CKD-Navigation System

Backend Deployment Manual

Written by Rahul Gautham Putcha
Backend Developer & Data architect
of CKD-Navigation System 2021-2022

(Draft Edition v2022.2.10)

1. Introduction

This manual contains the necessary information for deploying a backend application for a CKD-Navigation system. The application is written in NodeJS running on a server connected to the network and is accessible to any client-facing application. This manual also presents the complete setup of the database layer side using AWS RDS that is running a MySQL database as an API server endpoint.

Three-Tier application architecture

A general note on the application structure, the CKD-Navigation system follows a 3-tier application architecture [1] as shown below,

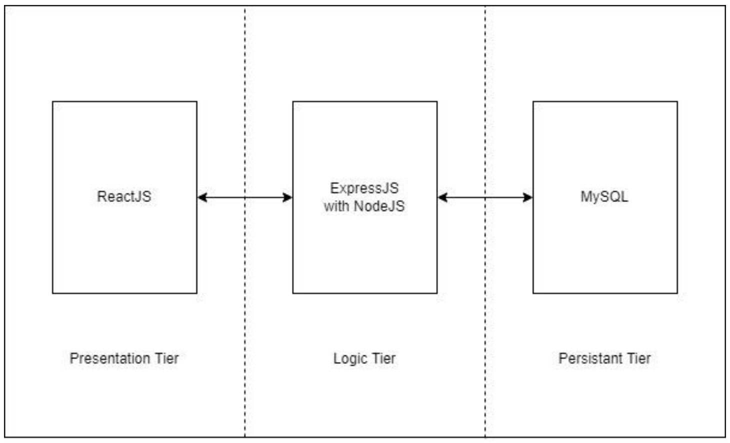


Fig 1. A simple three tier architecture for CKD-Navigation system

Above application design shows the following traits and technologies used at each tier,

Presentation Layer (Frontend Layer):

- Application built using React JS (v17.0.2 or latest)
- Static hosting on AWS S3 bucket or any server
- Connect to the Logic/ Backend Layer using Axios package

Logic Layer (API / Backend Layer):

- Application built using Express JS framework with Node JS platform
- AWS EC2 or any server running on port 3000. Requires port 3000 open to the outside world!
- CORS (Cross Origin Resource Sharing) enabled. If used outside of AWS on different platforms
- Connects the Persistence Layer running MySQL database via the Sequelize package

Persistence Layer (Storage Layer):

- MySQL database system
- AWS RDS for MySQL

Three-Tier architecture on AWS

As described in the previous section, the equivalent version of the application running on the AWS cloud platform is shown below,

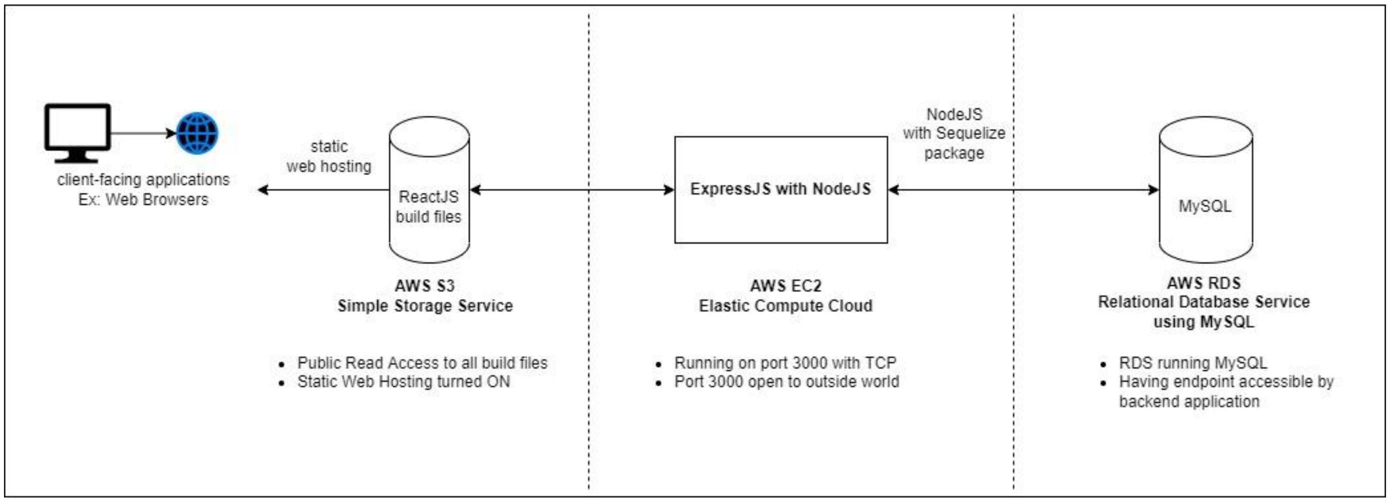

client-facing applications
Ex: Web Browsers
static
web hosting
ReactJS
build files
AWS S3
Simple Storage Service
Public Read Access to all build files
Static Web Hosting turned ON
ExpressJS with NodeJS
AWS EC2
Elastic Compute Cloud
Running on port 3000 with TCP
Port 3000 open to outside world
NodeJS
with Sequelize
package
MySQL
AWS RDS
Relational Database Service
using MySQL
RDS running MySQL
Having endpoint accessible by backend application

Fig 2. CKD-Navigation system application architecture on AWS cloud platform

In this manual we are focusing on setting up the logic and persistence part of the application tiers. Also, special focus is given to the deployment targeted on the AWS cloud platform.

2. Setting up an API Server on AWS

The API server is a NodeJS application that runs on port 3000. This port is set as a default and is configurable within the application's .env file, more on this in a later section. For setting up an API Server on AWS, the first thing one should do is to procure a compute instance / machine. On AWS, this can be done via an EC2 (Elastic Compute Cloud) instance.

Setting up an EC2 instance

Setting up a server on AWS requires a valid AWS account and with authorization access to create an EC2 instance. Below are the steps to create EC2 instances. Note that the UI may update frequently based on updates from AWS, but the overall process on EC2's configuration aspect would remain the same.

Step 1: Search for EC2 using the Search bar

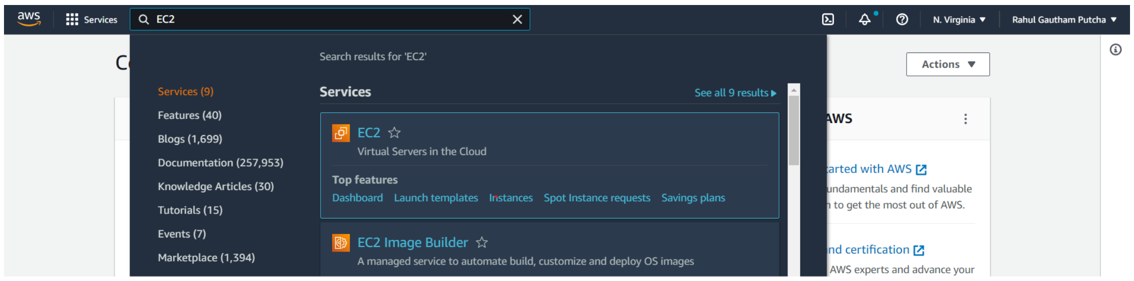


Fig 3. Searching for EC2 on AWS

Step 2: Choose a Region where you want to launch your AWS EC2 instance

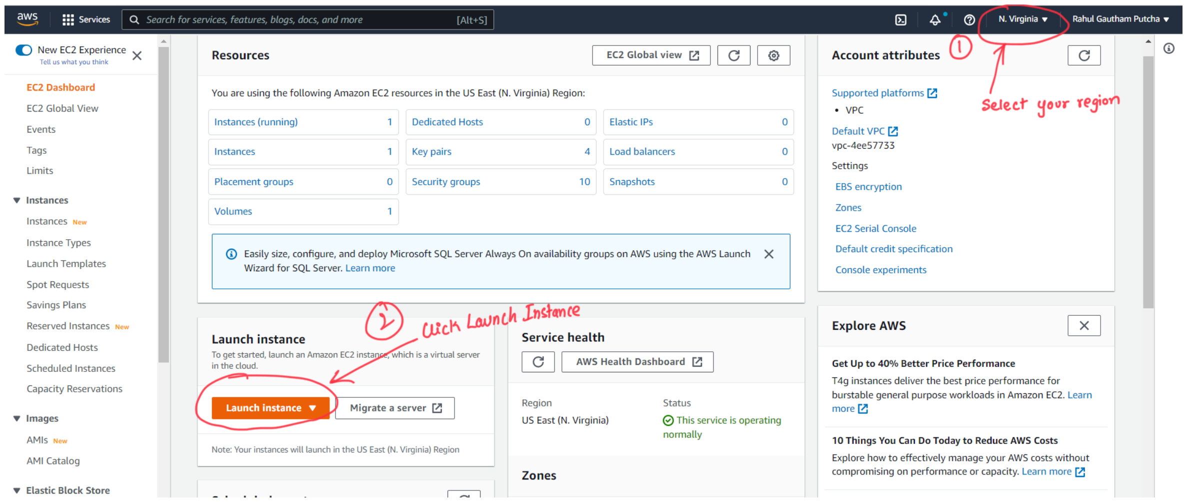
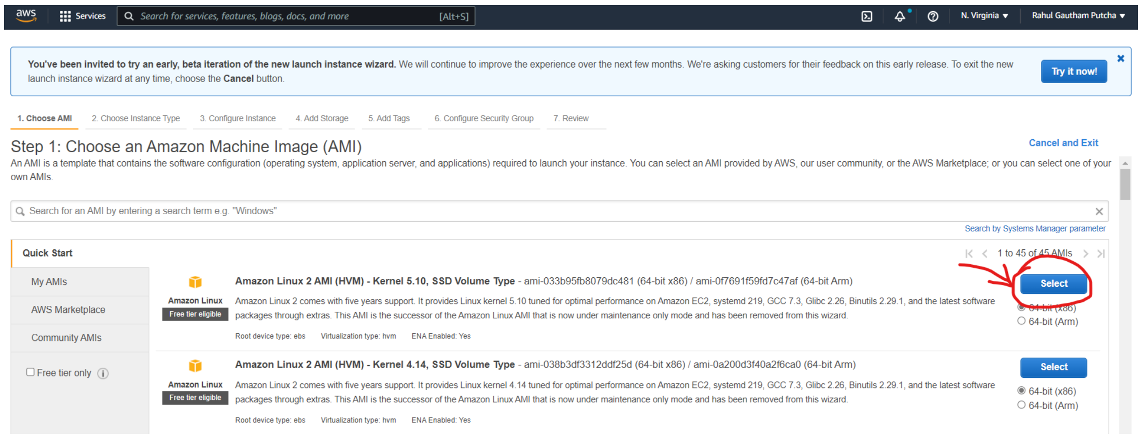


Fig 4. Requesting a EC2 instance on AWS

Step 3: Choose AMI to **Amazon Linux 2 AMI (HVM), SSD Volume Type**

Note: We have used Kernel 5.10 which is the latest version available at time of writing the manual. In future any latest version is satisfiable for selection. You can also select the available Ubuntu Server LTS.



You've been invited to try an early, beta iteration of the new launch instance wizard. We will continue to improve the experience over the next few months. We're asking customers for their feedback on this early release. To exit the new launch instance wizard at any time, choose the **Cancel** button.

1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Add Tags | 6. Configure Security Group | 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type - ami-033b95fb8079dc481 (64-bit x86) / ami-0f7691f59fd7c47af (64-bit Arm)

Amazon Linux — Free tier eligible

Amazon Linux 2 comes with five years support. It provides Linux kernel 5.10 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

[✓ 64-bit (x86)] 64-bit (Arm)

Amazon Linux 2 AMI (HVM) - Kernel 4.14, SSD Volume Type - ami-038b3df3312ddf25d (64-bit x86) / ami-0a200d3f40a2f6ca0 (64-bit Arm)

Amazon Linux — Free tier eligible

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.

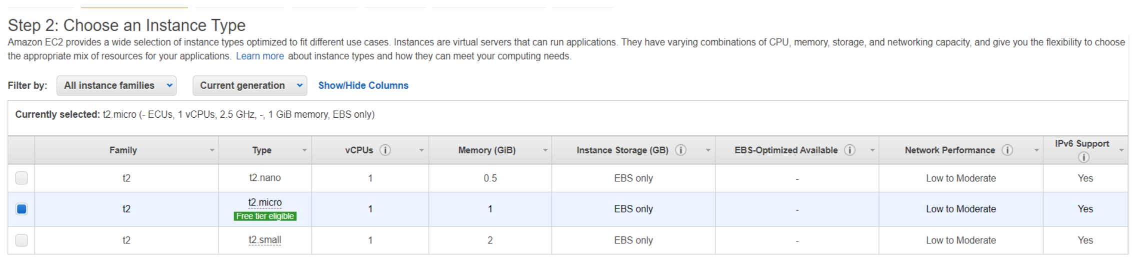
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

[✓ 64-bit (x86)] 64-bit (Arm)

Fig 5. Choosing a Operating System or AMI for EC2

Click on **Next**.

Step 4: Choose the instance type to **t2.micro** or higher depending on the need.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance families | Current generation

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes

Fig 6. Choosing and instance type

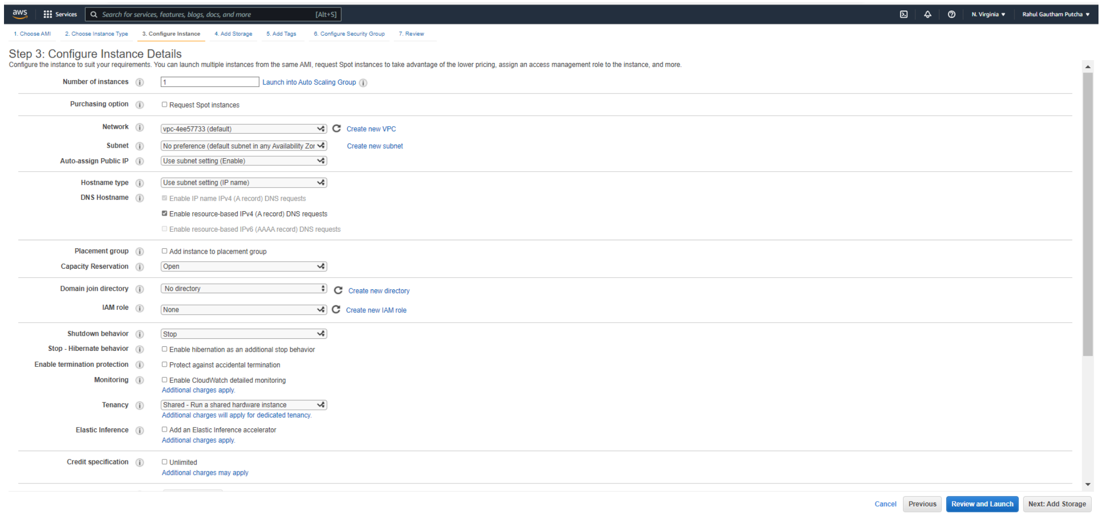
Click on **Next**.

Step 5: Configure Instance Details

We have set the instance details to default.

As part of the instance details for an advanced setting, one can configure:

- The Number of instances (connected to a Elastic Load Balancer within an Auto Scaling Group)
- Choosing the VPC and Subnet/Availability Zone for grouping instances
- Shutdown behavior
  - *Stop* will keep the instance but changes its IP address on relaunch
  - *Terminate* disposes the entire instance
- Enable termination protection to protect instance against accidental termination
- Monitoring to get analytics from CloudWatch



Step 3: Configure Instance Details

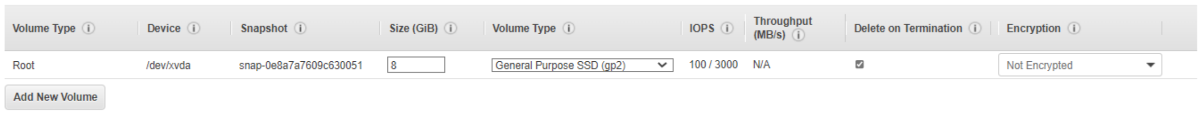
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1 · Launch into Auto Scaling Group
Purchasing option	☐ Request Spot instances
Network	vpc-4ee57733 (default) · Create new VPC
Subnet	No preference (default subnet in any Availability Zor · Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)
Hostname type	Use subnet setting (IP name)
DNS Hostname	☑ Enable IP name IPv4 (A record) DNS requests ☑ Enable resource-based IPv4 (A record) DNS requests ☐ Enable resource-based IPv6 (AAAA record) DNS requests
Placement group	☐ Add instance to placement group
Capacity Reservation	Open
Domain join directory	No directory · Create new directory
IAM role	None · Create new IAM role
Shutdown behavior	Stop
Stop - Hibernate behavior	☐ Enable hibernation as an additional stop behavior
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.
Tenancy	Shared - Run a shared hardware instance Additional charges will apply for dedicated tenancy.
Elastic Inference	☐ Add an Elastic Inference accelerator Additional charges apply.
Credit specification	☐ Unlimited Additional charges may apply

Fig 7. Configuring the instance details

After obtaining the required configuration in instance details, click on **Next.**

Step 6: Add a Storage (Set to default)



Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-0e8a7a7609c630051	8	General Purpose SSD (gp2)	100 / 3000	N/A	☑	Not Encrypted

Fig 8. Adding a Storage

Click on **Next**.

Step 7: Add Tags

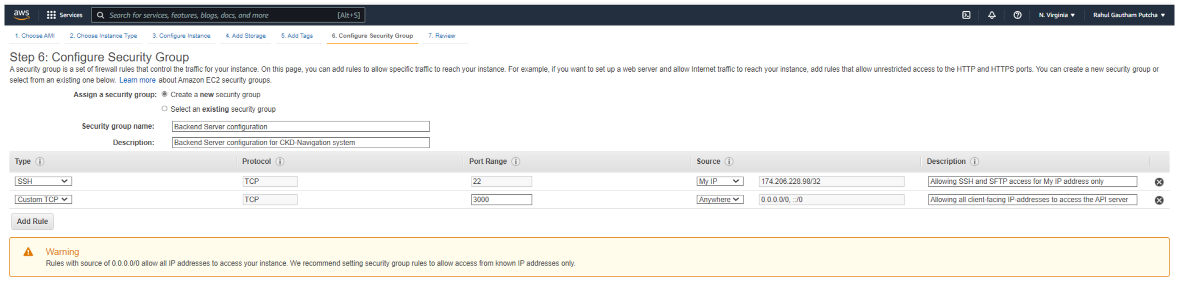
Tags are key value pairs that can hold details specific to EC2 instances. This step is optional.

Click **Next**.

Step 8: Configure Security Group

This is an important step in configuring EC2 instances. This step describes on:

- How is the EC2 server accessed by our computer?
- Allowing EC2 server on TCP port 3000.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

Security group name: Backend Server configuration

Description: Backend Server configuration for CKD-Navigation system

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	My IP	174.206.228.98/32	Allowing SSH and SFTP access for My IP address only
Custom TCP	TCP	3000	Anywhere	0.0.0.0/0, ::/0	Allowing all client-facing IP-addresses to access the API server

Warning

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Fig 9. Security Group - Securing the server access

Click on **Review and Launch.**

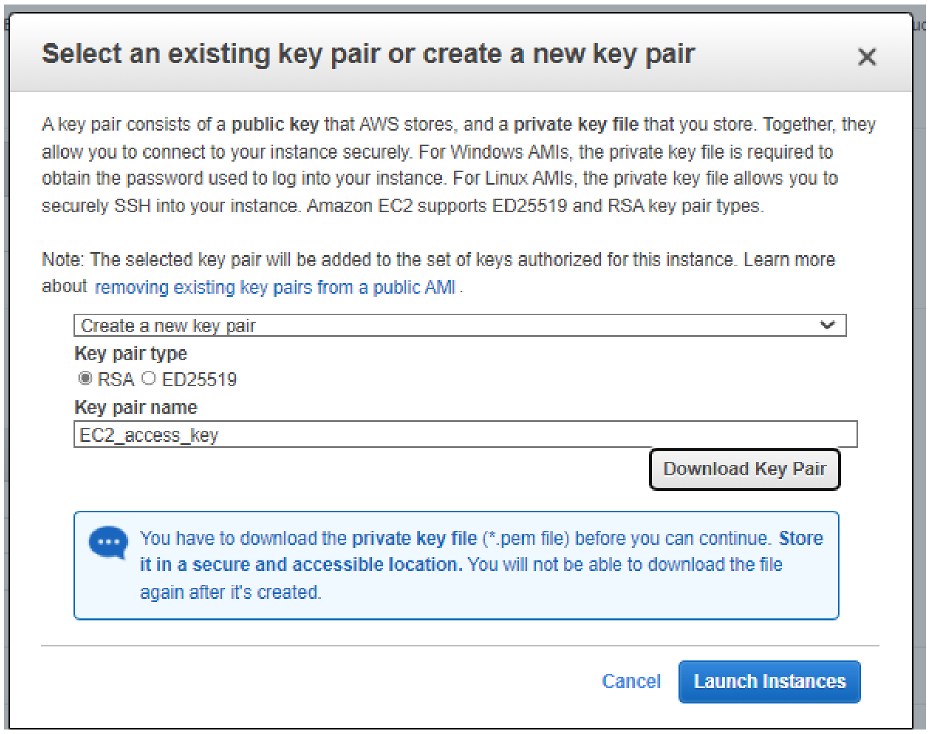
Step 9: Click on Launch after reviewing the changes.

You will see a dialog box open in front of you for choosing a key-pair.

Key Pairs are a secure way to access EC2. They are similar to passwords / secrets for accessing computers. If this is your first time then follow below steps,

- From the dropdown select *Create a new key pair* instead of *Choose an existing key pair*
- Set the *Key pair* type to *RSA*
- Enter any *Key pair name*, and Click on **Download Key Pair.** You will have a .pem file downloaded.
- You will see the **Launch Instance** button enabled to click. Click on **Launch Instance** button

Note: Keep the downloaded .pem safely for now. We will be using it in the later sections of this manual, for the sole purpose of accessing our EC2 instance from Command Prompt in Windows or any terminal of choice.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI .

Create a new key pair

Key pair type

[✓ RSA] ED25519

Key pair name

EC2_access_key

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

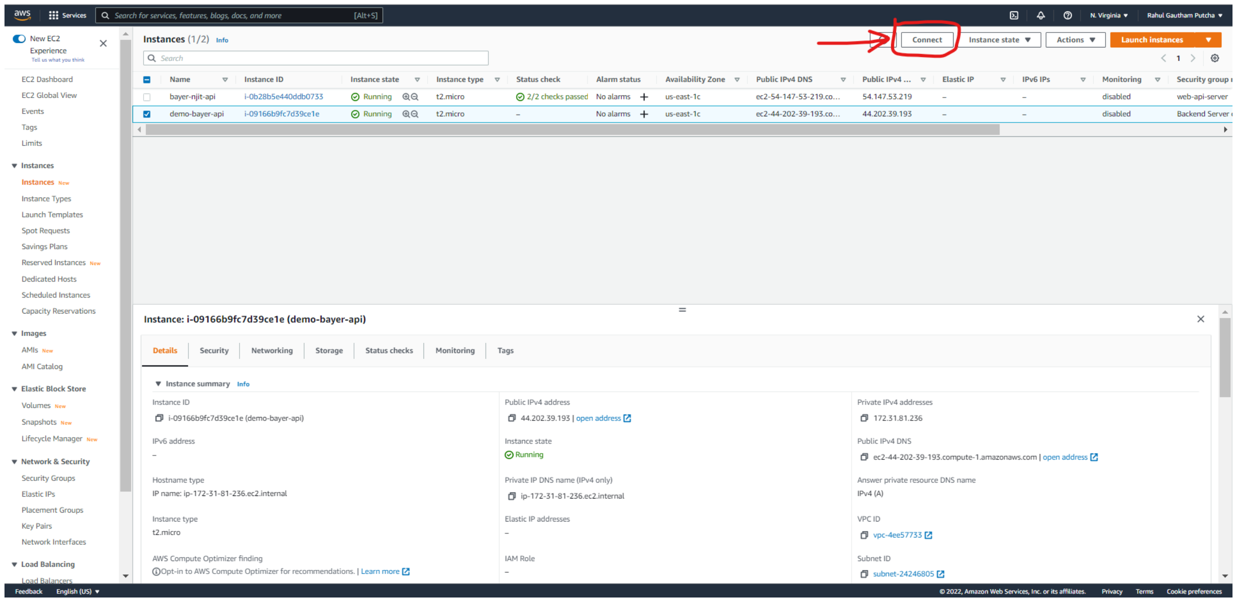
Fig 10. EC2 Key Pair and Launching EC2 instance

Connecting to the EC2 instance

In the previous section we have launched an EC2 server instance and downloaded a .pem file in order to access our EC2 server.

For Linux/Max users

On a general note for Linux/Mac users, connecting to an EC2 server is straightforward using a terminal. The instruction for accessing the EC2 on Linux/Mac can be found by choosing the instance to connect and then clicking the **Connect** button on the **EC2 instance page.**



Instances (1/2)

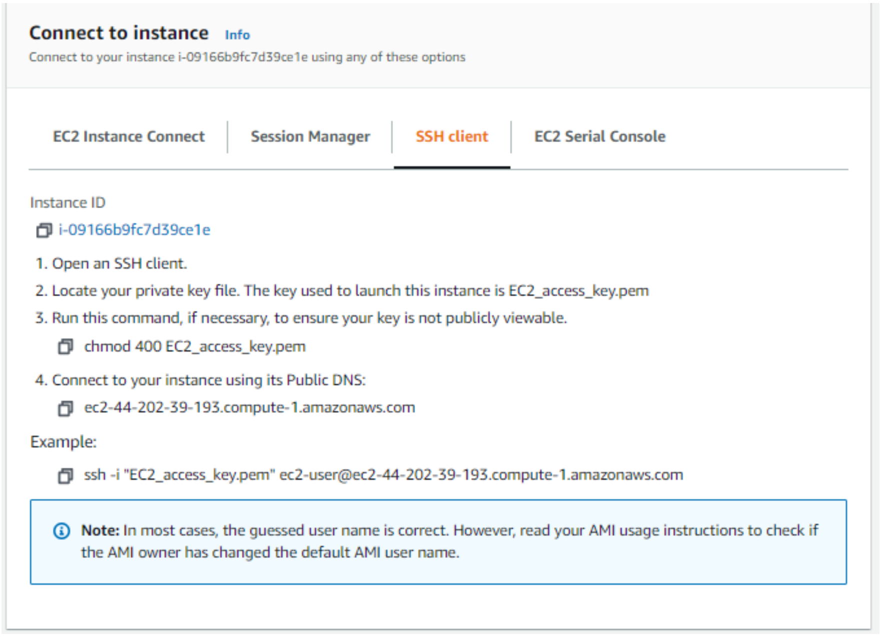
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security group
bayer-njit-api	i-0b28b5e440ddb0733	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-54-147-53-219.co...	54.147.53.219	–	–	disabled	web-api-server
demo-bayer-api	i-09166b9fc7d39ce1e	Running	t2.micro	–	No alarms	us-east-1c	ec2-44-202-39-193.co...	44.202.39.193	–	–	disabled	Backend Server

Instance: i-09166b9fc7d39ce1e (demo-bayer-api)

Instance summary

Instance ID: i-09166b9fc7d39ce1e (demo-bayer-api)
Public IPv4 address: 44.202.39.193 | open address
Private IPv4 addresses: 172.31.81.236
IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-44-202-39-193.compute-1.amazonaws.com | open address
Hostname type: IP name: ip-172-31-81-236.ec2.internal
Private IP DNS name (IPv4 only): ip-172-31-81-236.ec2.internal
Answer private resource DNS name: IPv4 (A)
Instance type: t2.micro
Elastic IP addresses: –
VPC ID: vpc-4ee57733
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role: –
Subnet ID: subnet-24246805

Fig 11. Connecting to the EC2 instance



Connect to instance

Connect to your instance i-09166b9fc7d39ce1e using any of these options

Instance ID
i-09166b9fc7d39ce1e

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is EC2_access_key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 EC2_access_key.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-44-202-39-193.compute-1.amazonaws.com
```

Example:

```
ssh -i "EC2_access_key.pem" ec2-user@ec2-44-202-39-193.compute-1.amazonaws.com
```

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Fig 12. Connecting to the EC2 instance (for Linux/Mac users)

(Note: Above picture may not resemble the details of your actual instance launched, but is meant for the demonstration purpose only.)

PuTTYgen for accessing using Command prompt (For Windows only)

For the PuTTYgen tool we must first install PuTTY for Windows, an open source tool for accessing remote servers. Installing PuTTY will automatically install the PuTTYgen tool. The main reason for using the PuTTYgen tool is for converting our .pem file to a privately accessible key on Windows.

Next we require the .pem file we have downloaded previously during the EC2 server setup. Keep a note on the location of the .pem file and open the PuTTYgen application on Windows.

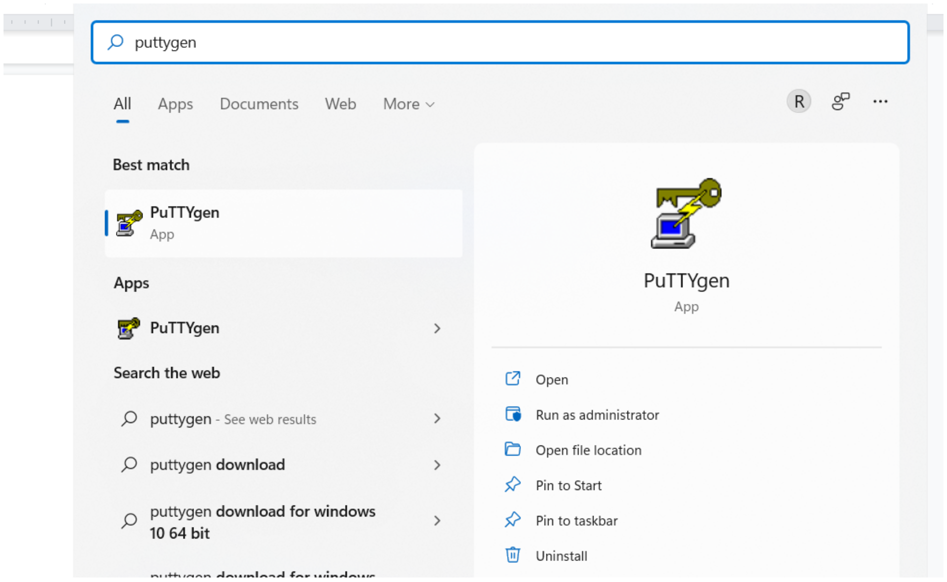
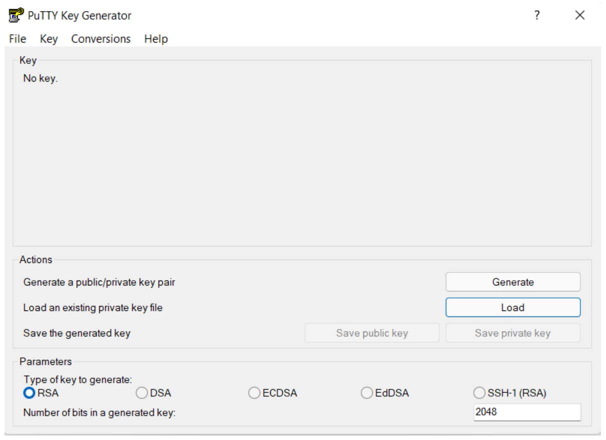
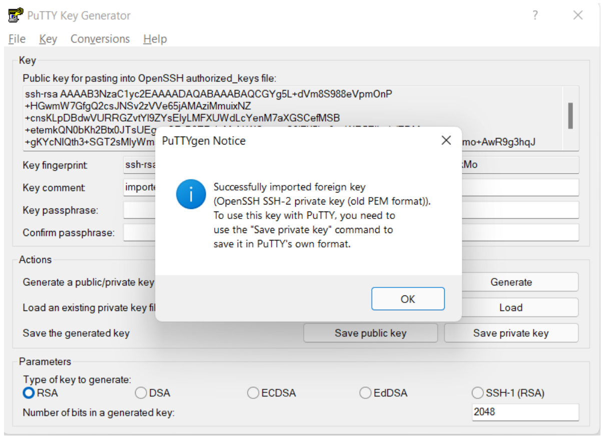


Fig 13. Search for PuTTYgen

Click on the **Load** button. A dialog box will open for selecting the pem file. In the dialog box set the file type to **All File (*.*)** type and select your downloaded Key Pair .pem file.





```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCGYg5L+dVm8S988eVpmOnP
+HGwmW7GfgQ2csJNSv2zVVe65jAMAziMmuixNZ
+cnsKLpDBdwVURRGZvtYl9ZYsElyLMFXUWdLcYenM7aXGSCefMSB
+etemkQN0bKh2Btx0JTsUEg
+gKYcNlQth3+SGT2sMlyWm                                mo+AwR9g3hqJ
```

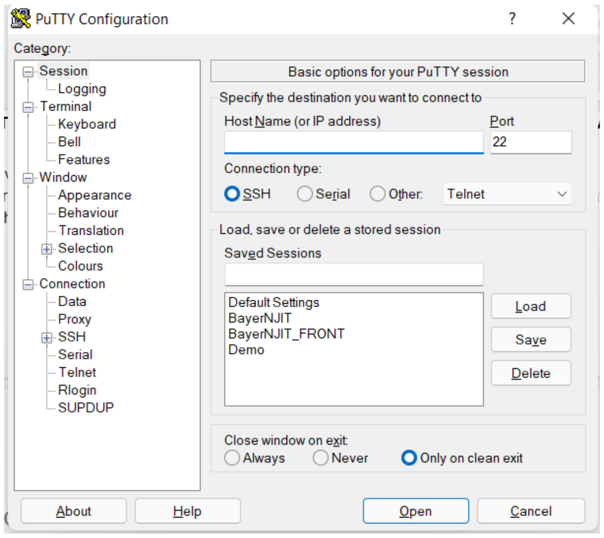
Successfully imported foreign key (OpenSSH SSH-2 private key (old PEM format)). To use this key with PuTTY, you need to use the "Save private key" command to save it in PuTTY's own format.

Fig 14. PuTTYgen Application tool **Fig 15. Converting .pem to .ppk file (Windows)**

Click on the **Save Private key** button and keep this file in a directory with all other files belonging to the CKD-Navigation project. We will be using this .ppk file as a secret file to access our EC2 server rather than the .pem file which we have originally downloaded earlier.

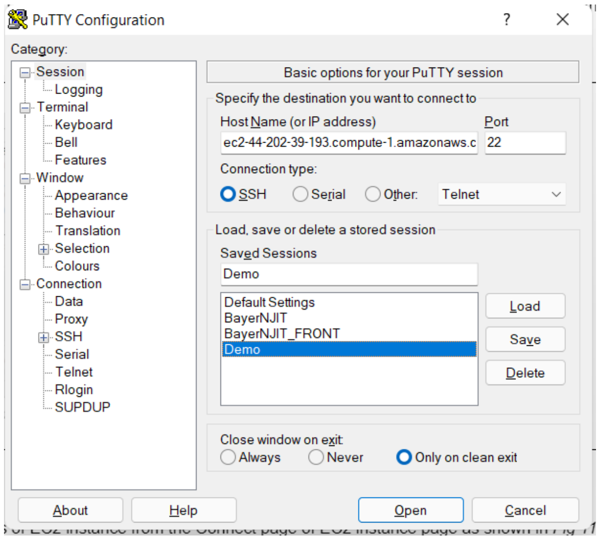
Accessing our EC2 instance (For Windows only)

After generating the private key file from the .pem file the next step is to connect to the instance. For this we use PuTTY tool. Open the **PuTTY** application.



Specify the destination you want to connect to
Host Name (or IP address):
Port: 22
Connection type: [✓ SSH] Serial Other: Telnet
Load, save or delete a stored session
Saved Sessions:
Default Settings
BayerNJIT
BayerNJIT_FRONT
Demo
Close window on exit: Always Never [✓ Only on clean exit]

Fig 16. PuTTY Application



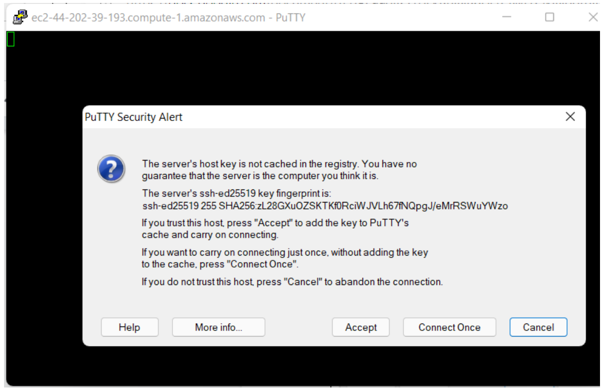
Specify the destination you want to connect to
Host Name (or IP address): ec2-44-202-39-193.compute-1.amazonaws.c
Port: 22
Connection type: [✓ SSH] Serial Other: Telnet
Load, save or delete a stored session
Saved Sessions: Demo
Default Settings
BayerNJIT
BayerNJIT_FRONT
[✓ Demo]
Close window on exit: Always Never [✓ Only on clean exit]

Fig 17. Setting EC2 Host address

Copy the IP address of EC2 instance from the Connect page of EC2 instance page, shown in *Fig 11* and *Fig 12* above. Inside of the PuTTY application, paste the IP address inside the **Host Name (or IP address) field.**

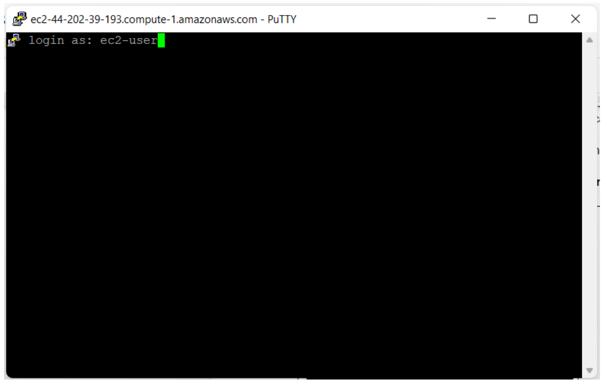
Next click on **SSH** > **Auth** from the **Category** section shown on the left part of the PuTTY application. In the *Auth* section, Click on **Browse** button that's part of the *Private key file for authentication* field and select the .ppk file, previously generated from PuTTYgen application, using the file dialog box.

Click on the **Open** button located on the bottom part of the application. This will open a terminal window. Click on either **Accept** or **Connect Once** button to proceed.



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's ssh-ed25519 key fingerprint is:
ssh-ed25519 255 SHA256:zL28GXuOZSKTKf0RciWJVLh67fNQpgJ/eMrRSWuYWzo
If you trust this host, press "Accept" to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, press "Connect Once".
If you do not trust this host, press "Cancel" to abandon the connection.

Fig 18. PuTTY terminal to your EC2 server

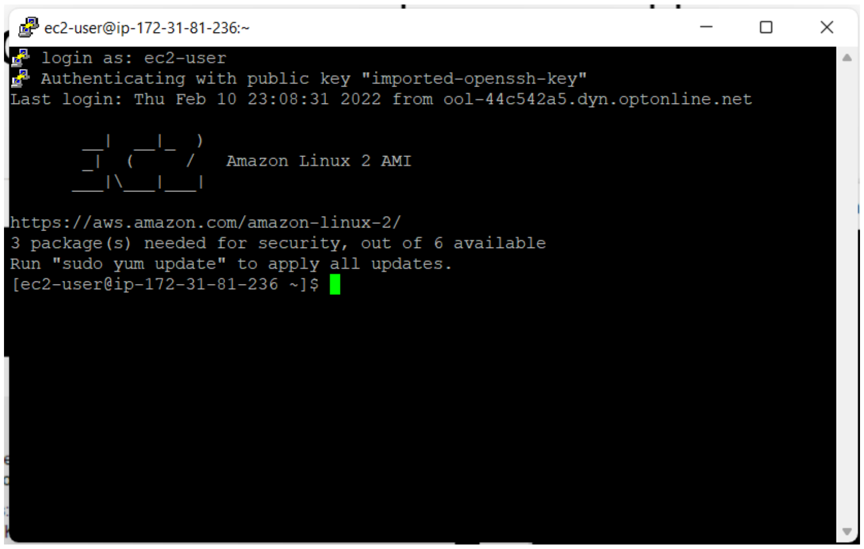


```
login as: ec2-user
```

Fig 19. EC2 user login

Enter the EC2 login user as **ec2-user** as shown above. You will be logged in on the terminal.

If you see the interface shown in *Fig 20* at your side, then Congratulations! you have successfully set up and you are now able to access the EC2 server on your machine.



```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Thu Feb 10 23:08:31 2022 from ool-44c542a5.dyn.optonline.net

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
3 package(s) needed for security, out of 6 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-81-236 ~]$
```

Fig 20. Accessing EC2 instance

Note: If you are facing any error during the setup process using PuTTY, try checking the Security Groups. Most of the problems arise due to unauthorized connection either by IP address mismatching or accessing wrong ports upon requests.

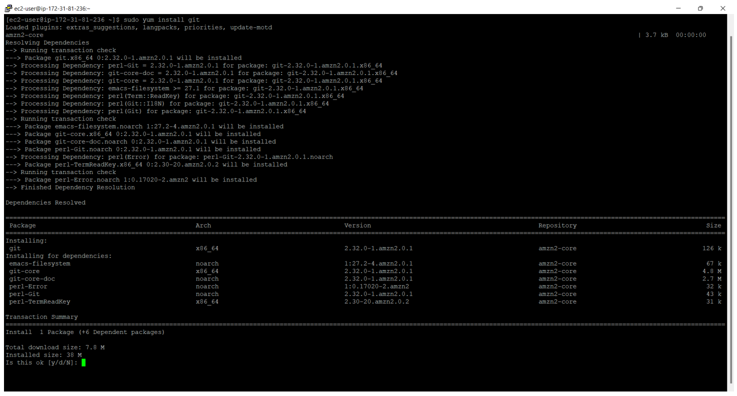
Transferring Backend Application Project files onto EC2 server

Uploading the project file to the EC2 server is a two step process,

1. Installing the git package on Amazon Linux 2 EC2 instance
2. Downloading the project files from the git repository

Installing the git package on Amazon Linux 2 EC2 instance

Amazon Linux 2 is a Red Hat Enterprise Linux (RHEL) instance which gets its external package available as a yum repository. One can also search for its counterpart Debian distribution such as Ubuntu from the apt repositories.



```
[ec2-user@ip-172-31-81-236 ~]$ sudo yum install git
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                                                  | 3.7 kB  00:00:00
Resolving Dependencies
--> Running transaction check
---> Package git.x86_64 0:2.32.0-1.amzn2.0.1 will be installed
--> Processing Dependency: perl-Git = 2.32.0-1.amzn2.0.1 for package: git-2.32.0-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core-doc = 2.32.0-1.amzn2.0.1 for package: git-2.32.0-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core = 2.32.0-1.amzn2.0.1 for package: git-2.32.0-1.amzn2.0.1.x86_64
--> Processing Dependency: emacs-filesystem >= 27.1 for package: git-2.32.0-1.amzn2.0.1.x86_64
--> Processing Dependency: perl(Term::ReadKey) for package: git-2.32.0-1.amzn2.0.1.x86_64
--> Processing Dependency: perl(Git::I18N) for package: git-2.32.0-1.amzn2.0.1.x86_64
--> Processing Dependency: perl(Git) for package: git-2.32.0-1.amzn2.0.1.x86_64
--> Running transaction check
---> Package emacs-filesystem.noarch 1:27.2-4.amzn2.0.1 will be installed
---> Package git-core.x86_64 0:2.32.0-1.amzn2.0.1 will be installed
---> Package git-core-doc.noarch 0:2.32.0-1.amzn2.0.1 will be installed
---> Package perl-Git.noarch 0:2.32.0-1.amzn2.0.1 will be installed
--> Processing Dependency: perl(Error) for package: perl-Git-2.32.0-1.amzn2.0.1.noarch
---> Package perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2 will be installed
--> Running transaction check
---> Package perl-Error.noarch 1:0.17020-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================
 Package                   Arch                Version                      Repository                    Size
================================================================================================================
Installing:
 git                       x86_64              2.32.0-1.amzn2.0.1           amzn2-core                   126 k
Installing for dependencies:
 emacs-filesystem          noarch              1:27.2-4.amzn2.0.1           amzn2-core                    67 k
 git-core                  x86_64              2.32.0-1.amzn2.0.1           amzn2-core                   4.8 M
 git-core-doc              noarch              2.32.0-1.amzn2.0.1           amzn2-core                   2.7 M
 perl-Error                noarch              1:0.17020-2.amzn2            amzn2-core                    32 k
 perl-Git                  noarch              2.32.0-1.amzn2.0.1           amzn2-core                    43 k
 perl-TermReadKey          x86_64              2.30-20.amzn2.0.2            amzn2-core                    31 k

Transaction Summary
================================================================================================================
Install  1 Package (+6 Dependent packages)

Total download size: 7.8 M
Installed size: 38 M
Is this ok [y/d/N]:
```

Fig 21. Installing packages using sudo yum

For installing git for RHEL OS based server/instances, we can type in the following command,

```
$ sudo yum install git
```

For Ubuntu OS based server/instances,

```
$ sudo apt-get install git
```

After installing we proceed to the next step of downloading project files from the git repository.

Downloading the project files from the git repository

The project files for the CKD-Navigation backend application are available at bayer-njit-backend. More details on the internal structure of the backend application is presented in the later sections of this manual.

In order to download files from this repository, you can clone the git repository directly from GitHub repo as shown below,

```
$ git clone https://github.com/RPG-coder/bayer-njit-backend.git
$ cd bayer-njit-backend # Open downloaded repository with this command
```

Now all that's left is to set up the application environment within the EC2 server in order to run the backend server using NodeJS, without any flaws.

Setting up the Application Environment

The application environment we are using for our backend application is in **NodeJS**. NodeJS provides us with an easy way to deploy a cross platform application. Due to its ability to have cross-platform compatibility and package management support, every dependency package is managed and stored within the package.json file once included in the project, without any need for us to further hunt any of these dependencies packages by ourselves. The procurement of node packages is done using the **Node Package Manager (npm)** tool that even has its own repository and also provides support from the open source developer community.

Following are the requirements we need to satisfy in order to run the NodeJS application.

1. Installing NodeJS and NPM
2. Downloading package dependencies for the backend application
3. Running the server (Testing only)
4. Running the server (on Production mode)

Installing NodeJS and NPM

Node JS and NPM can be installed using following

```
$ sudo yum update
```

For this project we have use NodeJS v17.0.5 version with npm v8.4.1 which can be fetched by below command,

```
$ curl -sL
https://raw.githubusercontent.com/nodesource/distributions/master/rpm/setup_17.
x | sudo bash - # This command ends here …
```

Also the above link may not work for Ubuntu servers. For Ubuntu please visit distributions/deb for more details on debian link and select the raw setup file for installation.

```
$ sudo yum install -y nodejs

$ npm -v
8.4.1
$ node -v
v17.5.0
```

If you received the above output then we have successfully set up Node JS and npm on the EC2 server.

Downloading package dependencies for the backend application

Next we have to install the npm dependencies for the backend application of the CKD-Navigation system. Execute the following command in the ***bayer-njit-backend*** directory inside the EC2 server. If you don't see this repository in the home directory of EC2 server via the ***ls*** (list files) cmd, please follow the section on Downloading the project files from the git repository.

```
   [bayer-njit-backend]$ npm install # Inside the bayer-njit-backend directory
```

Running the server (Testing only)

After successfully installing the application dependencies, let's check if the backend application runs without any errors from the environment setup. Enter the following command,

```
[bayer-njit-backend]$ npm start
```

This is only for testing purposes. Note that if you close the putty ssh terminal or any terminal you used for accessing the EC2 server, you will also be shutting down the server with it. To avoid this circumstance follow the steps given in Running the server (on Production mode) section.

Running the server (on Production mode)

In order to run the backend application even after the terminal is shutdown or closed, we need a way to make the application run as a daemon app rather than a session application. This can be easily done by the pm2 package in Linux. We can install pm2 via npm as follows,

```
$ sudo npm i -g pm2
```

After installation is complete, let's run the application as a daemon app.

```
          $ sudo npm i -g pm2
[bayer-njit-backend]$ sudo pm2 start ./bin/www
```

Stopping the backend server

To stop/pause the daemon running on background execute,

```
$ sudo pm2 stop www
```

For more information on pm2, please refer to PM2 - Quick Start.

3. Setting up the MySQL Database on AWS

Until now we haven't touched the persistence part of the application. Although the backend application written in NodeJS may be running flawlessly due to the Exception Handling features in the application. The application at this point will only respond with a "500 Internal server errors" for every request made via a browser or terminal. This is because the route expects the database to exist in order to send in or pull out some content. In this section we will set up the database using AWS RDS via MySQL.

Setting up MySQL on RDS

For setting up MySQL on RDS please refer

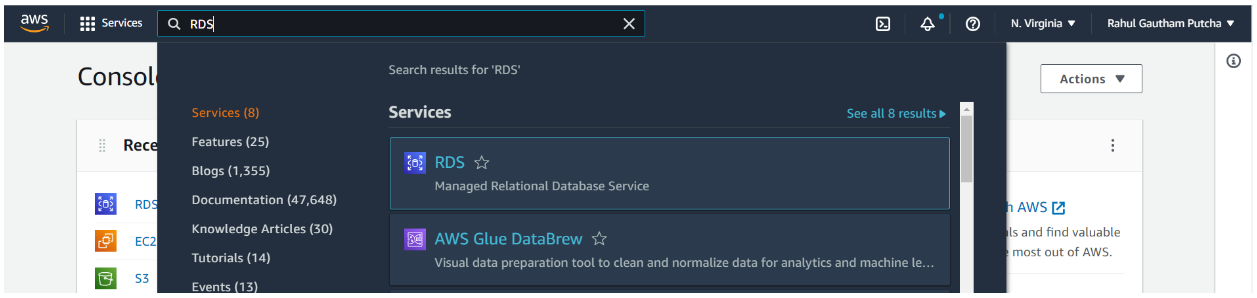
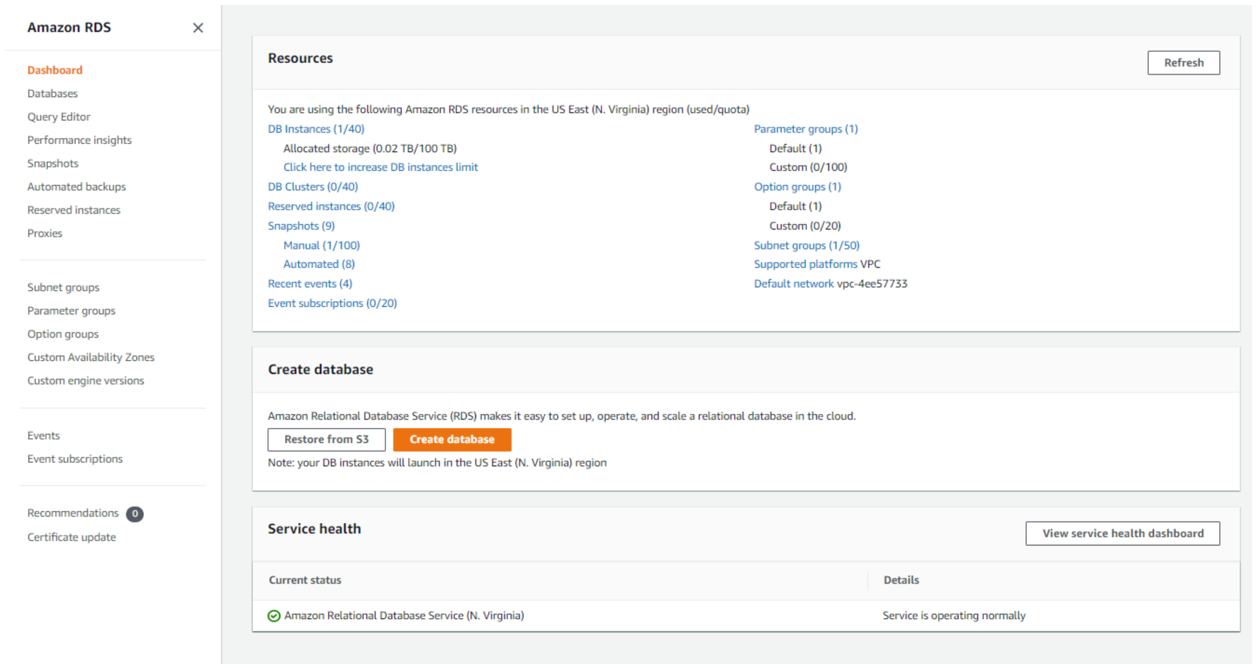


Fig 22. Searching for AWS RDS



Resources

You are using the following Amazon RDS resources in the US East (N. Virginia) region (used/quota)

DB Instances (1/40)	Parameter groups (1)
Allocated storage (0.02 TB/100 TB)	Default (1)
Click here to increase DB instances limit	Custom (0/100)
DB Clusters (0/40)	Option groups (1)
Reserved instances (0/40)	Default (1)
Snapshots (9)	Custom (0/20)
Manual (1/100)	Subnet groups (1/50)
Automated (8)	Supported platforms VPC
Recent events (4)	Default network vpc-4ee57733
Event subscriptions (0/20)	

Create database

Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud.

Note: your DB instances will launch in the US East (N. Virginia) region

Service health

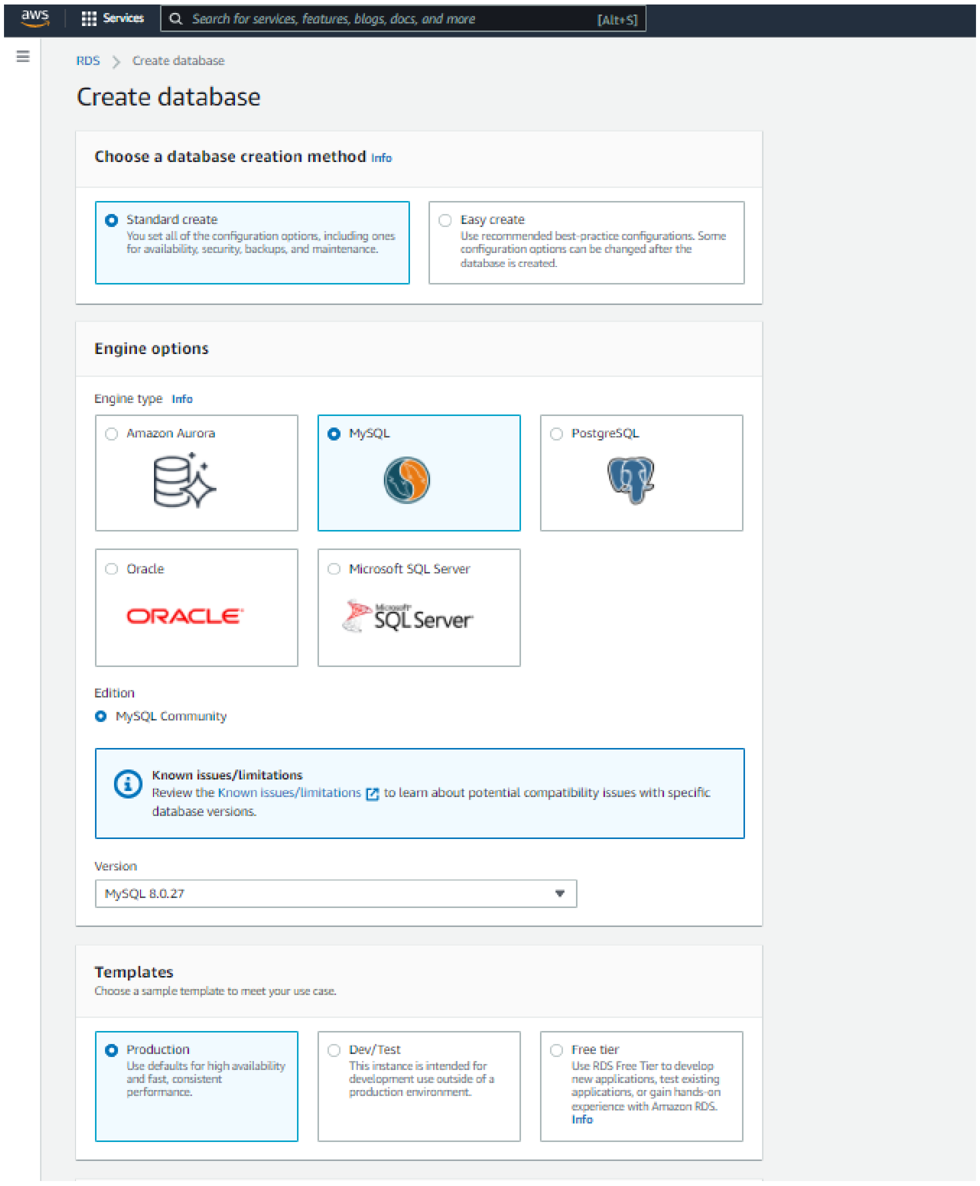
Current status	Details
Amazon Relational Database Service (N. Virginia)	Service is operating normally

Fig 23. Amazon RDS Home page

The backend application was designed on a free-tier database system, for actual production it is recommended working on a database that suits as per needs. The physical appearance of the database may not matter as the Database endpoint covers the complexity behind the internal working of the database. On overall the development configuration includes

1. Search for **RDS**
2. Click on the **Create database** button.

3. In order to create a MySQL database using Amazon RDS,
    a. Select **Standard create** option
    b. Engine options as **MySQL** and edition as **MySQL v8.0.27**
    c. Set template to **Production**
    d. Set Availability and Durability to **Single DB instance**
    (For actual production it is recommended to use a **Multi-AZ DB instance**. This makes the database replicated on multiple machines and hence data is protected from machine failure. This also solves the issue of high availability)



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

Amazon Aurora
MySQL [✓]
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.

Version
MySQL 8.0.27

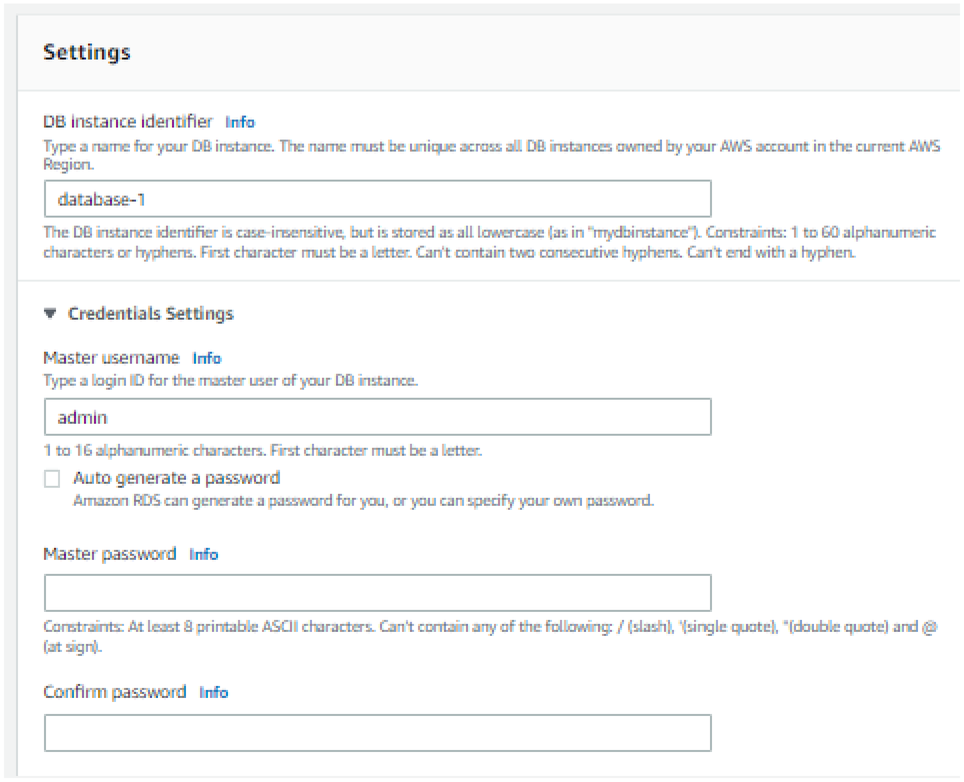
Templates
Choose a sample template to meet your use case.

Production [✓]
Use defaults for high availability and fast, consistent performance.

Dev/Test
This instance is intended for development use outside of a production environment.

Free tier
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.
Info

Fig 24. Creating a basic mysql database


Settings

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.

database-1

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.

admin

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

☐ 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 password Info

Fig 25. Setting up the User credentials

4. Next comes the DB instance identifier and credentials
   a. The instance identifier must be unique around AWS, so that a unique identifier endpoint is generated as HOSTNAME / HOSTADDRESS for the MySQL-based RDS database.
   b. The credentials must be decided as it must be used for accessing the database. Keep it as secure as possible.

   Note that above fields contents must be noted down / remembered as we are going to require it for connecting the AWS backend to the RDS database.

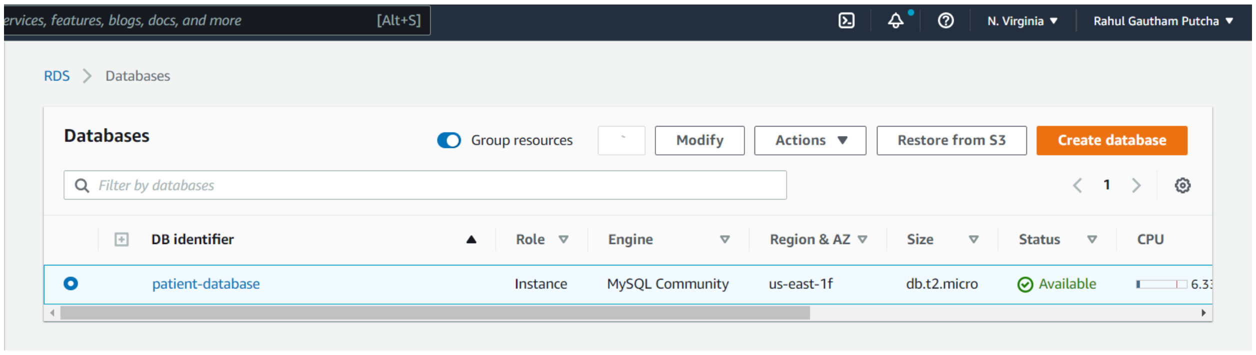
5. We can keep the rest as default or change it depending on the application requirements. For more details on AWS RDS, please refer to MySQL on Amazon RDS.
6. Click on **Create database** button

The RDS database is now created.

Accessing the endpoint information

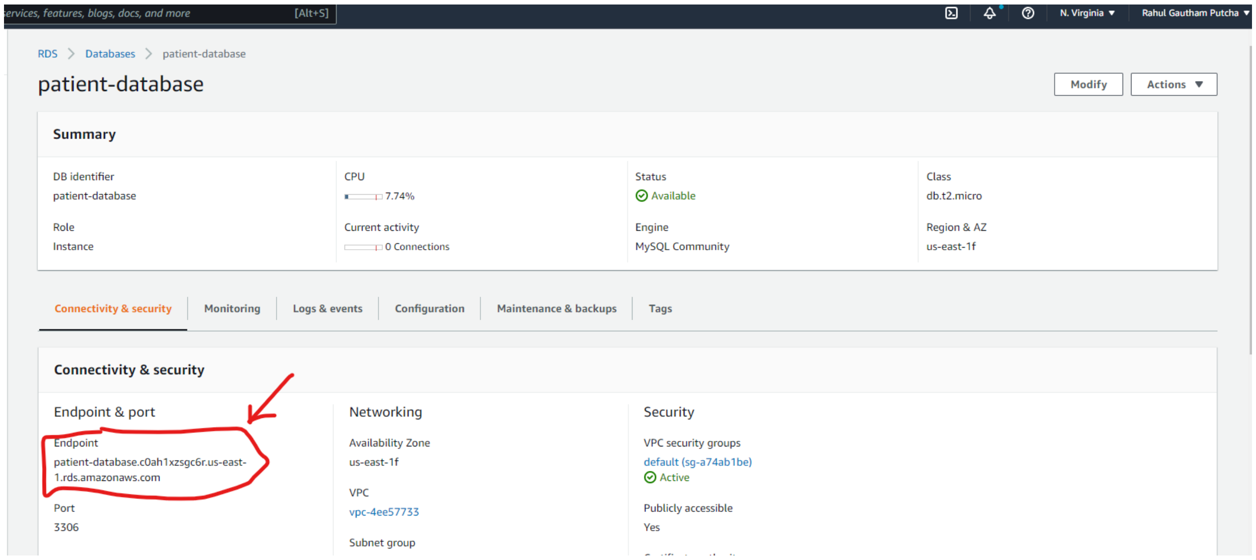
After having a database created we need a way to get the MySQL database endpoint for notifying our backend application our intention in using the database we have created in the previous section. Follow the steps below:

1. On the **RDS** > **Database** page, Select the database for which the endpoint is being searched
2. Under the **Connectivity and security** tab, find the endpoint that we need to connect backend to database



DB identifier	Role	Engine	Region & AZ	Size	Status	CPU
patient-database	Instance	MySQL Community	us-east-1f	db.t2.micro	Available	6.33

Fig 26. AWS RDS select the database using DB identifier



DB identifier: patient-database
CPU: 7.74%
Status: Available
Class: db.t2.micro
Role: Instance
Current activity: 0 Connections
Engine: MySQL Community
Region & AZ: us-east-1f

Endpoint: patient-database.c0ah1xzsgc6r.us-east-1.rds.amazonaws.com
Port: 3306
Availability Zone: us-east-1f
VPC: vpc-4ee57733
Subnet group
VPC security groups: default (sg-a74ab1be) Active
Publicly accessible: Yes

Fig 27. Get the endpoint for the Database

Setting up the initial MySQL database for CKD Navigation System

At initial level the application requires the label_info.sql and patients_info.sql in the patient-database.

For accessing the database execute the following command after substituting the AWS RDS credentials within the command syntax below.

Note that below is a single line command,

```
$ mysql
-h <<AWS Host address>>
-u <<AWS RDS username>> -p
```

Then you will be prompted a password. Enter the password for above RDS username,

```
Password:<<Enter the AWS RDS password>>
```

If you see the `mysql>` prompt then you are able to access the mysql using AWS RDS.

Note: Most accessibility issues may rise due to port issues or security issues dealt with mismatching IP addresses in Security Groups.

Exit the mysql prompt,
mysql> exit

Now transfer the .sql database import files to the RDS database using the following commands. Note that below commands must be executed in the directory that contains label_info.sql and patients_info.sql files.

```
$ mysql -h <<Host address>> -u <<username>> -p < label_info.sql
$ mysql -h <<Host address>> -u <<username>> -p < patients_info.sql
```

After running the NodeJS backend application on EC2 server using the pm2 command described in Running the server (on Production mode) section, you will also notice that additional tables are automatically created. These tables exist for storing the application user-specific data, such as user accounts, preferences, histories. ***Although this may not work at this point yet as the database endpoint changes aren't reflected back to the backend API on EC2 server.*** Let's do this next.

4. API and Database Connection

For the most part, the API and database connection is resolved internally within the backend part of the application using Sequelize package. In order to connect the intended database we are required to have the database endpoint and this endpoint must be configured within the **.env** file of the backend application. You may find a **.env.backup** file instead of .env. In this case rename the file to **.env**.

```
[bayer-njit-backend]$ mv .env.backup .env
```

The content of the .env file is shown below.

```
# Rename this file to .env

NODE_ENV=development
PORT=3000

# Development Env Settings
MYSQL_DEV_USER=rahul
MYSQL_DEV_PASSWORD=PASSword@123
MYSQL_DEV_HOST=localhost

# Testing Env Settings
MYSQL_TEST_USER=rahul
MYSQL_TEST_PASSWORD=PASSword@123
MYSQL_TEST_HOST=localhost

# Production Env Settings
MYSQL_PROD_USER=<HOST_MACHINE_SQL_USER>
MYSQL_PROD_PASSWORD=<YOUR_PASSWORD>
MYSQL_PROD_HOST=<HOST_MACHINE_ADDRESS>
```

For starters, the first two lines denote the application environment and server listening port set default on 3000. The next few lines with 3-lines pattern sequence (ignoring the comment and blank lines), include the username, password and host address. Substitute the lines with a MySQL endpoint that is used for each one of the development, testing or production environments.

Especially for the production mode, we need to set up the lines below for connecting Backend to AWS RDS.

```
# Production Env Settings
MYSQL_PROD_USER=<<HOST_MACHINE_SQL_USER>>
MYSQL_PROD_PASSWORD=<<YOUR_PASSWORD>>
MYSQL_PROD_HOST=<<HOST_MACHINE_ADDRESS>>
```

After running the NodeJS backend application on EC2 server using the pm2 command described in Running the server (on Production mode) section, you will also notice that additional tables are automatically created. These tables exist for storing the application user-specific data, such as user accounts, preferences, histories.

References

[1] Multitier architecture - Wikipedia
[2] Cloud Object Storage – Amazon S3 – Amazon Web Services
[3] Secure and resizable cloud compute – Amazon EC2 – Amazon Web Services
[4] Amazon RDS | Cloud Relational Database | Amazon Web Services

Future Planning for this documentation

1. API Testing using Postman
    a. Using Postman testing example file for testing purpose
2. Internal working of Backend Node JS application
    a. ExpressJS and MVC architecture
    b. General Structure of Sequelize ORM in Backend application