Configuring Reverse Proxy

AWS Internship Report Submitted

BY

Saitwadekar Valay Angar	(Roll No 49)
Dave Het Anand	(Roll No 05)
Nohwar Vaibhav	(Roll No 35)
Singh Anuj Pravin	(Roll No 56)

Supervisor/Guide

Prof. Sonali Padalkar

DEPARTMENT OF COMPUTER ENGINEERING

SHREE L. R. TIWARI COLLEGE OF ENGINEERING

KANAKIA PARK, MIRA ROAD (E), THANE -401 107, MAHARASHTRA.

University of Mumbai

Year 2022-23

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Saitwadekar Valay Angar

Roll No.: 49

Dave Het Anand

Roll No.: 05

Nohwar Vaibhav

Roll No.: 35

Singh Anuj Pravin

Roll No.: 56

Date: 8th August, 2022

Shree Rahul Education Society's (Regd.)

SHREE L. R. TIWARI COLLEGE OF ENGINEERING

Kanakia Park, Near Commissioner's Bungalow, Mira Road (East), Thane 401107, Maharashtra

(Approved by AICTE, Govt. of Maharashtra & Affiliated to University of Mumbai)

NAAC Accredited | ISO 9001:2015 Certified

Tel. No.: 022-28120144 / 022-28120145 | Email: slrtce@rahuleducation.com | Website: www.slrtce.in

CERTIFICATE

This is to certify that the project entitled "**Configuring Reverse Proxy**" is a bonafide work of

Saitwadekar Valay Angar	**(Roll No. 49)**
Dave Het Anand	**(Roll No. 05)**
Nohwar Vaibhav	**(Roll No. 35)**
Singh Anuj Pravin	**(Roll No. 56)**

submitted as a Project for AWS Internship **"Computer Engineering"**.

Signature of Supervisor/Guide
Prof. Sonali Padalkar

Table of Contents

1. Introduction

1.1 What is a Reverse Proxy?

A proxy server is a go-between or intermediary server that forwards requests for content from multiple clients to different servers across the Internet. A reverse proxy server is a type of proxy server that typically sits behind the firewall in a private network and directs client requests to the appropriate backend server. A reverse proxy provides an additional level of abstraction and control to ensure the smooth flow of network traffic between clients and servers.

1.2 How is a Reverse Proxy Different?

A reverse proxy is a server that sits in front of one or more web servers, intercepting requests from clients. This is different from a forward proxy, where the proxy sits in front of the clients. With a reverse proxy, when clients send requests to the origin server of a website, those requests are intercepted at the network edge by the reverse proxy server. The reverse proxy server will then send requests to and receive responses from the origin server.

The difference between a forward and reverse proxy is subtle but important. A simplified way to sum it up would be to say that a forward proxy sits in front of a client and ensures that no origin server ever communicates directly with that specific client. On the other hand, a reverse proxy sits in front of an origin server and ensures that no client ever communicates directly with that origin server.

Let's illustrate (Ref to the figure 1.1) by naming the computers involved:

D: Any number of users' home computers

E: This is a reverse proxy server

F: One or more origin servers

Typically all requests from D would go directly to F, and F would send responses directly to D. With a reverse proxy, all requests from D will go directly to E, and E will send its requests to and receive responses from F. E will then pass along the appropriate responses to D.

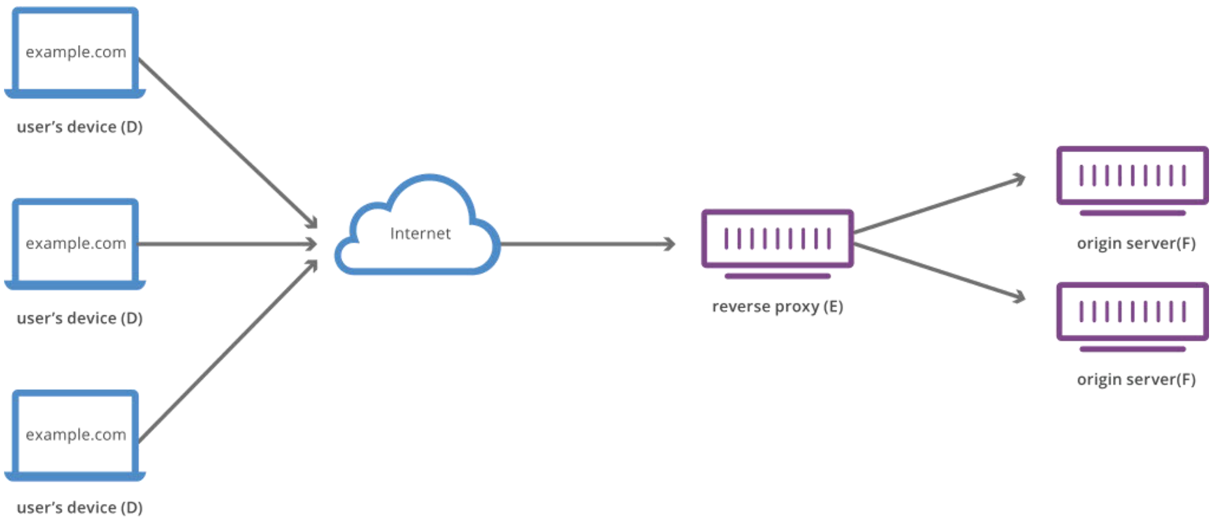


Fig 1.1: Reverse Proxy Flow

1.3 Benefits of Reverse Proxy:

Below are some of the benefits of a reverse proxy:

- Load balancing - A popular website that gets millions of users every day may not be able to handle all of its incoming site traffic with a single origin server. Instead, the site can be distributed among a pool of different servers, all handling requests for the same site. In this case, a reverse proxy can provide a load balancing solution which will distribute the incoming traffic evenly among the different servers to prevent any single server from becoming overloaded. In the event that a server fails completely, other servers can step up to handle the traffic.
- Protection from attacks - With a reverse proxy in place, a web site or service never needs to reveal the IP address of their origin server(s). This makes it much harder for attackers to leverage a targeted attack against them, such as a DDoS attack. Instead the attackers will only be able to target the reverse proxy.
- Global Server Load Balancing (GSLB) - In this form of load balancing, a website can be distributed on several servers around the globe and the reverse proxy will send clients to the server that's geographically closest to them. This decreases the distances that requests and responses need to travel, minimizing load times.
- Caching - A reverse proxy can also cache content, resulting in faster performance. For example, if a user in Paris visits a reverse-proxied website with web servers in Los Angeles, the user might actually connect to a local reverse proxy server in

Paris, which will then have to communicate with an origin server in L.A. The proxy server can then cache (or temporarily save) the response data. Subsequent Parisian users who browse the site will then get the locally cached version from the Parisian reverse proxy server, resulting in much faster performance.

- SSL encryption - Encrypting and decrypting SSL (or TLS) communications for each client can be computationally expensive for an origin server. A reverse proxy can be configured to decrypt all incoming requests and encrypt all outgoing responses, freeing up valuable resources on the origin server.

2. System Requirements

2.1 Hardware Requirements:

- Memory: 2 GB RAM
- CPU: Dual Core (in Physical Machine)/ 2 vCore (in Virtual Machine)
- Disk Space: 20 GB minimum

2.2 Software Requirements:

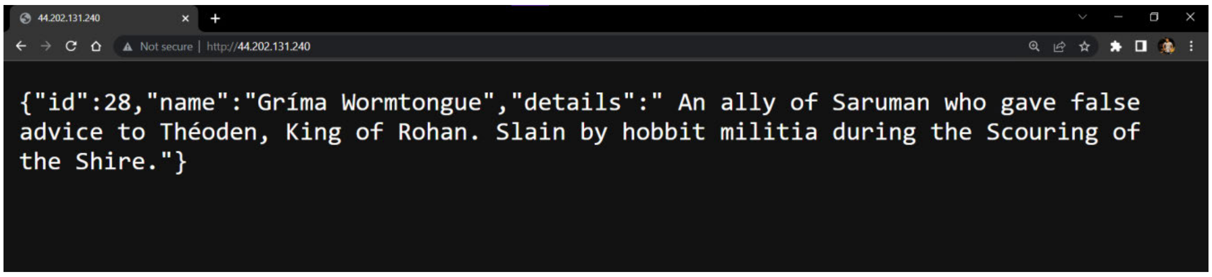
- OS: Amazon Linux AMI (Amazon Machine Language)
- Software: NGNIX
- Database: MySQL

3. Project Description

A reverse proxy is a server that sits in front of web servers and forwards client (e.g. web browser) requests to those web servers. Reverse proxies are typically implemented to help increase security, performance, and reliability. In order to better understand how a reverse proxy works and the benefits it can provide we made a basic node.js application which is connected to MYSQL database, which contains some information about Characters in LOTR (Lord of the Rings).

By default, the app is configured to the default PORT 8080 when connected, by using NGINX we have configured the application to reverse proxy which ultimately make the application capable of serving request on PORT 80.

This method enhances the security of the application by avoiding state or institutional browsing restrictions, blocking access to certain content and protecting their identity online.



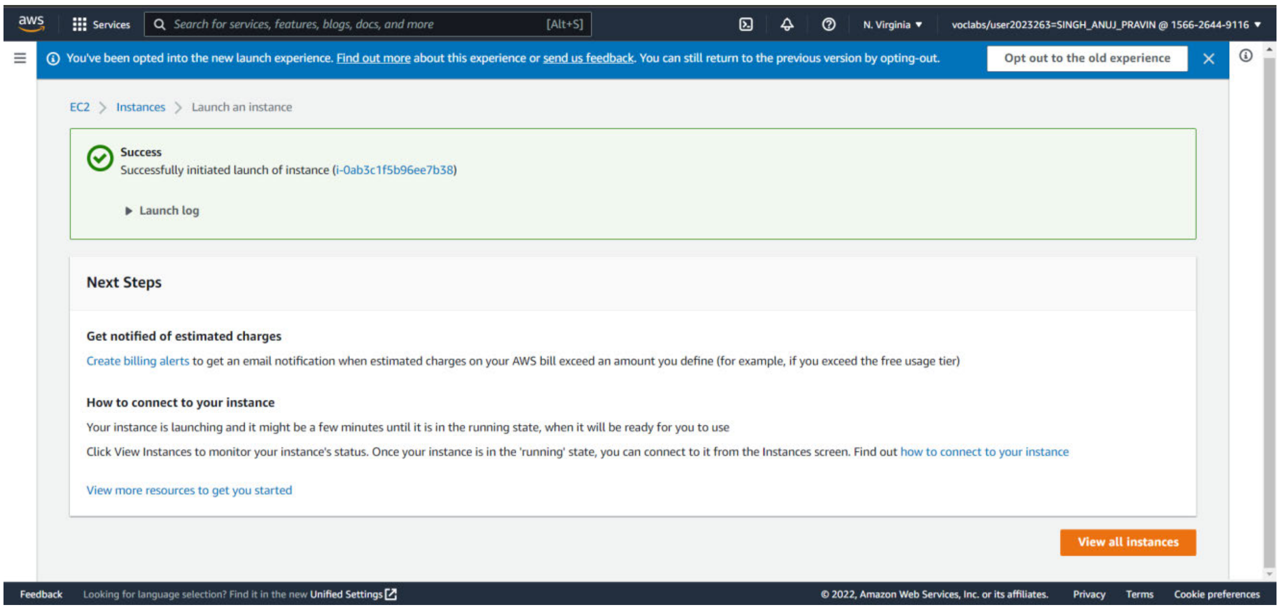
```
{"id":28,"name":"Gríma Wormtongue","details":" An ally of Saruman who gave false
advice to Théoden, King of Rohan. Slain by hobbit militia during the Scouring of
the Shire."}
```

4. Implementation:

<u>MYSQL – SERVER CREATION USING EC2 INSTANCE</u>

➔ Creating a MySQL Database Server

Launch Initiated



aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
voclabs/user2023263=SINGH_ANUJ_PRAVIN @ 1566-2644-9116
Opt out to the old experience
EC2 > Instances > Launch an instance
Launch log
View all instances
Feedback
Privacy
Terms
Cookie preferences

You've been opted into the new launch experience. Find out more about this experience or send us feedback. You can still return to the previous version by opting-out.

Success
Successfully initiated launch of instance (i-0ab3c1f5b96ee7b38)

Next Steps

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier)

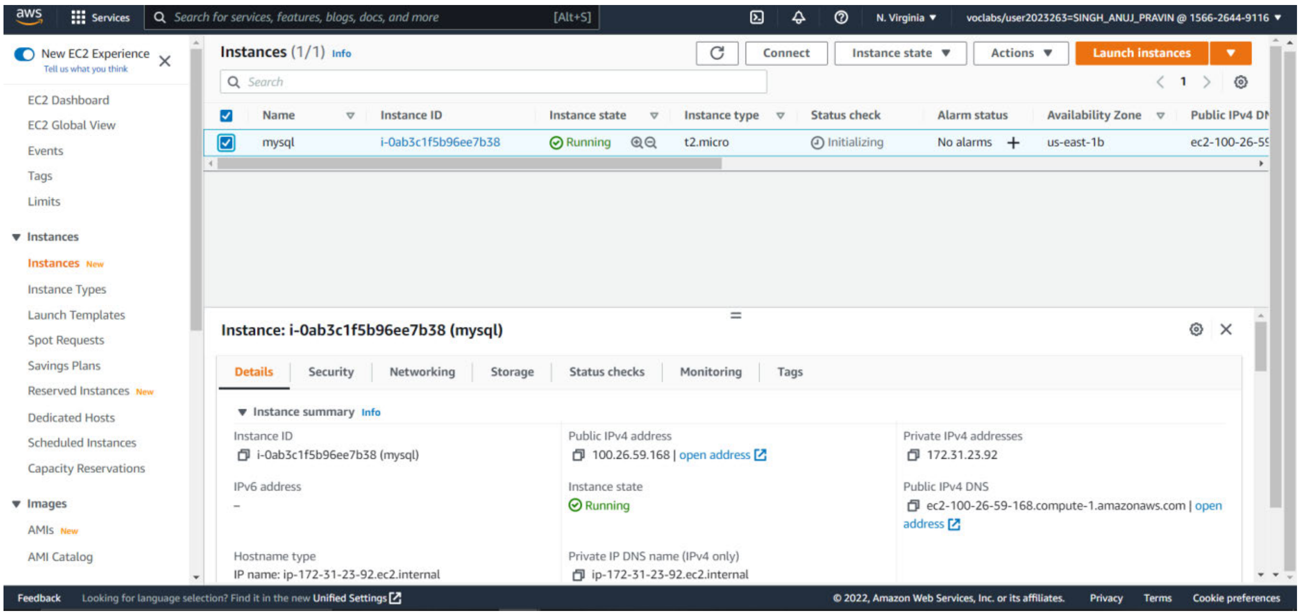
How to connect to your instance
Your instance is launching and it might be a few minutes until it is in the running state, when it will be ready for you to use
Click View Instances to monitor your instance's status. Once your instance is in the 'running' state, you can connect to it from the Instances screen. Find out how to connect to your instance

View more resources to get you started

Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates.

Successfully launched



Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
mysql	i-0ab3c1f5b96ee7b38	Running	t2.micro	Initializing	No alarms	us-east-1b	ec2-100-26-5

Instance: i-0ab3c1f5b96ee7b38 (mysql)

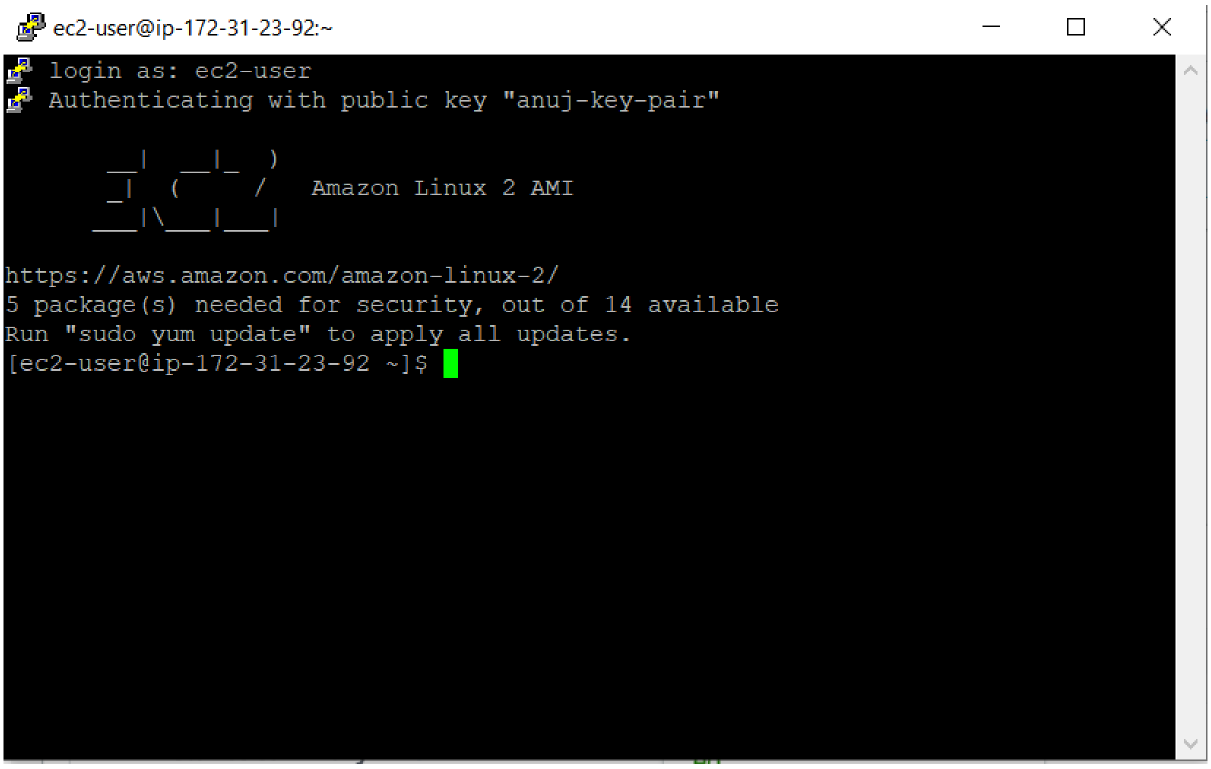
Instance summary Info

Instance ID i-0ab3c1f5b96ee7b38 (mysql)	Public IPv4 address 100.26.59.168 | open address	Private IPv4 addresses 172.31.23.92
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-100-26-59-168.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-23-92.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-23-92.ec2.internal	

Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates.

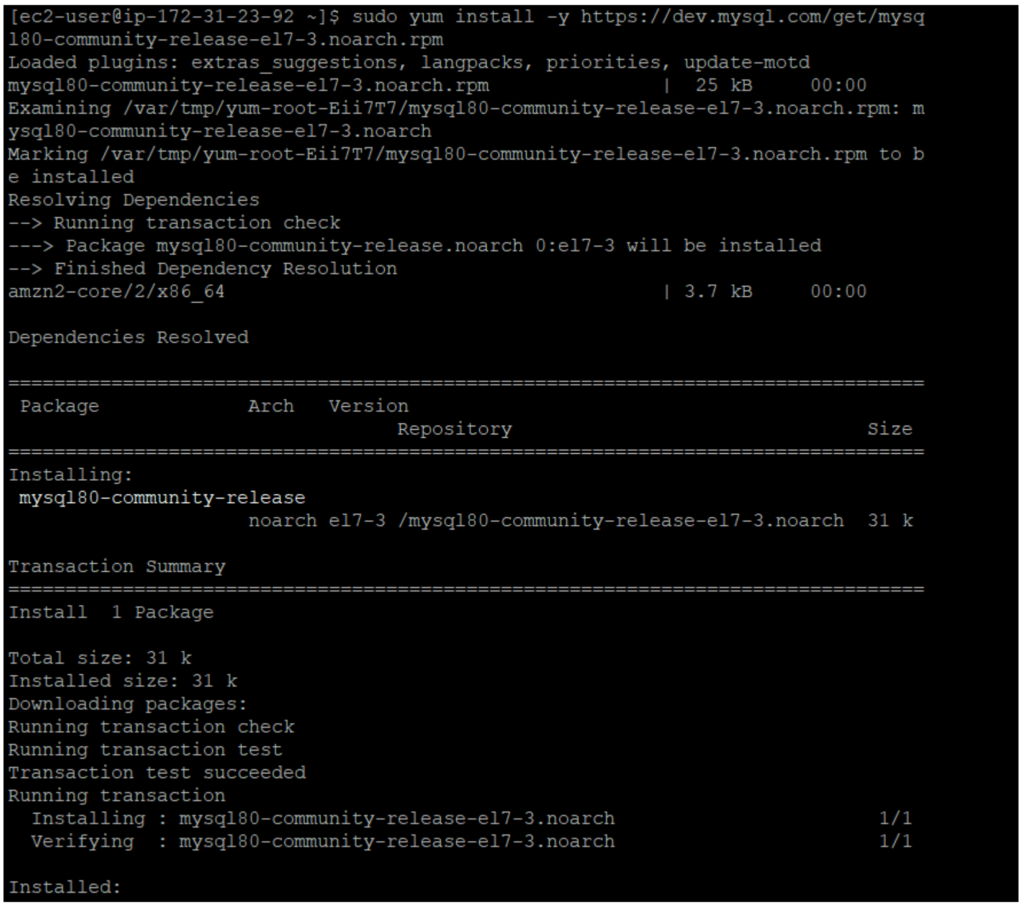
➔ **Connected EC2 Instance with Database**



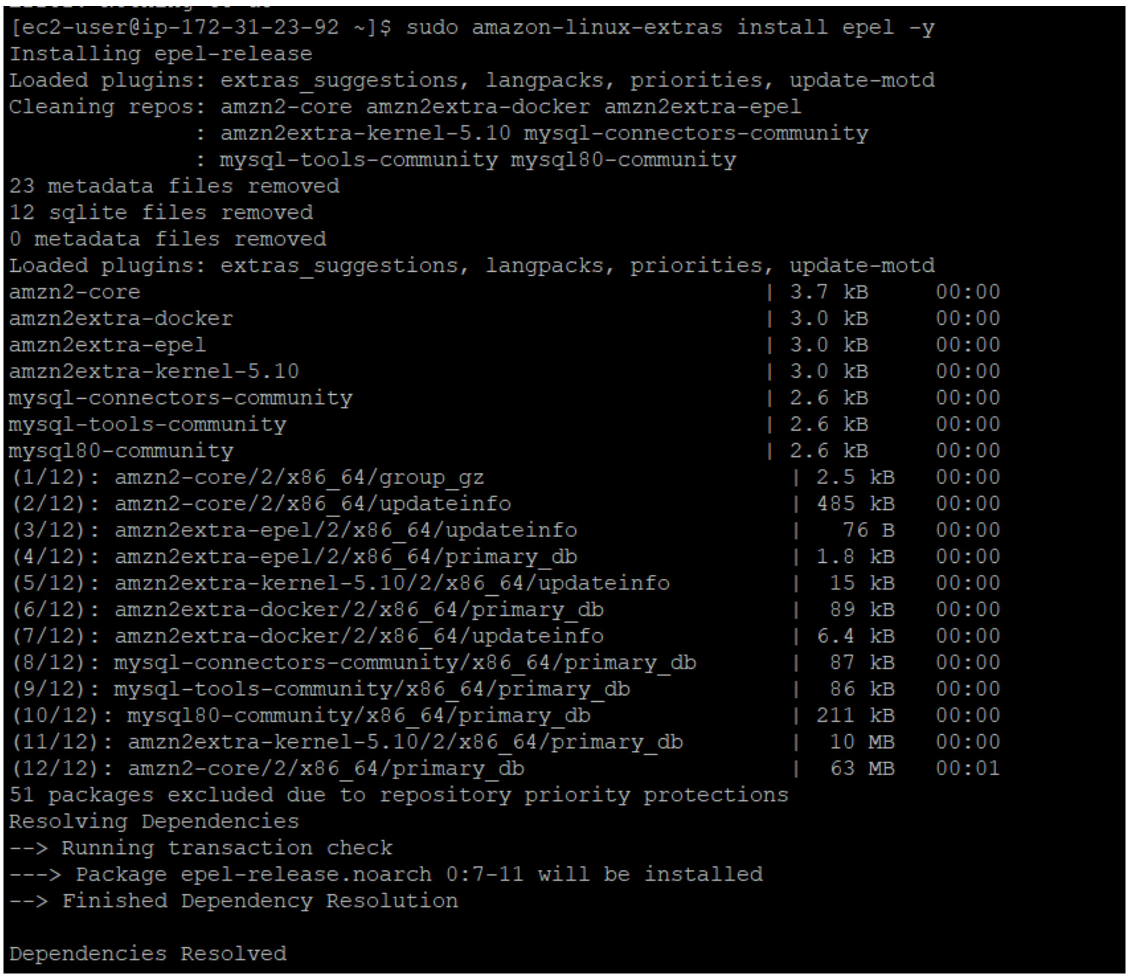
ec2-user@ip-172-31-23-92:~

```
login as: ec2-user
Authenticating with public key "anuj-key-pair"

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

https://aws.amazon.com/amazon-linux-2/
5 package(s) needed for security, out of 14 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-23-92 ~]$
```

➔ Installing **MySQL** on Amazon Linux EC2 instance

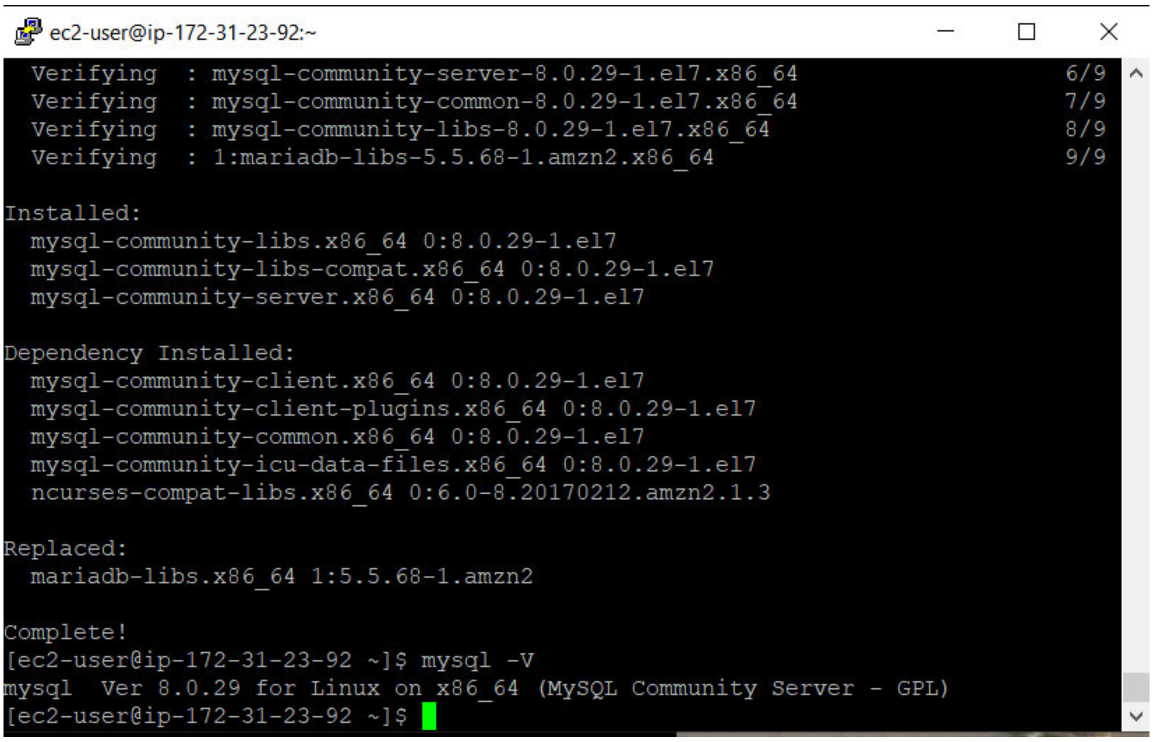


```
[ec2-user@ip-172-31-23-92 ~]$ sudo yum install -y https://dev.mysql.com/get/mysq
l80-community-release-el7-3.noarch.rpm
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
mysql80-community-release-el7-3.noarch.rpm               |  25 kB     00:00
Examining /var/tmp/yum-root-Eii7T7/mysql80-community-release-el7-3.noarch.rpm: m
ysql80-community-release-el7-3.noarch
Marking /var/tmp/yum-root-Eii7T7/mysql80-community-release-el7-3.noarch.rpm to b
e installed
Resolving Dependencies
--> Running transaction check
---> Package mysql80-community-release.noarch 0:el7-3 will be installed
--> Finished Dependency Resolution
amzn2-core/2/x86_64                                      | 3.7 kB     00:00

Dependencies Resolved

================================================================================
 Package                    Arch    Version
                                          Repository                        Size
================================================================================
Installing:
 mysql80-community-release
                             noarch el7-3 /mysql80-community-release-el7-3.noarch  31 k

Transaction Summary
================================================================================
Install  1 Package

Total size: 31 k
Installed size: 31 k
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : mysql80-community-release-el7-3.noarch                       1/1
  Verifying  : mysql80-community-release-el7-3.noarch                       1/1

Installed:
```



```
[ec2-user@ip-172-31-23-92 ~]$ sudo amazon-linux-extras install epel -y
Installing epel-release
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-epel
              : amzn2extra-kernel-5.10 mysql-connectors-community
              : mysql-tools-community mysql80-community
23 metadata files removed
12 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.7 kB     00:00
amzn2extra-docker                                        | 3.0 kB     00:00
amzn2extra-epel                                          | 3.0 kB     00:00
amzn2extra-kernel-5.10                                   | 3.0 kB     00:00
mysql-connectors-community                               | 2.6 kB     00:00
mysql-tools-community                                    | 2.6 kB     00:00
mysql80-community                                        | 2.6 kB     00:00
(1/12): amzn2-core/2/x86_64/group_gz                       | 2.5 kB   00:00
(2/12): amzn2-core/2/x86_64/updateinfo                     | 485 kB   00:00
(3/12): amzn2extra-epel/2/x86_64/updateinfo                |   76 B   00:00
(4/12): amzn2extra-epel/2/x86_64/primary_db                | 1.8 kB   00:00
(5/12): amzn2extra-kernel-5.10/2/x86_64/updateinfo         |   15 kB  00:00
(6/12): amzn2extra-docker/2/x86_64/primary_db              |   89 kB  00:00
(7/12): amzn2extra-docker/2/x86_64/updateinfo              | 6.4 kB   00:00
(8/12): mysql-connectors-community/x86_64/primary_db       |   87 kB  00:00
(9/12): mysql-tools-community/x86_64/primary_db            |   86 kB  00:00
(10/12): mysql80-community/x86_64/primary_db               | 211 kB   00:00
(11/12): amzn2extra-kernel-5.10/2/x86_64/primary_db        |   10 MB  00:00
(12/12): amzn2-core/2/x86_64/primary_db                    |   63 MB  00:01
51 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
---> Package epel-release.noarch 0:7-11 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

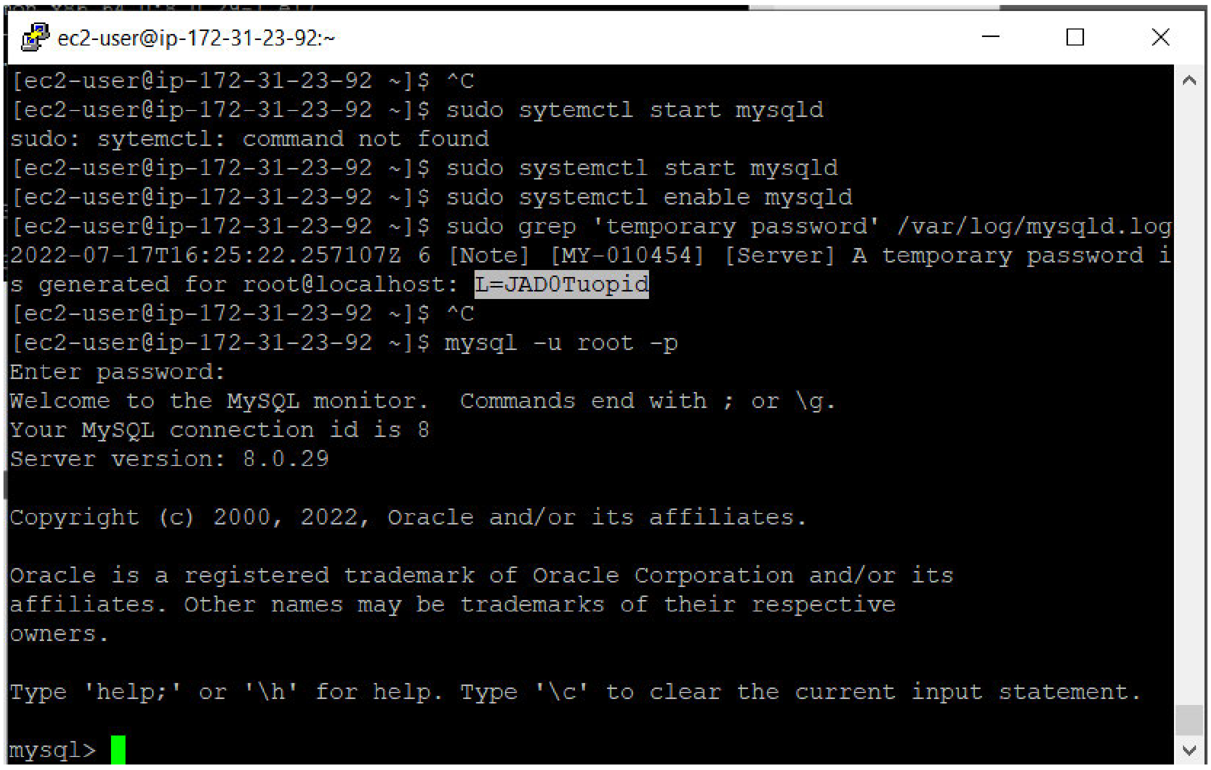
➔ **MySQL Successfully Installeds**



ec2-user@ip-172-31-23-92:~

```
  Verifying  : mysql-community-server-8.0.29-1.el7.x86_64                 6/9
  Verifying  : mysql-community-common-8.0.29-1.el7.x86_64                 7/9
  Verifying  : mysql-community-libs-8.0.29-1.el7.x86_64                   8/9
  Verifying  : 1:mariadb-libs-5.5.68-1.amzn2.x86_64                       9/9

Installed:
  mysql-community-libs.x86_64 0:8.0.29-1.el7
  mysql-community-libs-compat.x86_64 0:8.0.29-1.el7
  mysql-community-server.x86_64 0:8.0.29-1.el7

Dependency Installed:
  mysql-community-client.x86_64 0:8.0.29-1.el7
  mysql-community-client-plugins.x86_64 0:8.0.29-1.el7
  mysql-community-common.x86_64 0:8.0.29-1.el7
  mysql-community-icu-data-files.x86_64 0:8.0.29-1.el7
  ncurses-compat-libs.x86_64 0:6.0-8.20170212.amzn2.1.3

Replaced:
  mariadb-libs.x86_64 1:5.5.68-1.amzn2

Complete!
[ec2-user@ip-172-31-23-92 ~]$ mysql -V
mysql  Ver 8.0.29 for Linux on x86_64 (MySQL Community Server - GPL)
[ec2-user@ip-172-31-23-92 ~]$
```

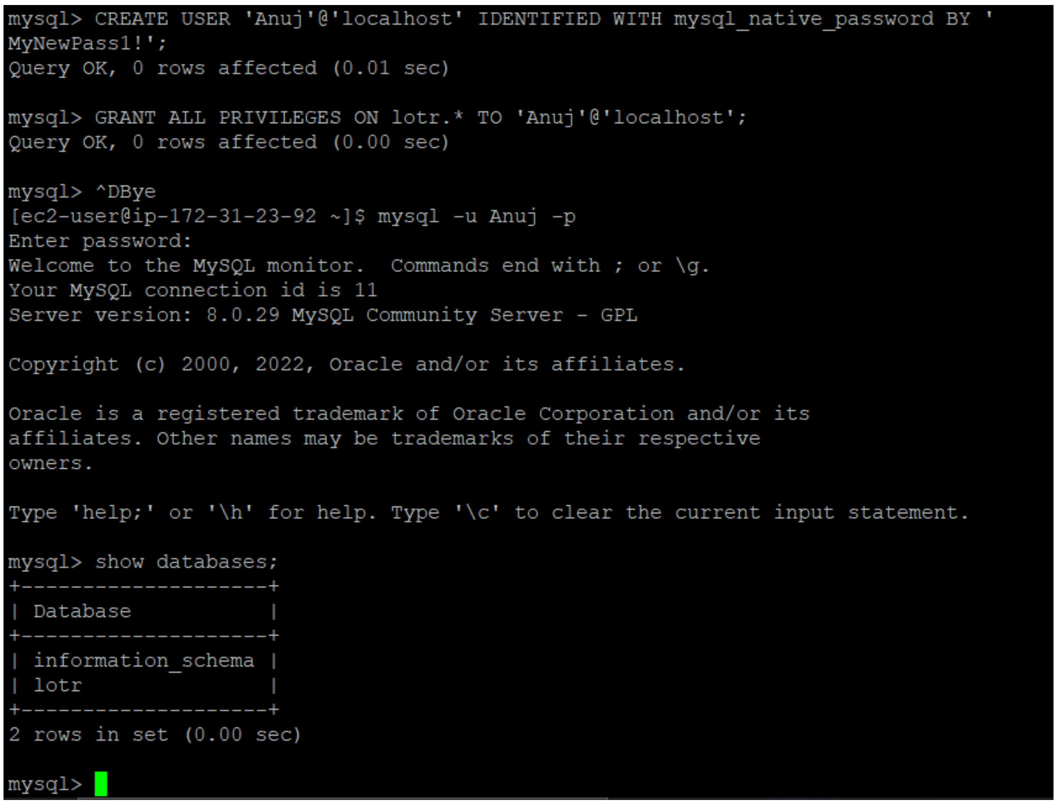
➔ **Configuring root user password for MySQL database to start always as EC2 instance start**



```
[ec2-user@ip-172-31-23-92 ~]$ ^C
[ec2-user@ip-172-31-23-92 ~]$ sudo sytemctl start mysqld
sudo: sytemctl: command not found
[ec2-user@ip-172-31-23-92 ~]$ sudo systemctl start mysqld
[ec2-user@ip-172-31-23-92 ~]$ sudo systemctl enable mysqld
[ec2-user@ip-172-31-23-92 ~]$ sudo grep 'temporary password' /var/log/mysqld.log
2022-07-17T16:25:22.257107Z 6 [Note] [MY-010454] [Server] A temporary password i
s generated for root@localhost: L=JAD0Tuopid
[ec2-user@ip-172-31-23-92 ~]$ ^C
[ec2-user@ip-172-31-23-92 ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.29

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

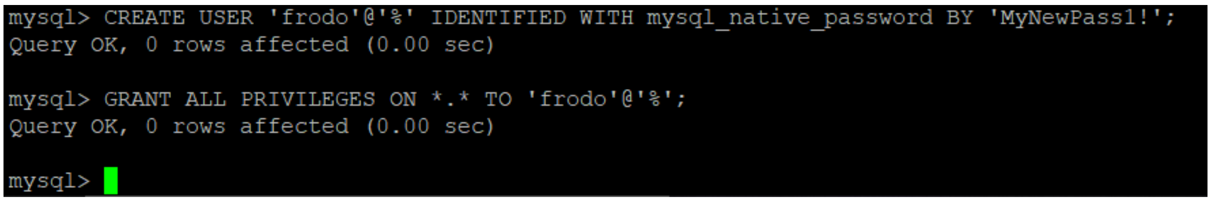
➔ **Creating a new user that can access the database from localhost:**

User name: Anuj, Password: MyNewPass1!



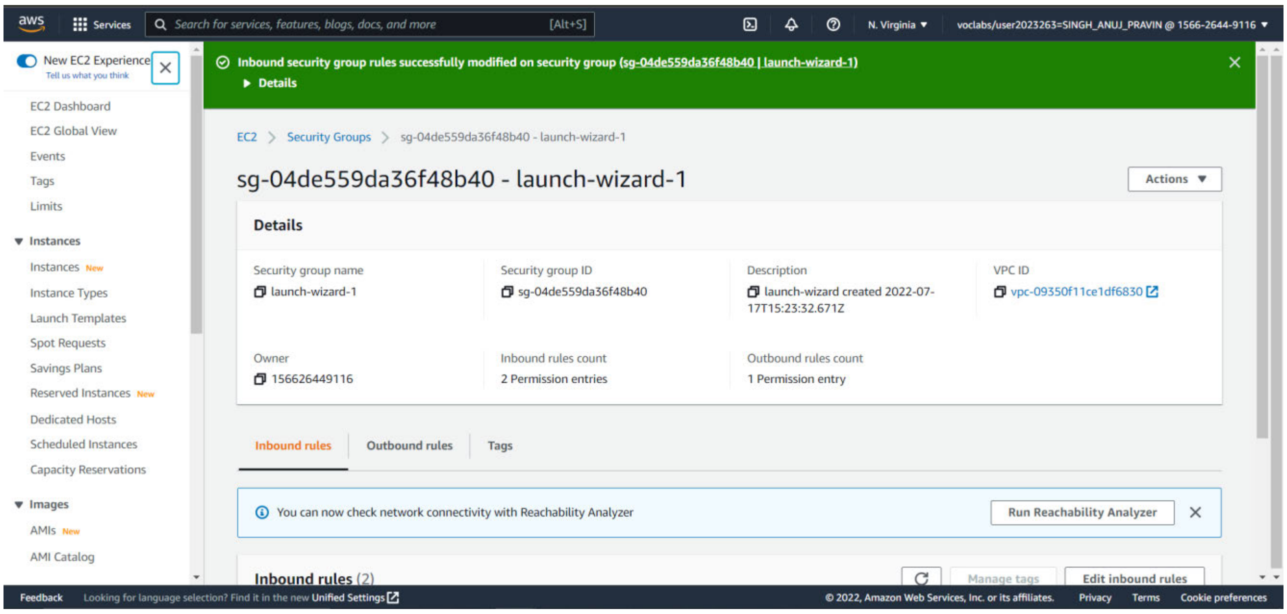
```
mysql> CREATE USER 'Anuj'@'localhost' IDENTIFIED WITH mysql_native_password BY '
MyNewPass1!';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON lotr.* TO 'Anuj'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> ^DBye
[ec2-user@ip-172-31-23-92 ~]$ mysql -u Anuj -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.29 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| lotr               |
+--------------------+
2 rows in set (0.00 sec)

mysql>
```

➔ Creating Another User which Access the Database All Over the World



```
mysql> CREATE USER 'frodo'@'%' IDENTIFIED WITH mysql_native_password BY 'MyNewPass1!';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'frodo'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> 
```

➔ Configuring Security Groups for MySQL/Aurora Connection Anywhere in The World



Inbound security group rules successfully modified on security group (sg-04de559da36f48b40 | launch-wizard-1)

Details

EC2 > Security Groups > sg-04de559da36f48b40 - launch-wizard-1

sg-04de559da36f48b40 - launch-wizard-1

Details

Security group name: launch-wizard-1
Security group ID: sg-04de559da36f48b40
Description: launch-wizard created 2022-07-17T15:23:32.671Z
VPC ID: vpc-09350f11ce1df6830
Owner: 156626449116
Inbound rules count: 2 Permission entries
Outbound rules count: 1 Permission entry

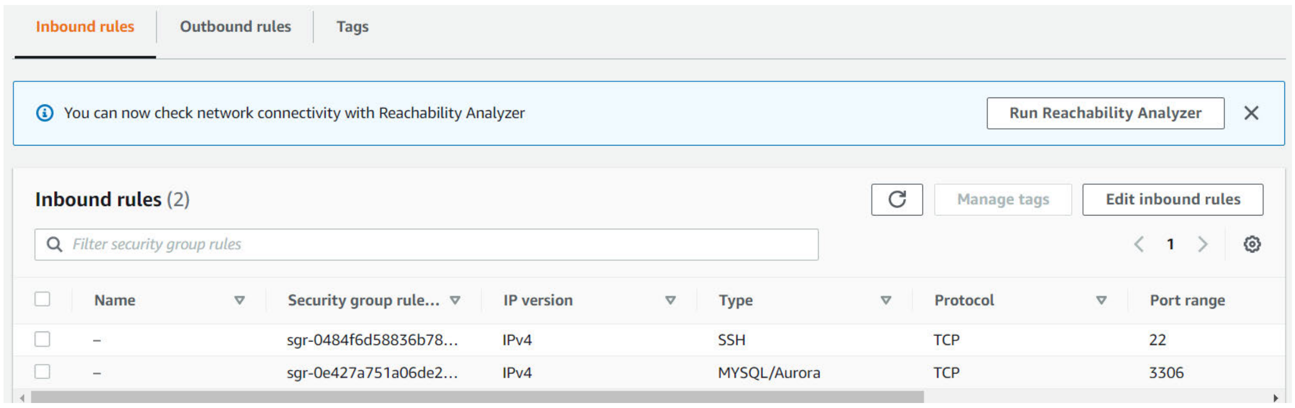
You can now check network connectivity with Reachability Analyzer

Inbound rules (2)

Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates.

Added Inbound Rule for port 3306



You can now check network connectivity with Reachability Analyzer

Inbound rules (2)

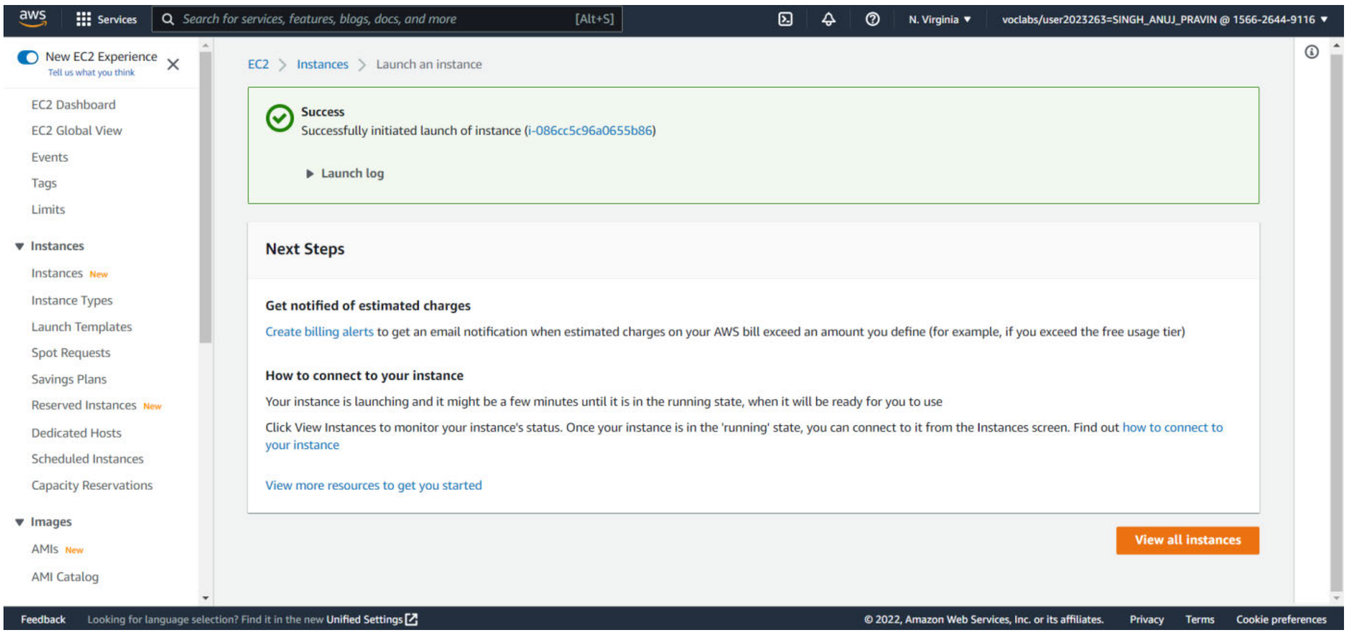
	Name	Security group rule...	IP version	Type	Protocol	Port range
☐	–	sgr-0484f6d58836b78...	IPv4	SSH	TCP	22
☐	–	sgr-0e427a751a06de2...	IPv4	MYSQL/Aurora	TCP	3306

5. Testing:

NODE.JS APP DEPLOYMENT USING EC2 INSTANCE

➔ Creating a EC2 Instance for App Deployment

Launch Initiated



Success
Successfully initiated launch of instance (i-086cc5c96a0655b86)

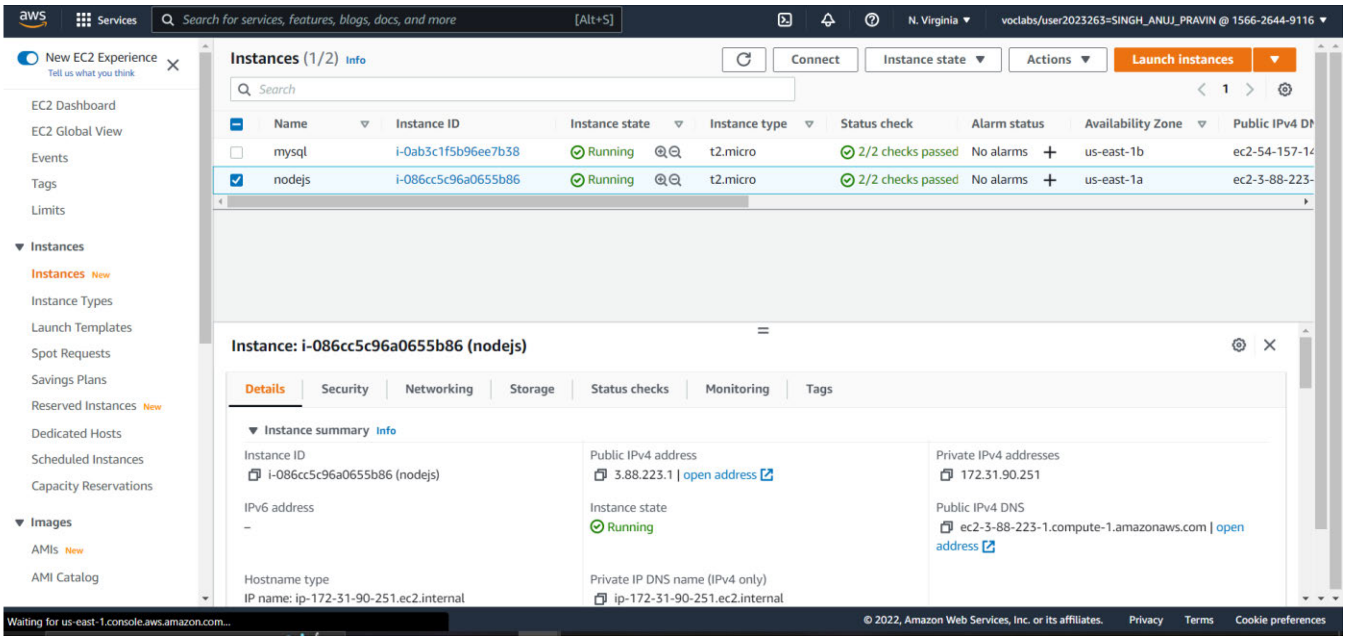
Next Steps

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier)

How to connect to your instance
Your instance is launching and it might be a few minutes until it is in the running state, when it will be ready for you to use
Click View Instances to monitor your instance's status. Once your instance is in the 'running' state, you can connect to it from the Instances screen. Find out how to connect to your instance

View more resources to get you started

Successfully Launched



Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
mysql	i-0ab3c1f5b96ee7b38	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-54-157-14
nodejs	i-086cc5c96a0655b86	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-3-88-223-

Instance: i-086cc5c96a0655b86 (nodejs)

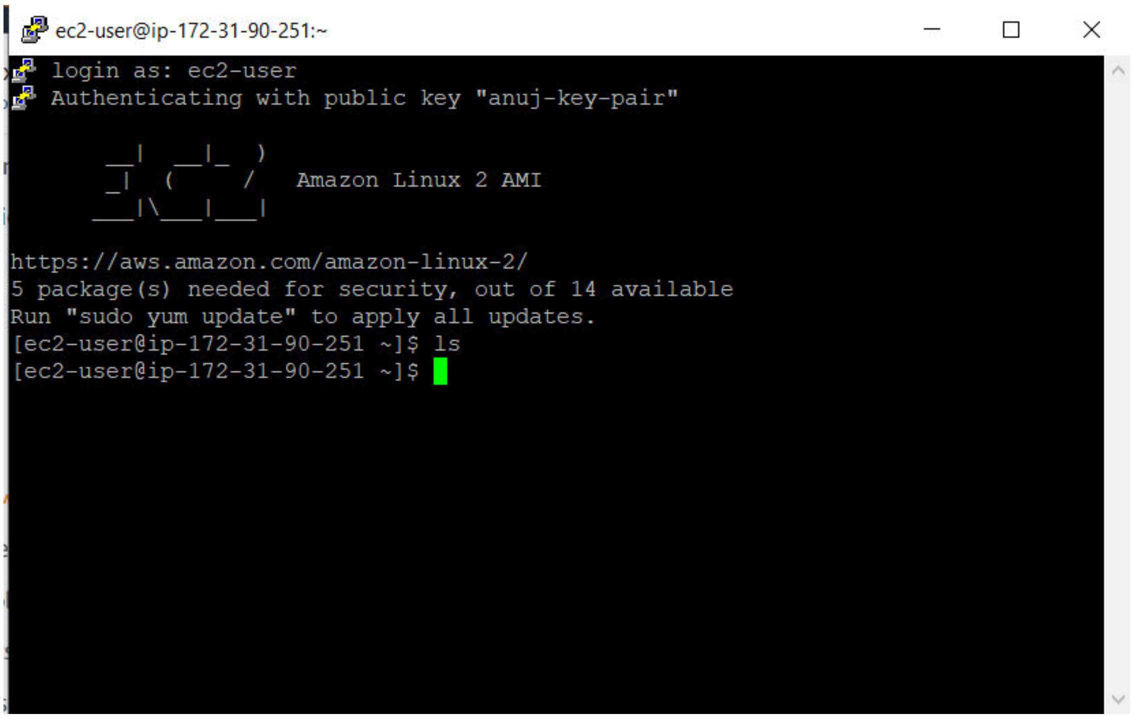
Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-086cc5c96a0655b86 (nodejs)	3.88.223.1 | open address	172.31.90.251

IPv6 address	Instance state	Public IPv4 DNS
–	Running	ec2-3-88-223-1.compute-1.amazonaws.com | open address

Hostname type	Private IP DNS name (IPv4 only)
IP name: ip-172-31-90-251.ec2.internal	ip-172-31-90-251.ec2.internal

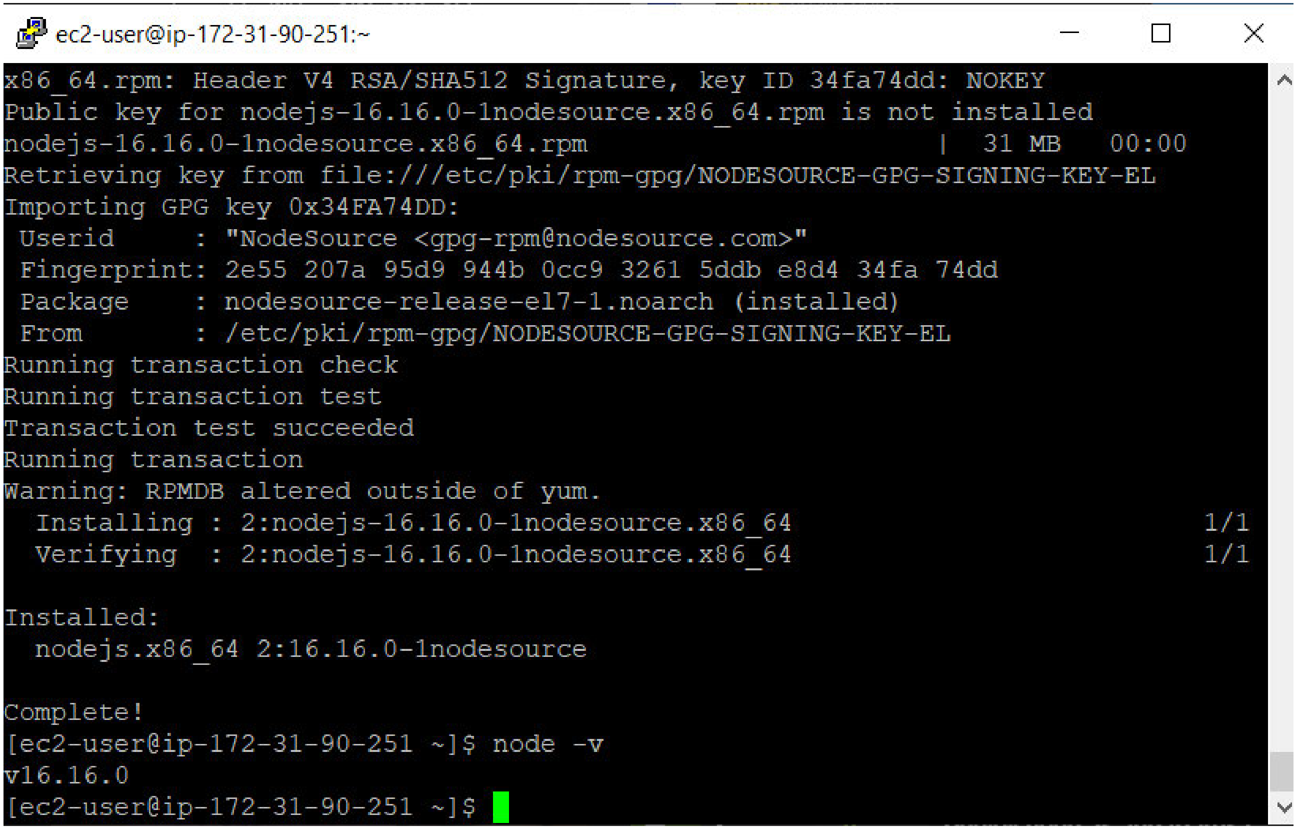
Connected to EC2 Instance



ec2-user@ip-172-31-90-251:~

```
login as: ec2-user
Authenticating with public key "anuj-key-pair"

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

https://aws.amazon.com/amazon-linux-2/
5 package(s) needed for security, out of 14 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-90-251 ~]$ ls
[ec2-user@ip-172-31-90-251 ~]$
```

➔ **Installing Node.JS On EC2 Instance**



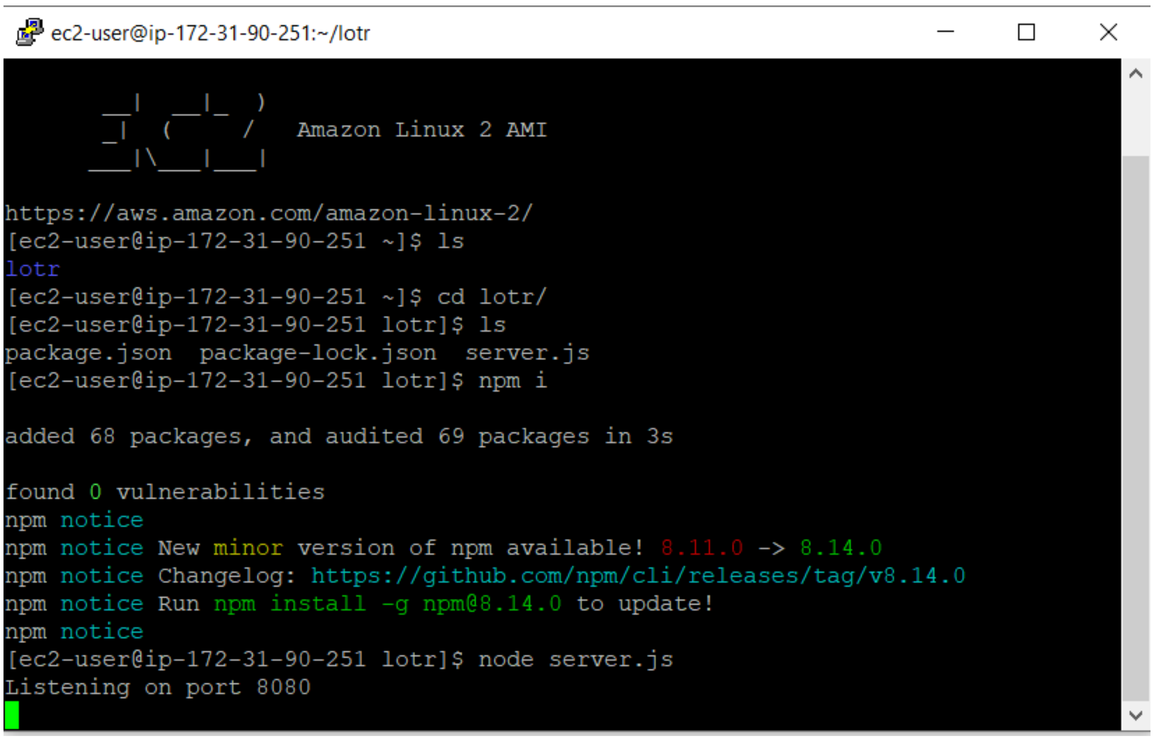
ec2-user@ip-172-31-90-251:~

```
x86_64.rpm: Header V4 RSA/SHA512 Signature, key ID 34fa74dd: NOKEY
Public key for nodejs-16.16.0-1nodesource.x86_64.rpm is not installed
nodejs-16.16.0-1nodesource.x86_64.rpm                      |  31 MB   00:00
Retrieving key from file:///etc/pki/rpm-gpg/NODESOURCE-GPG-SIGNING-KEY-EL
Importing GPG key 0x34FA74DD:
 Userid     : "NodeSource <gpg-rpm@nodesource.com>"
 Fingerprint: 2e55 207a 95d9 944b 0cc9 3261 5ddb e8d4 34fa 74dd
 Package    : nodesource-release-el7-1.noarch (installed)
 From       : /etc/pki/rpm-gpg/NODESOURCE-GPG-SIGNING-KEY-EL
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Warning: RPMDB altered outside of yum.
  Installing : 2:nodejs-16.16.0-1nodesource.x86_64                          1/1
  Verifying  : 2:nodejs-16.16.0-1nodesource.x86_64                          1/1

Installed:
  nodejs.x86_64 2:16.16.0-1nodesource

Complete!
[ec2-user@ip-172-31-90-251 ~]$ node -v
v16.16.0
[ec2-user@ip-172-31-90-251 ~]$
```

➔ Loading The Node.JS App On EC2 Instance

- Pushing the code from our device to **Version Control System (GitHub)**
- Cloning the code from **GitHub**

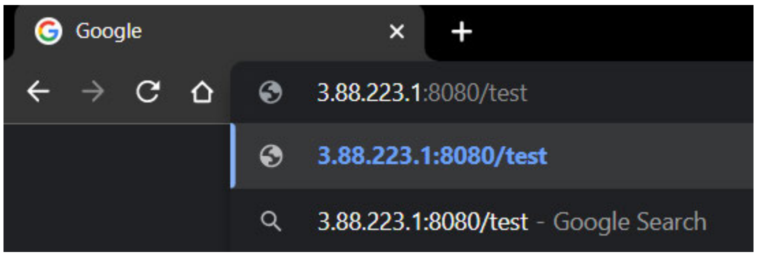
➔ Running the Node.JS Application On "/test route"



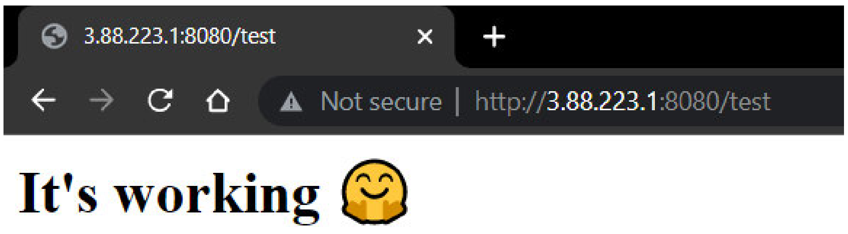
ec2-user@ip-172-31-90-251:~/lotr

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-90-251 ~]$ ls
lotr
[ec2-user@ip-172-31-90-251 ~]$ cd lotr/
[ec2-user@ip-172-31-90-251 lotr]$ ls
package.json  package-lock.json  server.js
[ec2-user@ip-172-31-90-251 lotr]$ npm i

added 68 packages, and audited 69 packages in 3s

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.11.0 -> 8.14.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.14.0
npm notice Run npm install -g npm@8.14.0 to update!
npm notice
[ec2-user@ip-172-31-90-251 lotr]$ node server.js
Listening on port 8080
```

Listening on Port 8080

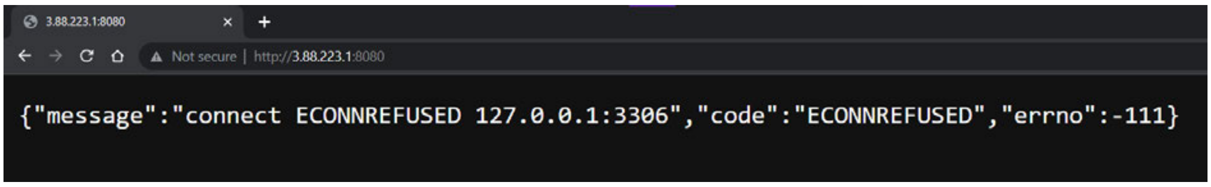


Output: -



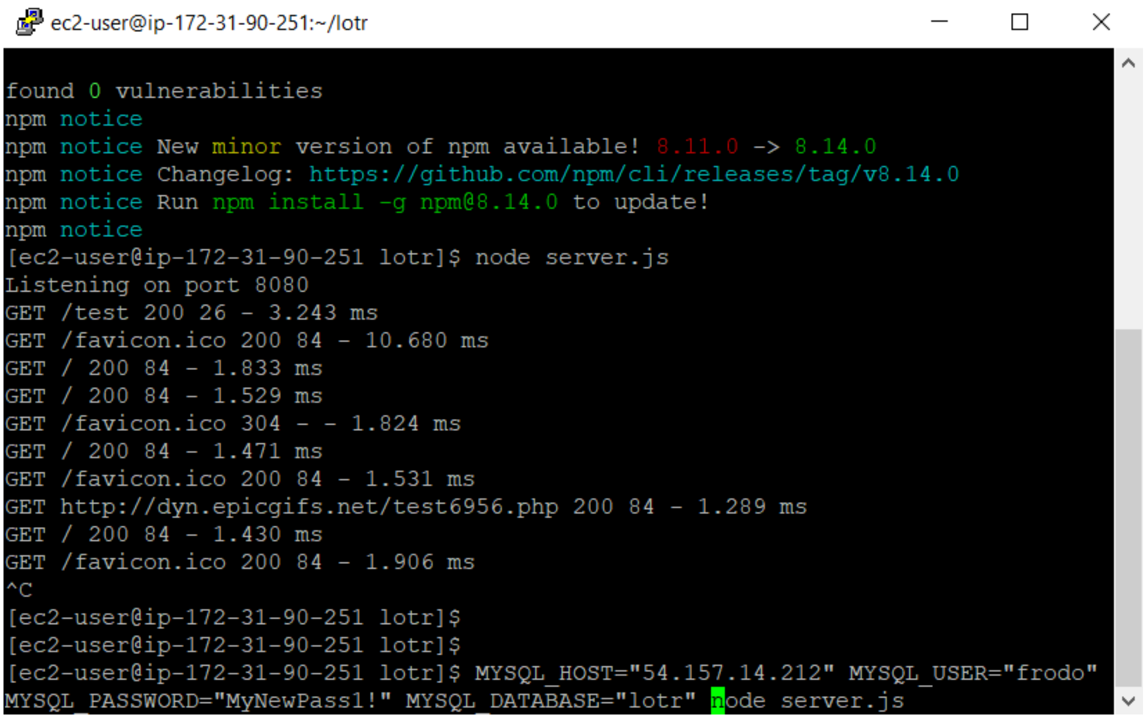
It's working 🤗

→On port 8080 before connecting to database



```
{"message":"connect ECONNREFUSED 127.0.0.1:3306","code":"ECONNREFUSED","errno":-111}
```

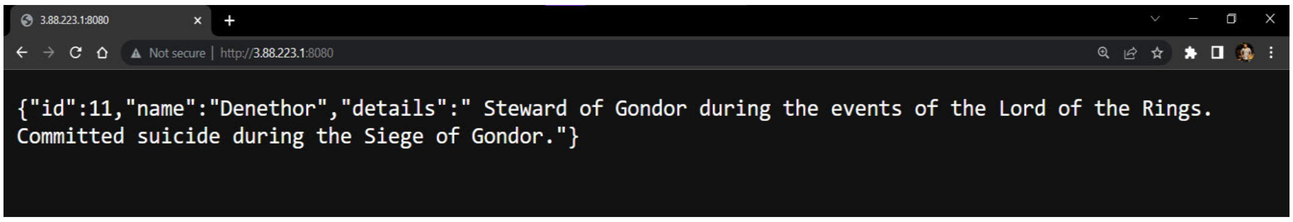
➔ **Defining Environment Variables for Node.JS Server**



```
found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.11.0 -> 8.14.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.14.0
npm notice Run npm install -g npm@8.14.0 to update!
npm notice
[ec2-user@ip-172-31-90-251 lotr]$ node server.js
Listening on port 8080
GET /test 200 26 - 3.243 ms
GET /favicon.ico 200 84 - 10.680 ms
GET / 200 84 - 1.833 ms
GET / 200 84 - 1.529 ms
GET /favicon.ico 304 - - 1.824 ms
GET / 200 84 - 1.471 ms
GET /favicon.ico 200 84 - 1.531 ms
GET http://dyn.epicgifs.net/test6956.php 200 84 - 1.289 ms
GET / 200 84 - 1.430 ms
GET /favicon.ico 200 84 - 1.906 ms
^C
[ec2-user@ip-172-31-90-251 lotr]$
[ec2-user@ip-172-31-90-251 lotr]$
[ec2-user@ip-172-31-90-251 lotr]$ MYSQL_HOST="54.157.14.212" MYSQL_USER="frodo"
MYSQL_PASSWORD="MyNewPass1!" MYSQL_DATABASE="lotr" node server.js
```

➔ **CONNECTED TO DATABASE: -** Database has some Lord of the Rings Characters defined within it. It consists of **LOTR (id, name, details)**. Refreshing the page always generates some new data.

Output: -



```
{"id":11,"name":"Denethor","details":" Steward of Gondor during the events of the Lord of the Rings.
Committed suicide during the Siege of Gondor."}
```

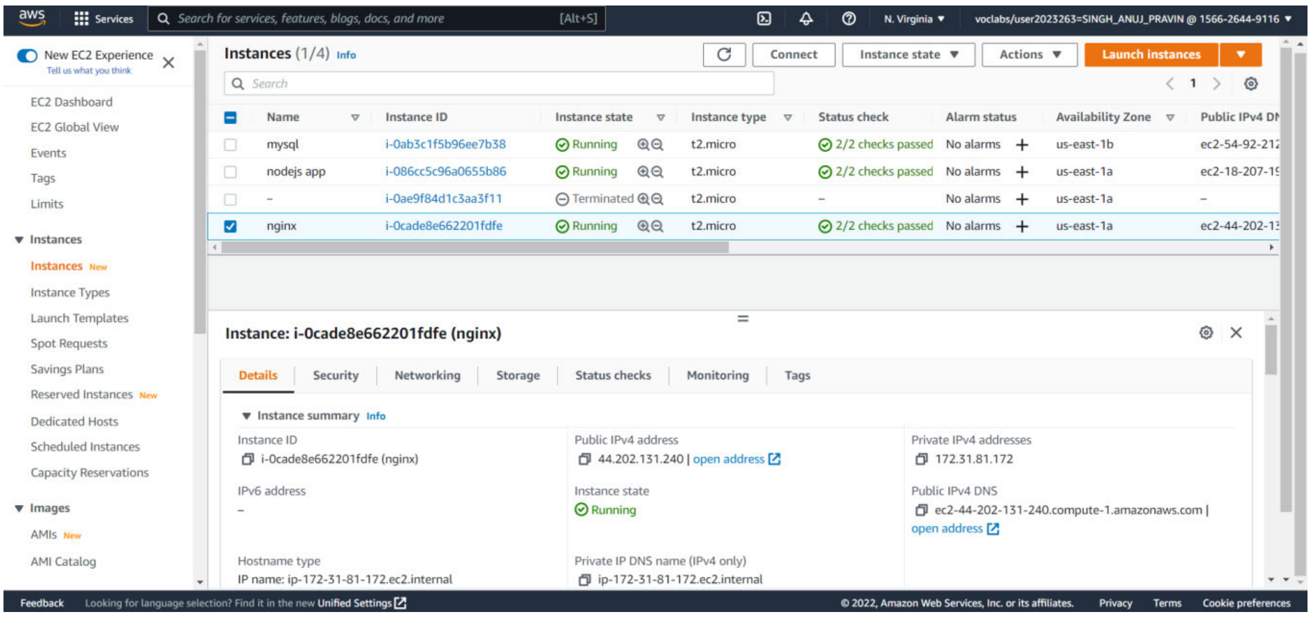
<u>USING NGINX TO SETUP REVERSE PROXY</u>

We are setting up reverse proxy here, so that instead of listening from **Port: - 8080** or some random port, our application will be capable to fetch the result from the default port i.e., **Port: - 80.**

What is NGINX?

NGINX is open-source software for web serving, reverse proxying, caching, load balancing, media streaming, and more. It started out as a web server designed for maximum performance and stability. In addition to its HTTP server capabilities, NGINX can also function as a proxy server for email (IMAP, POP3, and SMTP) and a reverse proxy and load balancer for HTTP, TCP, and UDP servers.

➔ Created EC2 Instance for NGINX



Instances (1/4) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
mysql	i-0ab3c1f5b96ee7b38	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-54-92-21
nodejs app	i-086cc5c96a0655b86	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-18-207-1
–	i-0ae9f84d1c3aa3f11	Terminated	t2.micro	–	No alarms	us-east-1a	–
nginx	i-0cade8e662201fdfe	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-44-202-1

Instance: i-0cade8e662201fdfe (nginx)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

Instance summary Info

Instance ID
i-0cade8e662201fdfe (nginx)

Public IPv4 address
44.202.131.240 | open address

Private IPv4 addresses
172.31.81.172

IPv6 address
–

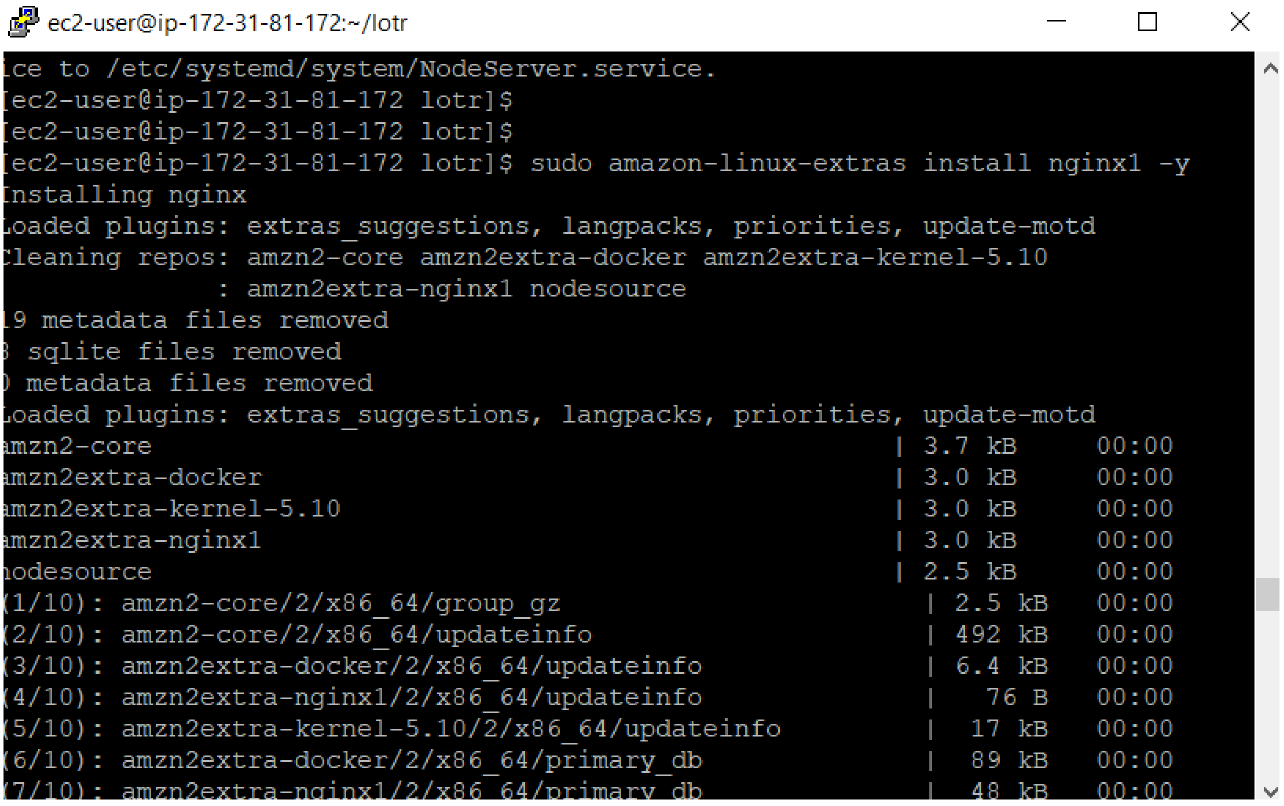
Instance state
Running

Public IPv4 DNS
ec2-44-202-131-240.compute-1.amazonaws.com | open address

Hostname type
IP name: ip-172-31-81-172.ec2.internal

Private IP DNS name (IPv4 only)
ip-172-31-81-172.ec2.internal

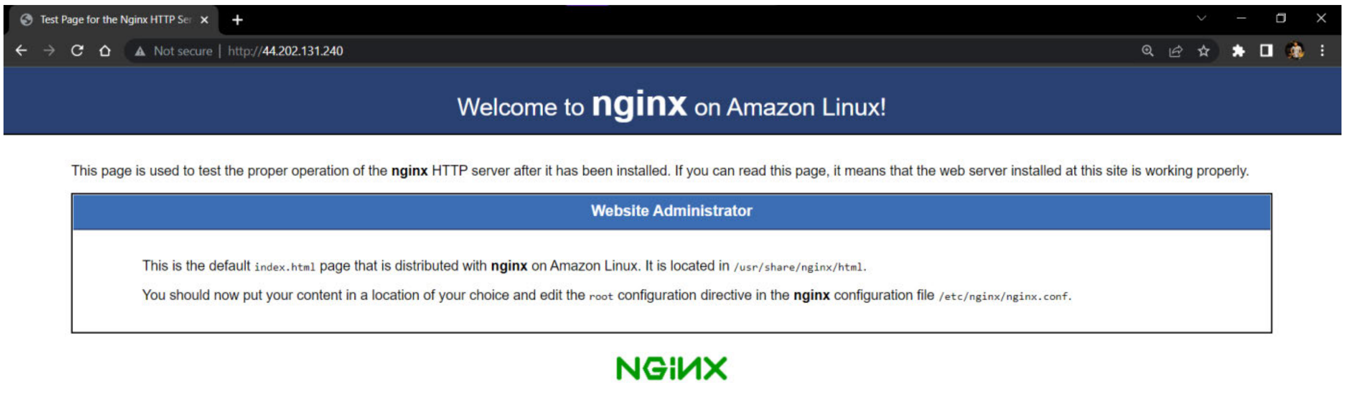
➔ Installing NGINX On EC2 Instance



ec2-user@ip-172-31-81-172:~/lotr

```
ice to /etc/systemd/system/NodeServer.service.
[ec2-user@ip-172-31-81-172 lotr]$
[ec2-user@ip-172-31-81-172 lotr]$
[ec2-user@ip-172-31-81-172 lotr]$ sudo amazon-linux-extras install nginx1 -y
Installing nginx
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10
              : amzn2extra-nginx1 nodesource
19 metadata files removed
8 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                              | 3.7 kB     00:00
amzn2extra-docker                                       | 3.0 kB     00:00
amzn2extra-kernel-5.10                                  | 3.0 kB     00:00
amzn2extra-nginx1                                       | 3.0 kB     00:00
nodesource                                              | 2.5 kB     00:00
(1/10): amzn2-core/2/x86_64/group_gz                      | 2.5 kB   00:00
(2/10): amzn2-core/2/x86_64/updateinfo                    | 492 kB   00:00
(3/10): amzn2extra-docker/2/x86_64/updateinfo             | 6.4 kB   00:00
(4/10): amzn2extra-nginx1/2/x86_64/updateinfo             |   76 B   00:00
(5/10): amzn2extra-kernel-5.10/2/x86_64/updateinfo        |  17 kB   00:00
(6/10): amzn2extra-docker/2/x86_64/primary_db             |  89 kB   00:00
(7/10): amzn2extra-nginx1/2/x86_64/primary_db             |  48 kB   00:00
```

➔ NGINX Installed On Default IP



Test Page for the Nginx HTTP Ser

Not secure | http://44.202.131.240

Welcome to **nginx** on Amazon Linux!

This page is used to test the proper operation of the **nginx** HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.

Website Administrator

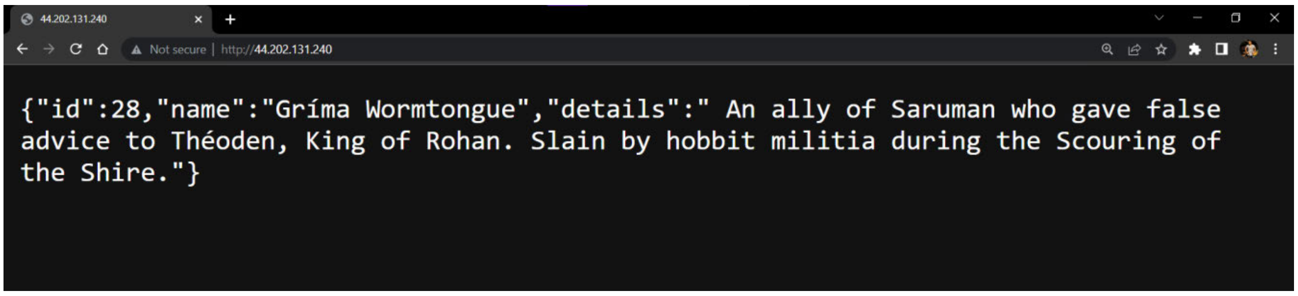
This is the default `index.html` page that is distributed with **nginx** on Amazon Linux. It is located in `/usr/share/nginx/html`.

You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

Now, we will configure NGINX so whenever we visit the default IP Address it will forward our request to our application instance running on port 8080

6. Output:

➔ **Node.JS Application Running On Default IP (through Reverse Proxy)**



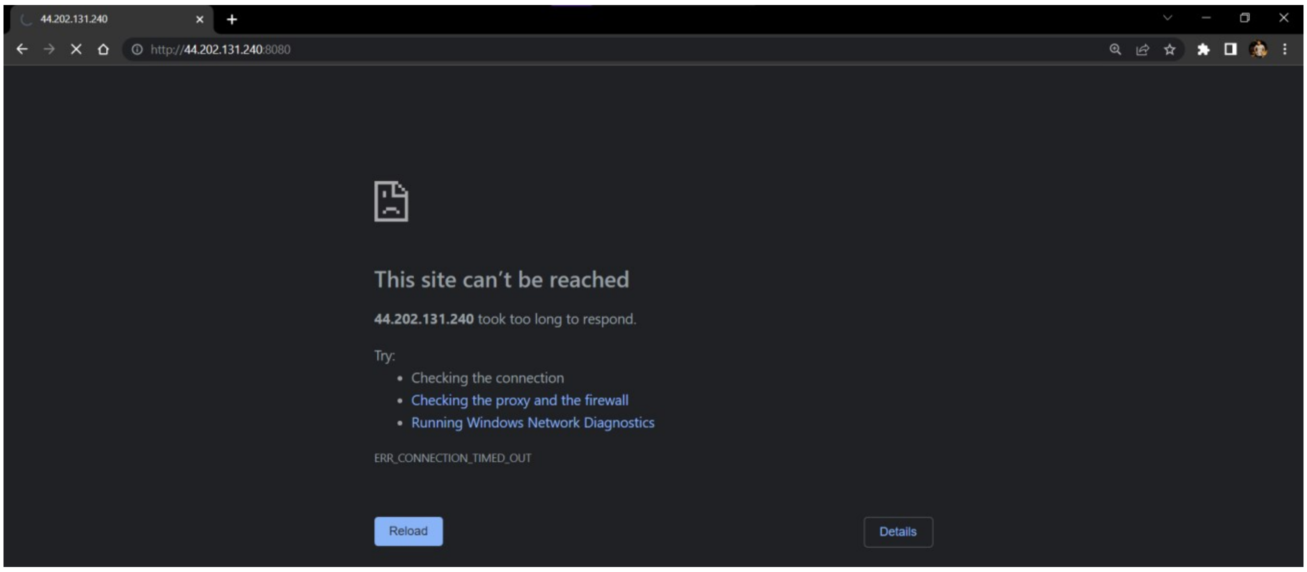
44.202.131.240

Not secure | http://44.202.131.240

```
{"id":28,"name":"Gríma Wormtongue","details":" An ally of Saruman who gave false advice to Théoden, King of Rohan. Slain by hobbit militia during the Scouring of the Shire."}
```

➔ **Configured Security Groups Allowing Access Only Through Port 80 and Port 22 SSH Client**

(Error while connecting through Port 8080)



44.202.131.240

http://44.202.131.240:8080

This site can't be reached

44.202.131.240 took too long to respond.

Try:

- Checking the connection
- Checking the proxy and the firewall
- Running Windows Network Diagnostics

ERR_CONNECTION_TIMED_OUT

7. Conclusion:

Implemented MySQL Database Server for a Node.JS application and then configured it to run on a reverse proxy using NGNIX.

References

- P. Wurzinger, C. Platzer, C. Ludl, E. Kirda and C. Kruegel, "SWAP: Mitigating XSS attacks using a reverse proxy," 2009 ICSE Workshop on Software Engineering for Secure Systems, 2009, pp. 33-39, doi: 10.1109/IWSESS.2009.5068456.

- Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Engin Kirda. 2006. An anomaly-driven reverse proxy for web applications. In Proceedings of the 2006 ACM symposium on Applied computing (SAC '06). Association for Computing Machinery, New York, NY, USA, 361–368.

- R. A. Muzaki, O. C. Briliyant, M. A. Hasditama and H. Ritchi, "Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall," 2020 International Workshop on Big Data and Information Security (IWBIS), 2020, pp. 85-90, doi: 10.1109/IWBIS50925.2020.9255601.

- Noviyanto, A. B., Erna, K. & Amir, H., 2015. PERANCANGAN DAN IMPLEMENTASI LOAD BALANCING REVERSE PROXY MENGGUNAKAN HAPROXY PADA APLIKASI WEB. Jurnal JARKOM ISBN:2338-6313, Volume III.

- Cloudflare.com

- NGINX.com

- AWS.Amazon.com