A CLOUD COMPUTING PROJECT

on

CLOUD - BASED WEB APPLICATION FOR UNIVERSITY DATA

By

Kavya Vuribindi
rw4674
kvuribindi@horizon.csueastbay.edu
California State University East Bay, Hayward
Department of Computer Science

and

Indu Paili
tp1739
ipaili@horizon.csueastbay.edu
California State University East Bay, Hayward
Department of Computer Science

Table of Contents

CHAPTER 1: INTRODUCTION

1.1 Problem Summary

In the contemporary world, with the advent of remote work, access to data from anywhere has become a necessity for universities in particular. The traditional approach of storing university data on-premises is outdated and needs a very efficient method of storing, accessing, and updating student data. In addition, maintaining an on-premise server is costly and is neither reliable nor scalable. Furthermore, universities need more skilled personnel to maintain its infrastructure and maintain student information security. In addition, once a server is owned, we cannot scale up or scale down if the needs change. Investing on resources like hardware, servers upfront is not a good idea in most of the scenarios. Also, year by year, as the amount of data increases and data retrieval will slow down with the on-premise data storage approach. So, it is crucial to have a system that provides a faster and efficient data management model for universities.

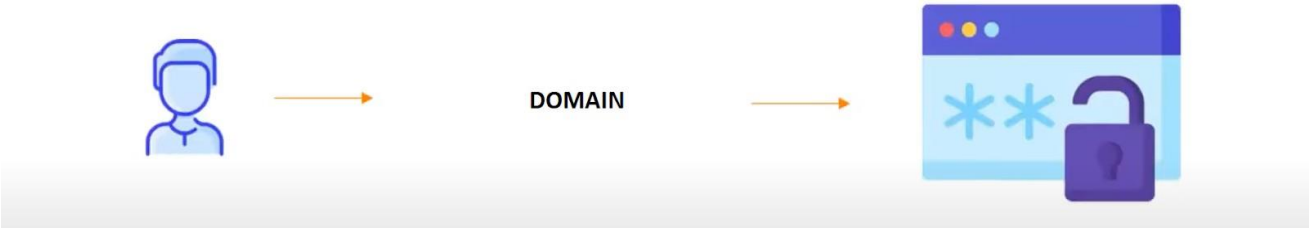
1.2 Solution

Cloud computing technology is used to resolve all the problems discussed above. With cloud-based storage, University administration can store files to a remote database instead of a local storage device. In most cases, an online university management system is being considered a safer alternative to the existing physical system since it is convenient and secure. Any user with the relevant credentials can access data stored in the cloud. There is no limit for the amount of data that can be stored and hence, this is an ideal solution. University data holds a lot of sensitive information, so any security breach can cause a lot of trouble. Cloud technology, however, eliminates such concerns when dealing with leading service providers, such as Amazon, GCP.

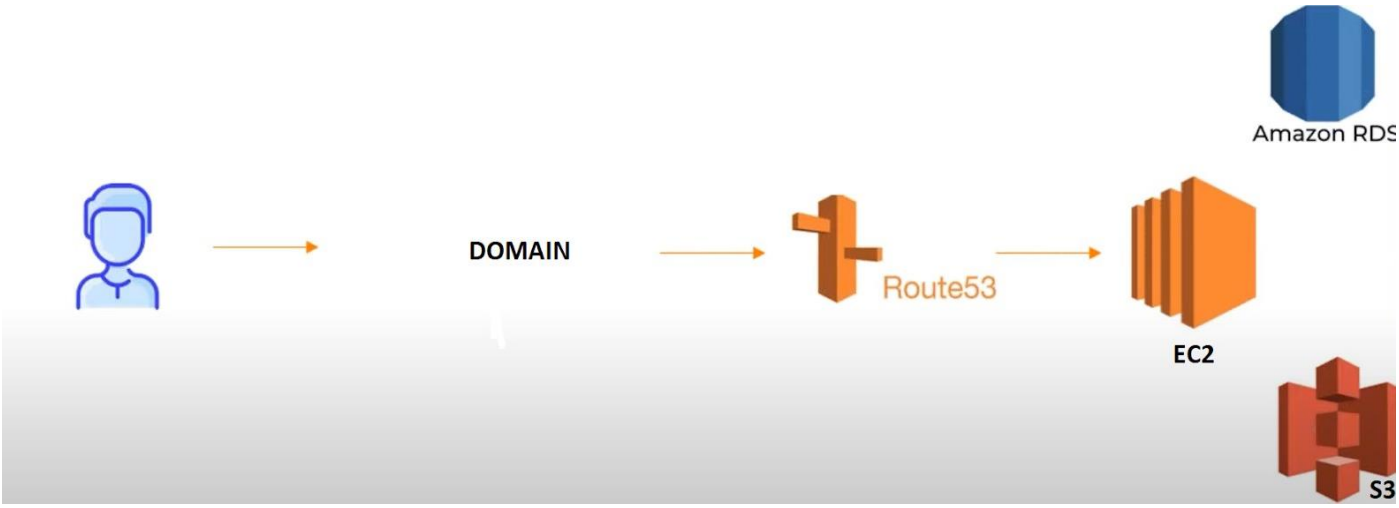
This will be implemented as a web application on AWS wherein application has functionalities for admin to login and perform CRUD operations i.e, Create, Retrieve, Update or Delete on student data as well as analyzing the stored data. Student data comprises of information such as First Name, Last Name, Net ID, Major, Location etc. Admin will be able to login to view or modify their personal information. On the AWS Cloud Platform, this web application uses EC2 to launch instances, S3 buckets to store data, such as images, and Relational Database Services to access databases. This project is about deploying web applications on Amazon EC2 instance that serves as Infrastructure as a Service (laaS). In the web application, we will be using search function to retrieve student information very quickly and efficiently. Overall, the system architecture can be broken down into three components i.e, Selecting and Configuring Amazon Services, Building Client-Side Application, Routing Domain to EC2 server.

To implement this system, we use a variety of AWS services including RDS, S3, EC2 and Python for programming. There are several benefits to implementing this system in the cloud. To name a few, the S3 service offers many advantages over traditional storage solutions, such as reliability, scalability, and affordability. By hosting on S3, we can leverage Scalability, Availability, Durability and Security offered by S3. Furthermore, configuring an EC2 server is easier than configuring a physical server, which requires a lot of effort. Scaling up or scaling down the resources is easy. When not needed, we can detach the resources and thus pay less. The S3 as well as EC2 pricing are pay-per-use, meaning that we only charge for what we use. To conclude, the web application will make administrators and university students lives much easier and more convenient.

CHAPTER 2: IMPLEMENTATION WORKFLOW



OVERVIEW DIAGRAM



WORKFLOW DIAGRAM

2.1 Architectural Overview

The following sections give the Architecture of the system and Relation between the services.

2.1.1 Architecture of the system

The overall architecture of the system can be roughly divided into three components:

1. Selecting and Configuring Amazon Services
2. Building client-side application
3. Routing Domain to EC2 server

In selecting and configuring Amazon services we are using three AWS namely -

1. RDS
2. S3
3. EC2

We need to select these services one by one from AWS dashboard and click appropriate options to set it up.

In building client-side applications, we are using Python and Html for the coding part.

In routing domain to EC2 server we need to use another Amazon service called Route 53, which is used for routing purposes and next we should purchase domain names and then create hosted zones and configure them.

2.1.2 Relation between services

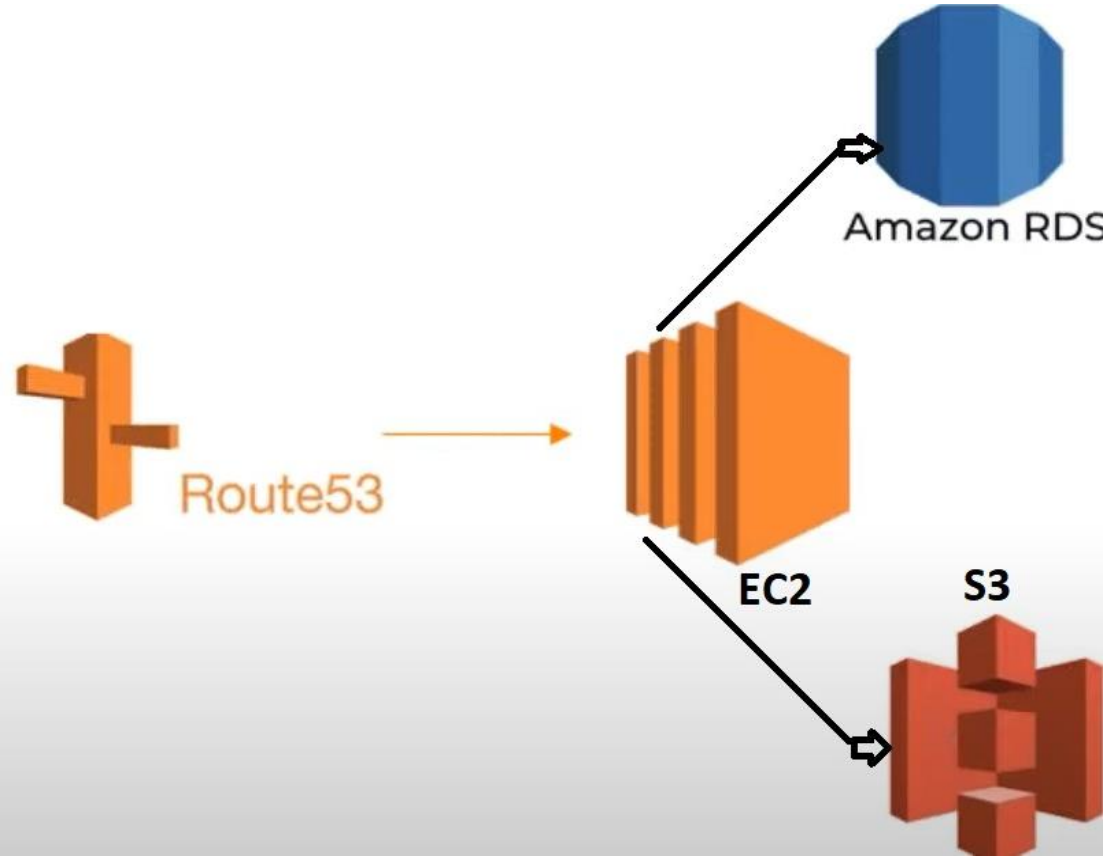
- We are using **Amazon RDS service**, which is used to operate and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. In this we are using MySQL database, and this is used to store the entries. This RDS service will be connected by the EC2 server, this relation is used to access the database.

- We are using **Amazon S3 service**, which is a Storage for the internet. We can use it to store and retrieve any amount of data at any time, from anywhere on the web. We should give EC2 instance access to S3, basically to upload the data. To make this relation we will give an IAM role to our instance.

- We are using **Amazon EC2 service**, it is a web service for launching and managing

Linux/UNIX and Windows Server instances in Amazon's data centers. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment. Here EC2 should be connected to RDS, S3 as described above, and the website is also deployed in EC2 server.

- We are using an **IAM service**, which is used when we connect with our RDS. We should give authentication to our S3 service, so we need to provide an IAM role for our instance. Then we are using Route 53 service which is used to route domain to EC2 server.



RELATION DIAGRAM

CHAPTER 3: DEVELOPMENT PROCESS

The description of the development process and challenges we faced during this project building are discussed below

3.1 Description of the Development Process

I. The first phase is to select and configure the Amazon Services.

We are using three Amazon services initially,

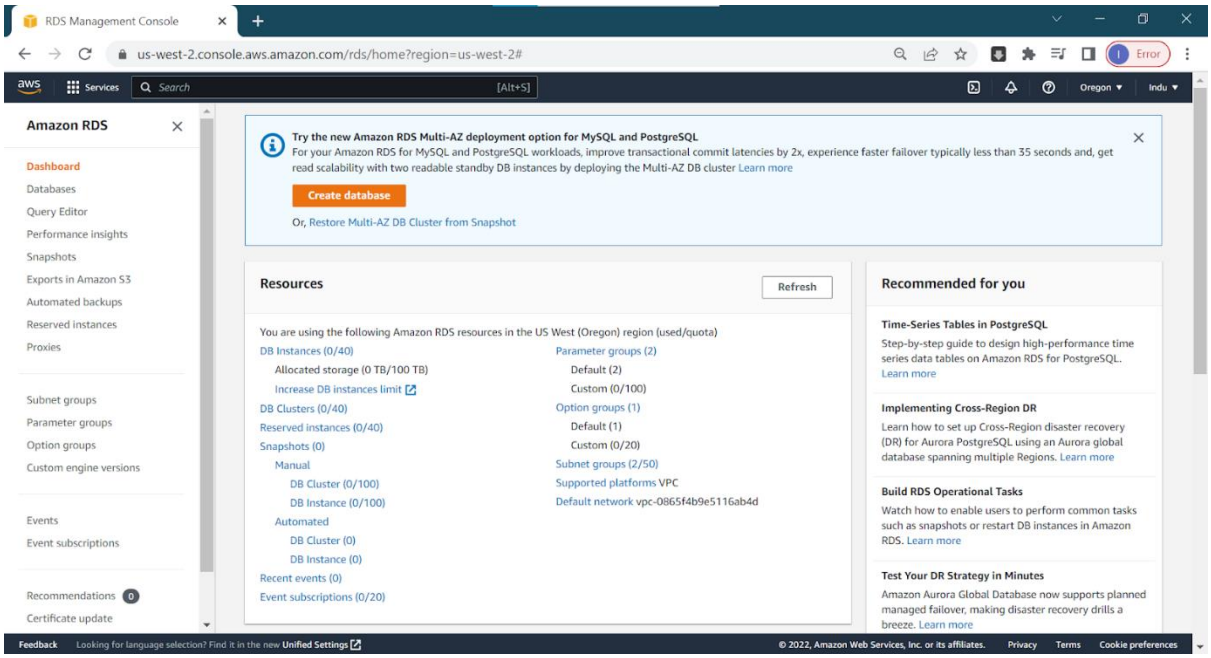
1. RDS
2. S3
3. EC2

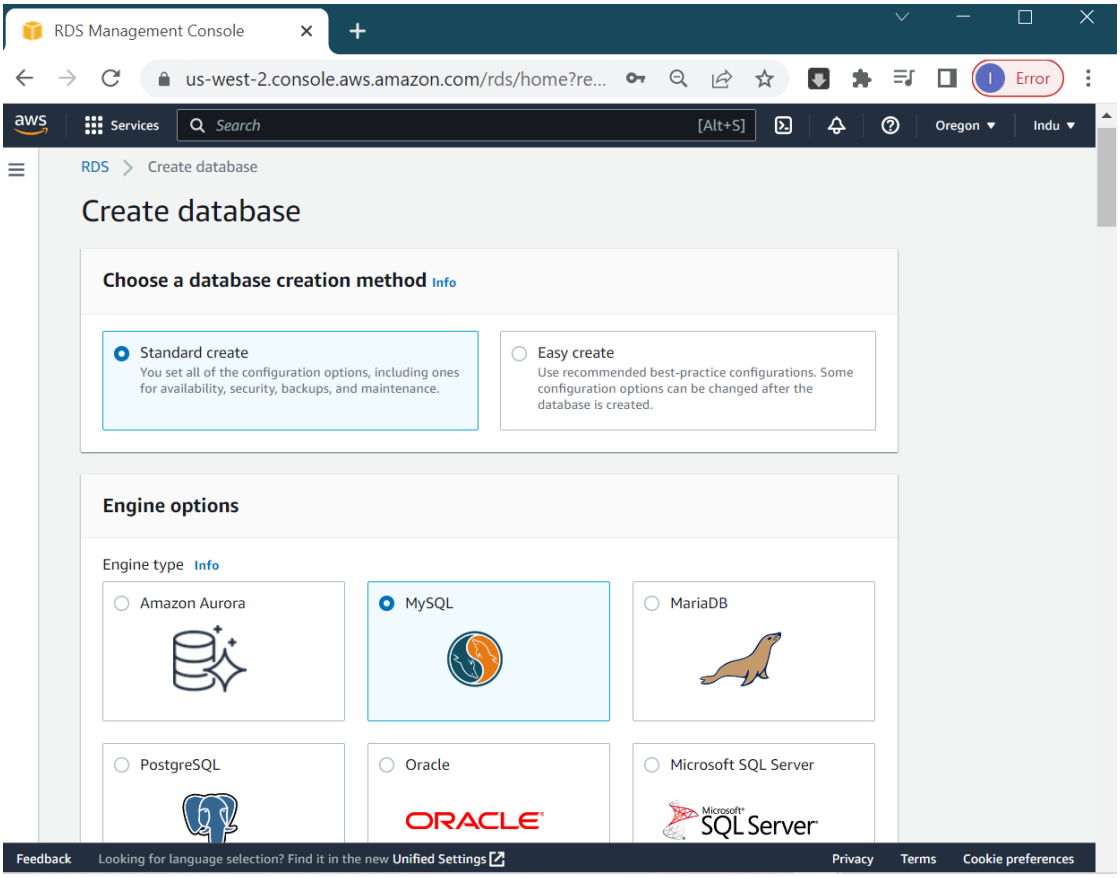
- **Deploy MySQL database on AWS**

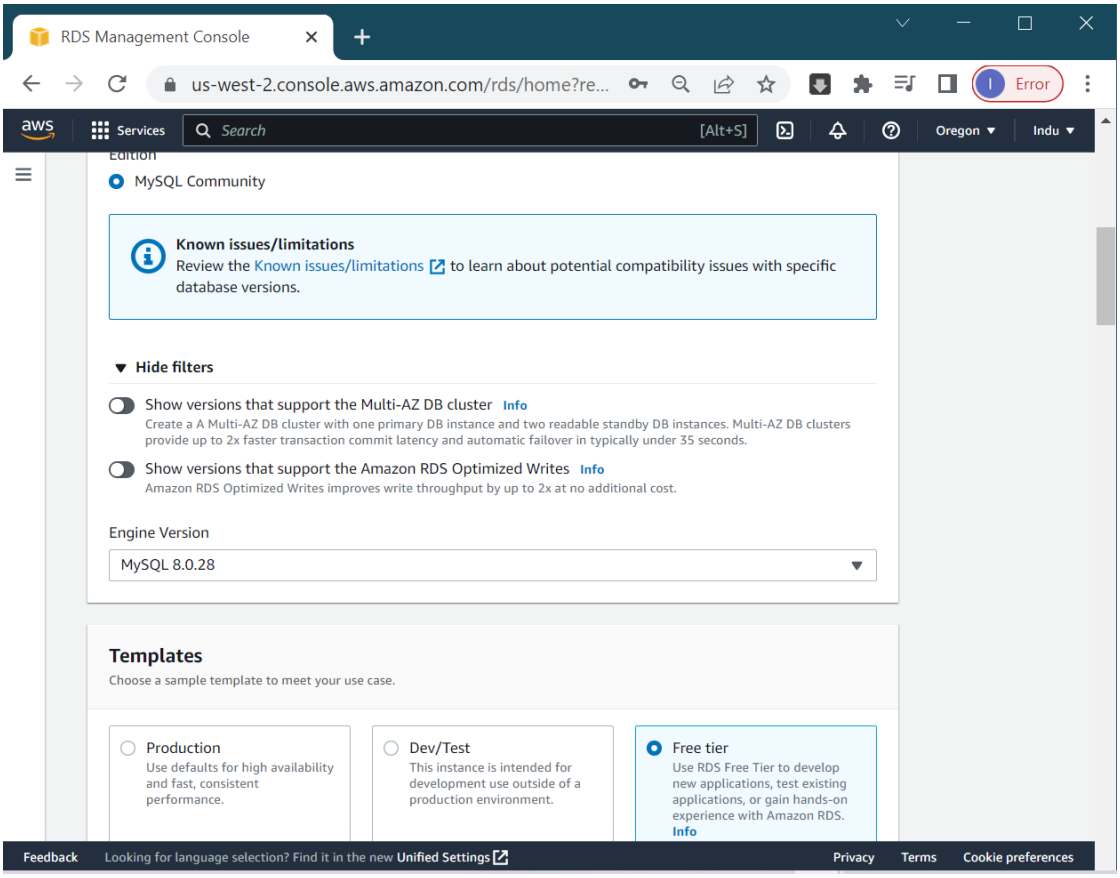
It is done by using Amazon RDS. For this, we should select RDS service from AWS management console dashboard and then select MySQL database engine. There are many engines for this project. We are using MySQL engine.

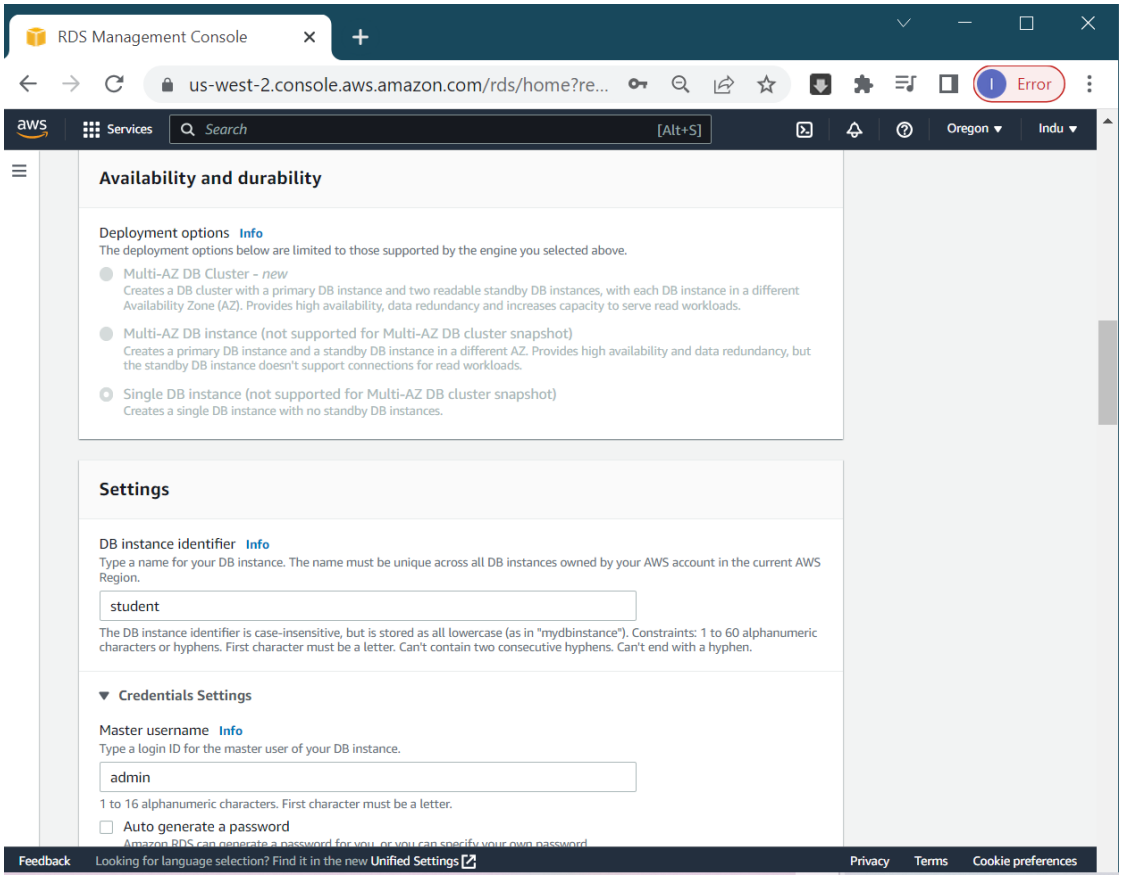
There are many advantages of using RDS, we don't need to launch a server, install MySQL on it and configure it manually. Instead of that we are just using AWS dashboard and select right options which will be managed automatically by AWS.

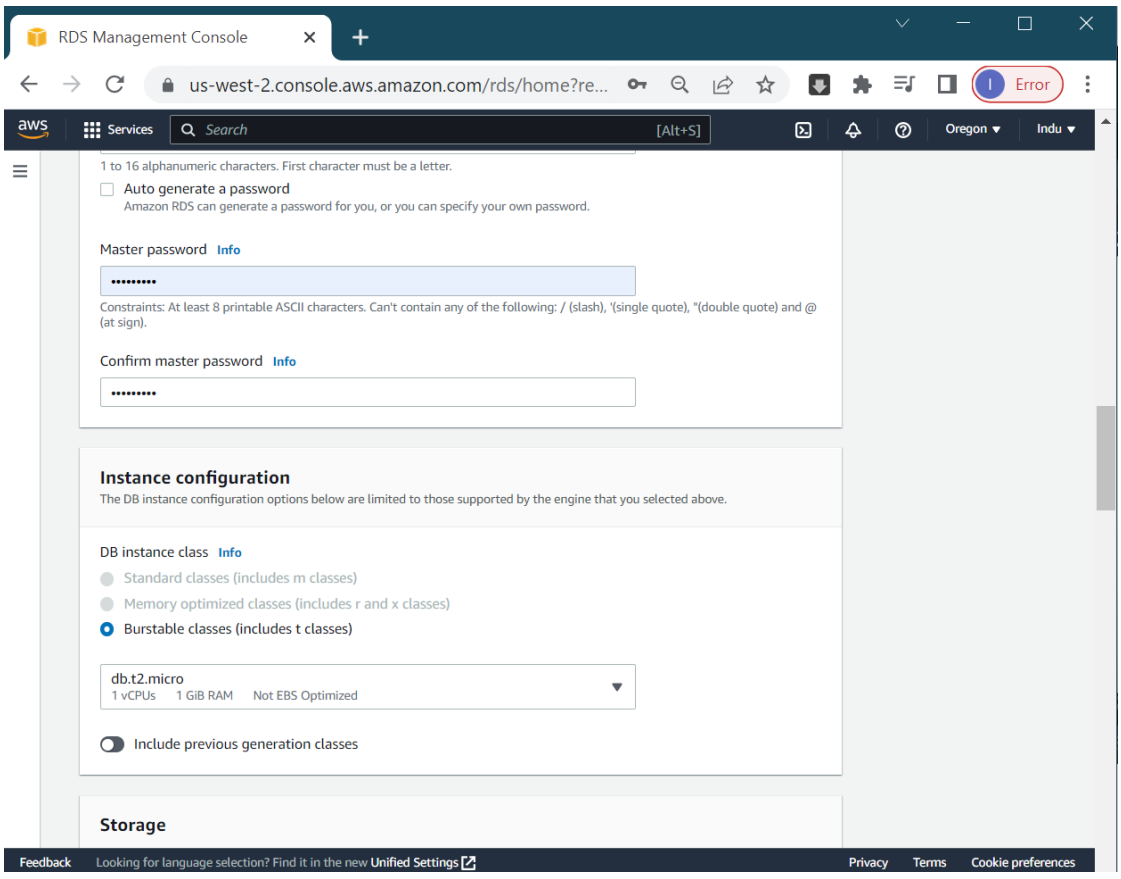
Some key advantages are easy to administer, highly scalable, available & durable, fast, secure, and inexpensive.

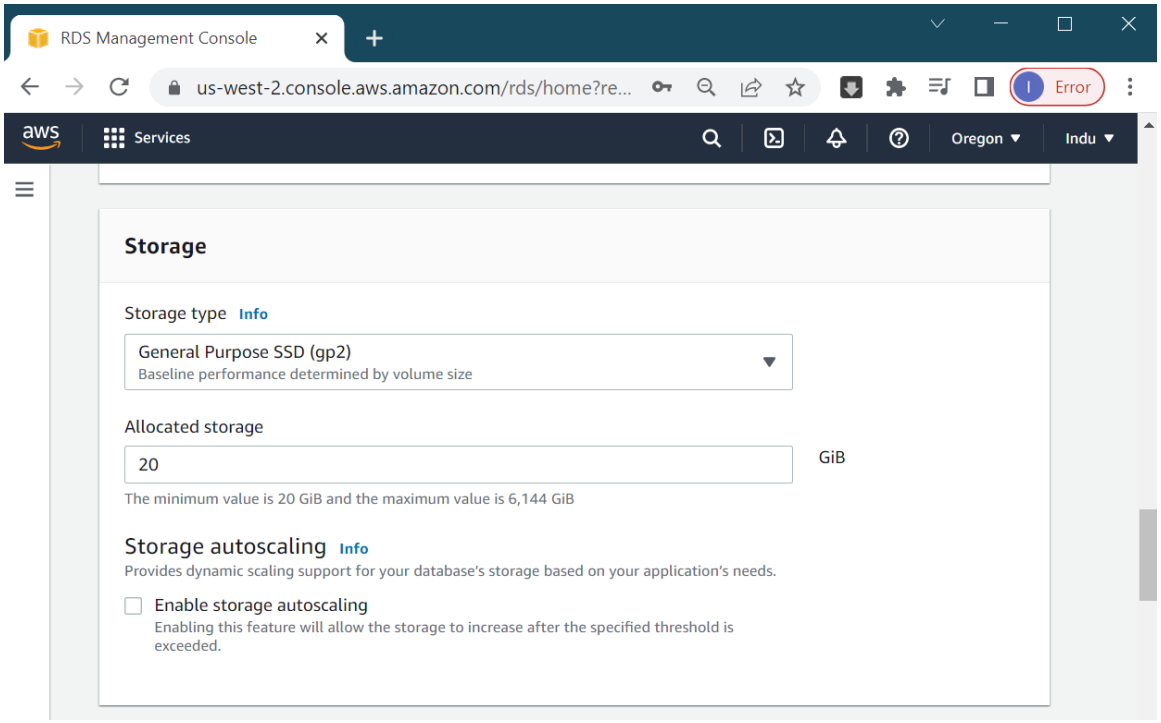


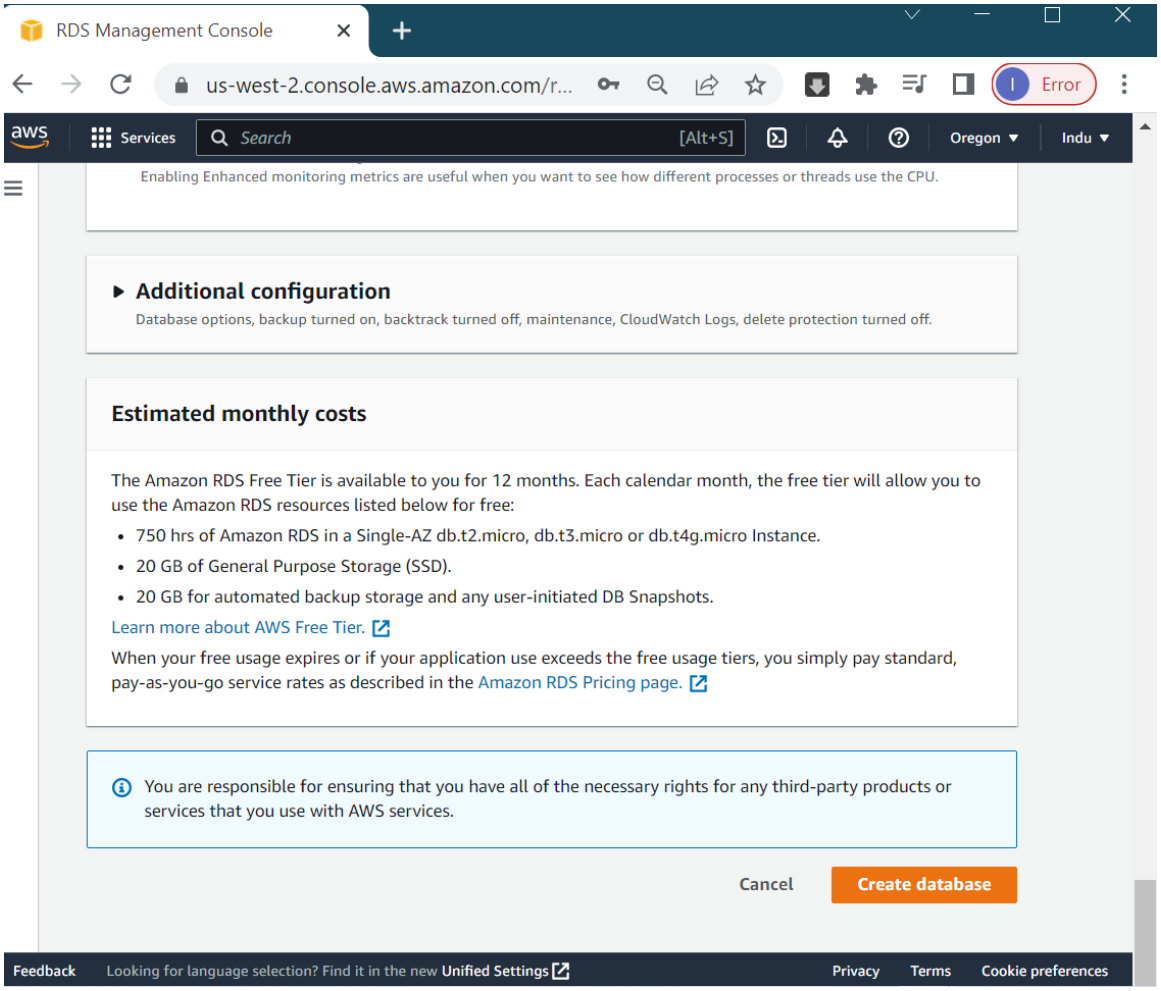

RDS Management Console
us-west-2.console.aws.amazon.com/rds/home?re...
Error
aws
Services
Search
[Alt+S]
Oregon
Indu
RDS
Create database
Create database
Choose a database creation method Info
Standard create
You set all of the configuration options, including ones for availability, security, backups, 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
MariaDB
PostgreSQL
Oracle
Microsoft SQL Server
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences

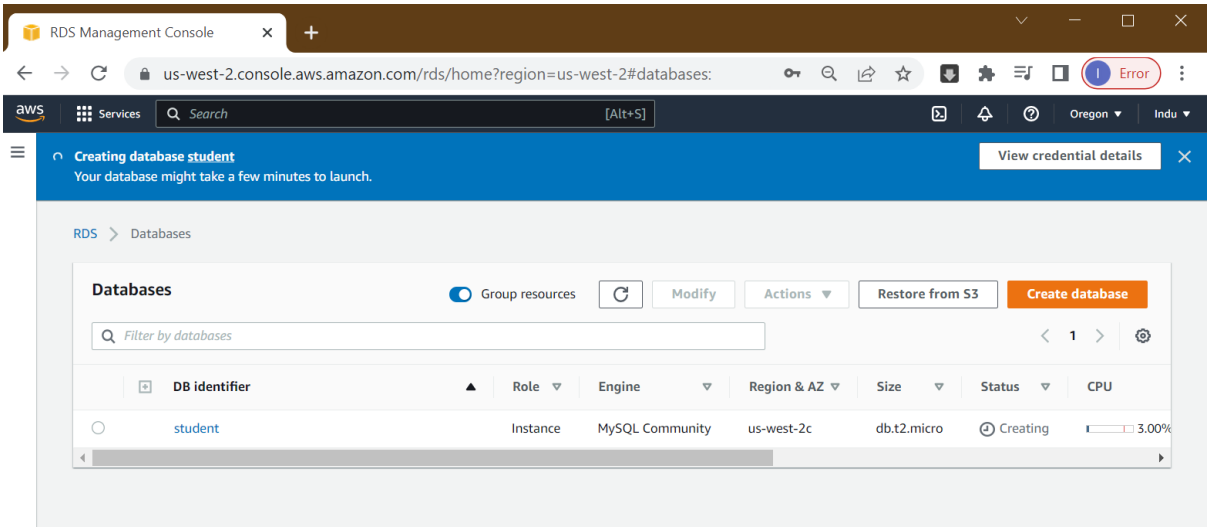

RDS Management Console
us-west-2.console.aws.amazon.com/rds/home?re...
Error
aws
Services
Search
[Alt+S]
Oregon
Indu
Edition
MySQL Community
Known issues/limitations
Review the Known issues/limitations to learn about potential compatibility issues with specific database versions.
Hide filters
Show versions that support the Multi-AZ DB cluster Info
Create a A Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.
Show versions that support the Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.
Engine Version
MySQL 8.0.28
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
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences


RDS Management Console
us-west-2.console.aws.amazon.com/rds/home?re...
Error
aws
Services
Search
[Alt+S]
Oregon
Availability and durability
Deployment options Info
The deployment options below are limited to those supported by the engine you selected above.
Multi-AZ DB Cluster - new
Creates a DB cluster with a primary DB instance and two readable standby DB instances, with each DB instance in a different Availability Zone (AZ). Provides high availability, data redundancy and increases capacity to serve read workloads.
Multi-AZ DB instance (not supported for Multi-AZ DB cluster snapshot)
Creates a primary DB instance and a standby DB instance in a different AZ. Provides high availability and data redundancy, but the standby DB instance doesn't support connections for read workloads.
Single DB instance (not supported for Multi-AZ DB cluster snapshot)
Creates a single DB instance with no standby DB instances.
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.
student
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
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences


RDS Management Console
us-west-2.console.aws.amazon.com/rds/home?re...
Error
aws
Services
Search
[Alt+S]
Oregon
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 master password Info
Instance configuration
The DB instance configuration options below are limited to those supported by the engine that you selected above.
DB instance class Info
Standard classes (includes m classes)
Memory optimized classes (includes r and x classes)
Burstable classes (includes t classes)
db.t2.micro
1 vCPUs 1 GiB RAM Not EBS Optimized
Include previous generation classes
Storage
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences


RDS Management Console
us-west-2.console.aws.amazon.com/rds/home?re...
Error
aws
Services
Oregon
Indu
Storage
Storage type Info
General Purpose SSD (gp2)
Baseline performance determined by volume size
Allocated storage
20
GiB
The minimum value is 20 GiB and the maximum value is 6,144 GiB
Storage autoscaling Info
Provides dynamic scaling support for your database's storage based on your application's needs.
Enable storage autoscaling
Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

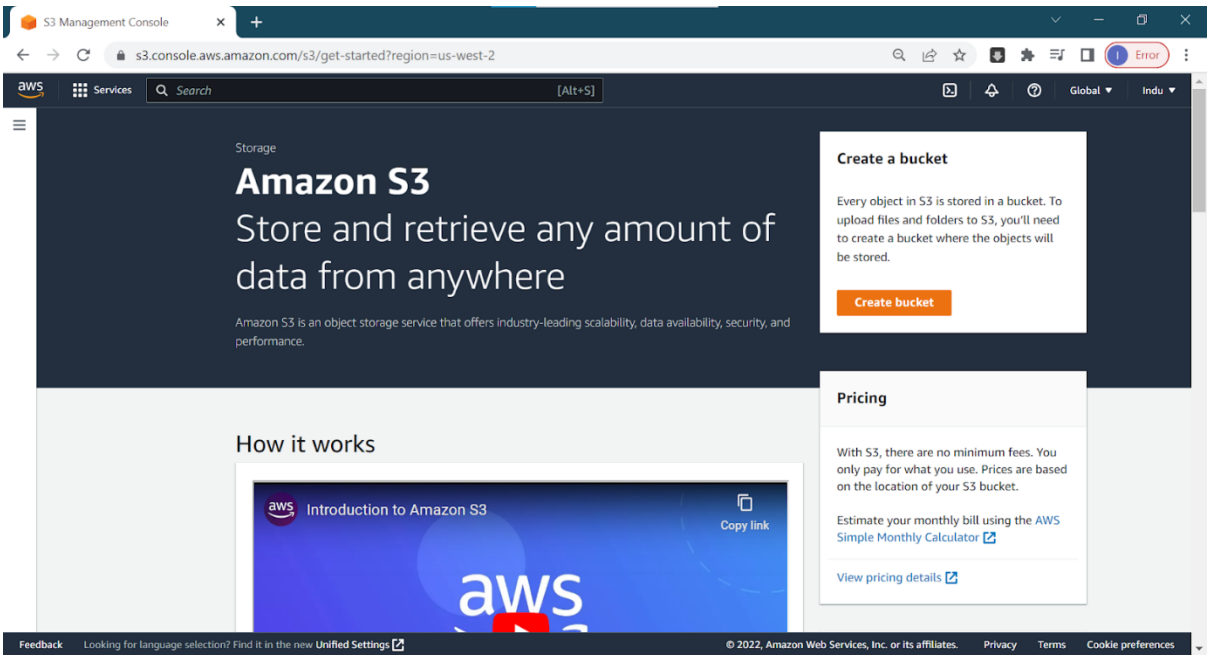

RDS Management Console
us-west-2.console.aws.amazon.com/r...
Error
aws
Services
Search
[Alt+S]
Oregon
Indu
Enabling Enhanced monitoring metrics are useful when you want to see how different processes or threads use the CPU.
Additional configuration
Database options, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.
Estimated monthly costs
The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:
750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance.
20 GB of General Purpose Storage (SSD).
20 GB for automated backup storage and any user-initiated DB Snapshots.
Learn more about AWS Free Tier.
When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the Amazon RDS Pricing page.
You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.
Cancel
Create database
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences

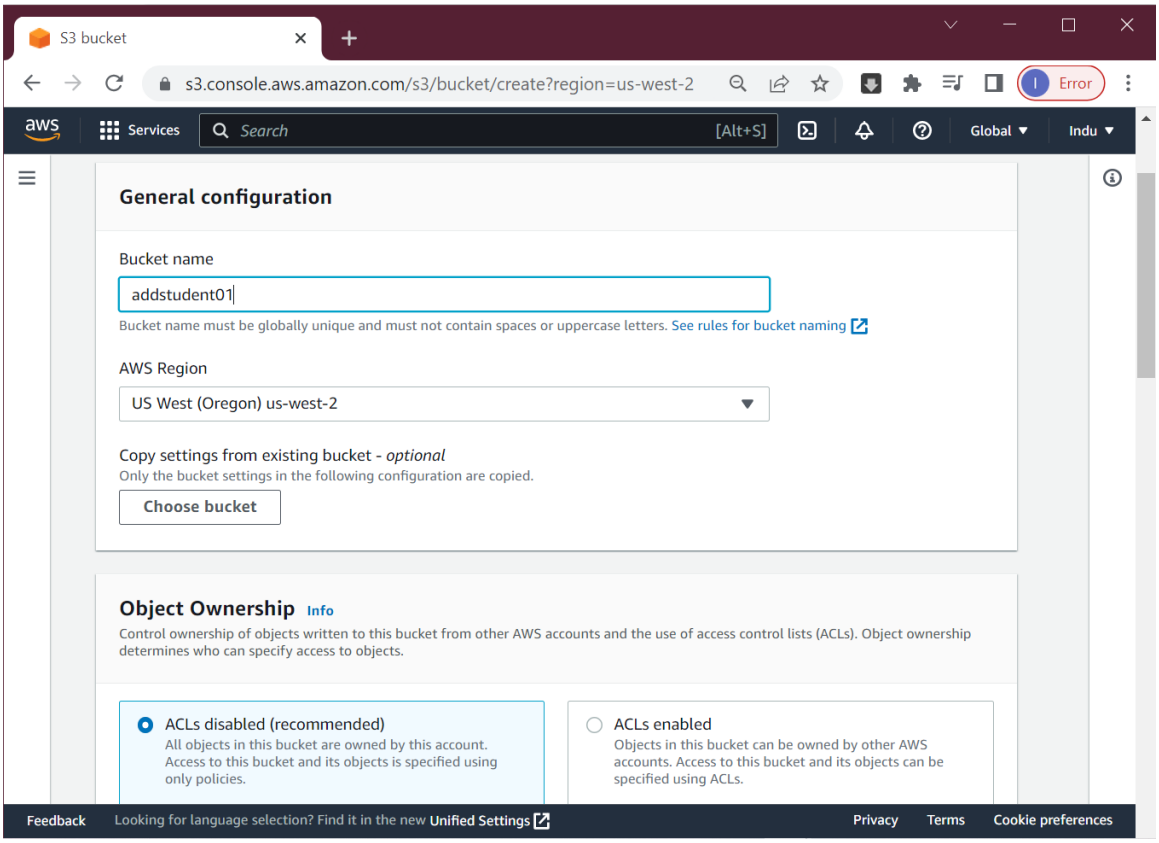


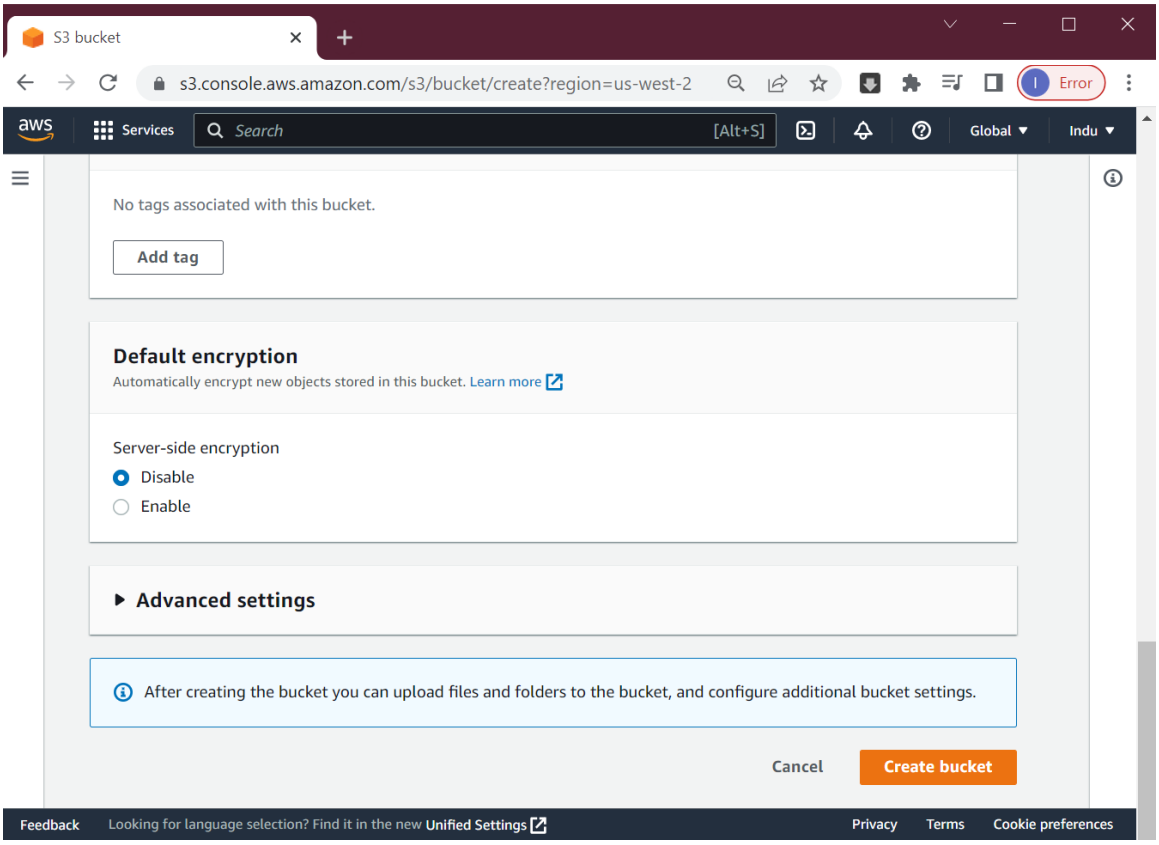
- **Create S3 bucket**

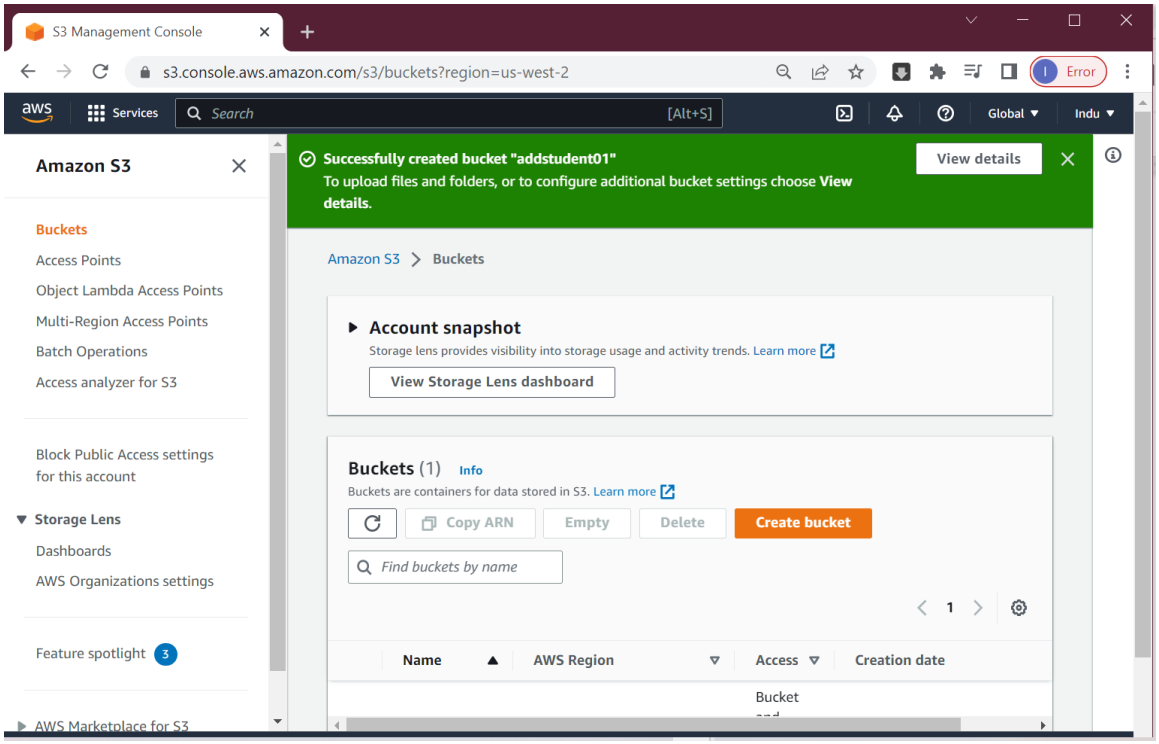
It is done by using Amazon S3, for this we should select S3 service from AWS management console dashboard and create it. Check whether your bucket has been deployed in the correct region.

Some key advantages are Cost effective storage classes, easily managed data and access controls, query-in-place and process on-request, performance, scalability, availability, and durability.



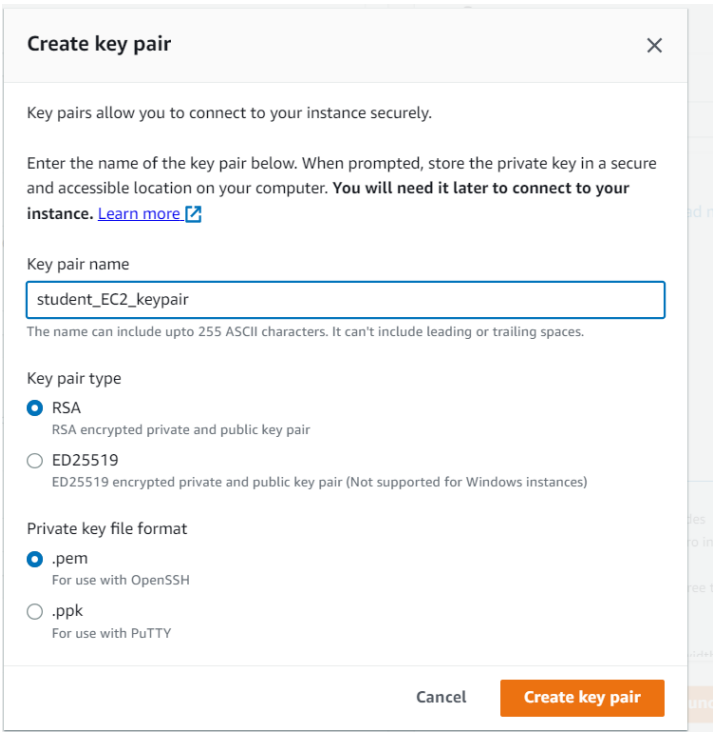

S3 bucket
s3.console.aws.amazon.com/s3/bucket/create?region=us-west-2
Error
aws
Services
Search
[Alt+S]
Global
Indu
General configuration
Bucket name
addstudent01
Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming
AWS Region
US West (Oregon) us-west-2
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.
ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences

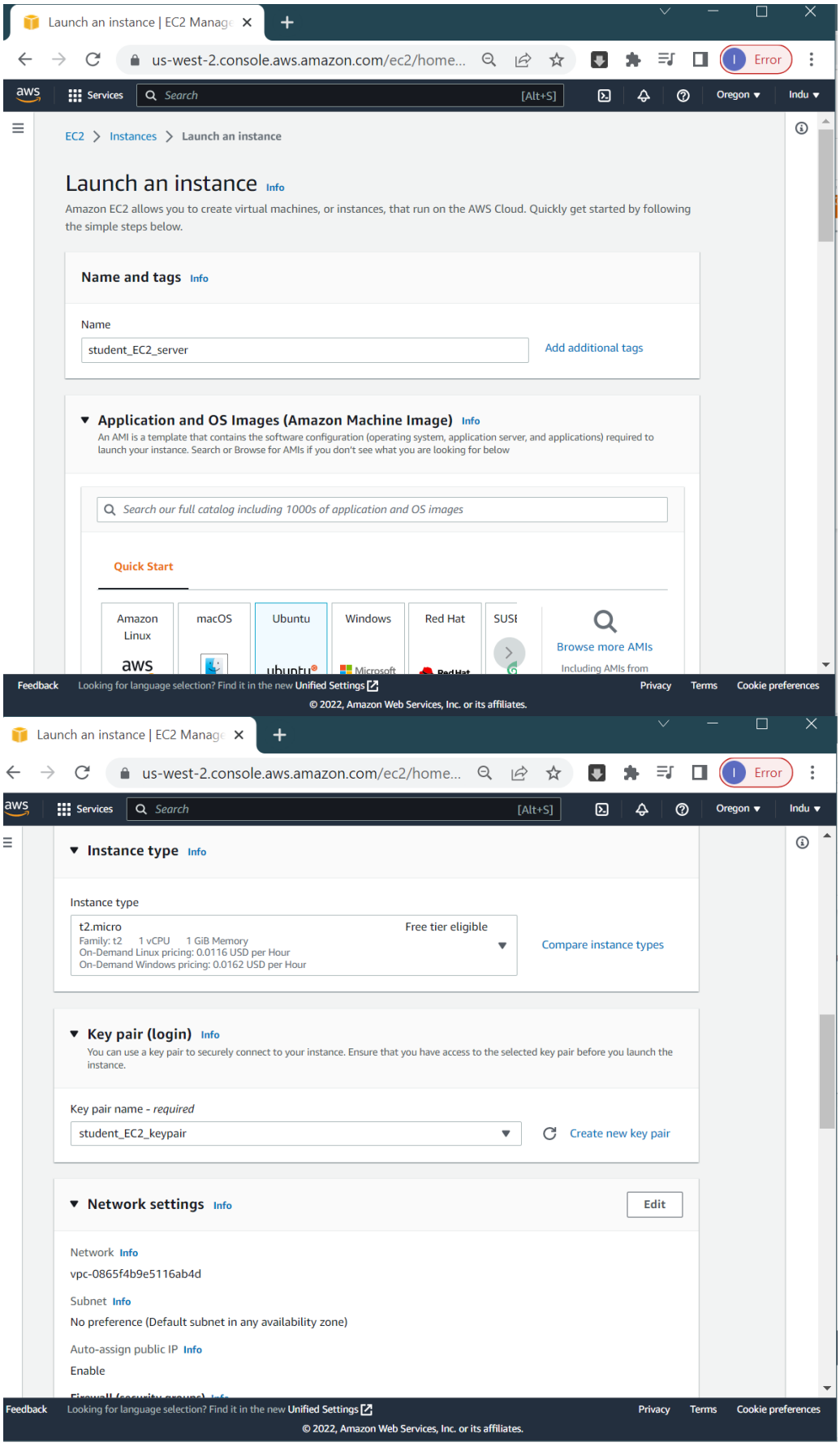

S3 bucket
s3.console.aws.amazon.com/s3/bucket/create?region=us-west-2
Error
aws
Services
Search
[Alt+S]
Global
Indu
No tags associated with this bucket.
Add tag
Default encryption
Automatically encrypt new objects stored in this bucket. Learn more
Server-side encryption
Disable
Enable
Advanced settings
After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.
Cancel
Create bucket
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences

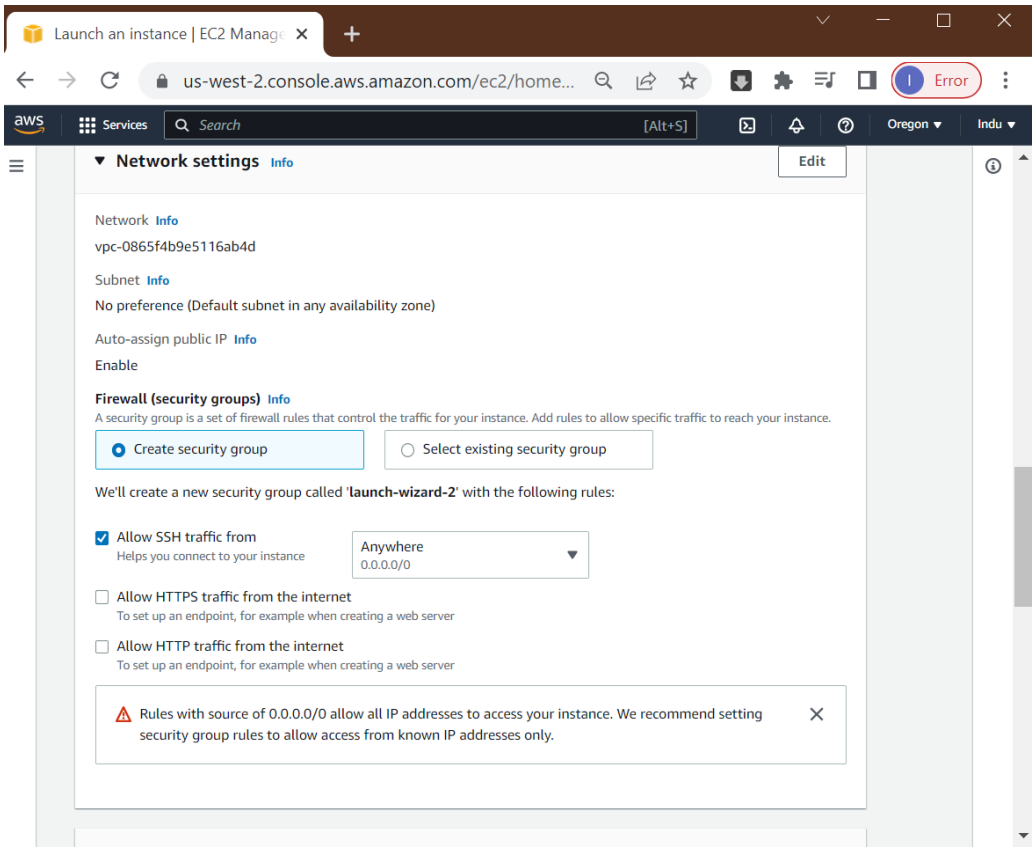


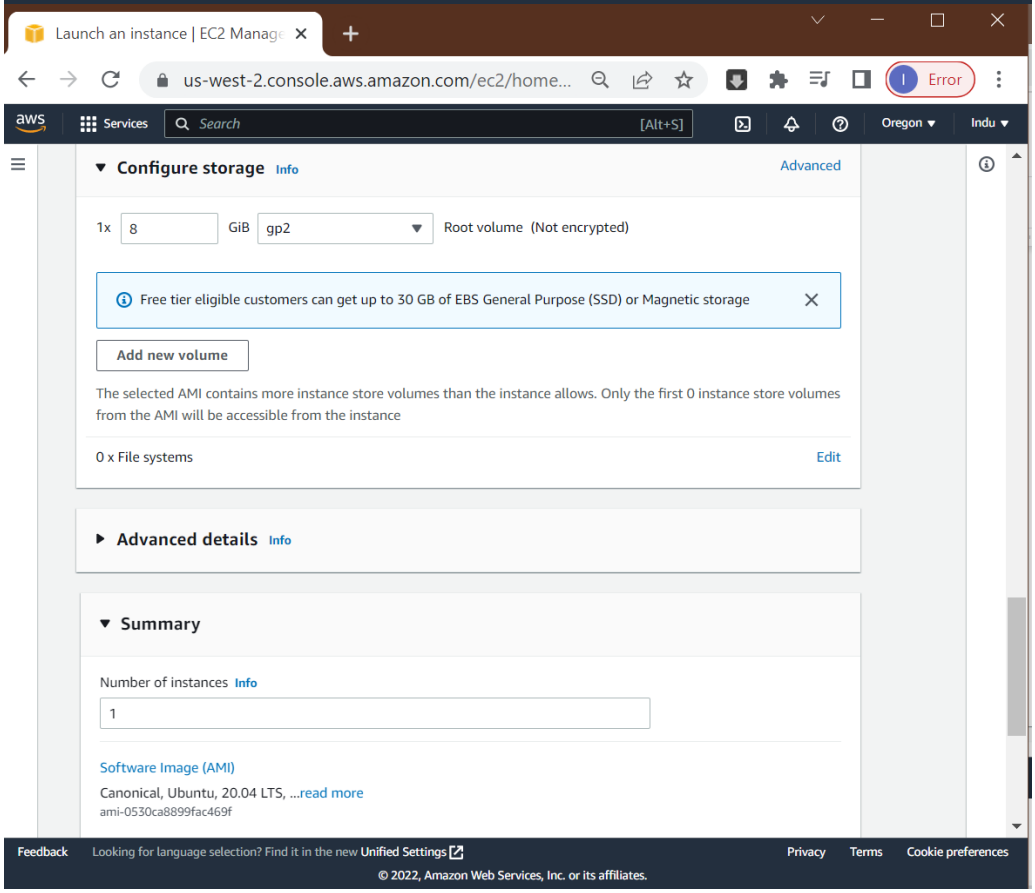
- **Deploy a EC2 server**

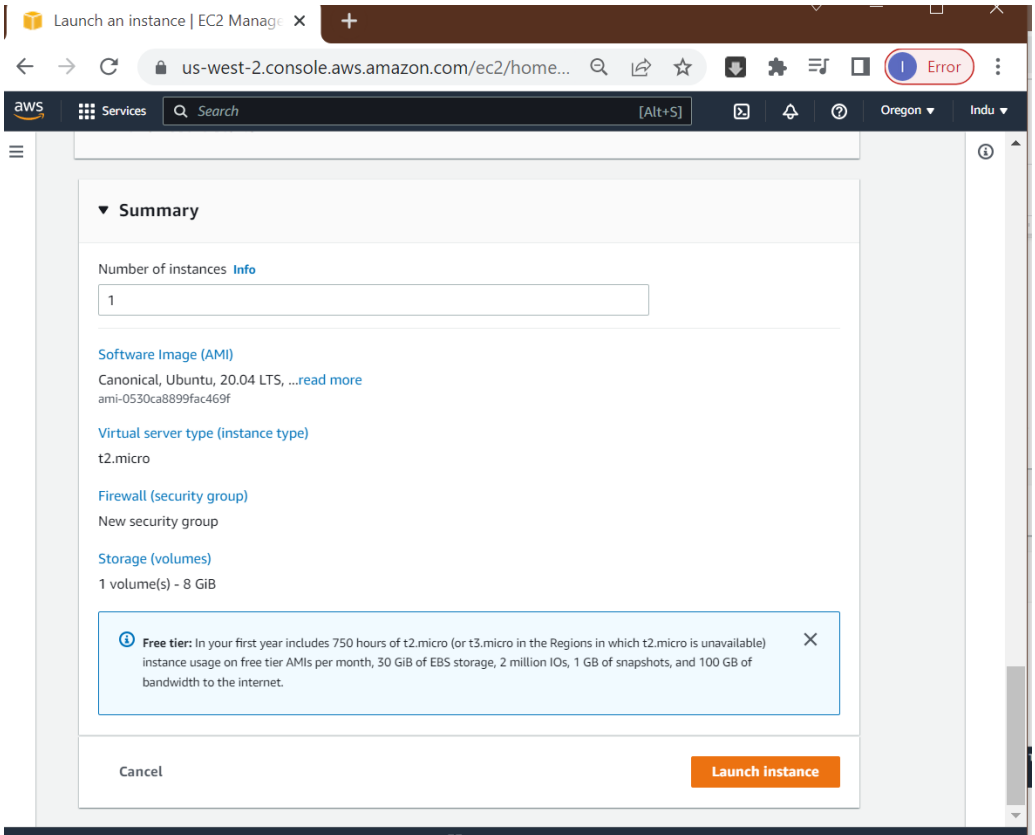
It is done by using Amazon EC2 service, for this we should select EC2 service from AWS management console dashboard. Then launch an instance, where the ubuntu server is used. Select all the desirable options, configure security groups, create a new key pair, or select existing one if you have one, and then review and launch. Some key advantages are reliable, scalable and infrastructure on demand.

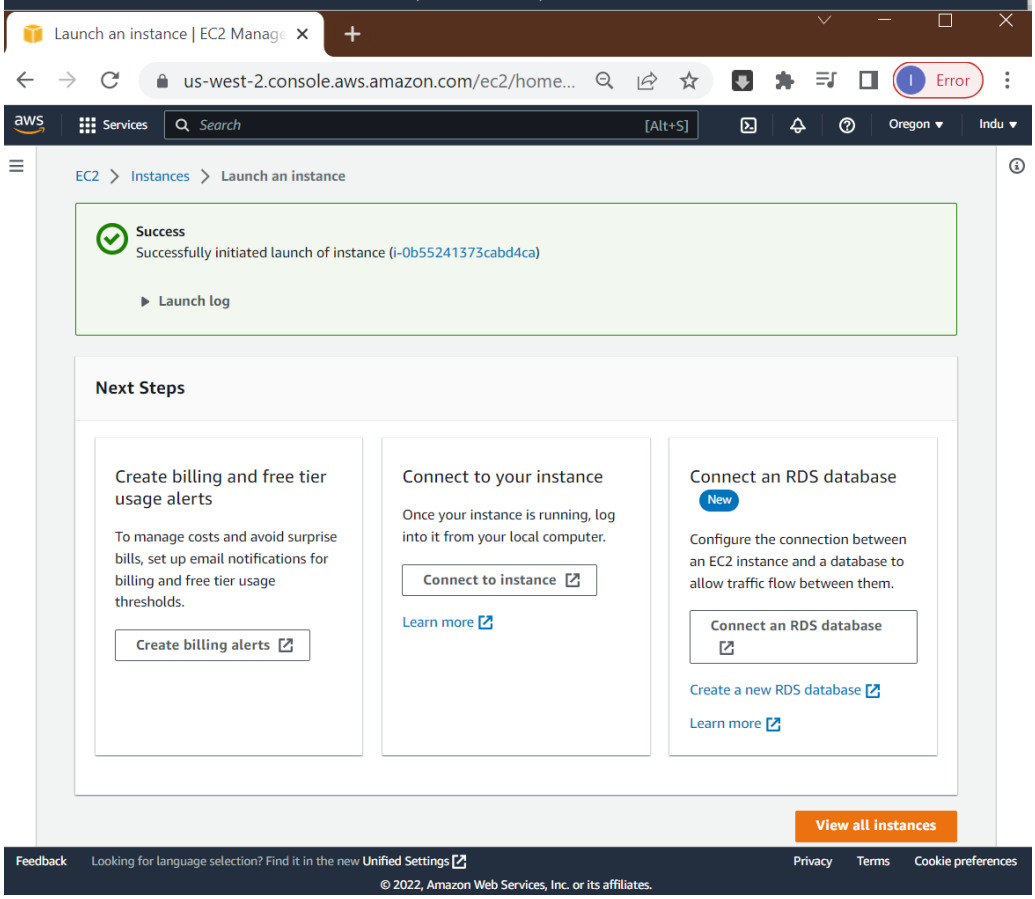


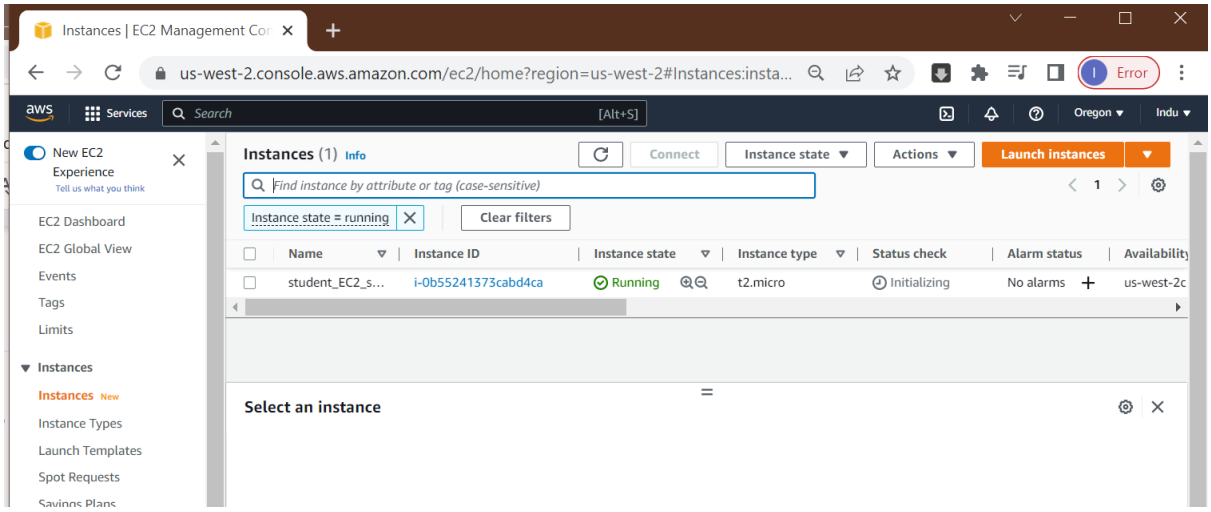

Launch an instance | EC2 Manage
us-west-2.console.aws.amazon.com/ec2/home...
Error
Services
Search
[Alt+S]
Oregon
Indu
EC2 > Instances > Launch an instance
Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
student_EC2_server
Add additional tags
Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Search our full catalog including 1000s of application and OS images
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE
Browse more AMIs
Including AMIs from
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences
© 2022, Amazon Web Services, Inc. or its affiliates.
Launch an instance | EC2 Manage
us-west-2.console.aws.amazon.com/ec2/home...
Error
Services
Search
[Alt+S]
Oregon
Indu
Instance type Info
Instance type
t2.micro
Free tier eligible
Family: t2 1 vCPU 1 GiB Memory
On-Demand Linux pricing: 0.0116 USD per Hour
On-Demand Windows pricing: 0.0162 USD per Hour
Compare instance types
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
student_EC2_keypair
Create new key pair
Network settings Info
Edit
Network Info
vpc-0865f4b9e5116ab4d
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences
© 2022, Amazon Web Services, Inc. or its affiliates.


Launch an instance | EC2 Manage
us-west-2.console.aws.amazon.com/ec2/home...
Error
aws
Services
Search
[Alt+S]
Oregon
Indu
Network settings Info
Edit
Network Info
vpc-0865f4b9e5116ab4d
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
We'll create a new security group called 'launch-wizard-2' with the following rules:
Allow SSH traffic from
Helps you connect to your instance
Anywhere
0.0.0.0/0
Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server
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.
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences
© 2022, Amazon Web Services, Inc. or its affiliates.


Launch an instance | EC2 Manage
us-west-2.console.aws.amazon.com/ec2/home...
Error
aws
Services
Search
[Alt+S]
Oregon
Indu
Configure storage Info
Advanced
1x
8
GiB
gp2
Root volume (Not encrypted)
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage
Add new volume
The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance
0 x File systems
Edit
Advanced details Info
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 20.04 LTS, ...read more
ami-0530ca8899fac469f
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences
© 2022, Amazon Web Services, Inc. or its affiliates.


Launch an instance | EC2 Manage
us-west-2.console.aws.amazon.com/ec2/home...
Error
aws
Services
Search
[Alt+S]
Oregon
Indu
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 20.04 LTS, ...read more
ami-0530ca8899fac469f
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Launch instance
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences
© 2022, Amazon Web Services, Inc. or its affiliates.

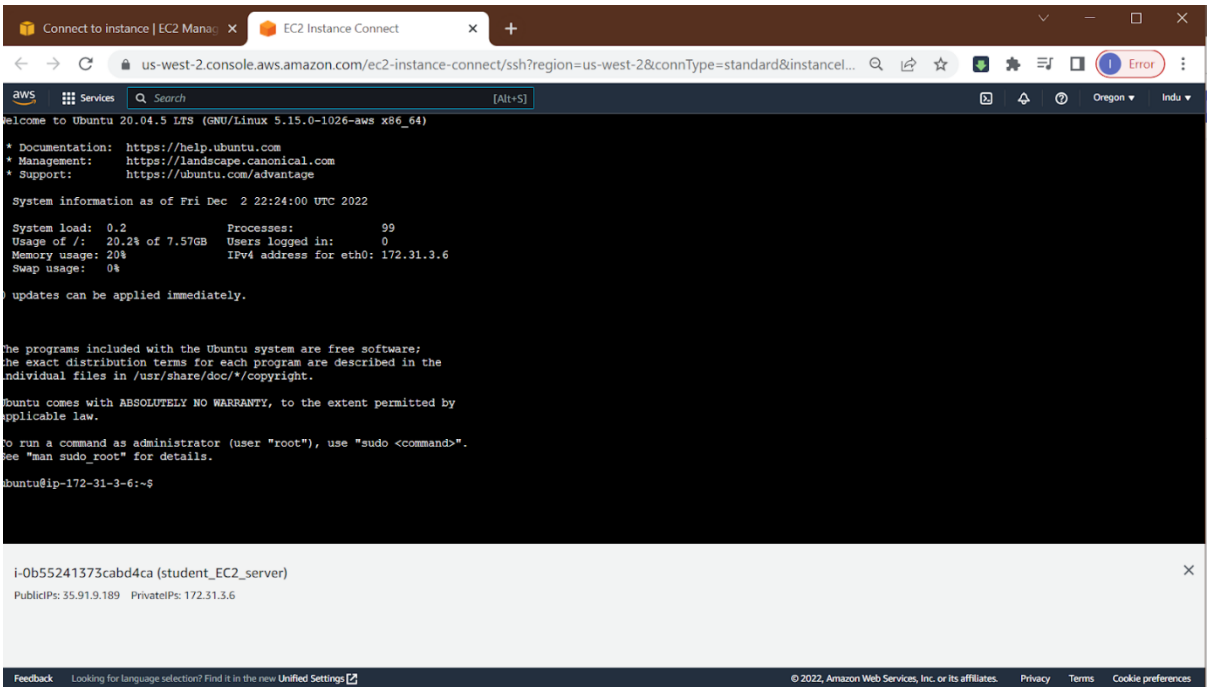

Launch an instance | EC2 Manage
us-west-2.console.aws.amazon.com/ec2/home...
Error
aws
Services
Search
[Alt+S]
Oregon
Indu
EC2 > Instances > Launch an instance
Success
Successfully initiated launch of instance (i-0b55241373cabd4ca)
Launch log
Next Steps
Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts
Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more
Connect an RDS database
New
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more
View all instances
Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie preferences
© 2022, Amazon Web Services, Inc. or its affiliates.

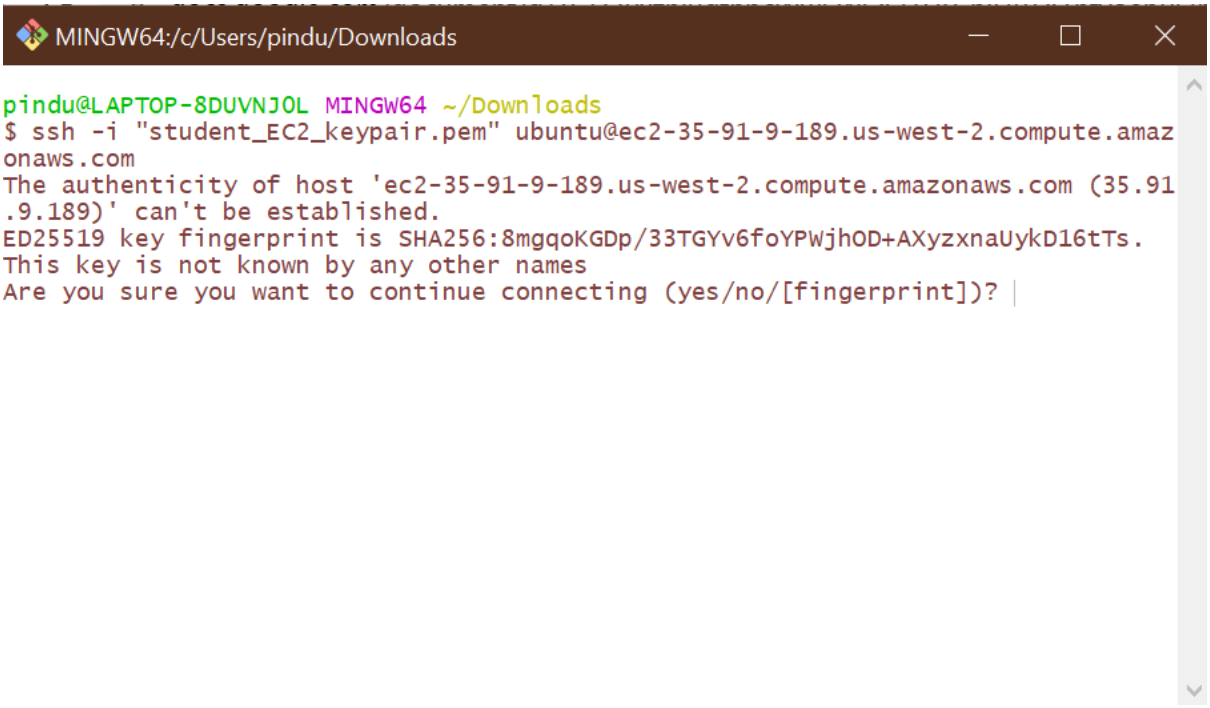


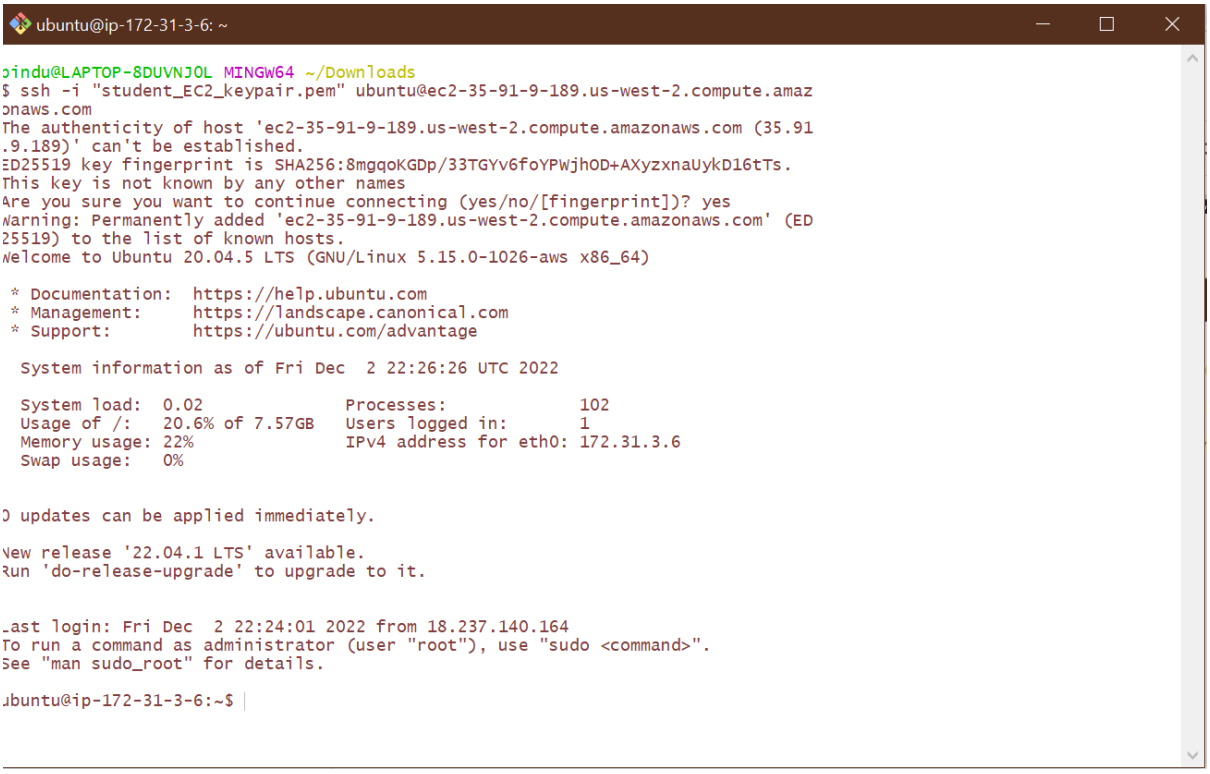
- **Next connect the server to our machine.**

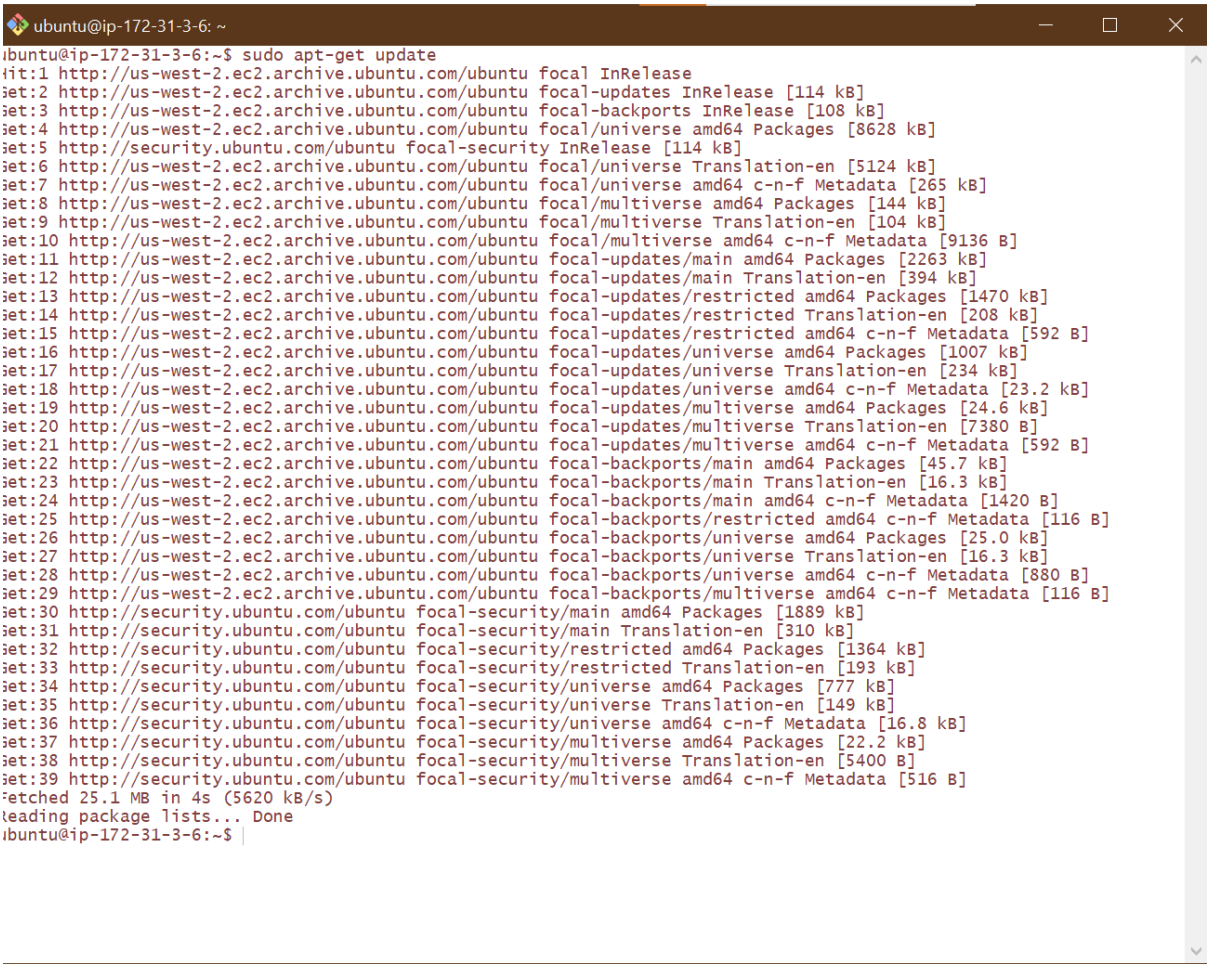
To connect from windows to server one can use Putty or Git bash or Super Putty which is a premium tool.

To connect from a mac, one can use the terminal itself, this will connect to an instance deployed in AWS.




MINGW64:/c/Users/pindu/Downloads
pindu@LAPTOP-8DUVNJOL MINGW64 ~/Downloads
$ ssh -i "student_EC2_keypair.pem" ubuntu@ec2-35-91-9-189.us-west-2.compute.amaz
onaws.com
The authenticity of host 'ec2-35-91-9-189.us-west-2.compute.amazonaws.com (35.91
.9.189)' can't be established.
ED25519 key fingerprint is SHA256:8mgqoKGDp/33TGYv6foYPWjhOD+AXyzxnaUykD16tTs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?


ubuntu@ip-172-31-3-6: ~
pindu@LAPTOP-8DUVNJOL MINGW64 ~/Downloads
$ ssh -i "student_EC2_keypair.pem" ubuntu@ec2-35-91-9-189.us-west-2.compute.amaz
onaws.com
The authenticity of host 'ec2-35-91-9-189.us-west-2.compute.amazonaws.com (35.91
.9.189)' can't be established.
ED25519 key fingerprint is SHA256:8mgqoKGDp/33TGYv6foYPWjhOD+AXyzxnaUykD16tTs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-35-91-9-189.us-west-2.compute.amazonaws.com' (ED
25519) to the list of known hosts.
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1026-aws x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Fri Dec 2 22:26:26 UTC 2022
System load: 0.02 Processes: 102
Usage of /: 20.6% of 7.57GB Users logged in: 1
Memory usage: 22% IPv4 address for eth0: 172.31.3.6
Swap usage: 0%
0 updates can be applied immediately.
New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Fri Dec 2 22:24:01 2022 from 18.237.140.164
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-3-6:~$



- **Next connect to our RDS**

First deploy MySQL client on the server which helps to connect to our RDS machine, for this install MySQL client.

```
ubuntu@ip-172-31-3-6:~$ mysql -h student.cmojbf7a9vmd.us-west-2.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 21
Server version: 8.0.28 Source distribution

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>
```

After installing MySQL client software, now we connect to the RDS, so that now we will be in a MySQL shell.

- **Next deploy database on server**

For this first create a database and then create a table in that database for the information to be stored.

// commands

After logging into databases:

`Show databases;` // gives list of Databases present

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql>
```

(need to create a database now)

`create database <database-name>;` // will create a database with the given name.

```
mysql> create database student;
Query OK, 1 row effected(0.05 sec)
```

(need to switch to created database)

`use <databasename>;`

```
mysql> use student;
Database changed
```

(now we can create tables in the created database)

```
create table <table-name>(<required arguments>);
```
// creates a table inside the database.

```
mysql> create table student(
    -> netid varchar(20) NOT NULL,
    -> fname varchar(20) NOT NULL,
    -> lname varchar(20),
    -> major varchar(20),
    -> location varchar(20),
    -> PRIMARY KEY(netid));
Query OK, 0 rows affected (0.03 sec)

mysql>
```

```
show tables;
```

```
mysql> show tables;
+--------------------+
| Tables_in_student  |
+--------------------+
| student            |
+--------------------+
1 row in set (0.00 sec)

mysql>
```

```
Select * from student;
```
//Initially it displays an empty set since it does not have any student data in it.

```
mysql> select * from student;
Empty set (0.01 sec)
```

II. The second phase is to build the website, configure the website code and deploy the website on EC2 server.

- **Build the website**

Here to build the website we have used the Python Flask Framework. Flask is known as a simple but extensible framework.

- **Configure the Website Code**
  i. Provide Host name of RDS in host region of config code.
  ii. Provide username, password, and database name in config code.
  iii. Provide bucket name and region in the configuration code.

- **Deploy Website on EC2 Server**
  Make the website available on EC2 server.

After connecting to EC2 instance we need to clone a git repository.

Then navigate to the project folder and install the required packages.

Execute all the below commands on the server to install them on server.

- **`sudo apt-get install python3`**

Python3 is the language that is used for coding the website, we must install it on an existing system where we are running the website.

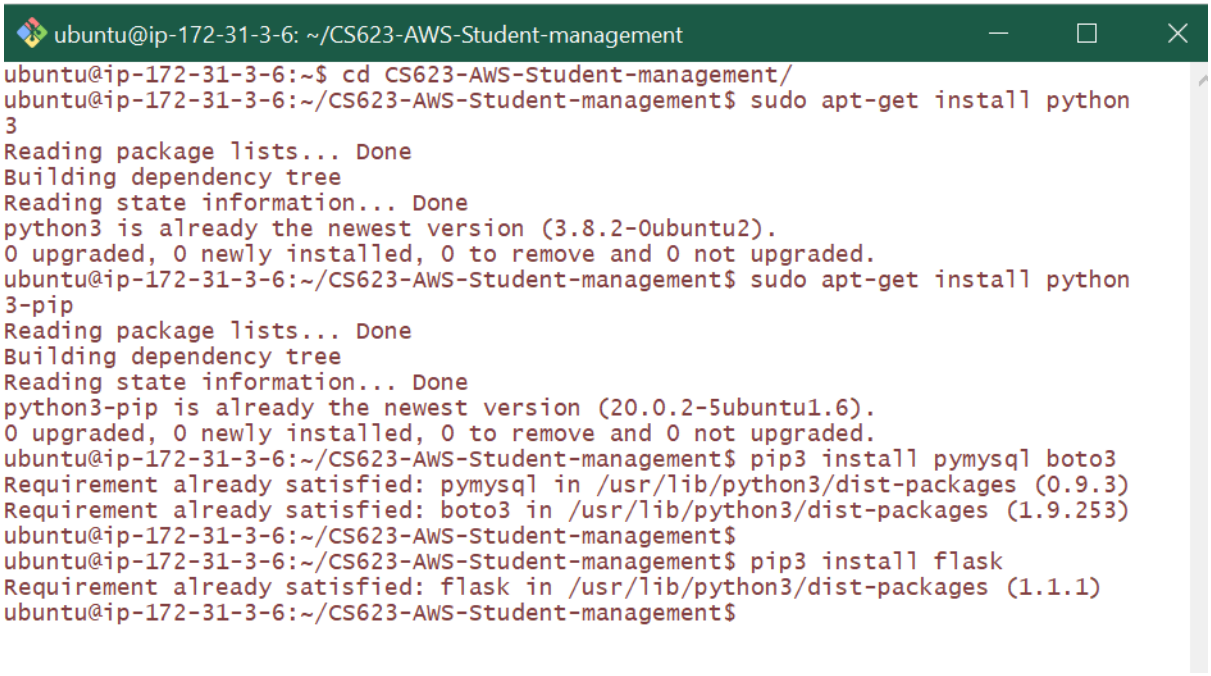
- **`sudo apt-get install python3-pip`**

- **`pip3 install pymysql boto3`**

It is an SDK which helps to connect to S3 service where we upload files.

- **`pip3 install flask`**

Flask helps to go ahead and publish a website on to that server.



Then give EC2 instance or server access to basically upload data to S3. We need to do this because on the server we are running some code and that code is trying to interact with the AWS S3 service, when we connect with our RDS we should give authentication to our S3 service, so for that we must give the IAM role to our instance.

To give IAM role to our instance, follow the below steps -

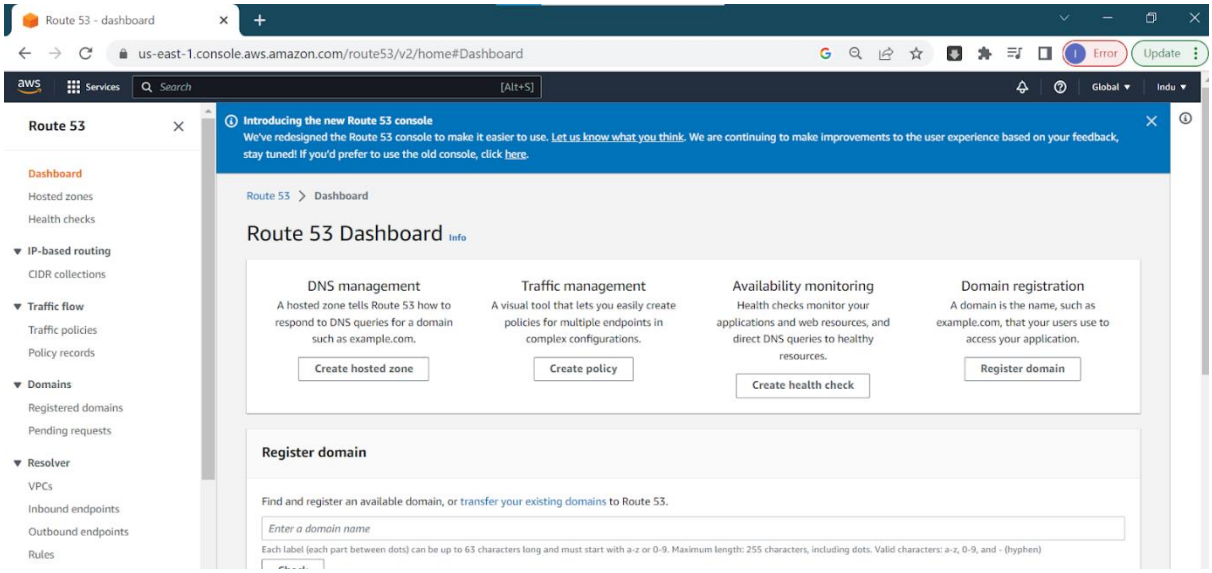
i. Select IAM service from AWS management console dashboard.
ii. Then create an admin role for our AWS account.
iii. Then select roles for our EC2 instance.
iv. Then we should go to our EC2 service and click on the attach/replace IAM role and select the role which we created from it. Now EC2 is connected to RDS, EC2 is connected to S3, and website deployed in EC2 server.

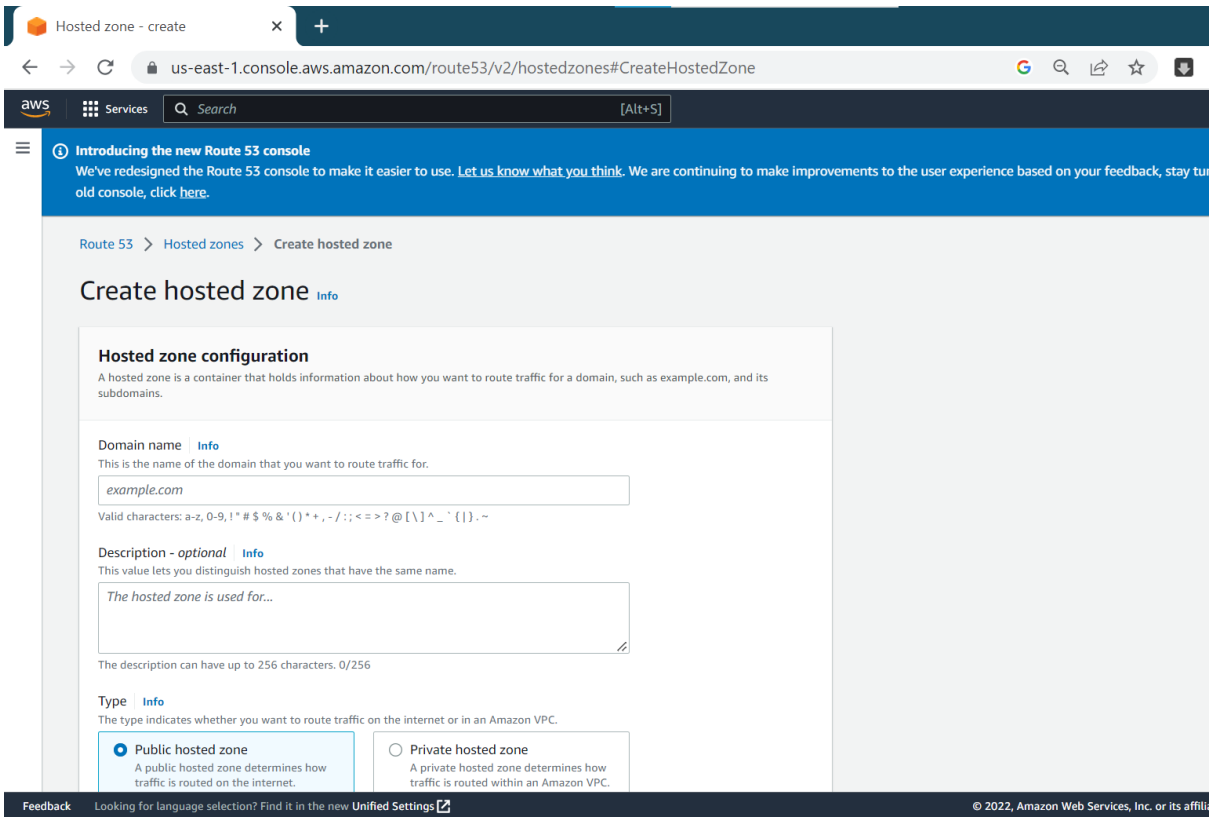
III. The third phase is routing domain to EC2 server.

To route domain to EC2 server we need to use Route 53 service from AWS Management console dashboard and then connect Route 53 service to EC2 server.

i. Route 53 service

First, select Route 53 service from AWS management console dashboard. Then, purchase a domain name for the website. Then, create a hosted zone and mention the purchased domain name here.

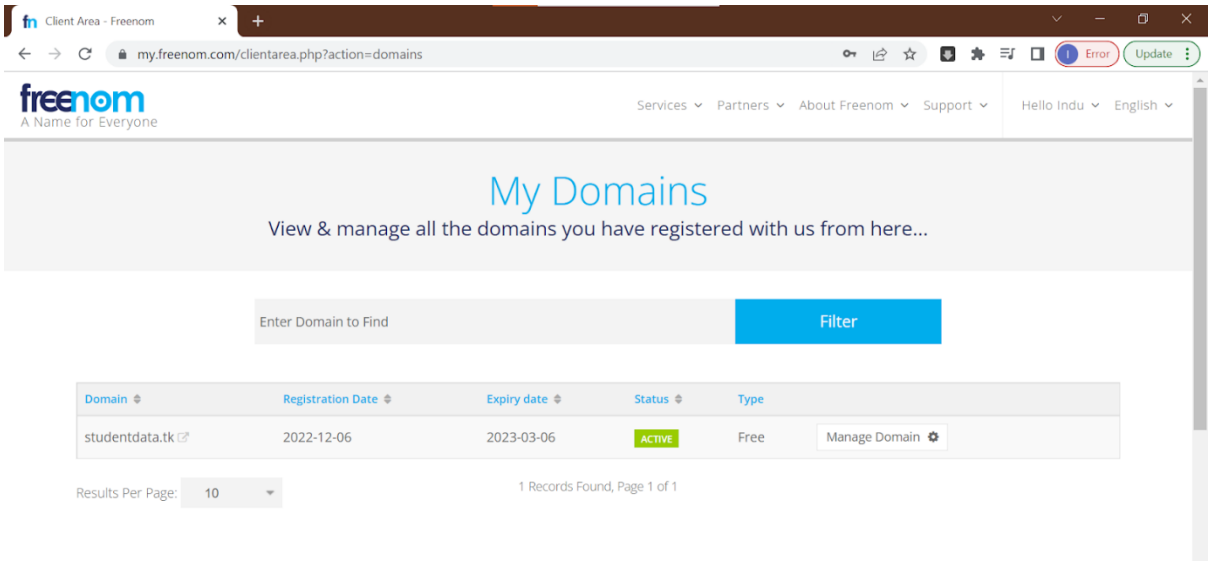

Route 53 - dashboard
us-east-1.console.aws.amazon.com/route53/v2/home#Dashboard
Error
Update
aws
Services
Search
[Alt+S]
Global
Indu
Route 53
Dashboard
Hosted zones
Health checks
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Pending requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Introducing the new Route 53 console
We've redesigned the Route 53 console to make it easier to use. Let us know what you think. We are continuing to make improvements to the user experience based on your feedback, stay tuned! If you'd prefer to use the old console, click here.
Route 53 > Dashboard
Route 53 Dashboard Info
DNS management
A hosted zone tells Route 53 how to respond to DNS queries for a domain such as example.com.
Create hosted zone
Traffic management
A visual tool that lets you easily create policies for multiple endpoints in complex configurations.
Create policy
Availability monitoring
Health checks monitor your applications and web resources, and direct DNS queries to healthy resources.
Create health check
Domain registration
A domain is the name, such as example.com, that your users use to access your application.
Register domain
Register domain
Find and register an available domain, or transfer your existing domains to Route 53.
Enter a domain name
Each label (each part between dots) can be up to 63 characters long and must start with a-z or 0-9. Maximum length: 255 characters, including dots. Valid characters: a-z, 0-9, and - (hyphen)


Hosted zone - create
us-east-1.console.aws.amazon.com/route53/v2/hostedzones#CreateHostedZone
aws
Services
Search
[Alt+S]
Introducing the new Route 53 console
We've redesigned the Route 53 console to make it easier to use. Let us know what you think. We are continuing to make improvements to the user experience based on your feedback, stay tu
old console, click here.
Route 53 > Hosted zones > Create hosted zone
Create hosted zone Info
Hosted zone configuration
A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.
Domain name Info
This is the name of the domain that you want to route traffic for.
example.com
Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~
Description - optional Info
This value lets you distinguish hosted zones that have the same name.
The hosted zone is used for...
The description can have up to 256 characters. 0/256
Type Info
The type indicates whether you want to route traffic on the internet or in an Amazon VPC.
Public hosted zone
A public hosted zone determines how traffic is routed on the internet.
Private hosted zone
A private hosted zone determines how traffic is routed within an Amazon VPC.
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affili

Domain name has to be purchased:

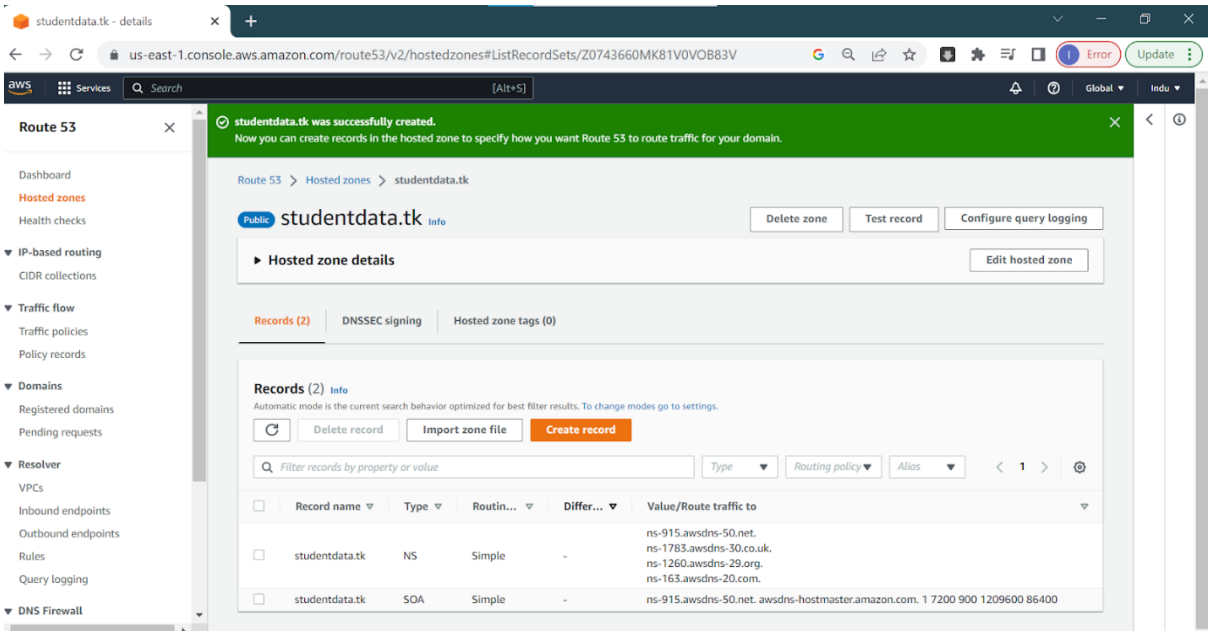
freenom.com offers domains for free which are valid for 3 Months.

Myservices->my domains

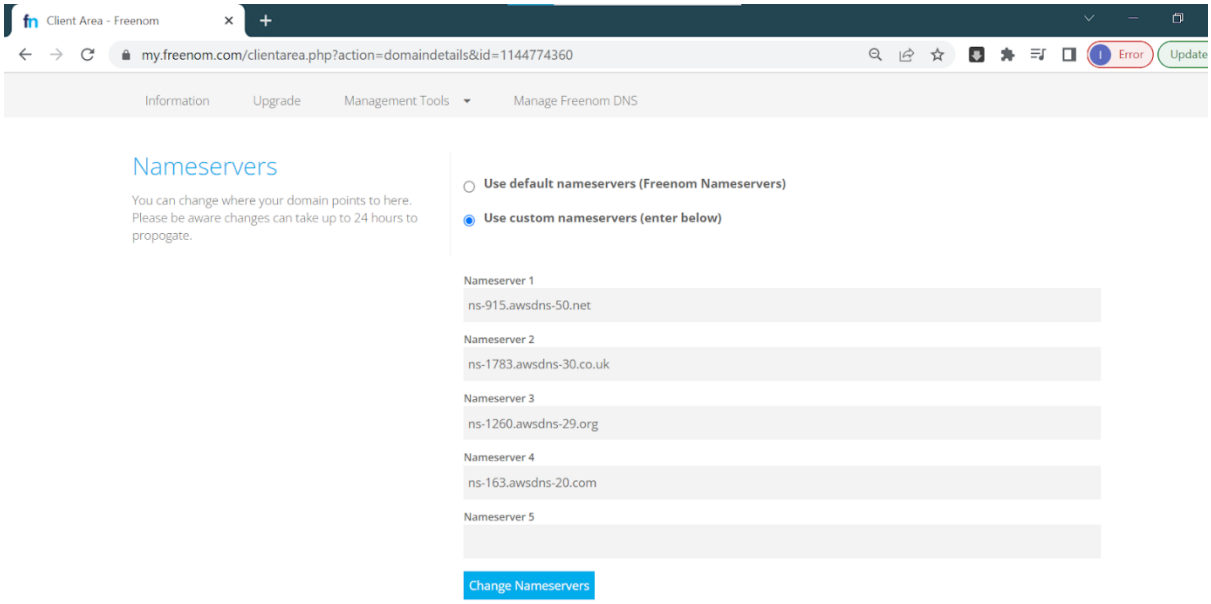


Now go to Route53 services->create a hosted zone

Specify domain name “Studentdata.tk”

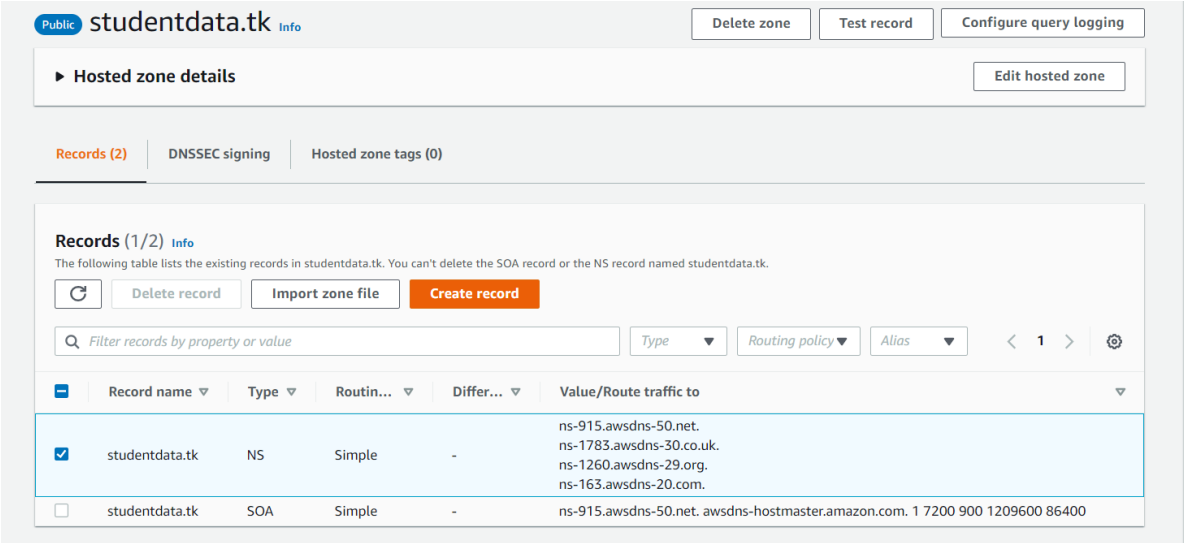


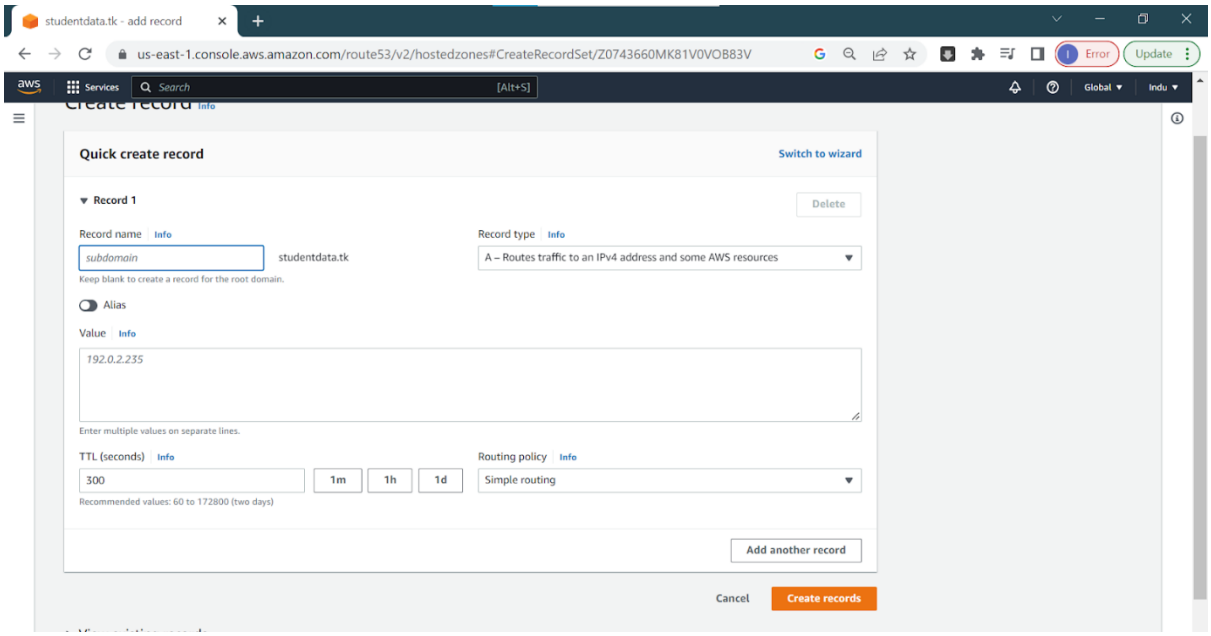
Now, configure the Name Servers on freenom.com



ii. Connect Route 53 service to EC2 server

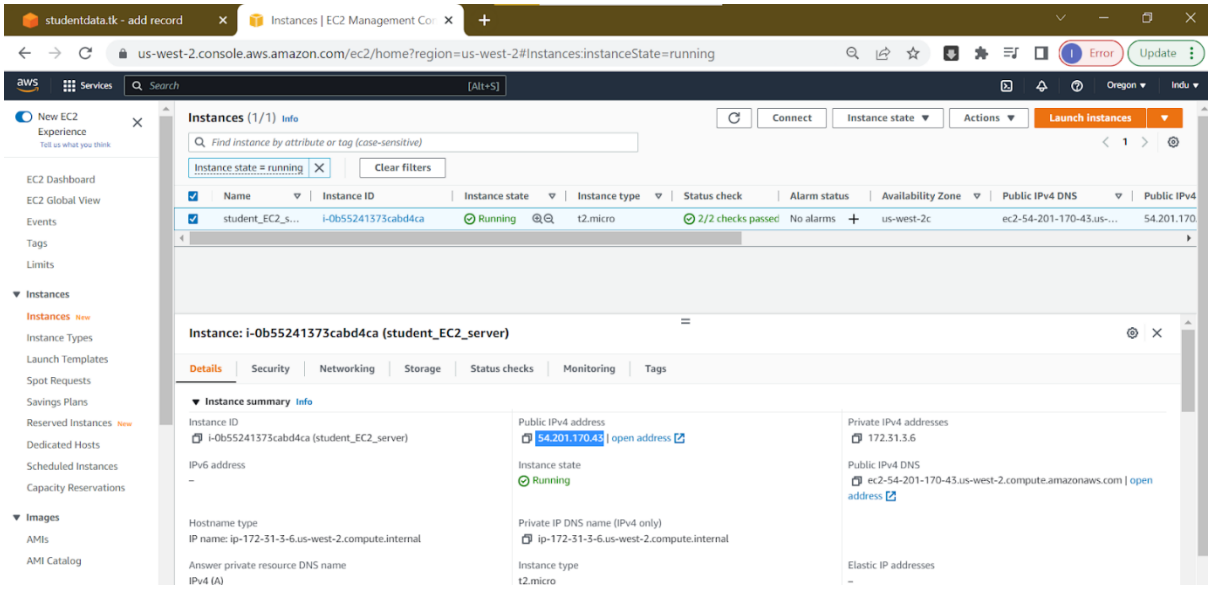
For this, first create a record set and mention EC2 IP Address in the value box.



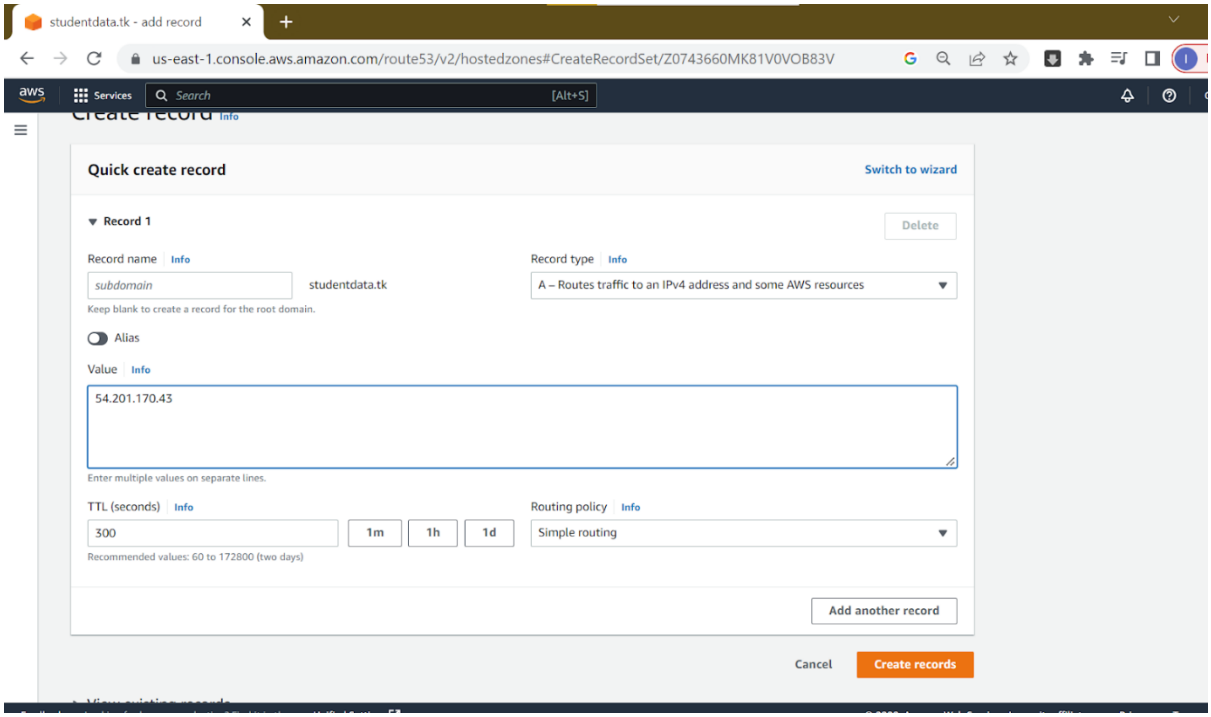


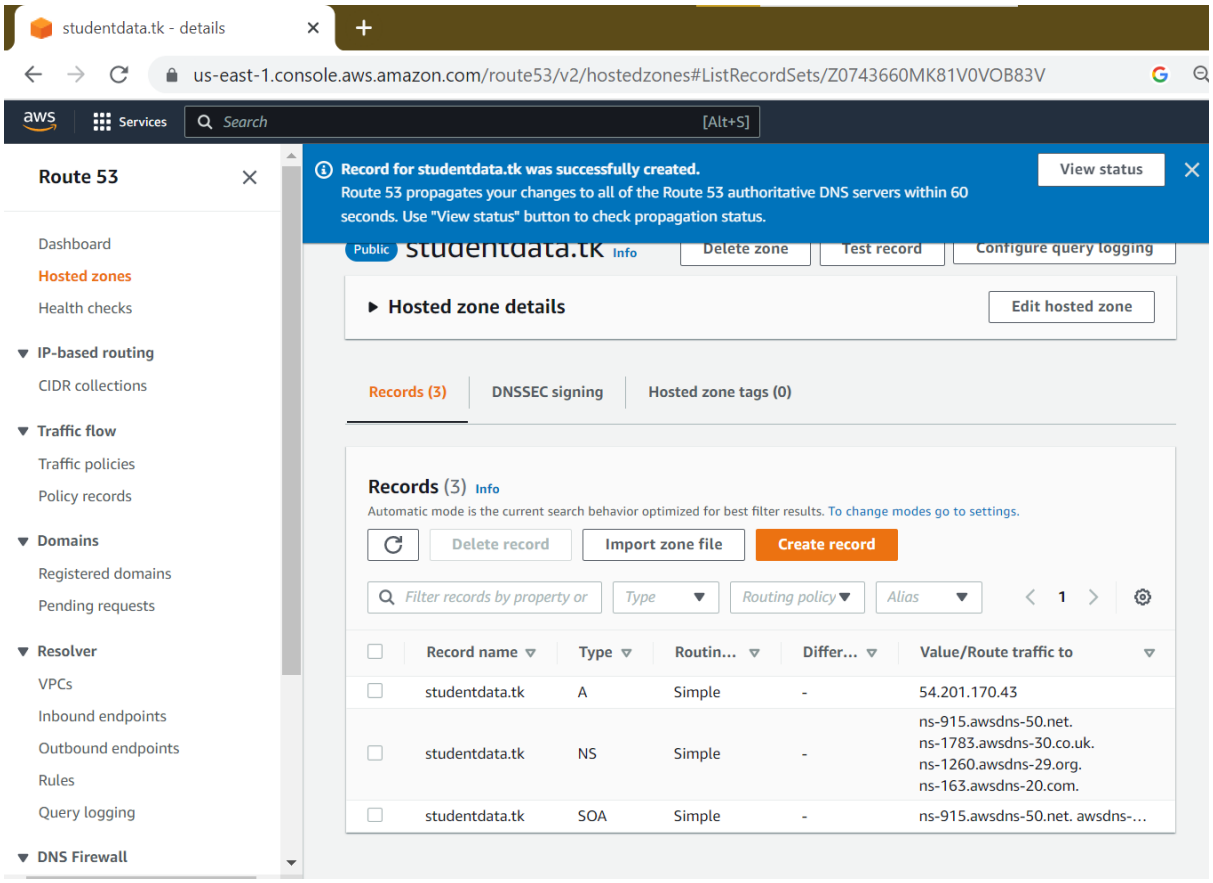
Now we need to specify an IP address in the ‘value’ box.

For this go to EC2 instance

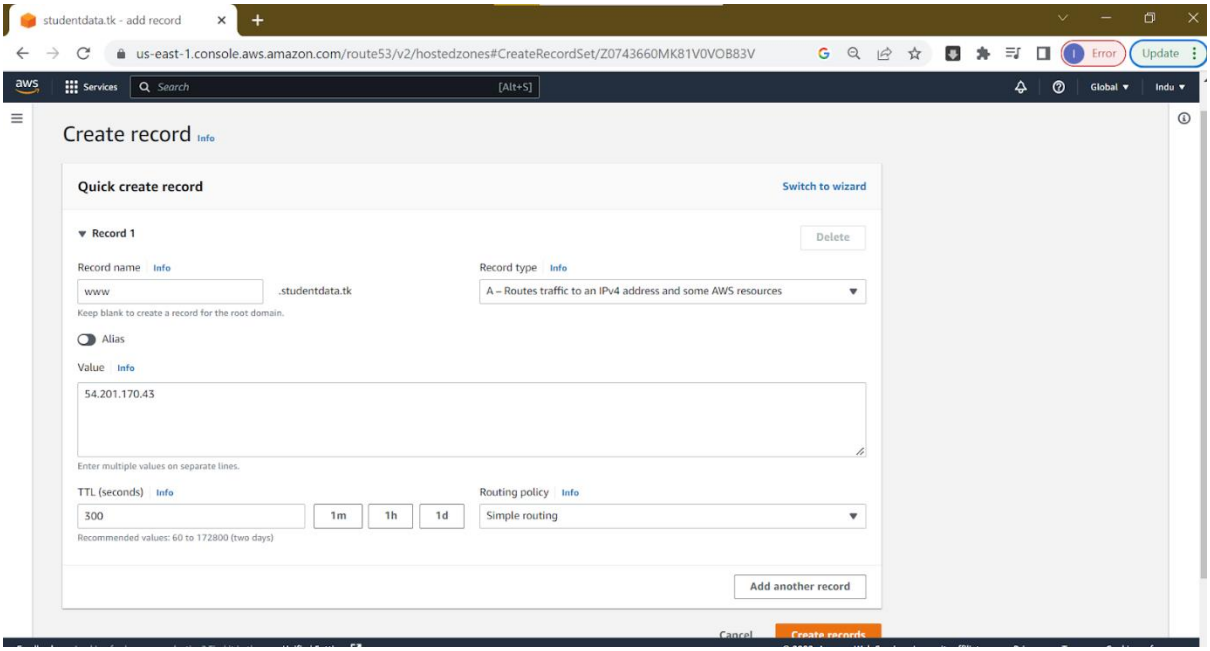


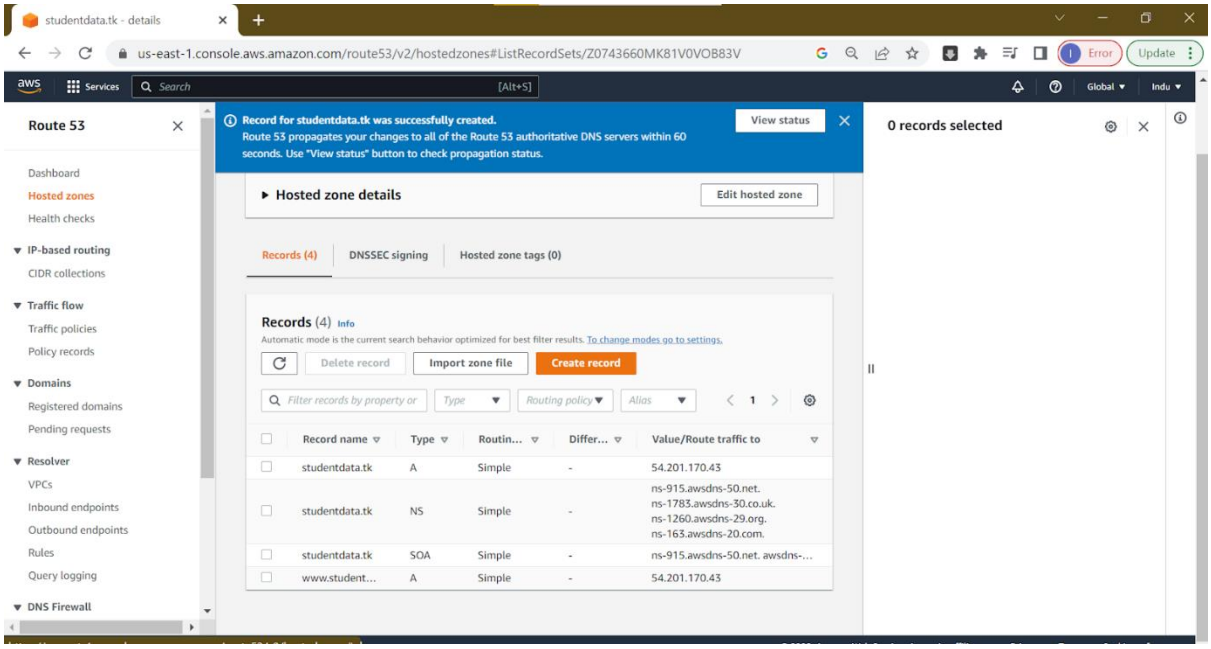
Copy the above IPV4 address and paste it in the value box as shown below.



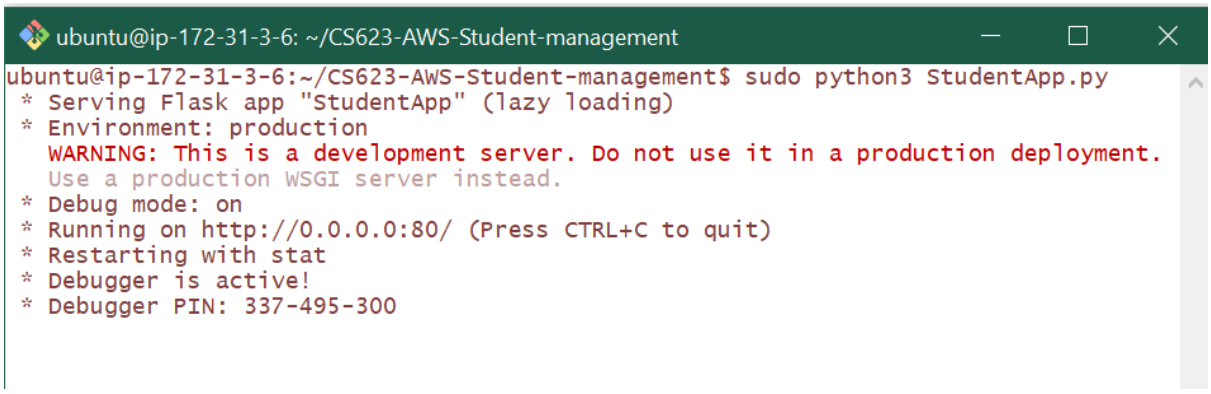


Now, create another record set and mention the name as **www** and put the same IP Address in the value box.

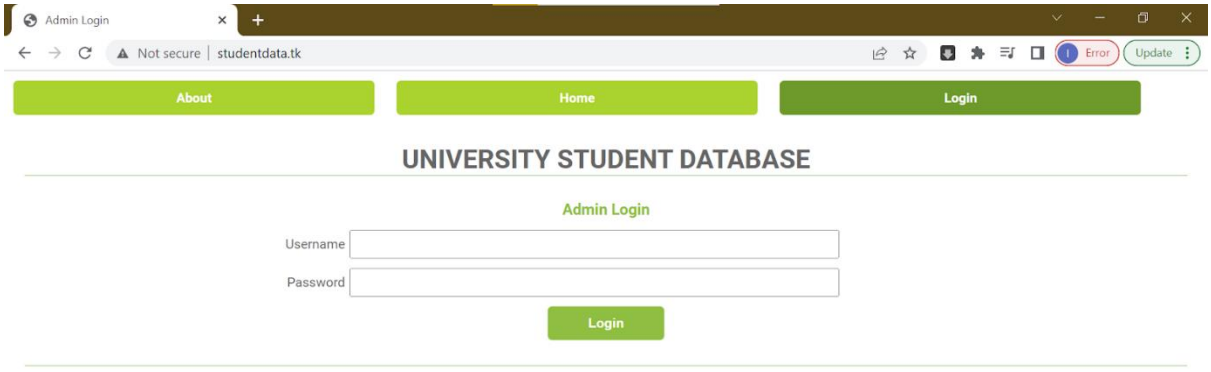




Now, try to run the project from EC2 Server.



Now go to search bar and type studentdata.tk - http://studentdata.tk/



3.2 Challenges we faced during building this project

- Initially, we created EC2 and RDS in different regions due to which we had challenges in establishing connection between them.
- We had to add new security group rules to access S3 from EC2 public IPV4 address
- We also faced challenges in connecting EC2 to S3 which was later fixed by creating necessary IAM roles for EC2 access to S3 service.
- Initially we created RDS with default configuration instead of choosing free tier setup that resulted in added costs. We had to then delete it to replace it with free tier options.
- We created a hosted zone in Route 53 service to route domain to EC2 server. Though we had an option of creating a domain while setting up hosted zone, we used a third party website to create a free domain instead of purchasing it from Route 53.
- Every time we stop EC2 instance, the IPV4 address changes and hence need to reconfigure Route53 with new IP address in such instances. In the future scope of the project, we can overcome this challenge by using Elastic IP address.
- We had trouble installing all the necessary packages like flask, boto and pymysql in the server.

CHAPTER 4: CODE & TECHNOLOGY

4.1 Programming Language

1. **Frontend**

In frontend, we used Python and HTML.

2. **Backend**

In backend, we used RDS (MySQL Database)

4.2 Cloud services

1. **Amazon RDS**:
   We are using Amazon RDS service, which is used to operate and scale a relational database in the cloud. In this we are using MySQL database, and this is used to store the entries. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

2. **Amazon S3**:
   We are using Amazon S3 service, which is a Storage for the internet. We can use it to store and retrieve any amount of data at any time, from anywhere on the web.

3. **Amazon EC2**:
   We are using Amazon EC2 service, it is a web service for launching and managing Linux/UNIX and Windows Server instances in Amazon's Data Centers.

4. **Amazon IAM**
   AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. When we connect with our RDS we should give authentication to our S3 service, so that we must give the IAM role to our instance.

5. **Amazon Route 53**
   AWS Route 53 is a service we are using to route domain to EC2 server.

CHAPTER 5: GIT ACCESS & SOURCE CODE

Below is the GitHub Link for our project:

https://github.com/in-d-web/CS623-AWS-Student-management

We have a component where students can provide their information and when they click to update the database, this code will run and will establish a connection to store data and to retrieve the data as well.

Python Code:

StudentApp.py is the entry point of our application:

```
import io

from flask import Flask, render_template, request
from pymysql import connections
import sys
import boto3
import os
from config import *
from PIL import Image
import json

app = Flask(__name__)

bucket = awsbucket
region = awsregion

db_conn = connections.Connection(
    host=apphost,
    port=3306,
    user=appuser,
    password=apppass,
    db=appdb
)


output = {}
table = 'student'


@app.route("/", methods=['GET', 'POST'])
def home():
```

```
    return render_template('admin.html')


@app.route("/gotoadd", methods=['GET', 'POST'])
def gotoadd():
    return render_template('AddStudent.html')


@app.route("/about", methods=['GET', 'POST'])
def about():
    return render_template('about.html')


@app.route("/addstudent", methods=['POST'])
def AddStudent():
    net_id = request.form['net_id']
    first_name = request.form['first_name']
    last_name = request.form['last_name']
    major = request.form['major']
    location = request.form['location']
    student_image_file = request.files['student_image_file']

    insert_sql = "INSERT INTO student VALUES (%s, %s, %s, %s, %s)"
    cursor = db_conn.cursor()

    try:

        cursor.execute(insert_sql, (net_id, first_name,
                       last_name, major, location))
        db_conn.commit()
        student_name = "" + first_name + " " + last_name
        # Upload image file to S3 bucket #
        student_image_file_name_in_s3 = "net-id-" + str(net_id) + "_image_file"
        s3 = boto3.resource('s3')

        try:
            print("Data inserted in MySQL RDS... uploading image to S3...")
            s3.Bucket(awsbucket).put_object(
                Key=student_image_file_name_in_s3, Body=student_image_file)
            bucket_location = boto3.client(
                's3').get_bucket_location(Bucket=awsbucket)
            s3_location = (bucket_location['LocationConstraint'])

            if s3_location is None:
                s3_location = ''
            else:
                s3_location = '-' + s3_location
```

```
            object_url = "https://s3{0}.amazonaws.com/{1}/{2}".format(
                s3_location,
                awsbucket,
                student_image_file_name_in_s3)

        except Exception as e:
            print(e)
            return render_template('Error1.html')

    finally:
        cursor.close()

    print("all modification done...")
    return render_template('StudentAdd_Success.html', name=student_name)


@app.route("/admin", methods=['GET', 'POST'])
def admin():
    return render_template("admin.html")


@app.route("/getstudent", methods=['GET', 'POST'])
def GetStudent():
    return render_template("GetStudentInfo.html")


@app.route("/fetchdata", methods=['POST'])
def FetchStudent():
    net_id = request.form['net_id']

    output = {}
    select_sql = "SELECT netid, fname, lname, major, location from student where netid=%s"
    cursor = db_conn.cursor()
    student_image_file_name_in_s3 = "net-id-" + str(net_id) + "_image_file"
    s3 = boto3.resource('s3')

    bucket_location = boto3.client(
        's3').get_bucket_location(Bucket=awsbucket)
    s3_location = (bucket_location['LocationConstraint'])

    if s3_location is None:
        s3_location = ''
    else:
        s3_location = '-' + s3_location

    image_url = "https://s3{0}.amazonaws.com/{1}/{2}".format(
        s3_location,
        awsbucket,
```

```
        student_image_file_name_in_s3)

    try:
        cursor.execute(select_sql, (net_id))
        result = cursor.fetchone()

        output["net_id"] = result[0]
        print('EVERYTHING IS FINE TILL HERE')
        output["first_name"] = result[1]
        output["last_name"] = result[2]
        output["major"] = result[3]
        output["location"] = result[4]
        print(output["net_id"])

        return render_template("StudentInfo_Output.html", id=output["net_id"],
fname=output["first_name"],
                               lname=output["last_name"], major=output["major"],
location=output["location"], image_url=image_url)

    except Exception as e:
        print(e)
        return render_template('Error2.html')


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80, debug=True)
```

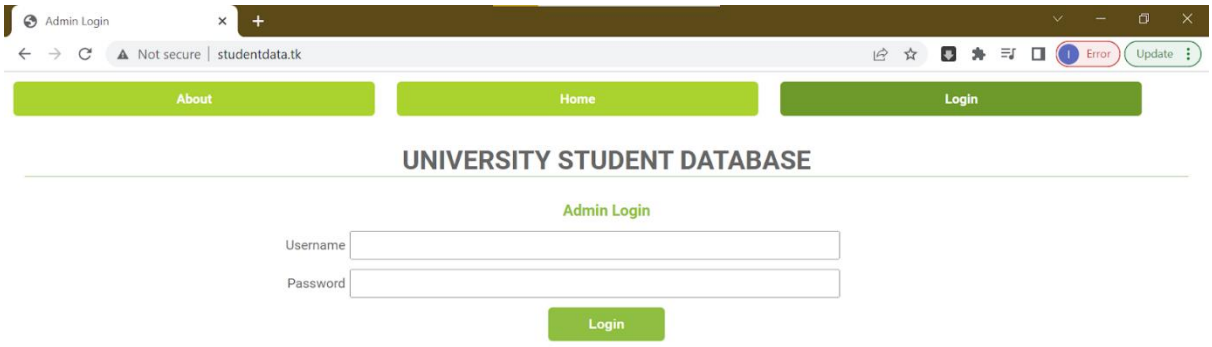
There is a small python code below for the configuration - **config.py**

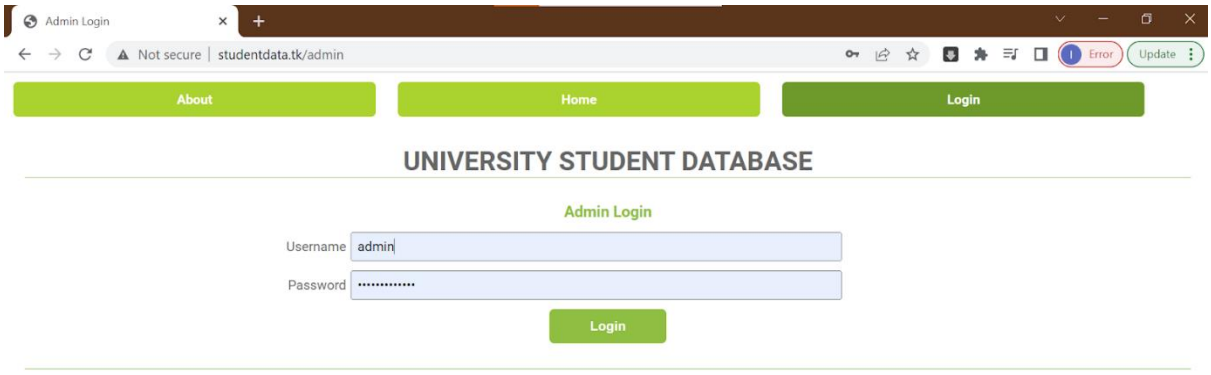
```
import os

apphost = "student.cmojbf7a9vmd.us-west-2.rds.amazonaws.com"
appuser = "admin"
apppass = "qazqazqaz"
appdb = "student"
awsbucket = "addstudent01"
awsregion ="us-west-2c"
adminusername= os.environ.get('adminpassword') or "admin"
adminpassword= os.environ.get('adminpassword')
```

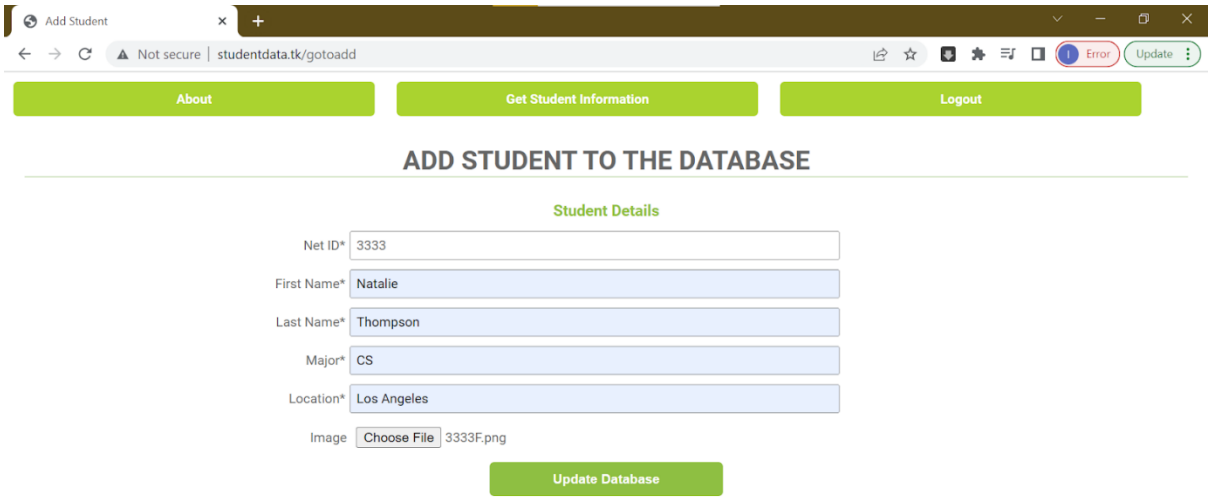
CHAPTER 6: DEMO SCREENSHOTS FROM WEBSITE

Home Page:

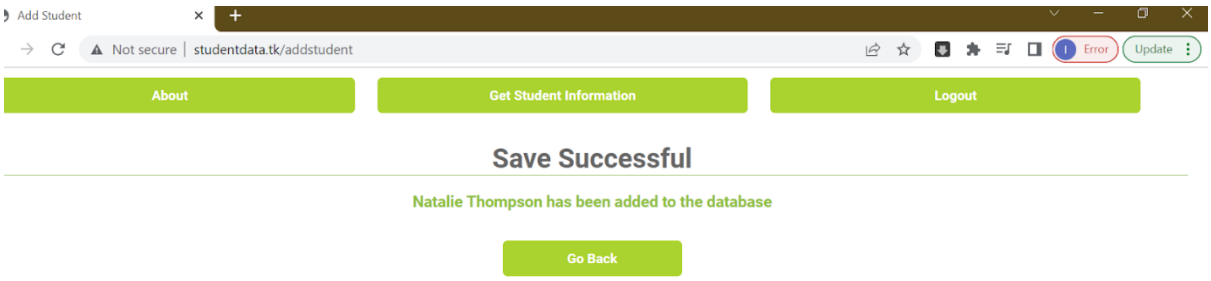




Once logged in, add student to the Database Page displays.



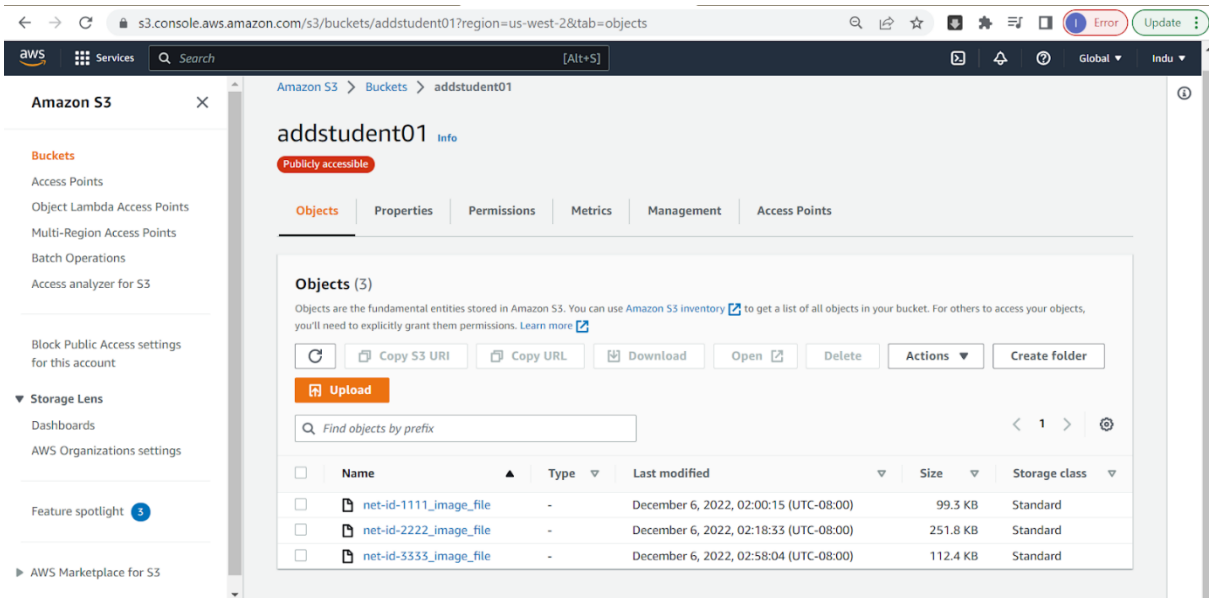
When successfully a student's data is added, the following page displays.



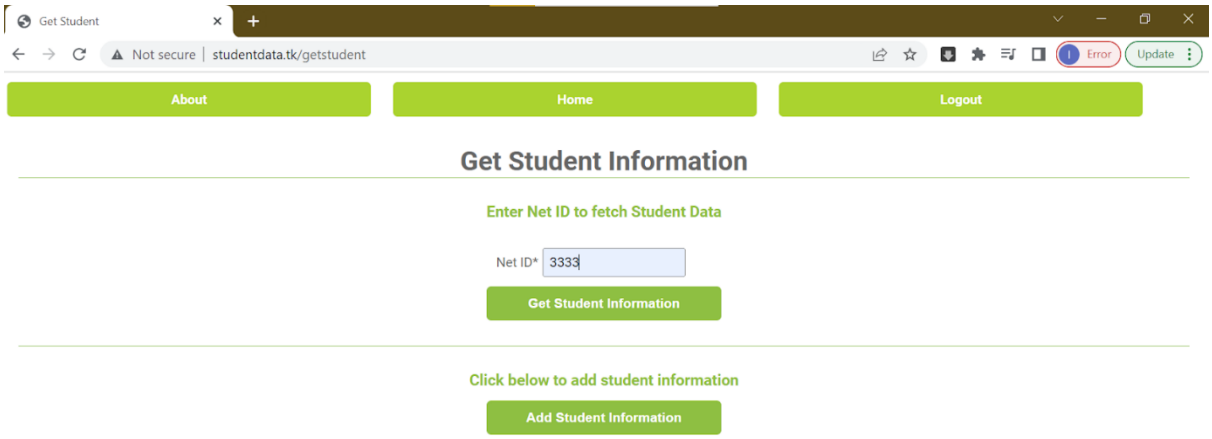
Once a student data is added on the website, the same data gets added to the RDS.

```
mysql> select * from student;
+-------+---------+----------+-------+-------------+
| netid | fname   | lname    | major | location    |
+-------+---------+----------+-------+-------------+
| 1111  | Emma    | Watson   | CS    | New york    |
| 2222  | Nick    | Jonas    | Arts  | Hayward     |
| 3333  | Natalie | Thompson | CS    | Los Angeles |
+-------+---------+----------+-------+-------------+
3 rows in set (0.00 sec)
```

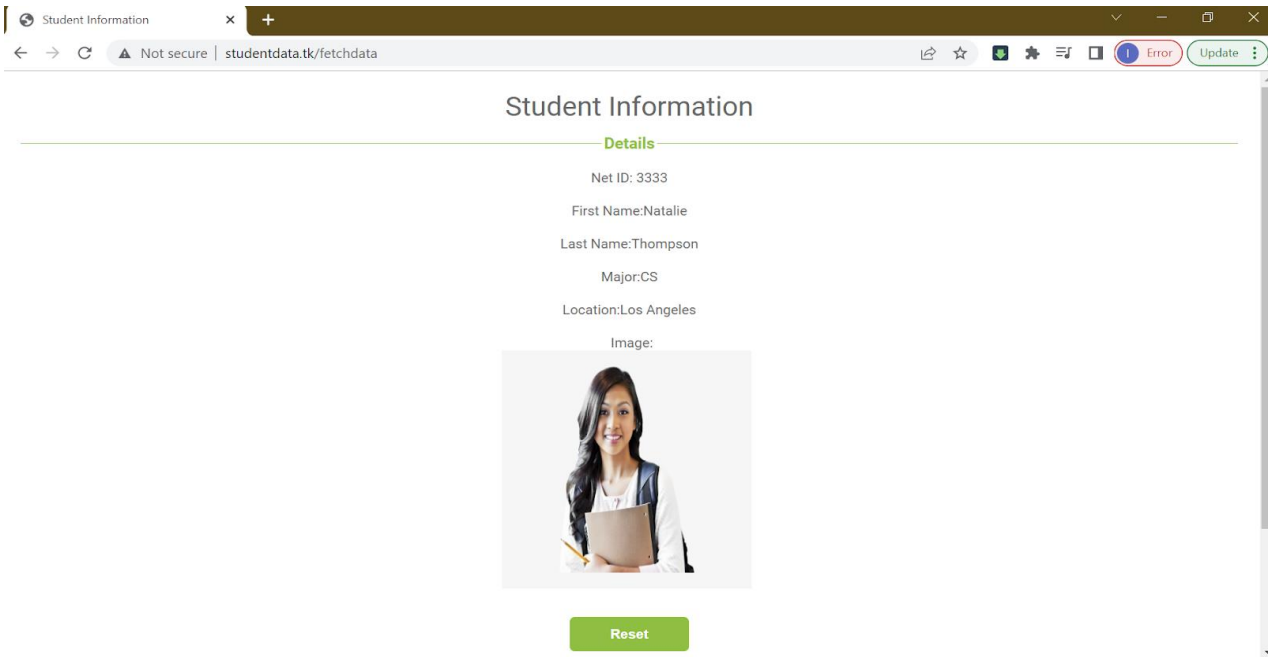
And the images get stored in S3 bucket as shown below.



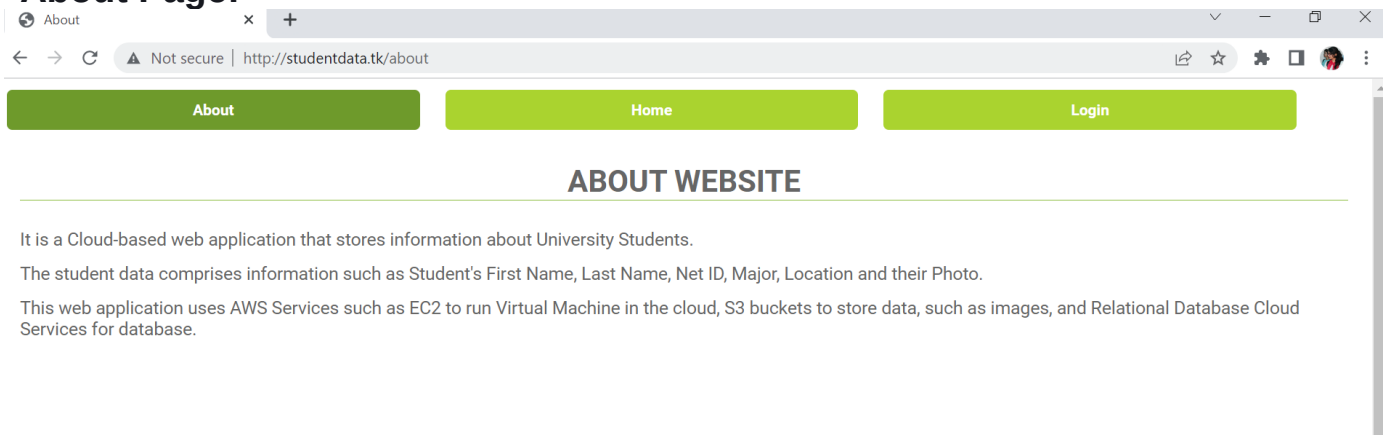
Get Student Information Page:



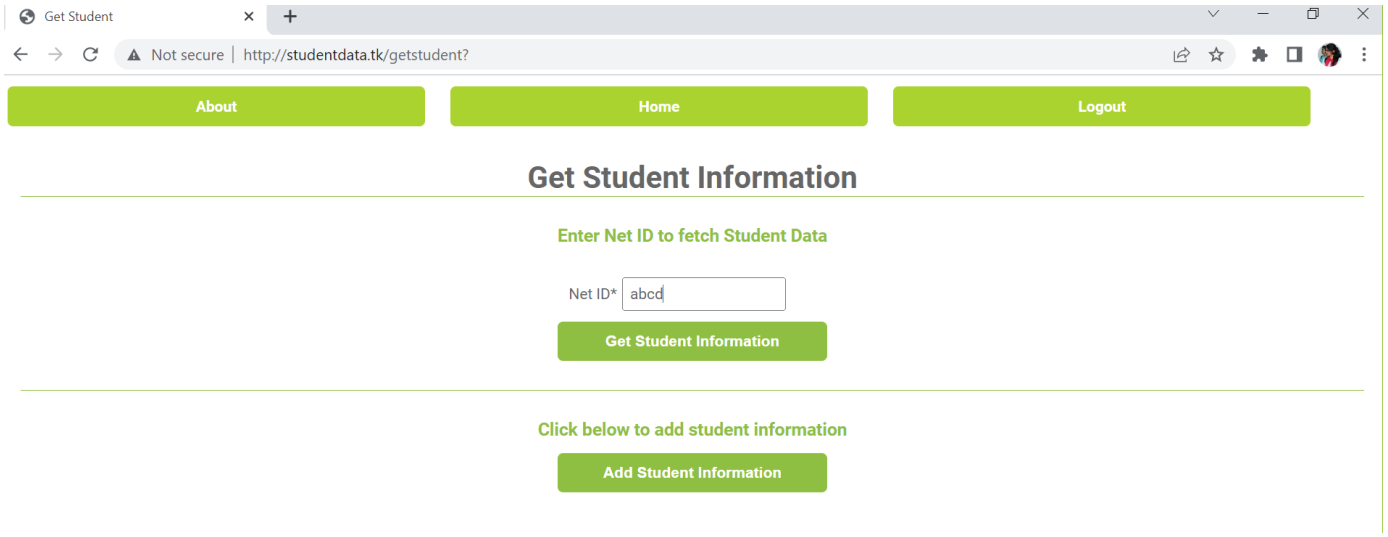
Once the net ID is given, the student data is fetched from RDS and S3 Bucket and is displayed as shown below.

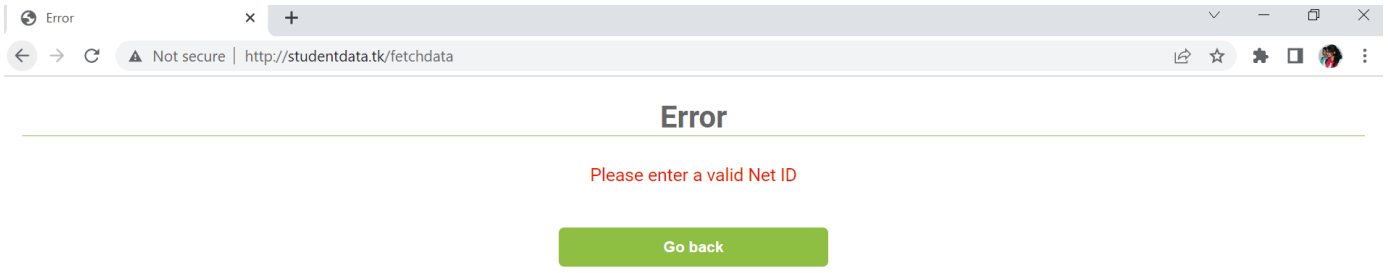


About Page:



When a wrong Net ID is given, an error is displayed.





CHAPTER 7: KEY FEATURES OF THE PROJECT

1. Usage of different cloud services ie, EC2, S3, RDS, Route 53, IAM
2. Elimination of Infrastructure, and upfront investment.
3. Enhanced level of flexibility and scalability in comparison to traditional data centers.
4. Storing and retrieving of data in a fast and efficient manner.
5. Simple Project management.
6. Accessible from anywhere in the world and Increased productivity.

CHAPTER 8: FUTURE ENHANCEMENTS

1. Every time we stop EC2 instance, the IPV4 address changes and hence need to reconfigure Route53 with new IP address in such instances. In the future scope of the project, we can overcome this challenge by using Elastic IP address.

2. Implementation of AWS DynamoDB instead of MYSQL database, as AWS DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second. So, AWS DynamoDB can be one of our future enhancements.

CHAPTER 9: REFERENCES

1. https://docs.aws.amazon.com/
2. https://docs.aws.amazon.com/s3/index.html
3. https://docs.aws.amazon.com/pythonsdk/
4. https://docs.aws.amazon.com/route53/
5. https://aws.amazon.com/rds/features/
6. https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html
7. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
8. https://aws.amazon.com/ec2/instance-types/?trk=36c6da98-7b20-48fa-8225-4784bced9843&sc_channel=ps&s_kwcid=AL!4422!3!467723097970!e!!g!!ec2&ef_id=Cj0KCQiAkMGcBhCSARIsAIW6d0ACH4wfakeYsWwP1Pjq4LMxg4oeWtcp36hK8CA9GndlHvNTNg46NmEaAlJjEALw_wcB:G:s&s_kwcid=AL!4422!3!467723097970!e!!g!!ec2
9. [1] “Flask web development, one drop at a time”, https://flask.palletsprojects.com/en/1.1.x/
   This is the official documentation of the Flask web application framework.
10. [2] "Introducing Python: modern computing in simple packages (Second edition.)", Bill Lubanovic, O'Reilly Media, November, 2019., Chapter 18.

    This is a book that discusses the basic usage of Python programming language within various topics, including the web systems. Bill Lubanovic is a senior software engineer currently working for Internet Archive. O'Reilly is a popular publisher that has published various books related to software development.