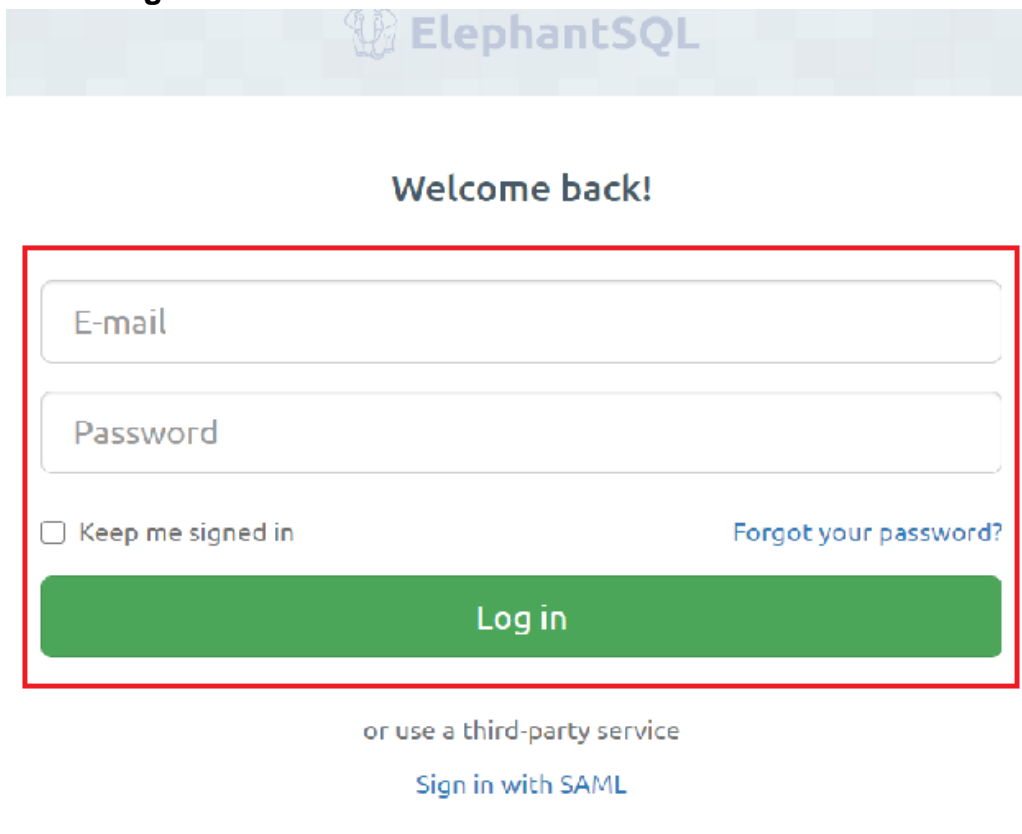
4.1.1 | Create a new instance

1. Navigate to the website: <https://www.elephantsql.com/>

2. Click on the Login button.



3. Key in your account credential to login or click the button **Sign in with GitHub/ Sign in with Google**.

The image displays the login form on the ElephantSQL website. At the top, there is a banner with the ElephantSQL logo. Below this, the text 'Welcome back!' is centered. The login form itself is a white box with a red border. It contains two input fields: 'E-mail' and 'Password'. Below these fields is a checkbox labeled 'Keep me signed in' and a link 'Forgot your password?'. A large green 'Log in' button is positioned at the bottom of the form. Below the form, the text 'or use a third-party service' is centered, followed by a link 'Sign in with SAML'.

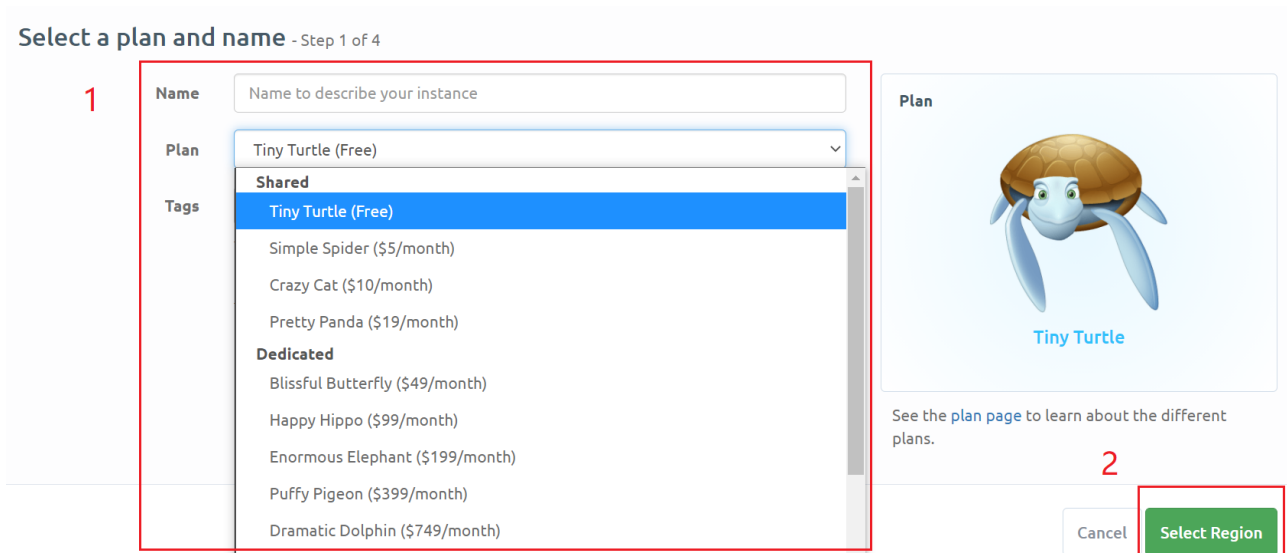
 Sign in with GitHub

 Sign in with Google

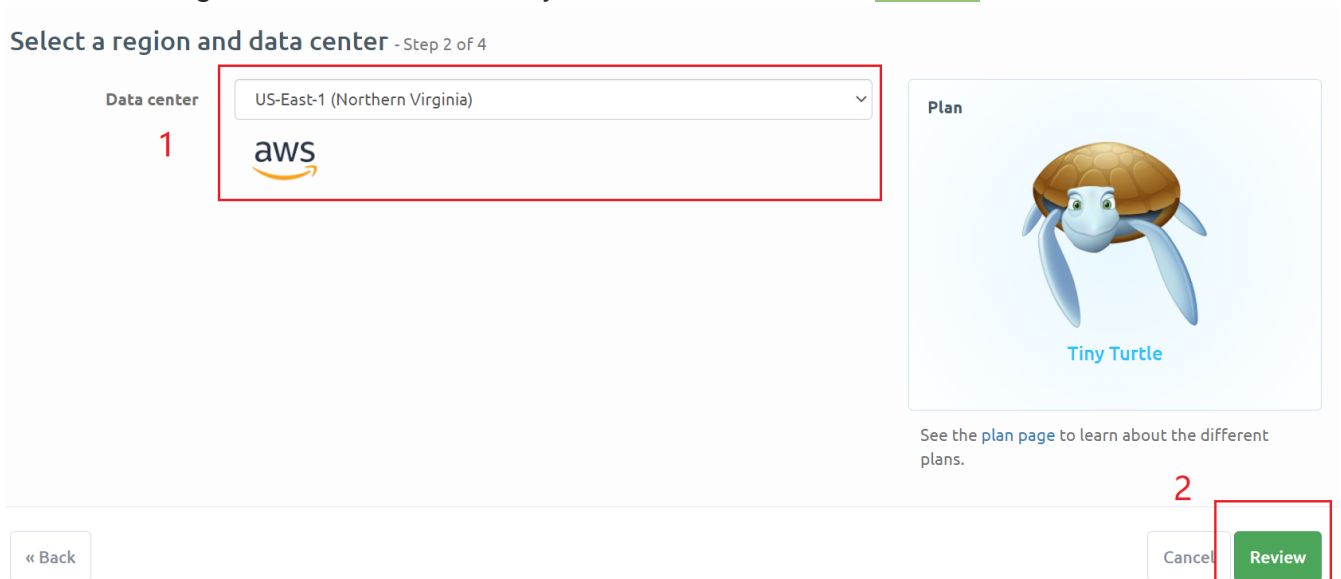
4. Click **+ Create New Instance** on top right of the page.



5. Fill in the **Name** for your instance and select a **Plan**. You can use the **Tiny Turtle** plan because it's free, but you can also choose the other plan. Then, click on the **Select Region** button.




7. Select a region and data centre for your instance and click **Review**.



8. Once all the information is filled out, click **Create instance** on the bottom right of the page.

Confirm new instance - Step 4 of 4

Plan	
	
Tiny Turtle	
Total: Free	
Name:	2
Provider:	Amazon Web Services
Region:	US-East-1 (Northern Virginia)

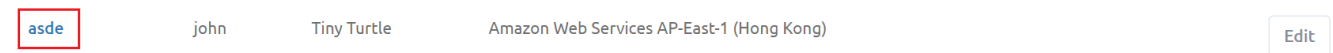
« Back

Cancel

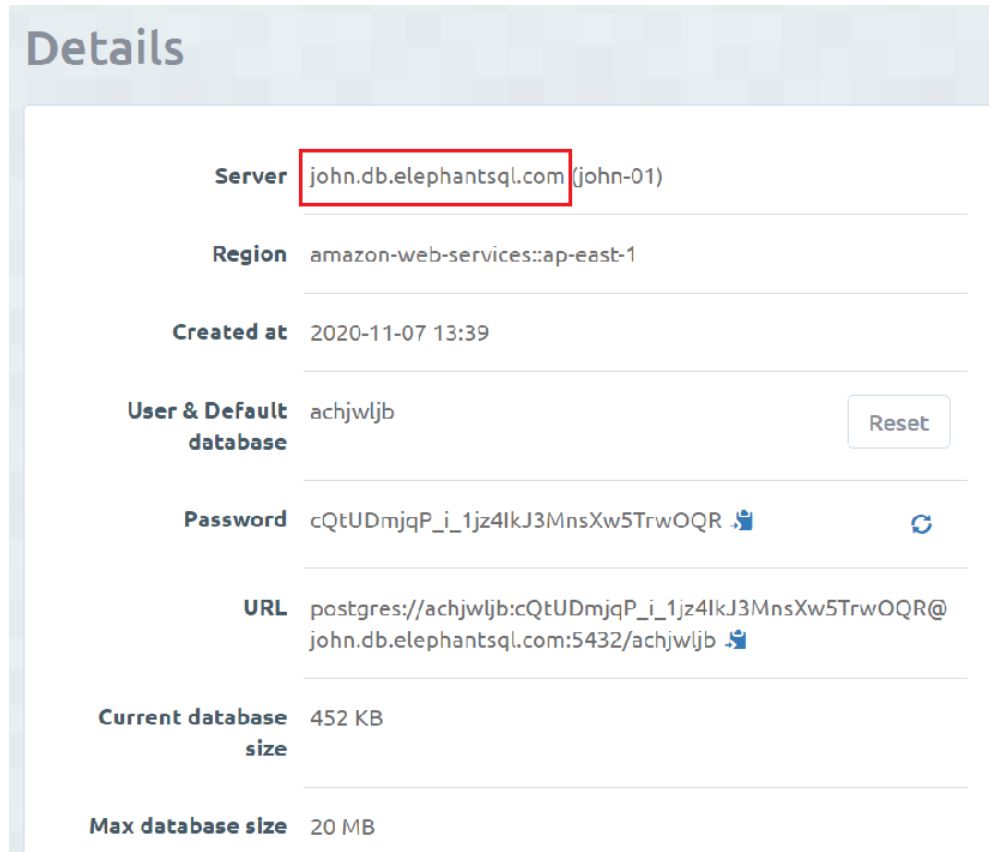
Create instance ✓

4.1.2 | Connect application to database

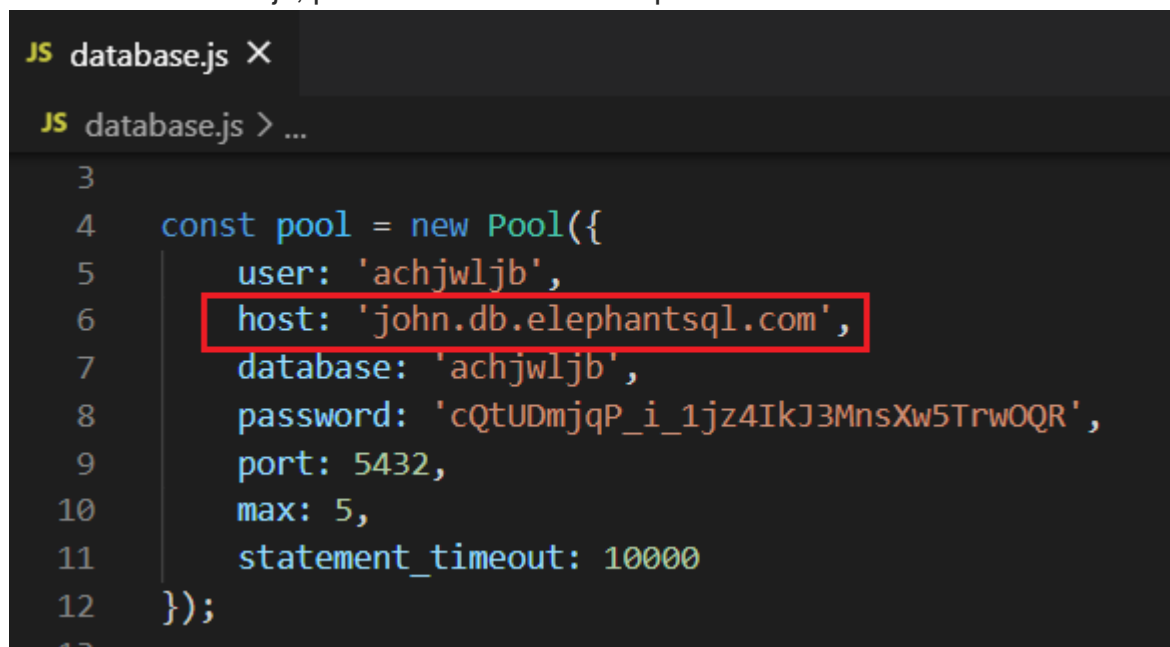
1. Click on the instance you just created to view its details. For example:



2. Copy the information on **Server**.






3. Go to database.js, paste the information copied on the **host** section.



4. Copy the **User & Default database**.

Details

Server	john.db.elephantsql.com (john-01)	
Region	amazon-web-services::ap-east-1	
Created at	2020-11-07 13:39	
User & Default database	achjwljb	<button>Reset</button>
Password	cQtUDmjqP_i_1jz4IkJ3MnsXw5TrwOQR  	
URL	postgres://achjwljb:cQtUDmjqP_i_1jz4IkJ3MnsXw5TrwOQR@john.db.elephantsql.com:5432/achjwljb 	
Current database size	452 KB	
Max database size	20 MB	

5. Go to database.js, paste the information copied on **user** and **database**.

```
JS database.js X
JS database.js > ...
3
4  const pool = new Pool({
5      user: 'achjwljb',
6      host: 'john.db.elephantsql.com',
7      database: 'achjwljb',
8      password: 'cQtUDmjqP_i_1jz4IkJ3MnsXw5TrwOQR',
9      port: 5432,
10     max: 5,
11     statement_timeout: 10000
12 });
13
```

6. Copy the **Password**. You will need to click on the little eye icon to expand the full URL.

Details

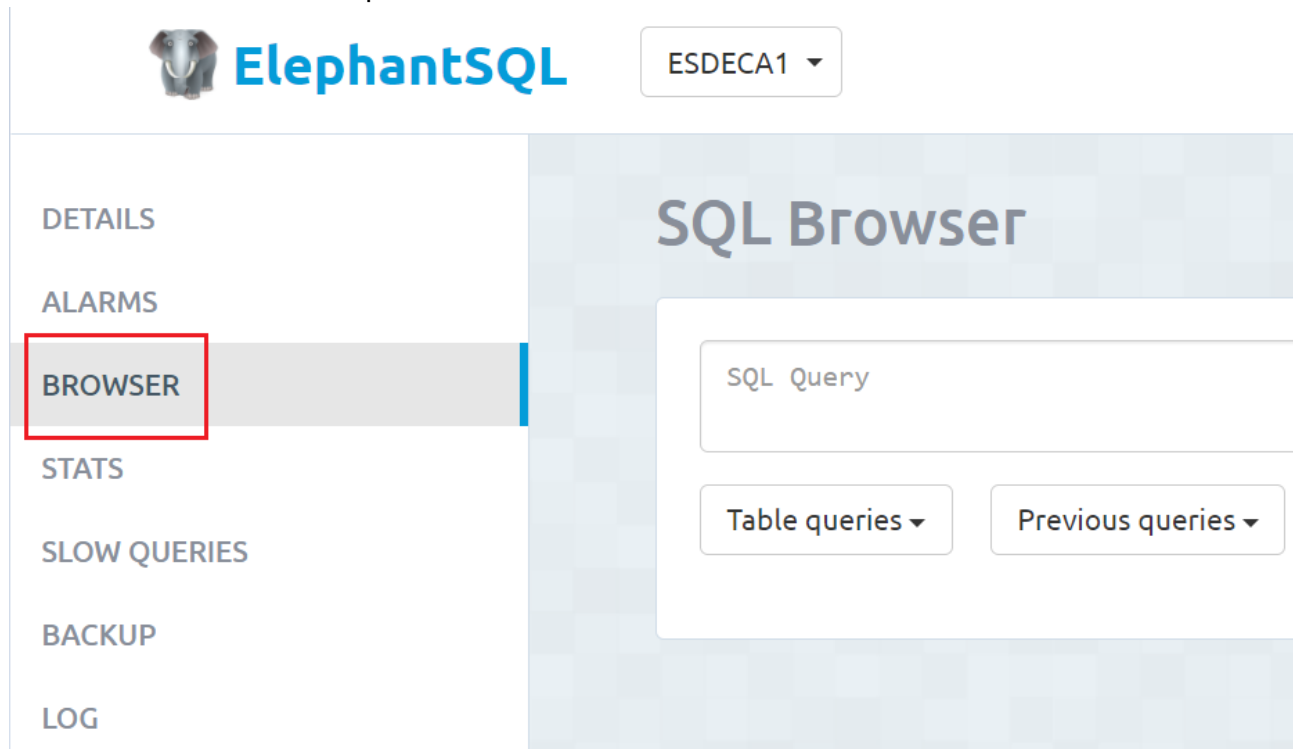
Server	john.db.elephantsql.com (john-01)		
Region	amazon-web-services::ap-east-1		
Created at	2020-11-07 13:39		
User & Default database	achjwljb	<button>Reset</button>	
Password	cQtUDmjQp_i_1jz4IkJ3MnsXw5TrwOQR		
URL	postgres://achjwljb:cQtUDmjQp_i_1jz4IkJ3MnsXw5TrwOQR@john.db.elephantsql.com:5432/achjwljb		
Current database size	452 KB		
Max database size	20 MB		

7. Go to database.js, paste the information copied on **password**.

```
JS database.js X
JS database.js > ...
3
4  const pool = new Pool({
5    user: 'achjwljb',
6    host: 'john.db.elephantsql.com',
7    database: 'achjwljb',
8    password: 'cQtUDmjQp_i_1jz4IkJ3MnsXw5TrwOQR',
9    port: 5432,
10   max: 5,
11   statement_timeout: 10000
12 });
```

4.1.3 | Create table on database created on ElephantSQL

1. Go to the browser on ElephantSQL.



2. Copy the SQL statements below.

```
CREATE TABLE Queue (  
  queue_id VARCHAR(10) PRIMARY KEY,  
  current_queue_number INT NOT NULL,  
  status BOOL NOT NULL,  
  company_id BIGINT NOT NULL  
);
```

```
CREATE TABLE CustomerQueueNumber (  
  id SERIAL PRIMARY KEY,  
  queue_number INT NOT NULL,  
  queue_id VARCHAR(10) references Queue (queue_id) NOT NULL,  
  customer_id BIGINT NOT NULL,  
  time_joined timestamp with time zone NOT NULL DEFAULT (current_timestamp(0) AT  
TIME ZONE 'UTC')  
);
```


3. Paste the SQL statements in SQL Browser and hit **Execute**

SQL Browser

achjwlyb

```
CREATE TABLE Queue (  
  queue_id VARCHAR(10) PRIMARY KEY,  
  current_queue_number INT NOT NULL,  
  status BOOL NOT NULL,  
  company_id BIGINT NOT NULL  
);  
  
CREATE TABLE CustomerQueueNumber (  
  id SERIAL PRIMARY KEY,  
  queue_number INT NOT NULL,  
  queue_id VARCHAR(10) references Queue (queue_id) NOT NULL,  
  customer_id BIGINT NOT NULL,  
  time_joined timestamp with time zone NOT NULL DEFAULT (current_timestamp(0) AT TIME ZONE 'UTC')  
);
```

Table queries ▾

Previous queries ▾

Execute ▶

4. A prompt message: Query completed indicates that you successfully launch the sql statement.

SQL Browser

achjwlyb

✓

Query completed

5. Click on **Table queries** and tables called customerqueuenumner and queue should be created.

SQL Browser

achjwlyb

```
SELECT * FROM "public"."queue" LIMIT 100
```

Table queries ▾

Previous queries ▾

Execute ▶

customerqueuenumner (public) (4 KiB, ~2 rows)
queue (public) (0 KiB, ~0 rows)

No rows returned

4.1.4 | Insert records into tables

1. Copy the sample SQL statements below. (you can create your own records and insert.)

```
INSERT INTO Queue (queue_id,current_queue_number,status,company_id)
VALUES ( 'QUEUE12341', 0, '0',1234567899);
INSERT INTO Queue (queue_id,current_queue_number,status,company_id)
VALUES ( 'QUEUE12342', 0, '0',1234567890);
INSERT INTO Queue (queue_id,current_queue_number,status,company_id)
VALUES ( 'QUEUE12343', 0, '0',1234567891);
INSERT INTO Queue (queue_id,current_queue_number,status,company_id)
VALUES ( 'QUEUE12344', 0, '0',1234567892);
INSERT INTO Queue (queue_id,current_queue_number,status,company_id)
VALUES ( 'QUEUE12345', 0, '0',1234567893);
INSERT INTO Queue (queue_id,current_queue_number,status,company_id)
VALUES ( 'QUEUE12346', 0, '0',1234567894);
INSERT INTO Queue (queue_id,current_queue_number,status,company_id)
VALUES ( 'QUEUE12347', 0, '0',1234567895);
INSERT INTO Queue (queue_id,current_queue_number,status,company_id)
VALUES ( 'QUEUE12348', 0, '0',1234567896);
```

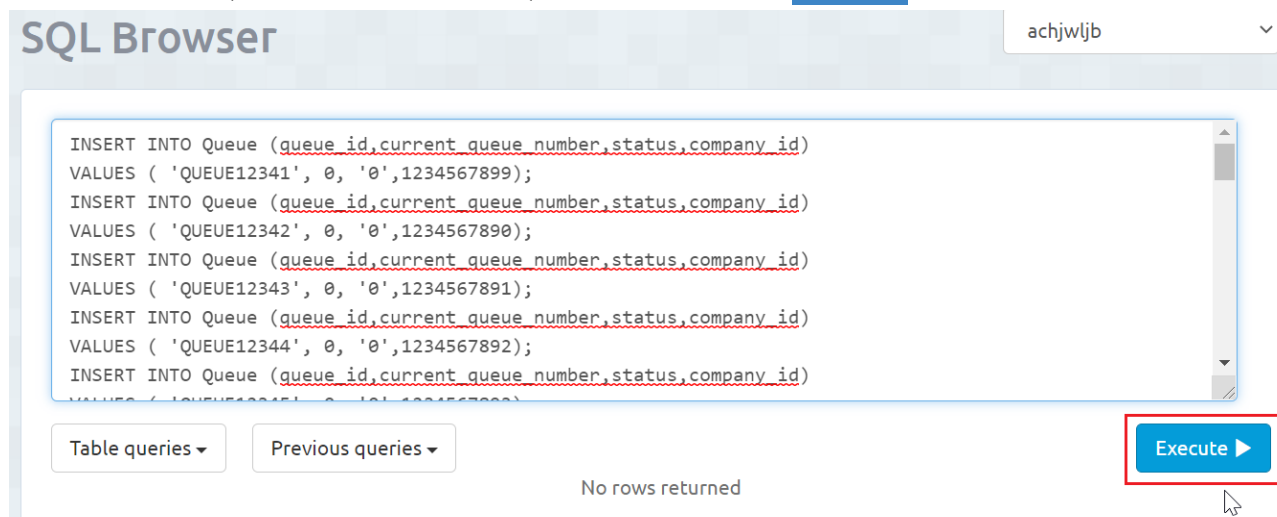
```
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 1, 'QUEUE12341', 1134567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 1, 'QUEUE12342', 1234567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 1, 'QUEUE12343', 1334567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 2, 'QUEUE12343', 1434567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES (3, 'QUEUE12343', 1534567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 2, 'QUEUE12341', 1634567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 3, 'QUEUE12341', 1734567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 2, 'QUEUE12342', 1834567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 4, 'QUEUE12343', 1934567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 5, 'QUEUE12343', 1114567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 1, 'QUEUE12344', 1124567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 1, 'QUEUE12345', 1134567899);
```

```

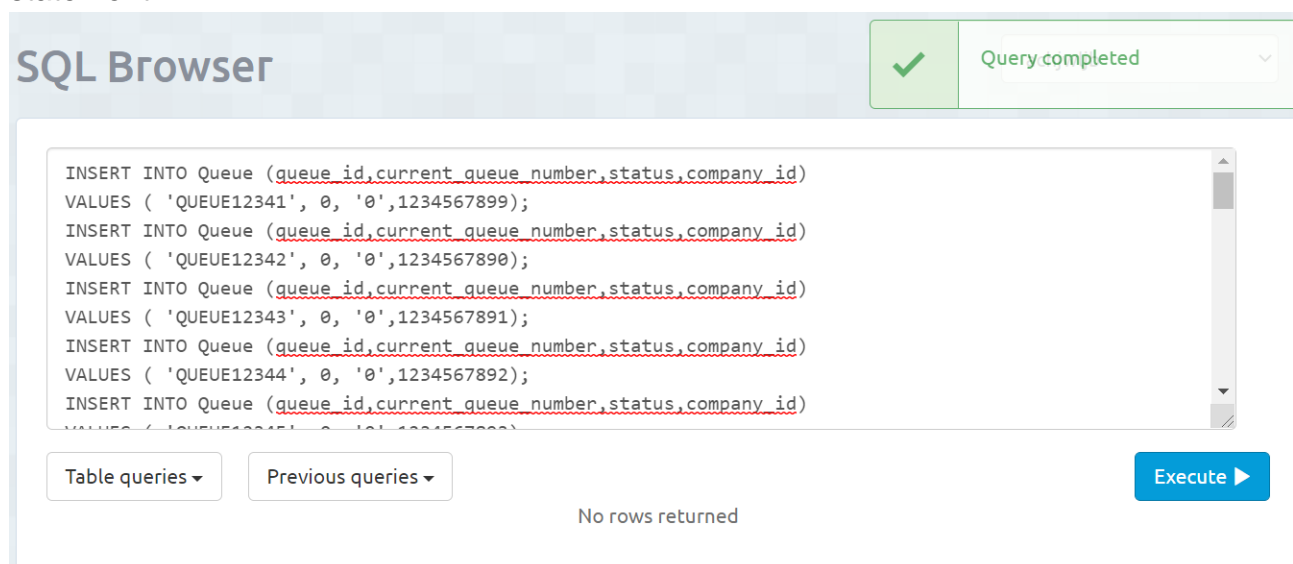
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 2, 'QUEUE12344', 1144567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 3, 'QUEUE12344', 1154567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 2, 'QUEUE12345', 1164567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 1, 'QUEUE12346', 1174567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 2, 'QUEUE12346', 1184567899);
INSERT INTO CustomerQueueNumber (queue_number,queue_id,customer_id)
VALUES ( 3, 'QUEUE12346', 1194567899);

```

2. Paste the SQL statements into SQL Browser and hit **Execute**.



3. A prompt message: Query completed indicates that you successfully launch the sql statement.



4. Click on **Table queries** and click on one of the tables created called queue. Then, hit **Execute**.

Table queries ▾

Previous queries ▾

customerqueue number (public) (4 KiB, ~2 rows)

queue (public) (0 KiB, ~0 rows)

2

Execute ▶

queue_id	current_queue_number	status	company_id
		false	1234567899

5. You should see the records created in the table.

SQL Browser

✓

Query completed

SELECT * FROM "public"."queue" LIMIT 100

Table queries ▾

Previous queries ▾

Execute ▶

queue_id	current_queue_number	status	company_id
QUEUE12341	0	false	1234567899
QUEUE12342	0	false	1234567890
QUEUE12343	0	false	1234567891
QUEUE12344	0	false	1234567892
QUEUE12345	0	false	1234567893
QUEUE12346	0	false	1234567894

5 | Start the Application

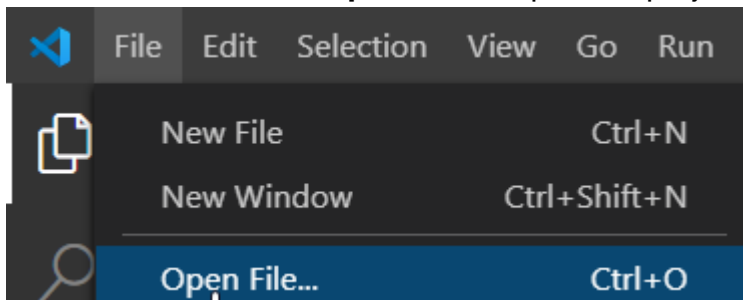
5.1 | node server.js

1. Open Visual Studio Code.

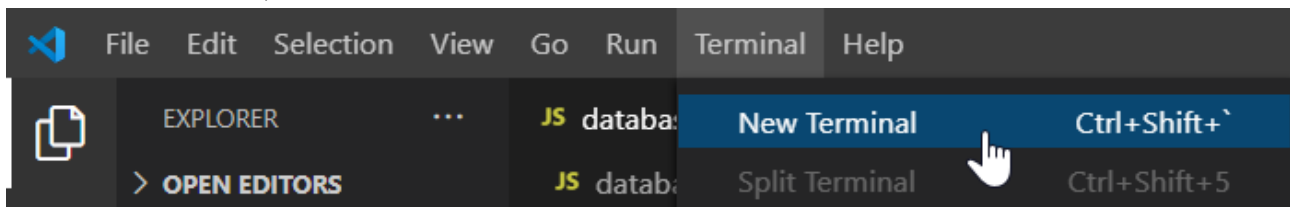


Visual Studio Code
App

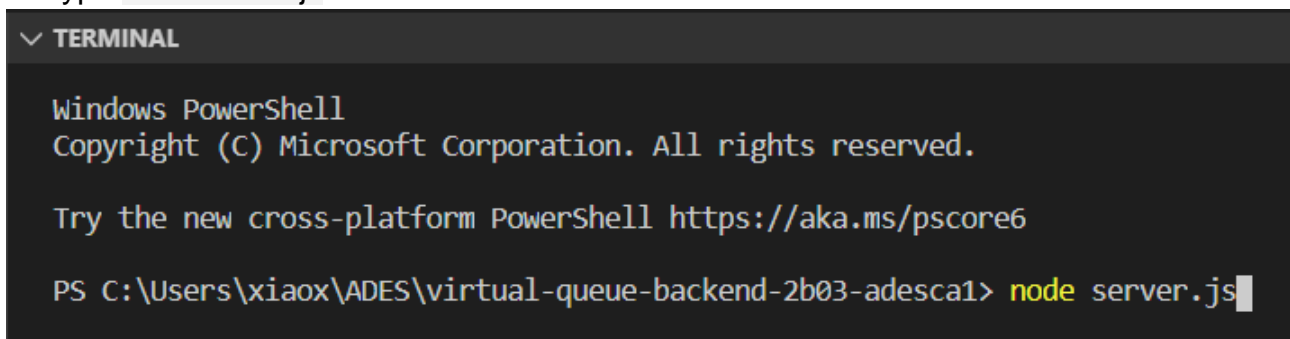
2. Under File, click on **Open File** to open the project you stored previously.



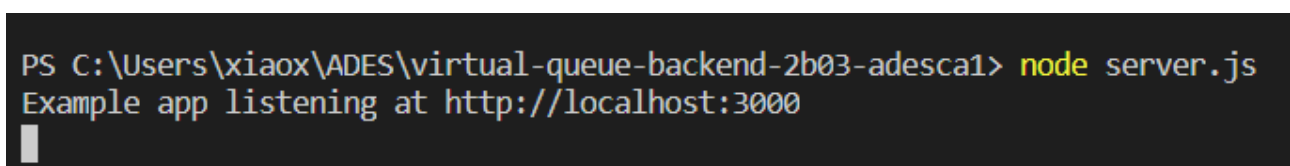
3. Under Terminal, click on **New Terminal**.



4. Type `node server.js` and hit Enter.



5. You will see a url like example below, it means you have successfully started the app.



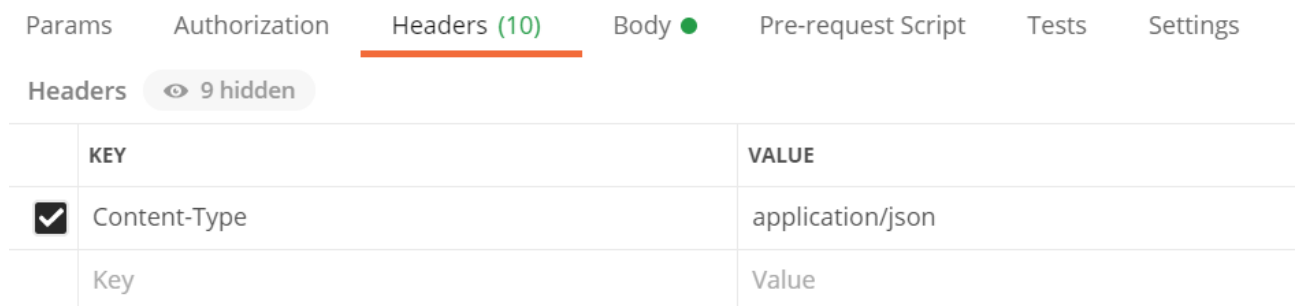
6 | Testing the Project

6.1 | Test using Postman

We will test one of the APIs which is Create Queue API. A successful request made to the server should create a new entry in the queue id with the correct parameters.

1. Check your database, tables should be created with data inserted. (Can refer to 4.1.3 & 4.1.4) Queue Id ("QUEUE33333") does not already exist in the database.

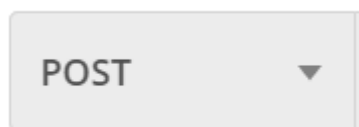
2. Go to Postman, under Headers, set Content-Type to application/json.



The screenshot shows the Postman interface with the 'Headers' tab selected. The 'Headers' tab is highlighted with an orange underline. Below the tab, there is a button that says '9 hidden'. A table with two columns, 'KEY' and 'VALUE', is visible. The first row has a checked checkbox in the 'KEY' column, the text 'Content-Type' in the 'KEY' column, and 'application/json' in the 'VALUE' column. Below this table, there are labels 'Key' and 'Value'.

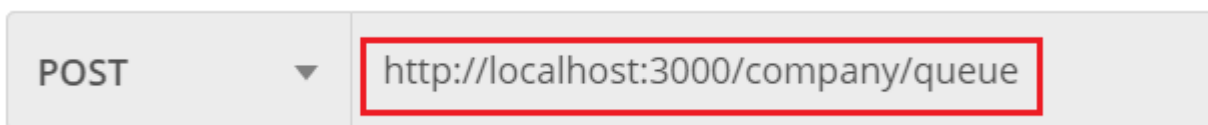
	KEY	VALUE
<input checked="" type="checkbox"/>	Content-Type	application/json
	Key	Value

3. Choose **POST** for the method.



The screenshot shows a dropdown menu with the word 'POST' selected and a downward arrow next to it.

4. Enter request url: `http://localhost:3000/company/queue`.



The screenshot shows the Postman interface with the 'POST' method selected. The URL field is highlighted with a red border and contains the text 'http://localhost:3000/company/queue'.

5. Under **Body**, key in information below:

```
{  
  "company_id": 1234567890,  
  "queue_id": "QUEUE33333"  
}
```

POST

http://localhost:3000/company/queue

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2   "company_id": 1234567890,
3   "queue_id": "QUEUE33333"
4 }
```

6. Then, Hit **Send**.

POST

http://localhost:3000/company/queue

Send

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2   "company_id": 1234567890,
3   "queue_id": "QUEUE33333"
4 }
```

7. You will receive a 201 Created response from Postman.

POST

http://localhost:3000/company/queue

Send

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2   "company_id": 1234567890,
3   "queue_id": "QUEUE33333"
4 }
```

Body

Cookies

Headers (8)

Test Results

Status: 201 Created

Time: 419 ms

Size: 365 B

Sav

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   {
3     "queue_id": "QUEUE33333",
4     "current_queue_number": 0,
5     "status": false,
6     "company_id": "1234567890"
7   }
8 }
```

8. Go to SQL Browser on ElephantSQL, Click on Table queries. Then, click on one of the tables (I will click on the table queue in this example). After that, hits **Execute**.

SQL Browser

achjwljb

SQL Query

1

Table queries

Previous queries

customerqueuenumber (public) (0 KiB, ~0 rows)

queue (public) (0 KiB, ~0 rows)

2

3

Execute

9. You should see a prompt message: Query completed indicates successfully executed the sql statement. You should also see that queue_id(QEUE33333) is created. It means the Create Queue API works. (which means you have successfully set up the app!)

SQL Browser

✓

Query completed

SELECT * FROM "public"."queue" LIMIT 100

Table queries

Previous queries

Execute

queue_id	current_queue_number	status	company_id
QUEUE12345	0	true	1234567890
QUEUE33333	0	false	1234567890