DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

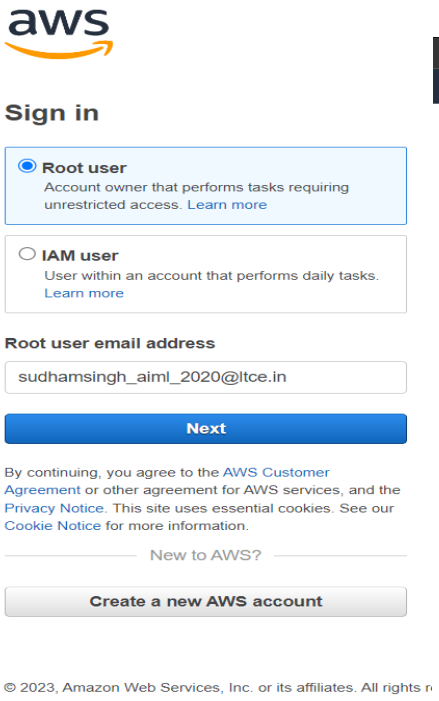
NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**CLOUD COMPUTING LAB**
COURSE CODE	**CSL605**
PRACTICAL NO.	**01**
DOP	**24/01/2023**
DOS	

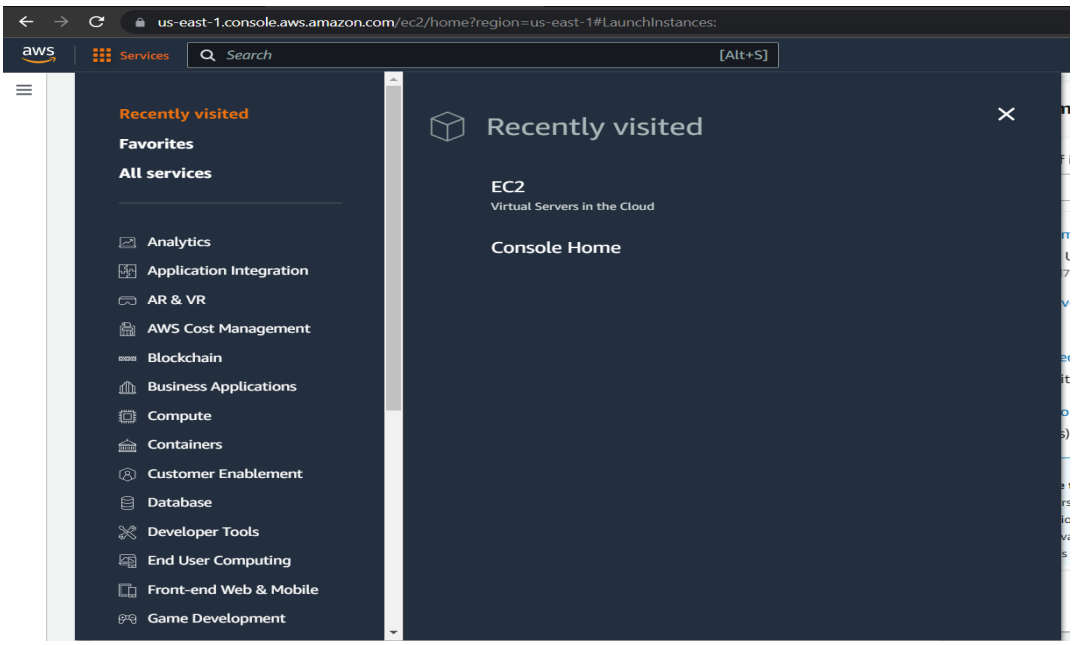
AWS (EC2) Installation steps for Linux instance

Please find the AWS account creation steps in the link.

- https://aws.amazon.com/premiumsupport/knowledge-center/create-andactivate-aws-account/
- https://signin.aws.amazon.com/signin?redirect_uri=https%3A%2F%2Fconsole.aws.amazon.com%2Fsupport%2Fhome%2F%3Fnc2%3Dh_ql_cu%26state%3DhashArgs%2523%26isauthcode%3Dtrue&client_id=arn%3Aaws%3Aiam%3A%3A015428540659%3Auser%2Fsupportcenter&forceMobileApp=0

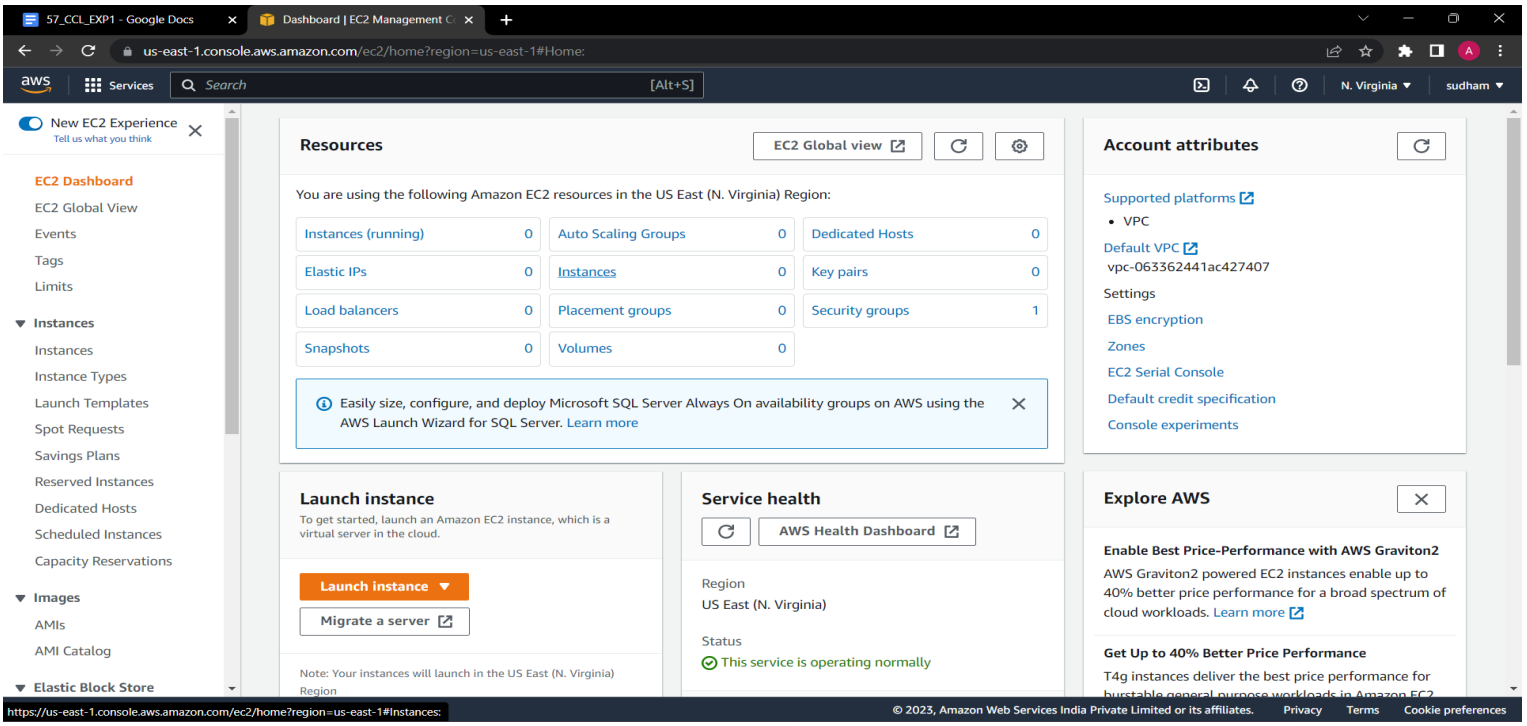
Requirement – Amazon web service account



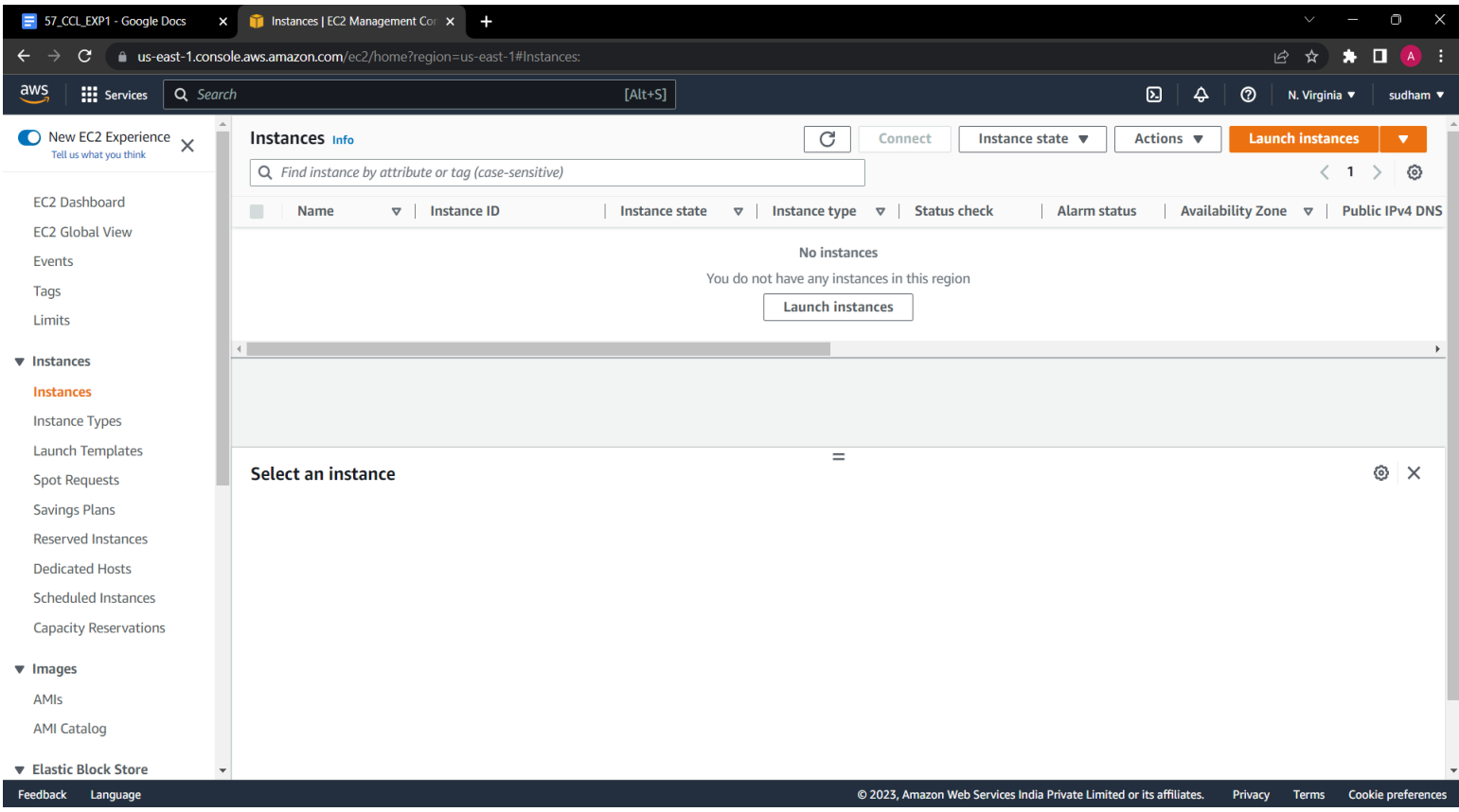


Step 0 : Login

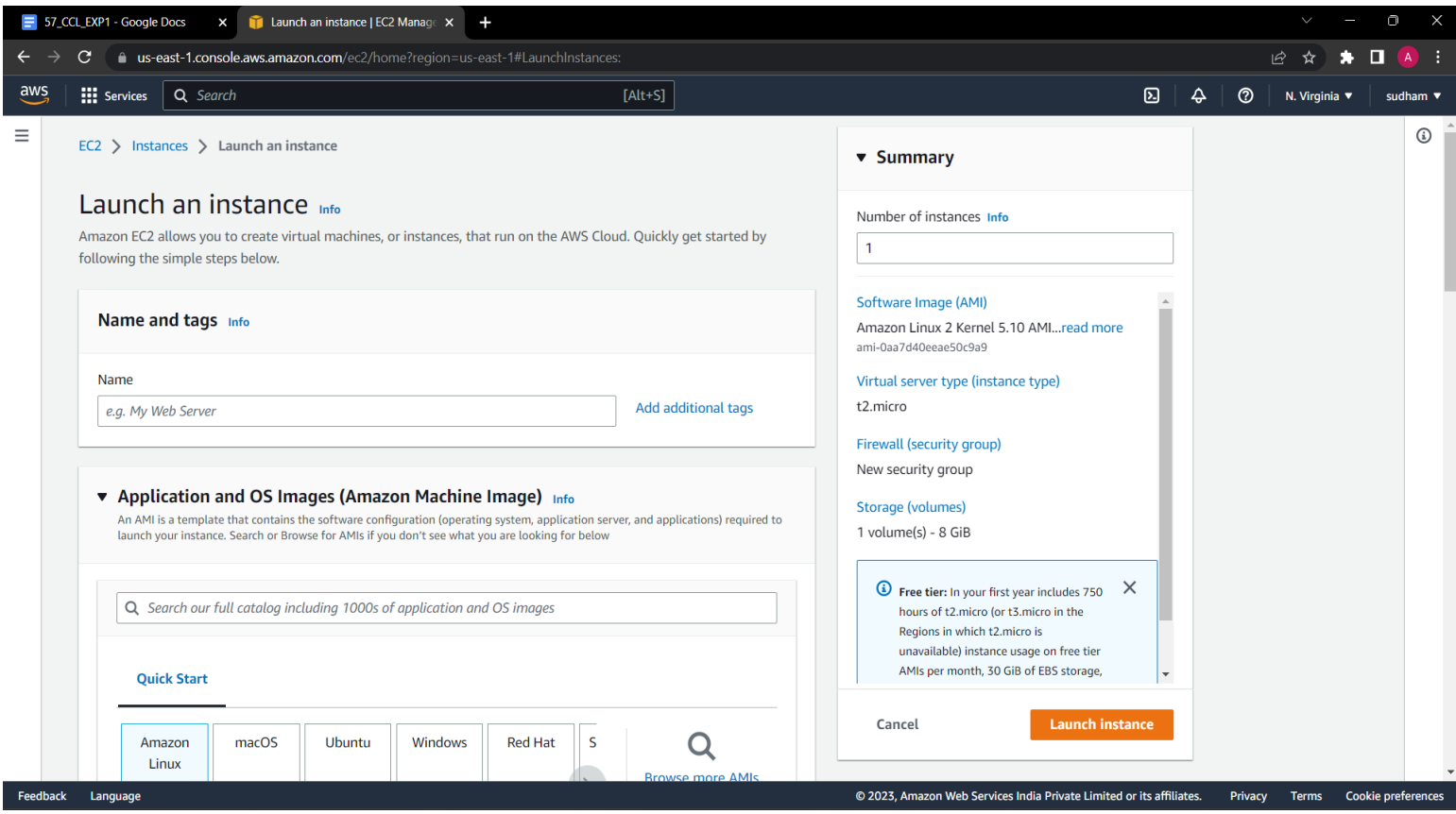
Step 1 : Open **AWS services** and select **EC2**



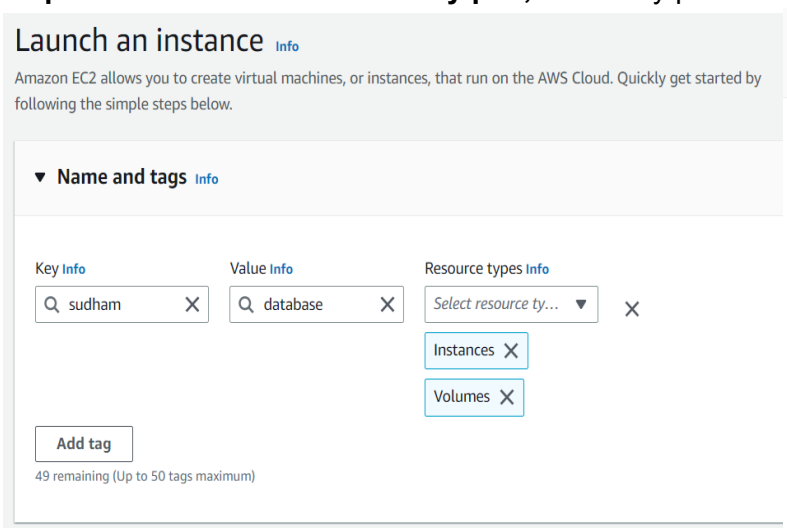
Step 2 : CLICK ON **instance (running)** in above screen

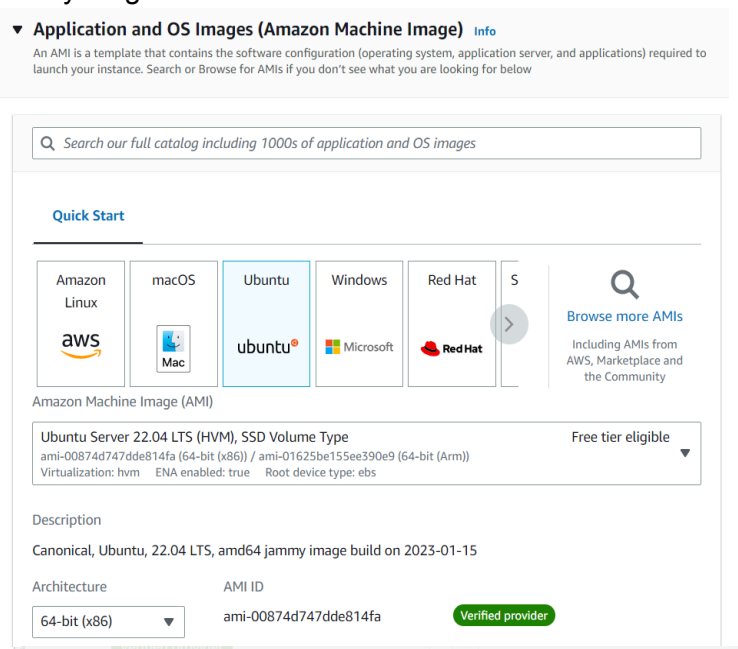


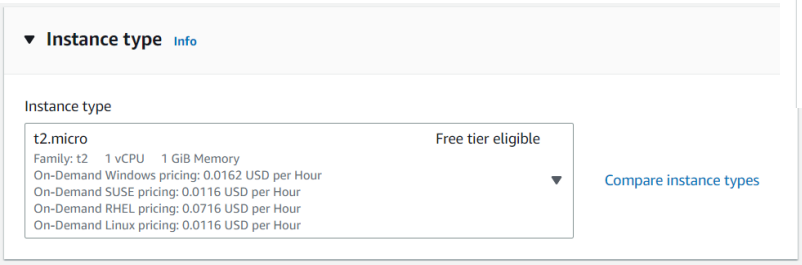
Step 3 : On above screen click on **launch Instances**

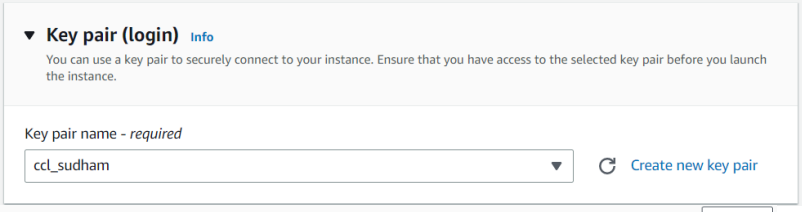


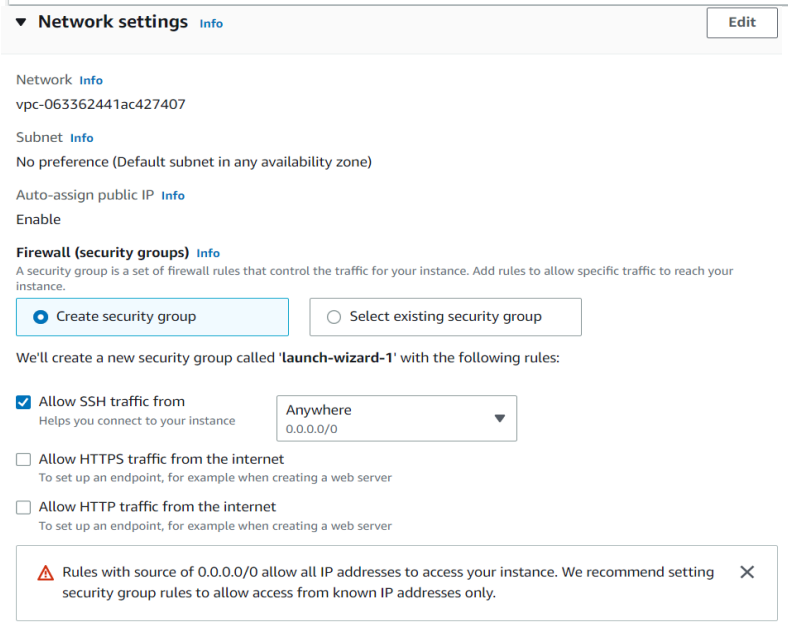
Step 4 : Select **Ubuntu server 22.04** and Select **free tier eligible** and click on button– **Next:**configure instance details
Step 5 : Don't change any setting directly click on button – **Next:** Add storage
Step 6 : Check Volume type: **General purpose SSD (gp 2)** and then click on button – next: **Add Tags**
Step 7 : Need to add key so click on **Add Tag,** In key tab give **any name**, value – **database**
Step 8 : Click on **Create a new key pair,** Insert key pair **name** anything like – test

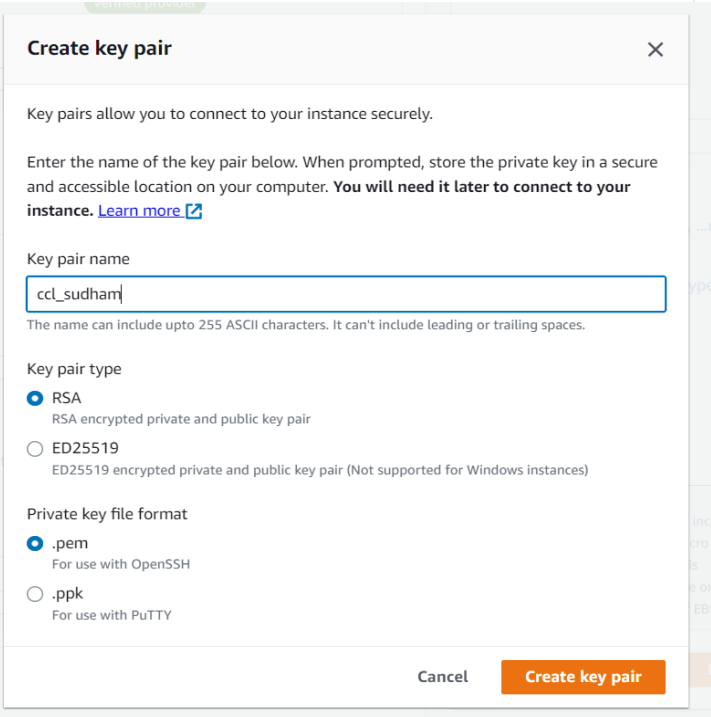


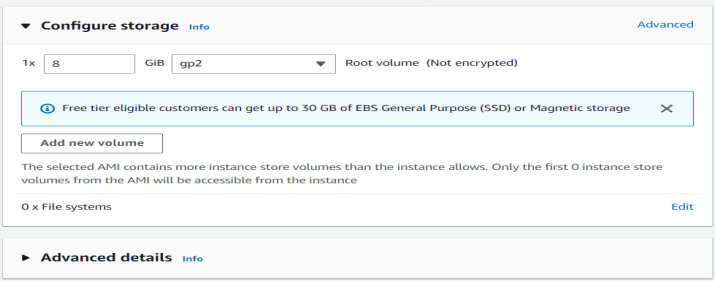




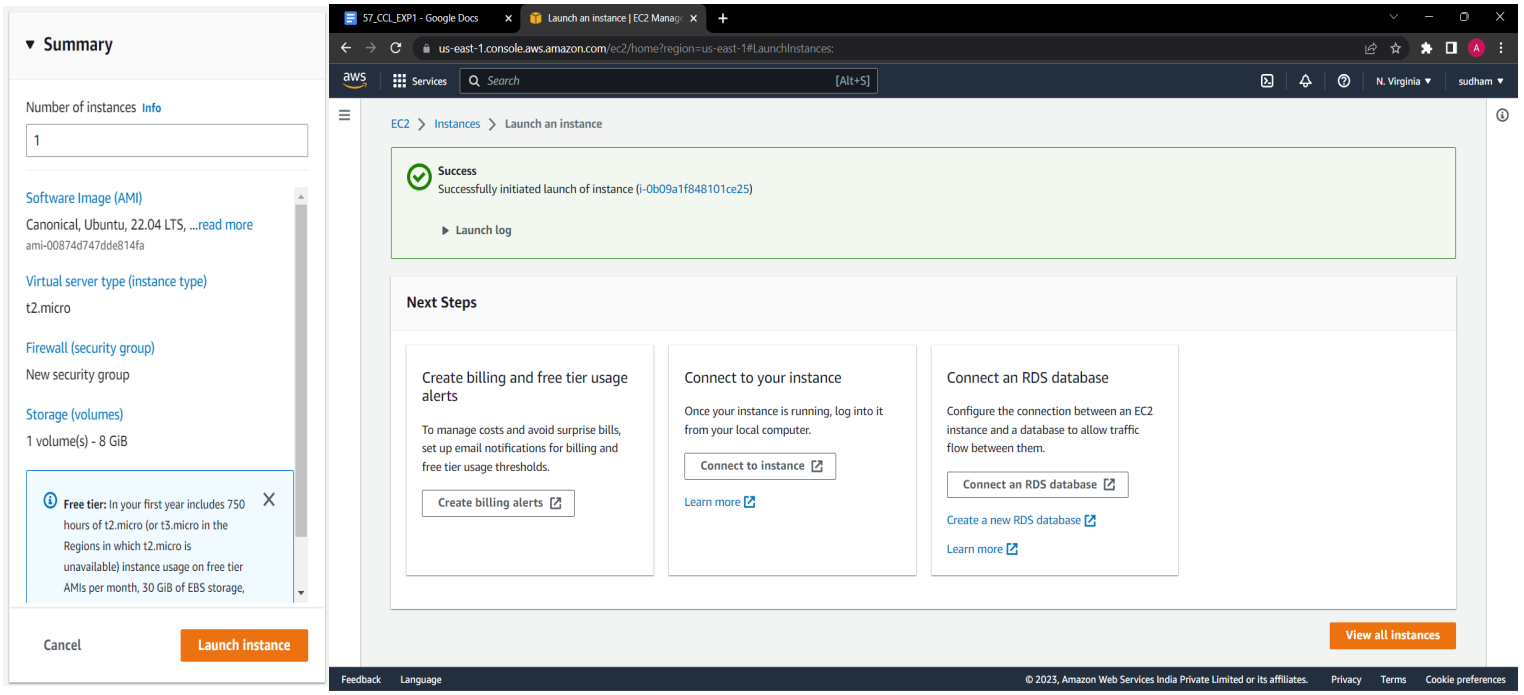








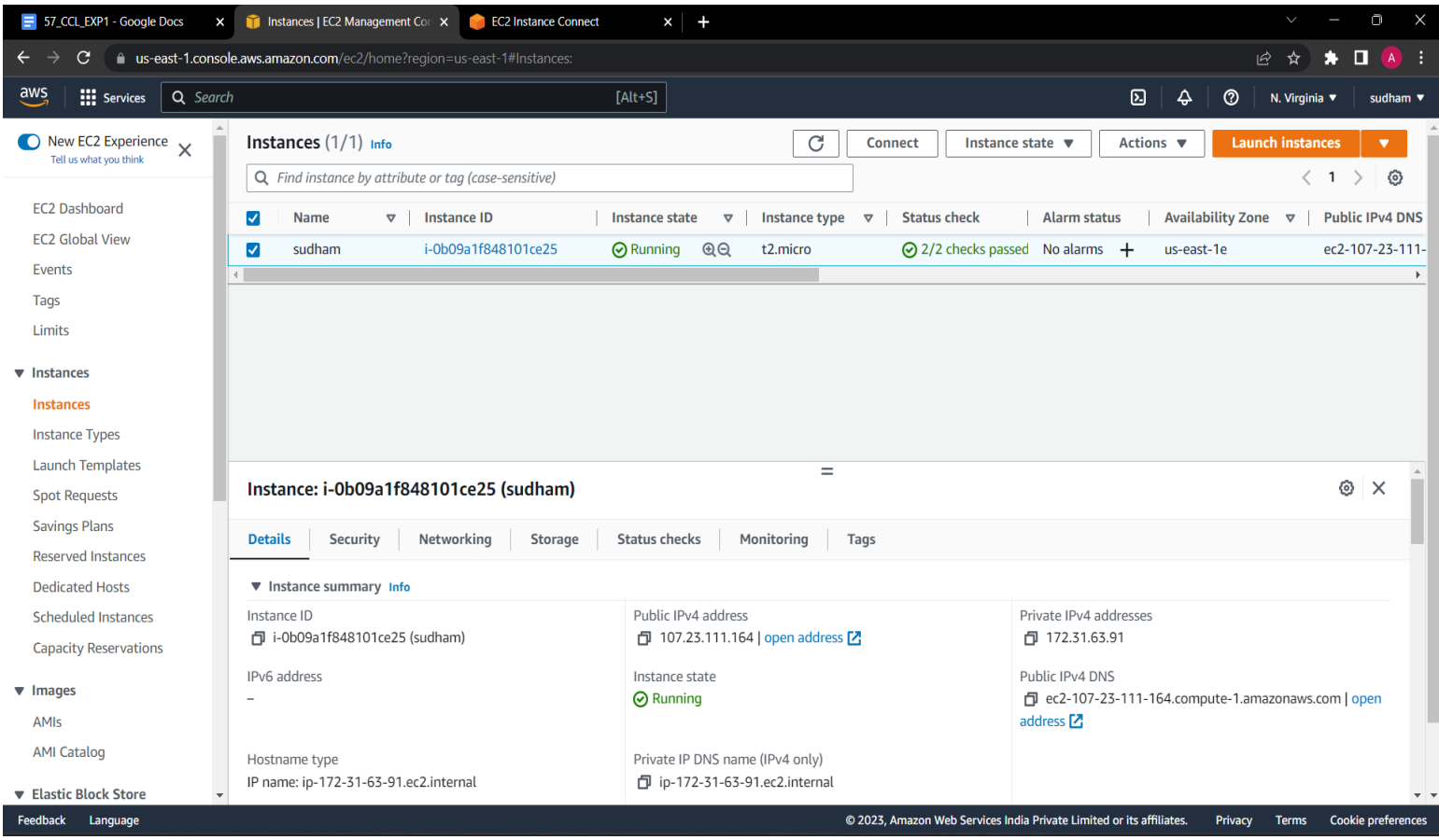
Step 9 : Click on **Launch Instances**



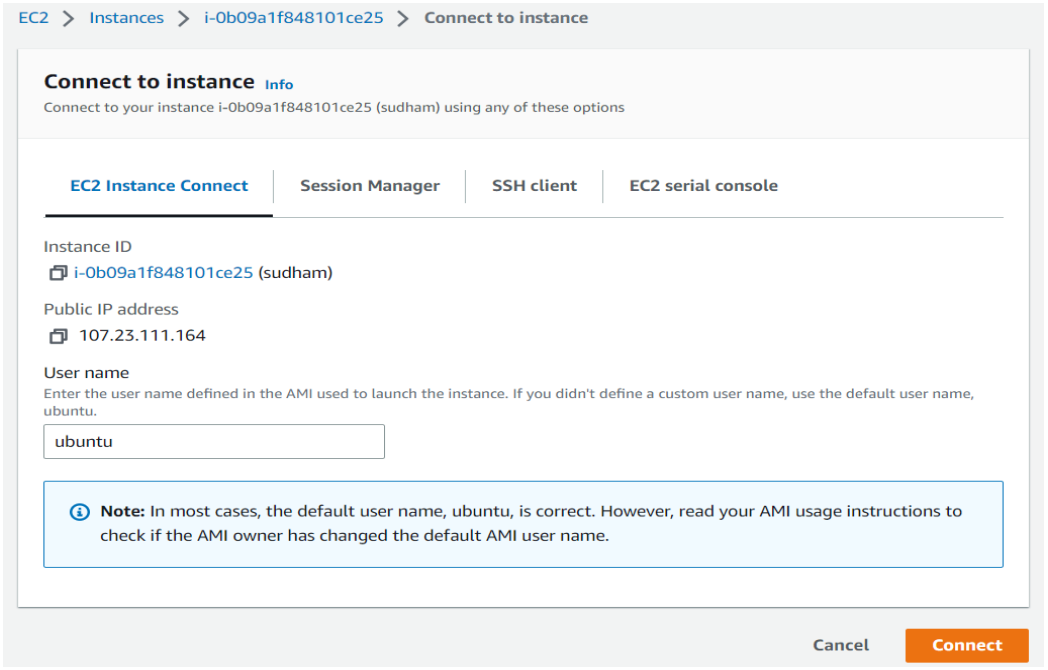
Step 10 : You get the mgs in green color – click on **View Instances**

Step 11 : Click on refresh button because we need Status check tab – **2/2 check pass**

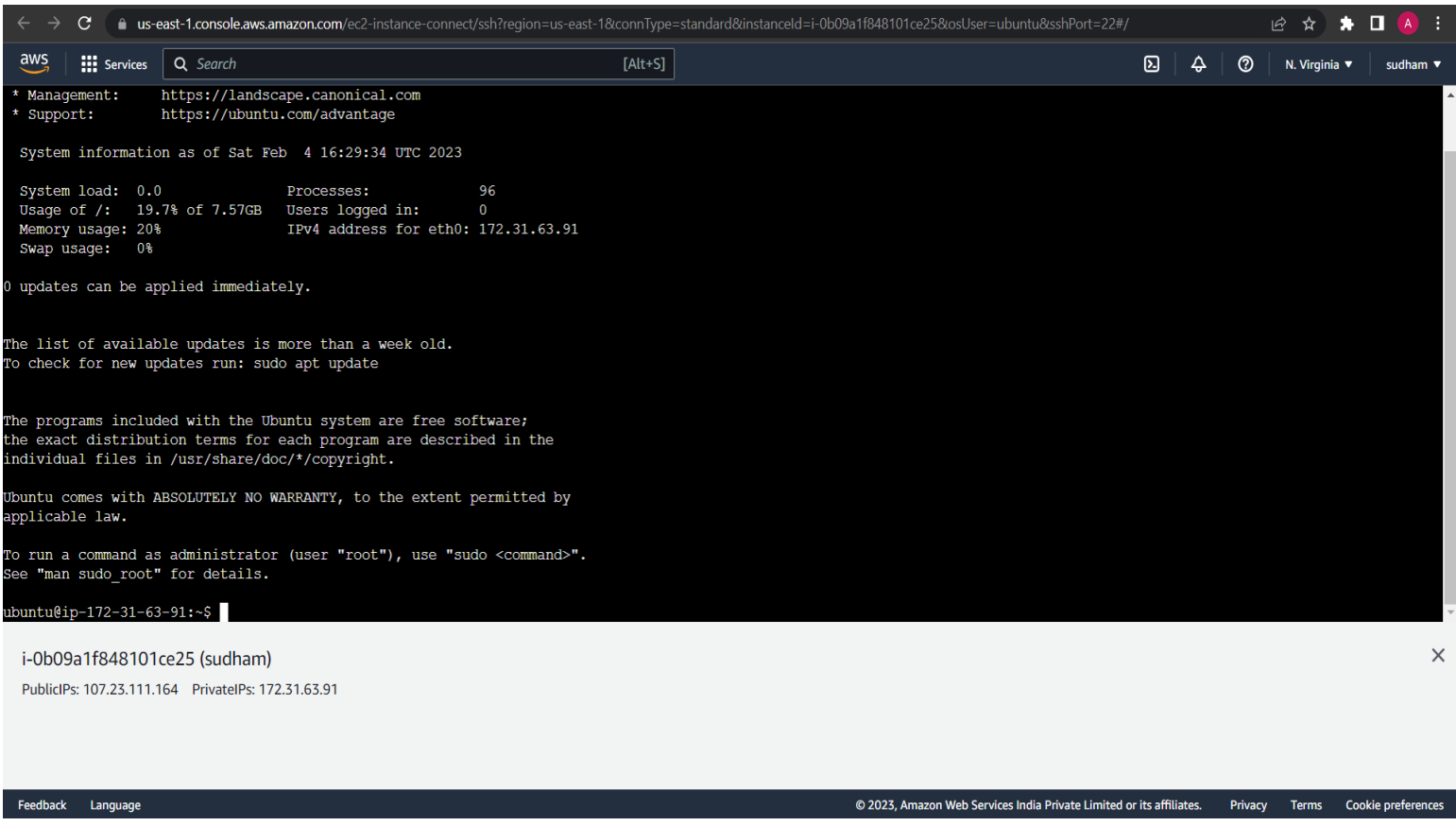
Step 12 : Select check box which in present at before name DATABASE and click on **connect Button**



Step 13 : Here you can change the user name if you want – like db_sg1, Step- click on **Connect**

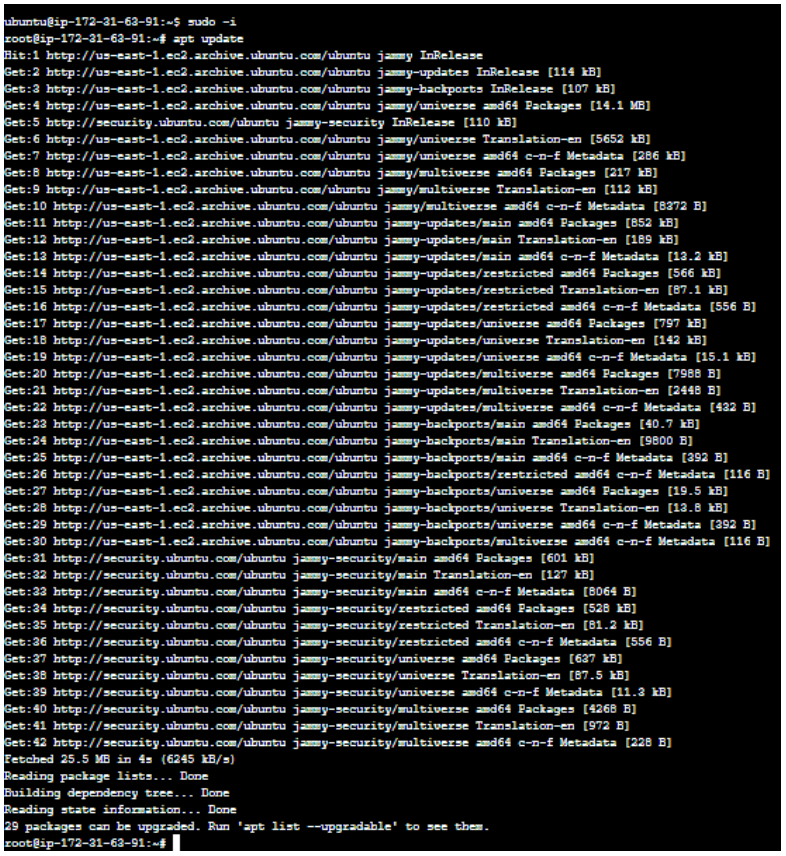


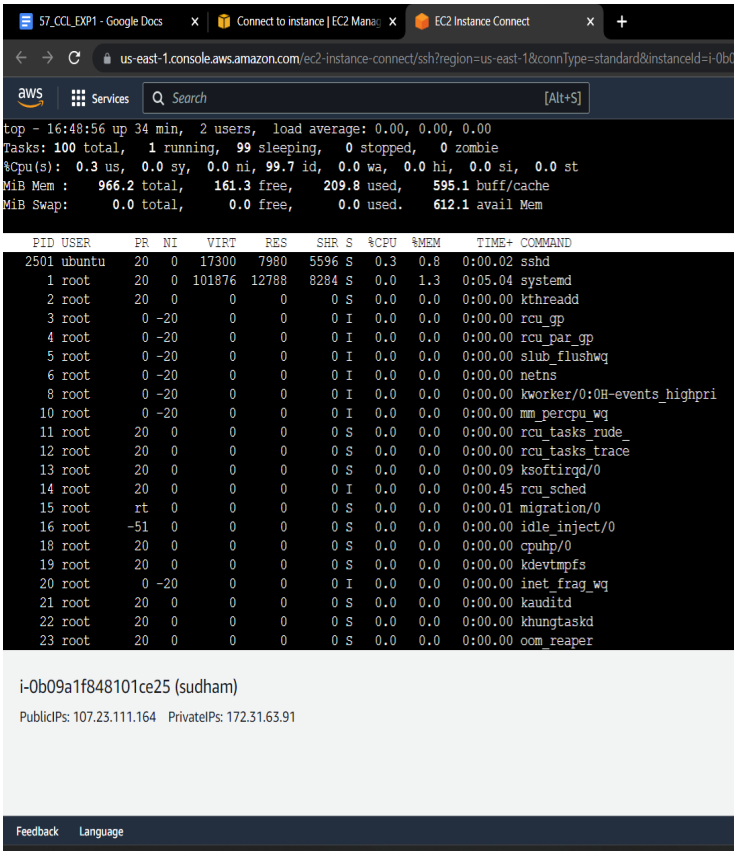
Step 14 : Next step perform some **command** on your Ubuntu server-
$ **sudo –i** (root login)
# **apt update** (update your system)



top (table of processes, which are the running processes in our system and also check usage management)

Press **Ctrl+c** or press **q** for the end top command.





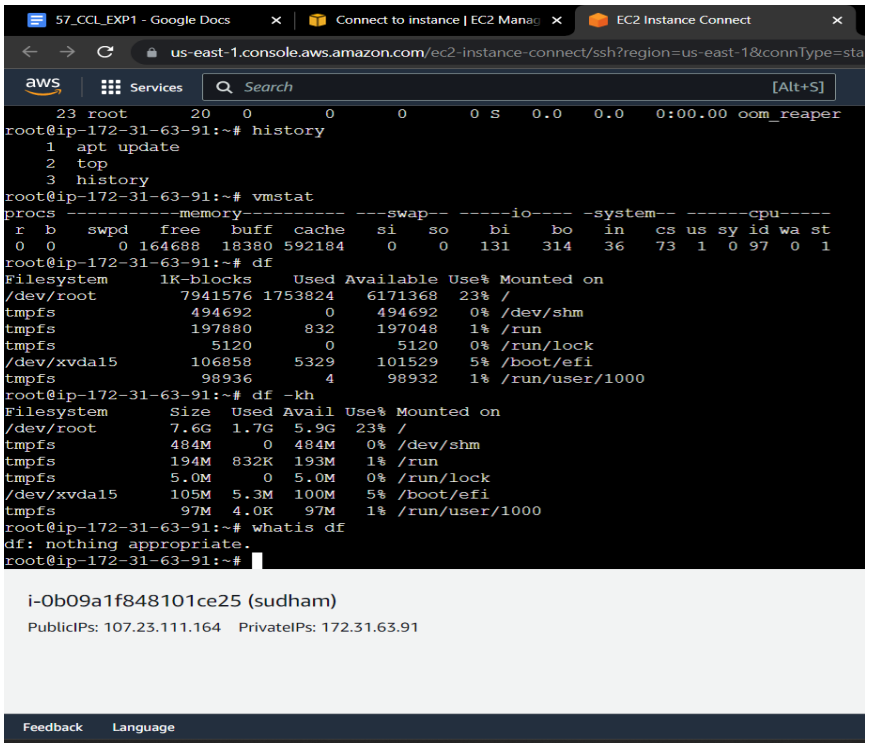
history

vmstat (virtual memory static ,how much memory in the buffer,in the cache, what is in the input,output,systems and the cpu)

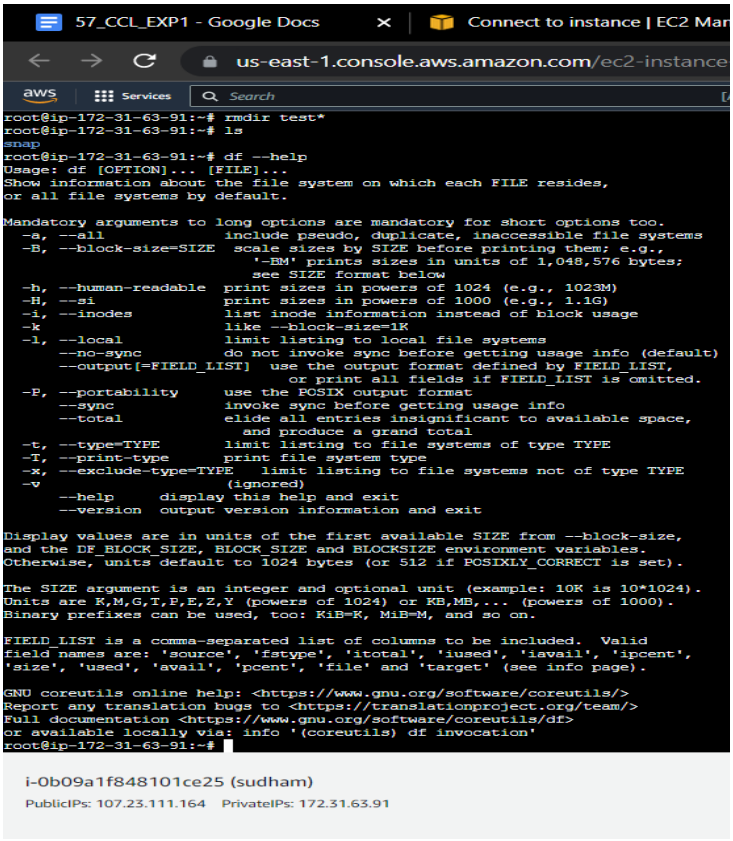
df (disk file system)

#df –kh (k-kilobyte h-human readable)

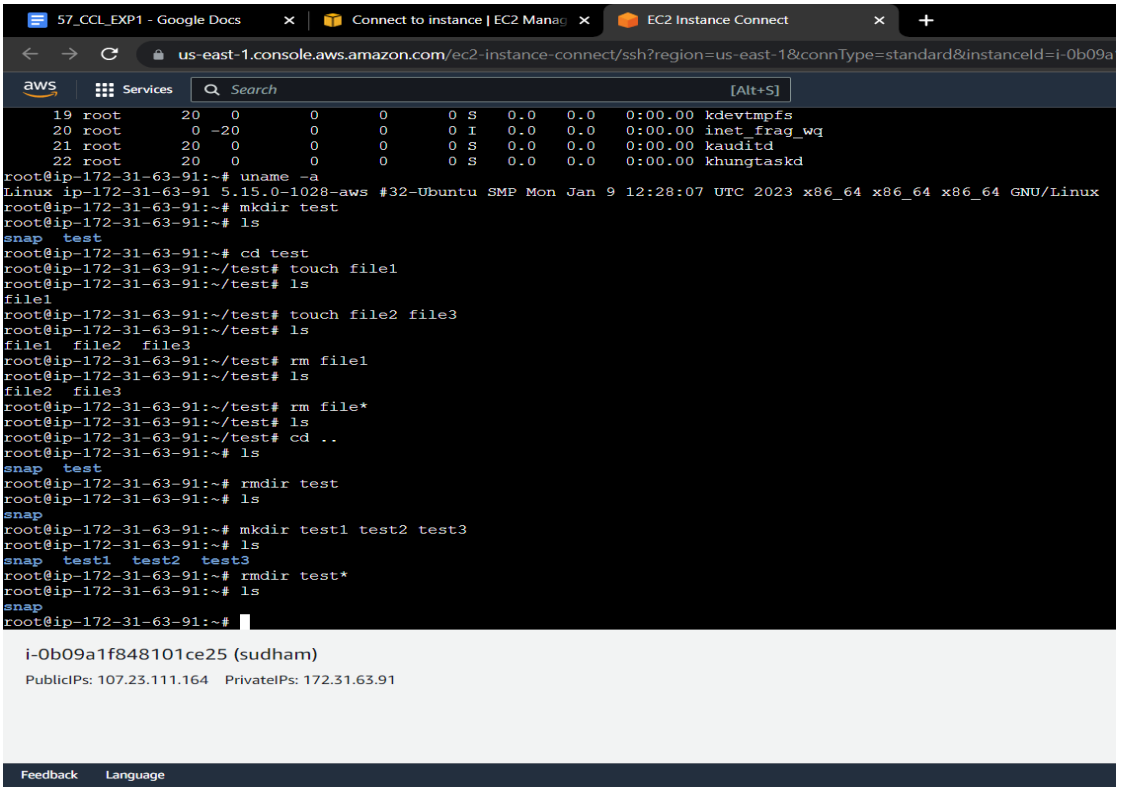
#whatis df (using whatis command take help from system)



#df - - help (help command)
 # ctrl + l (clear the screen)
#uname -a (information related to ip, kernel version)
All are Validation steps for checking your EC2 instance working properly or not(check system performance).

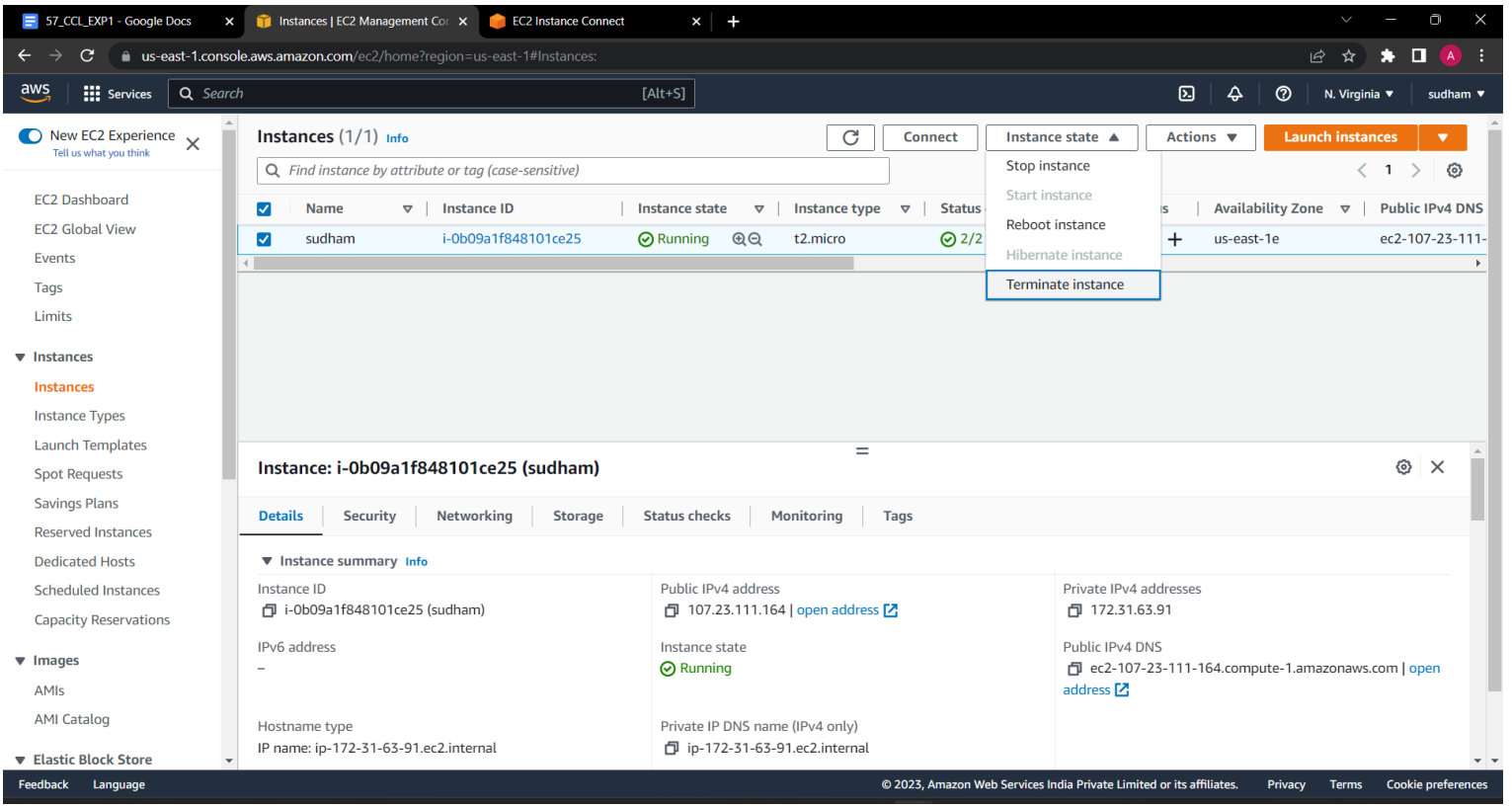


 # mkdir test
 # ls
 # cd test
test# **touch file1** (create file)
 # ls
 # touch file2 file3
 # ls
 # rm file1 (remove file)
 # ls
 # rm file* (remove all files)
 # ls
 # cd ..
 # ls
 # rmdir test
 # ls
 # mkdir test1 test2 test3
#ls
 # rmdir test*
 # ls

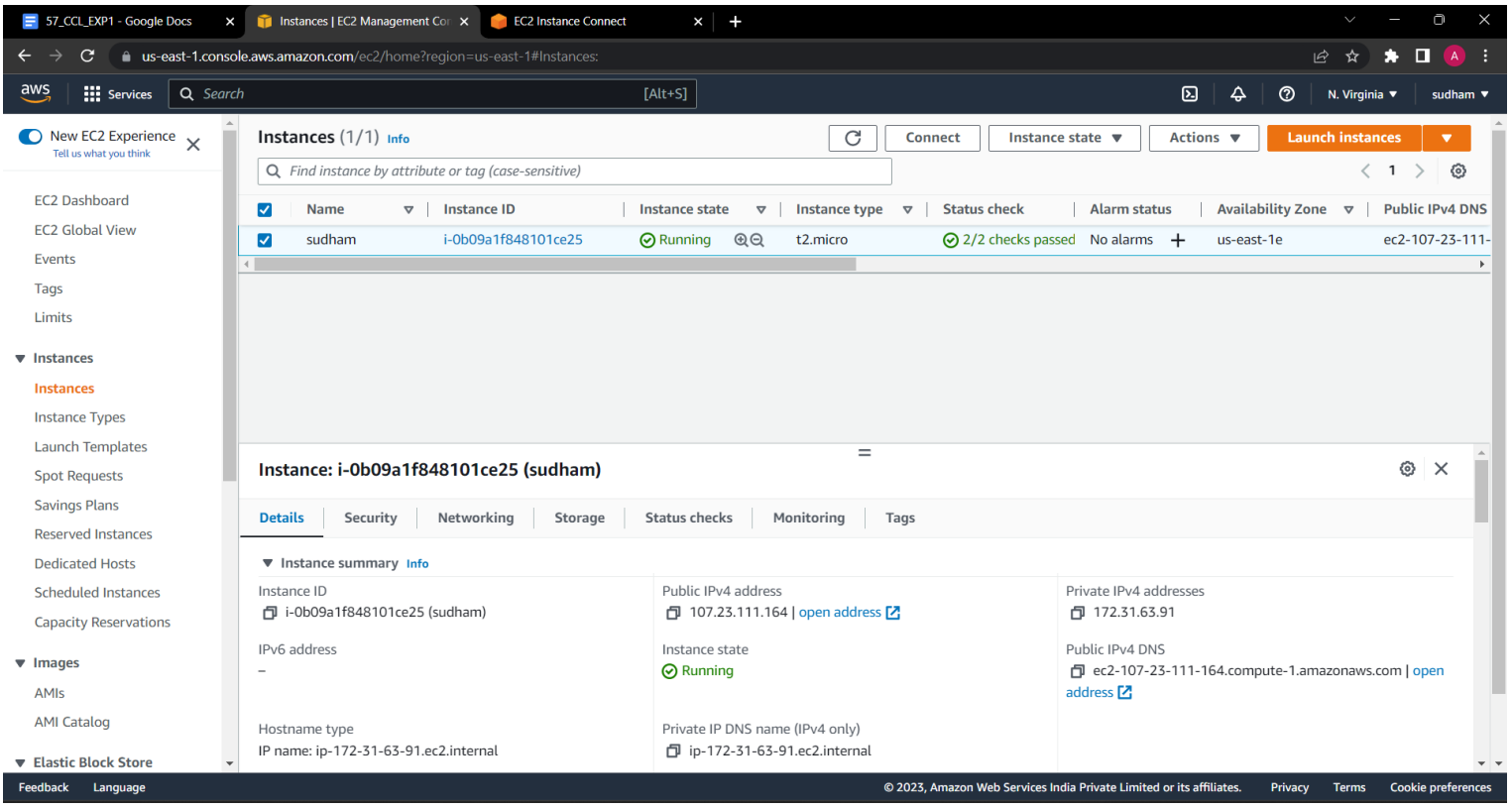


Step 15 : To terminate the instance -

Select the checkbox which is available at the start of your name of instant , then click on **Instant State** Button on the top and select **Terminal Instance**



Step 16 : If you want to terminate then this will, and if not the following instance will show as it is !



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

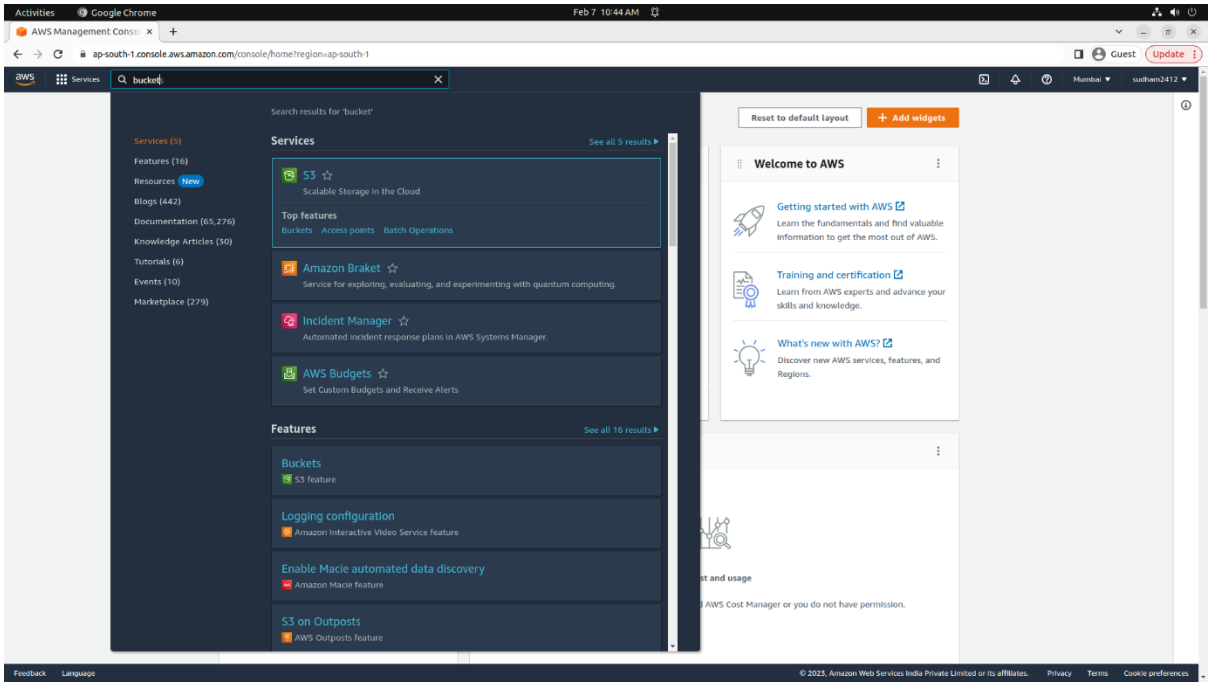
T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**CLOUD COMPUTING LAB**
COURSE CODE	**CSL605**
PRACTICAL NO.	**3**
DOP	
DOS	

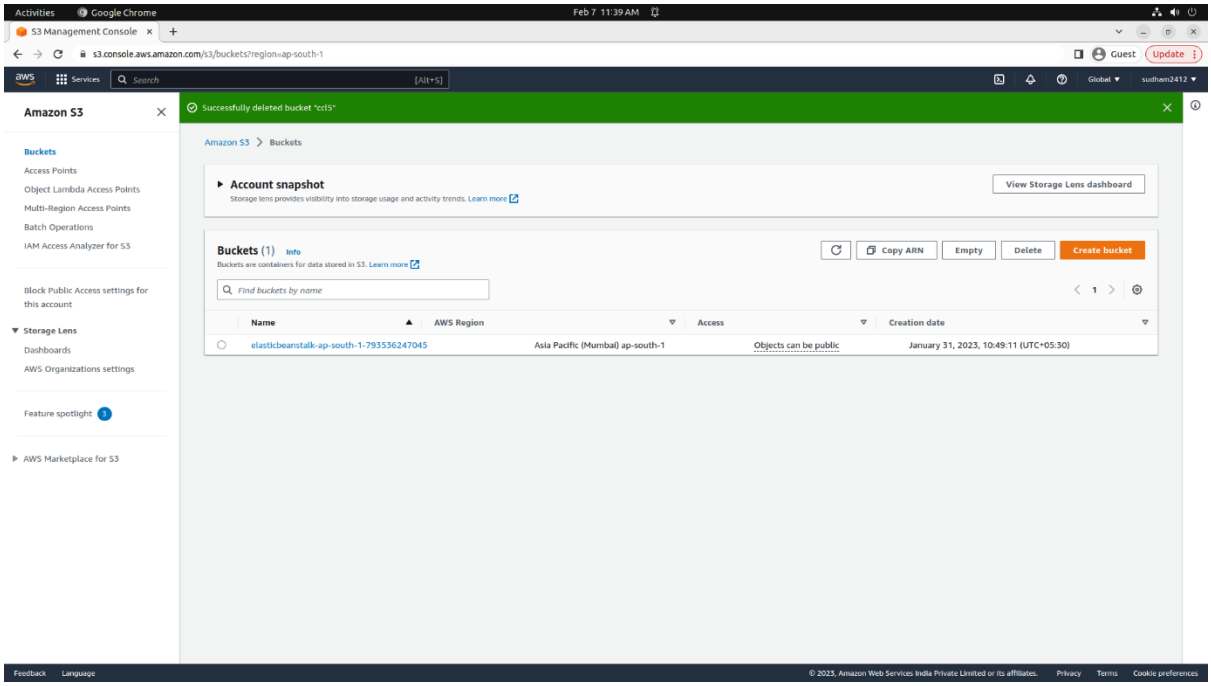
AMAZON S3 (BUCKET)

OUTPUT:

STEP 1: Login to AWS console and go to **Amazon S3 BUCKET**.

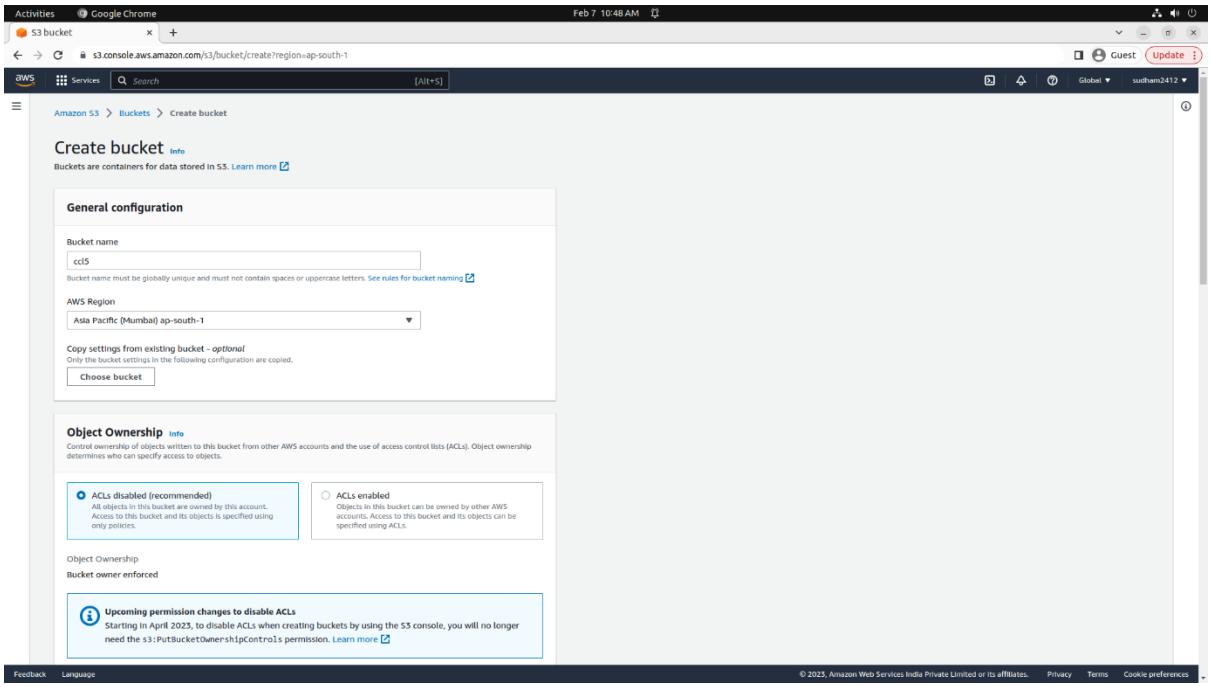


Click on **Create Bucket**.



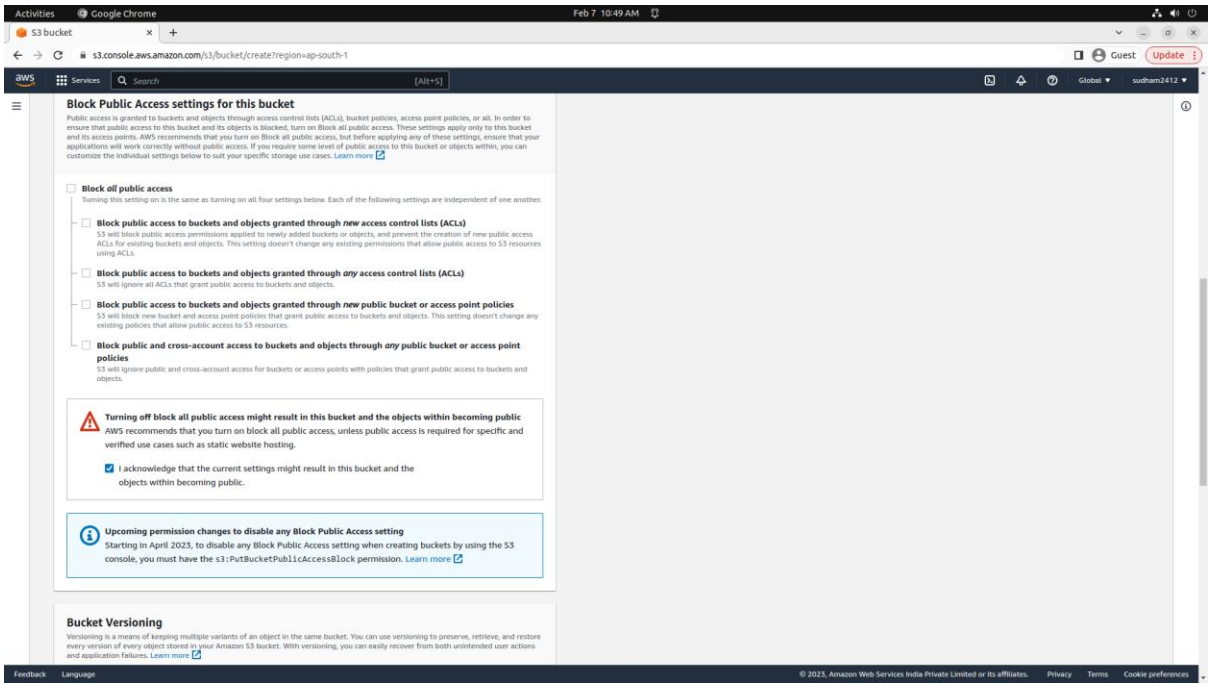
STEP 2: Give **Bucket name** & select **region** for storage.

STEP 3: Keep object ownership setting as **ACLs Disabled** as by-default.



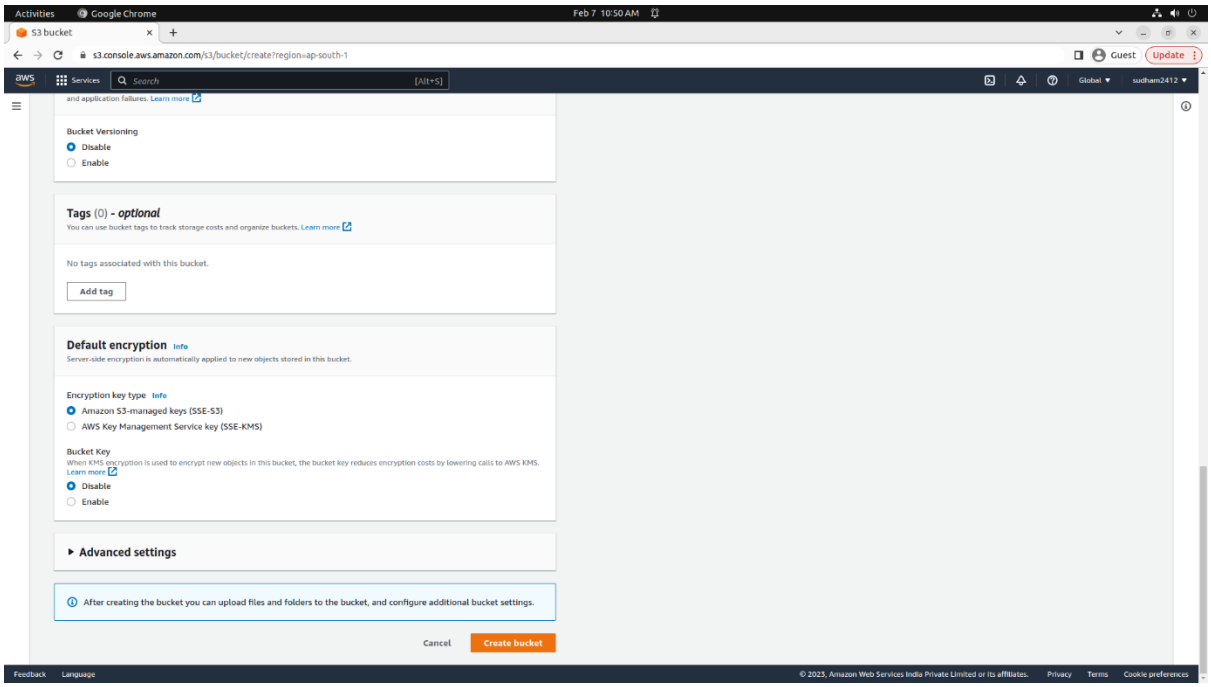
STEP 4: Disable block all **public access** checkbox.

STEP 5: Select the checkbox for **Turning off block all public access might result in this bucket and the objects with in becoming public**.

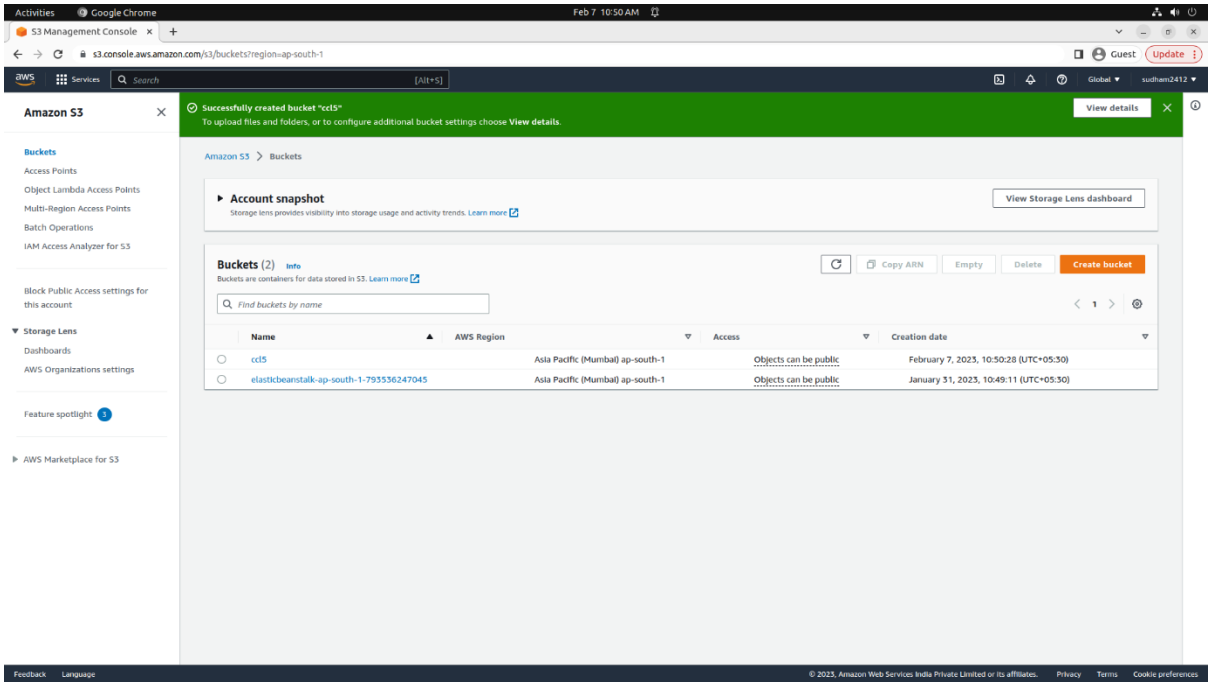


STEP 6: Keep **Bucket Versioning** as **Disabled** and **add tags** if required.

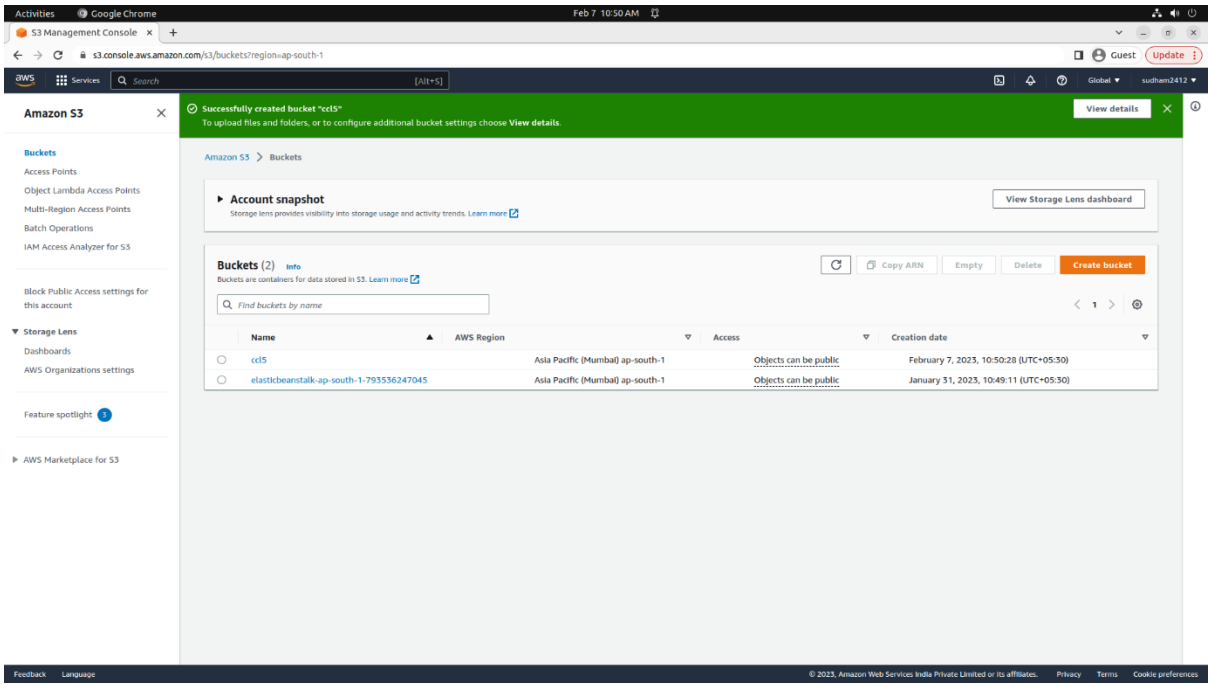
STEP 7: Keep **Default encryption** Disabled and click on **create bucket** button.



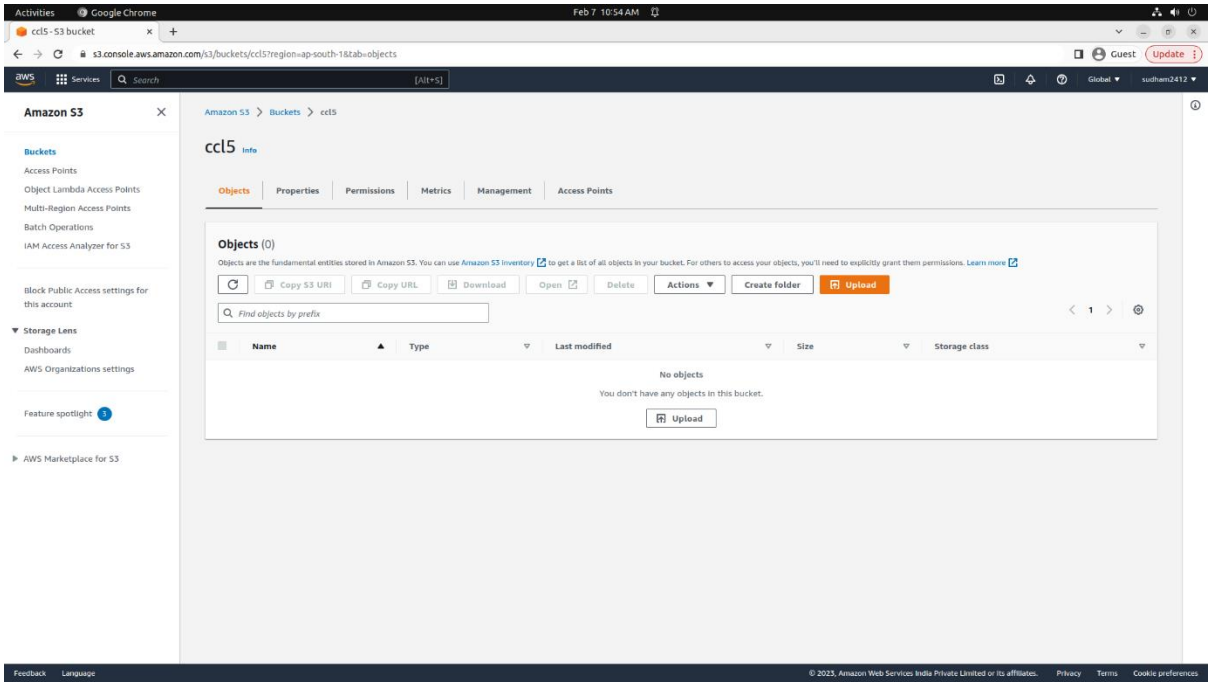
You can now see the successful creation of your **Bucket**.



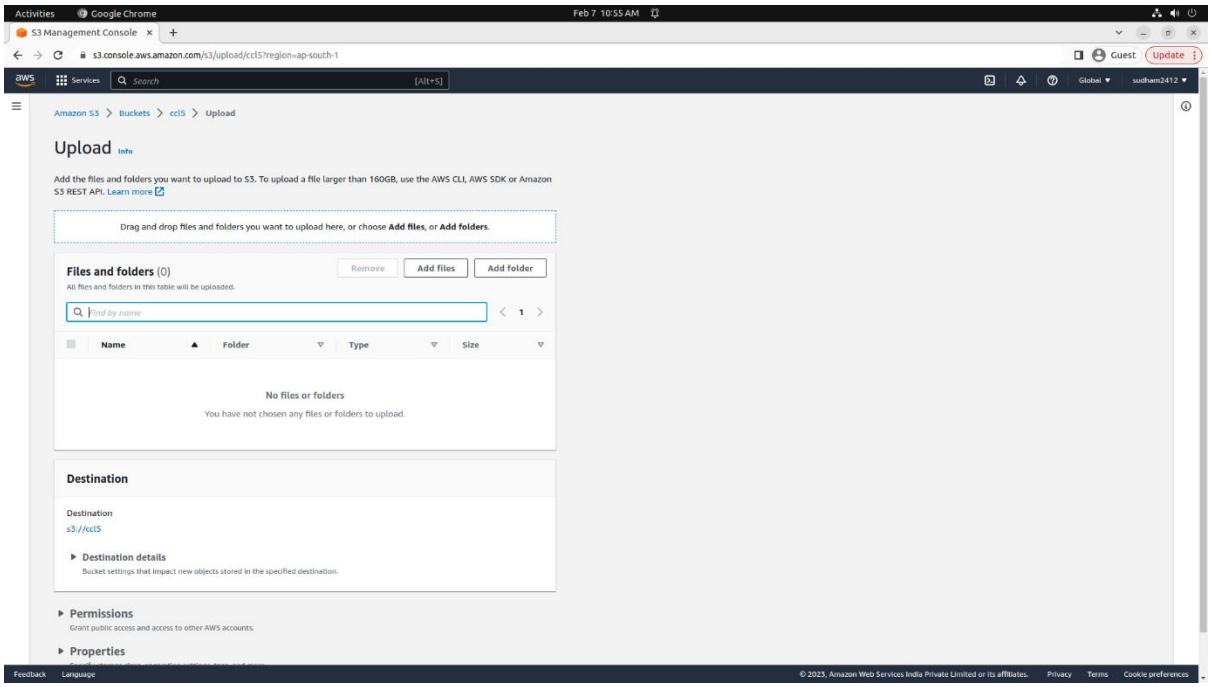
STEP 8: Now click on the **Bucket** that you have created.



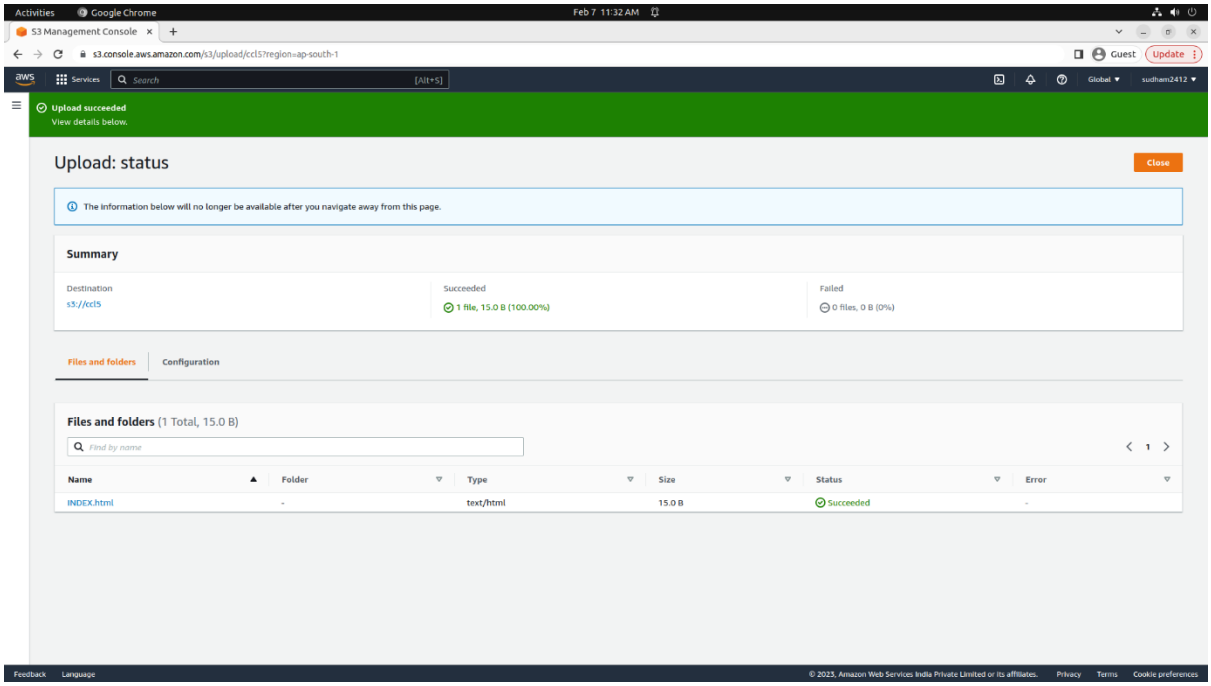
STEP 9: You can either **create a folder** here or **upload** an existing file in the Bucket.



STEP10: Now click on **upload** button and click on **add files** button browse your local machine and select which file you need to upload on S3 next click on **upload button** at bottom right end.

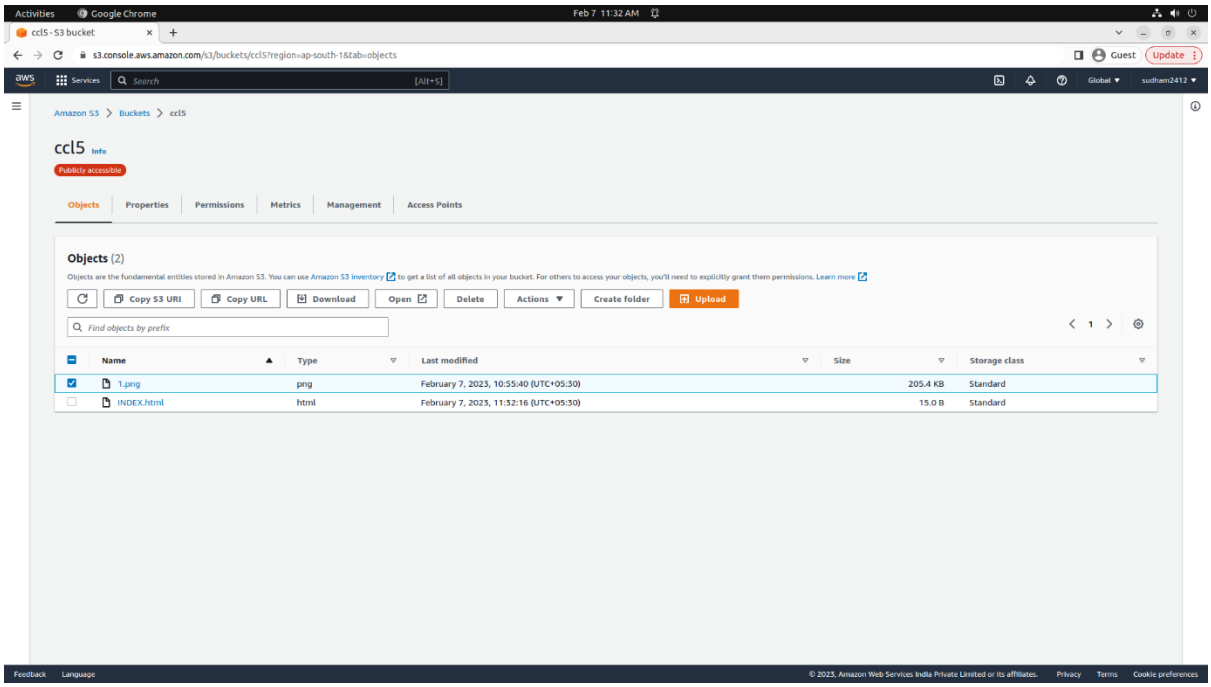


Now you can check the **upload status** screen.

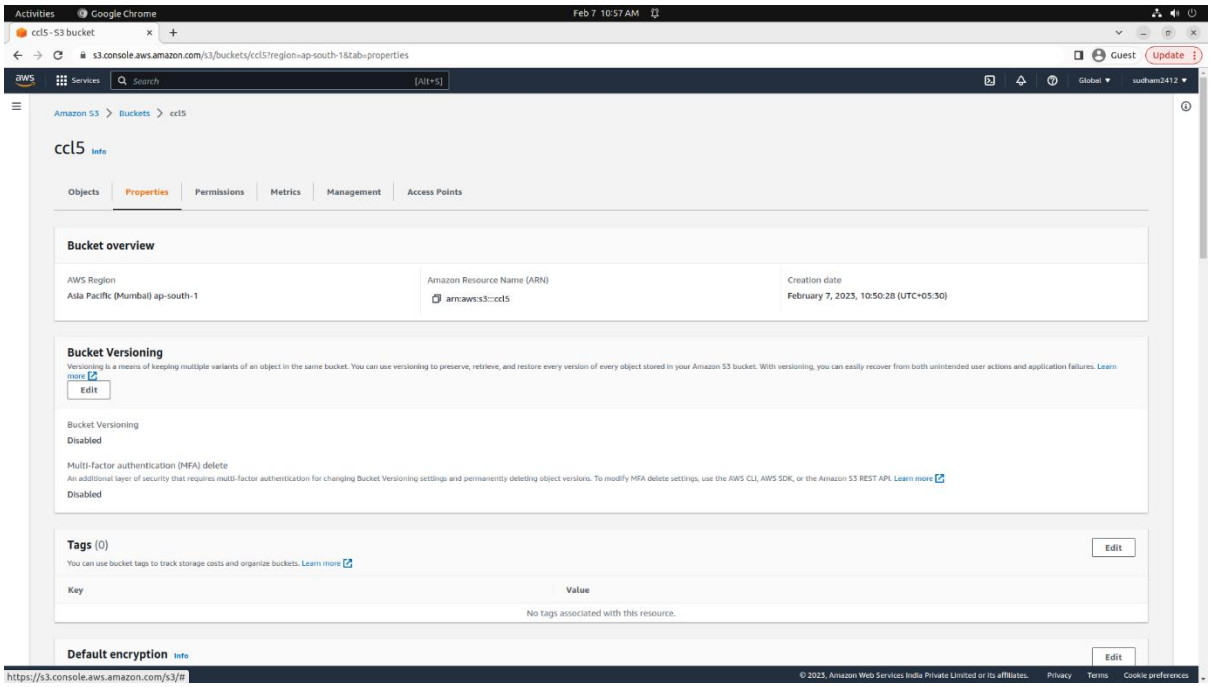


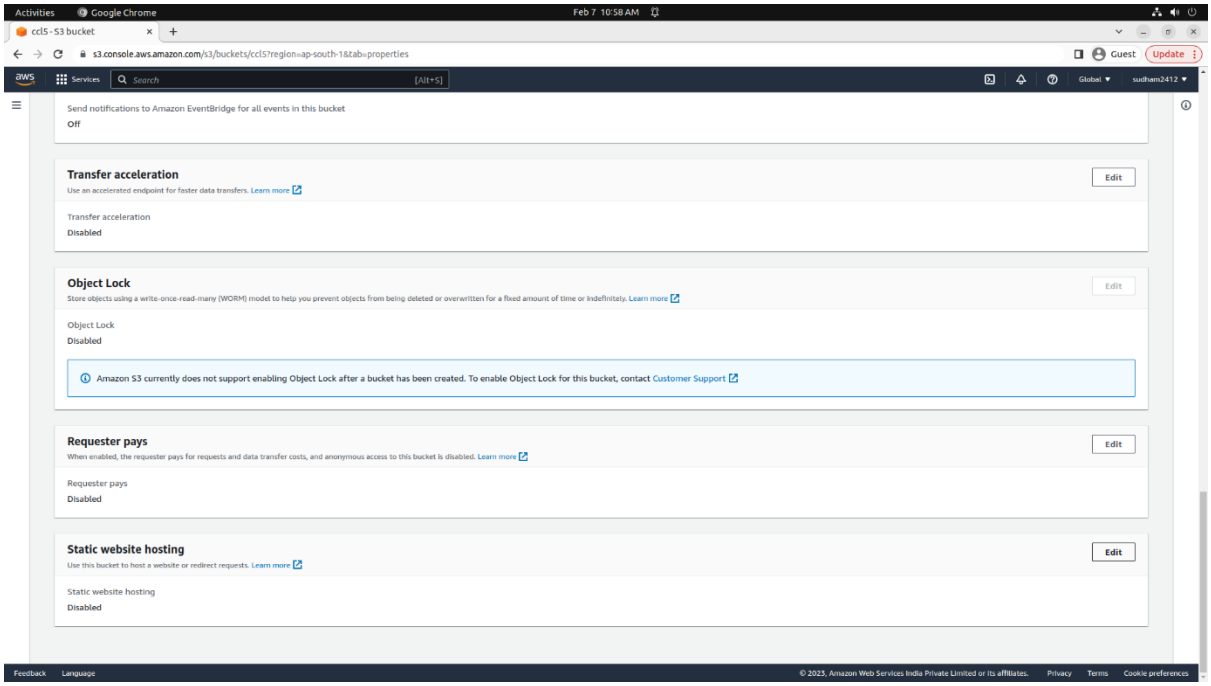
Now click on **close** button.

The screen will appear as below

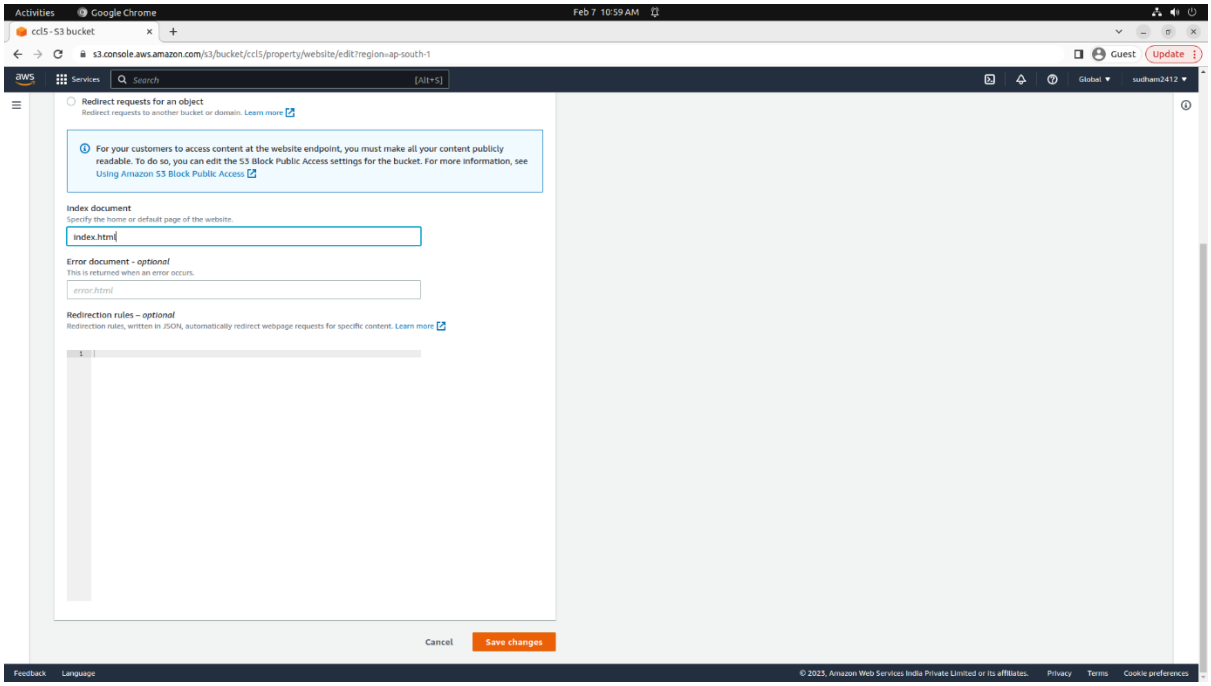


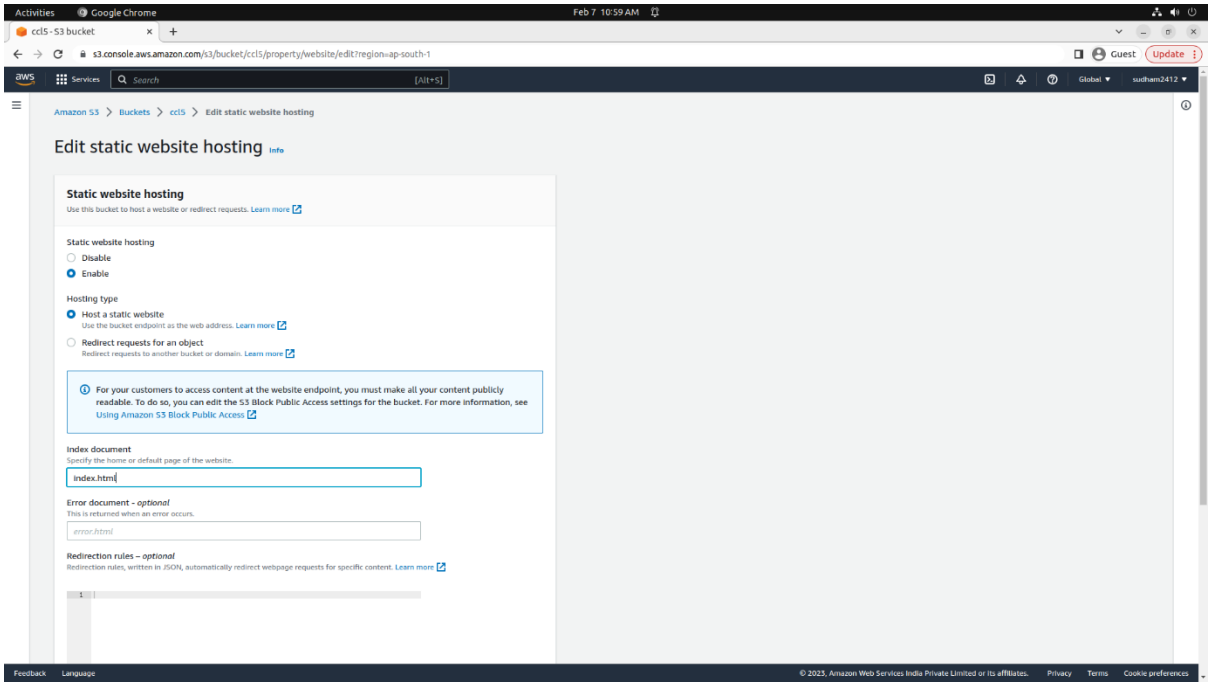
STEP 11: Select **Properties** and scroll down to **Static website hosting** option which is Disabled now click on **Edit** option on right side.





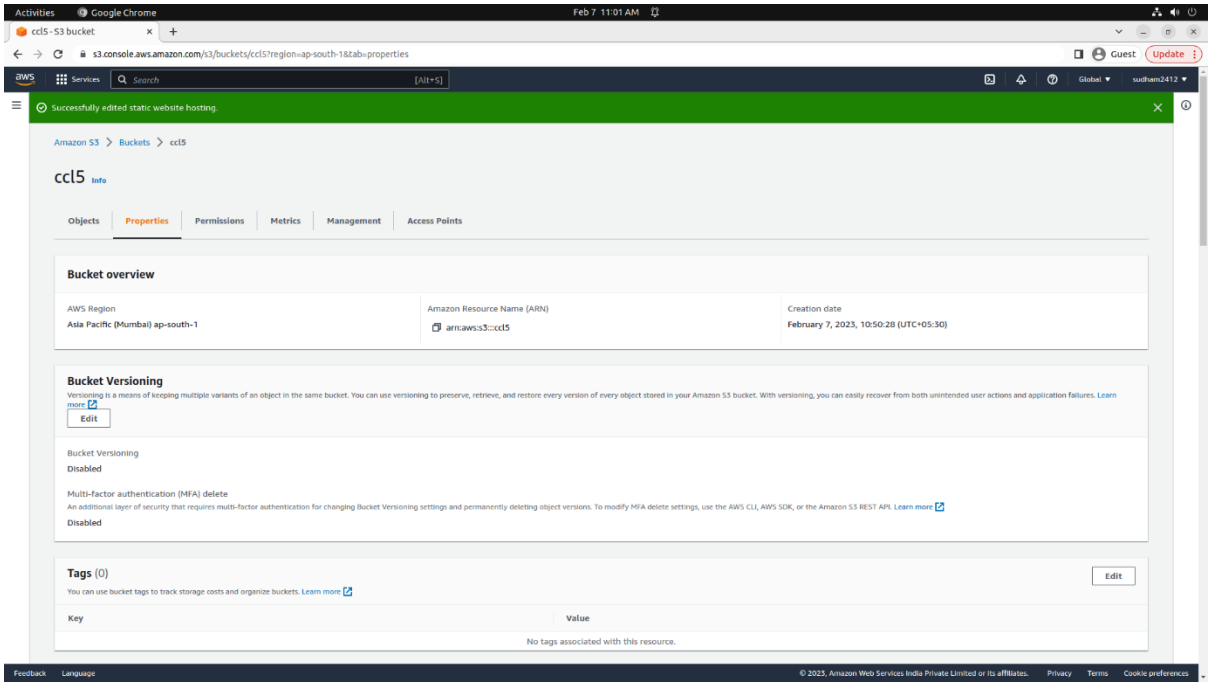
STEP 12: Enable the radio button and specify the file name in Index document which you have added in S3.





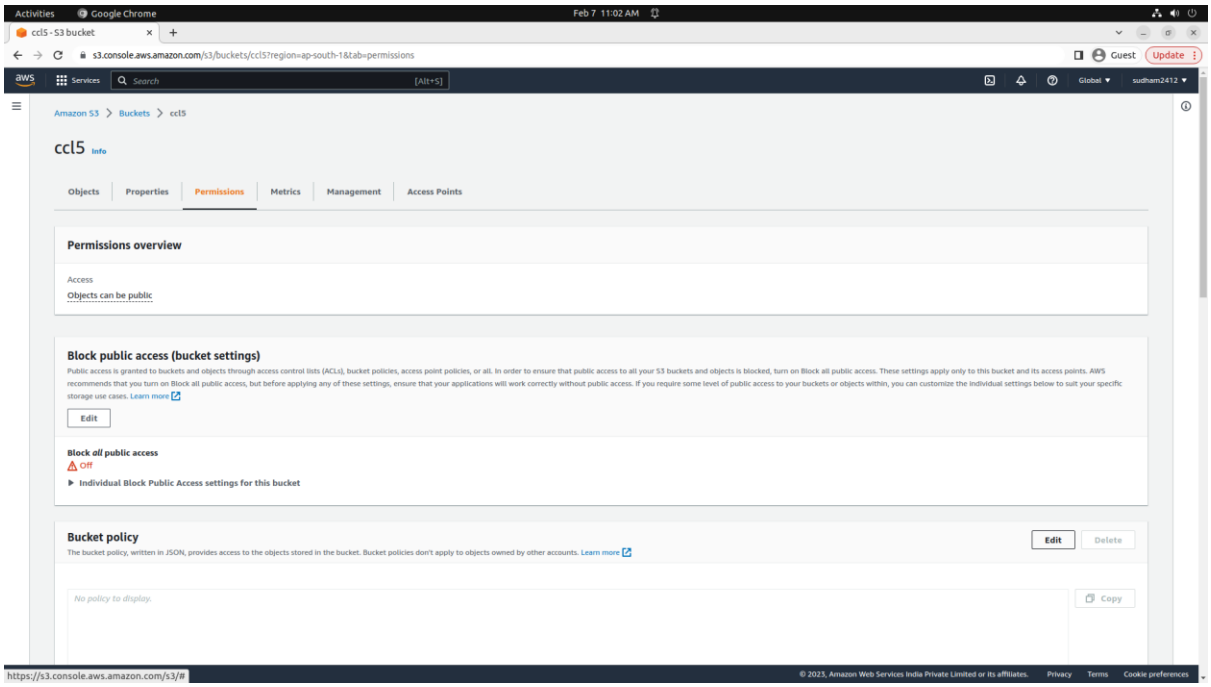
Scroll down and save the changes at bottom right.

Following screen will appear.

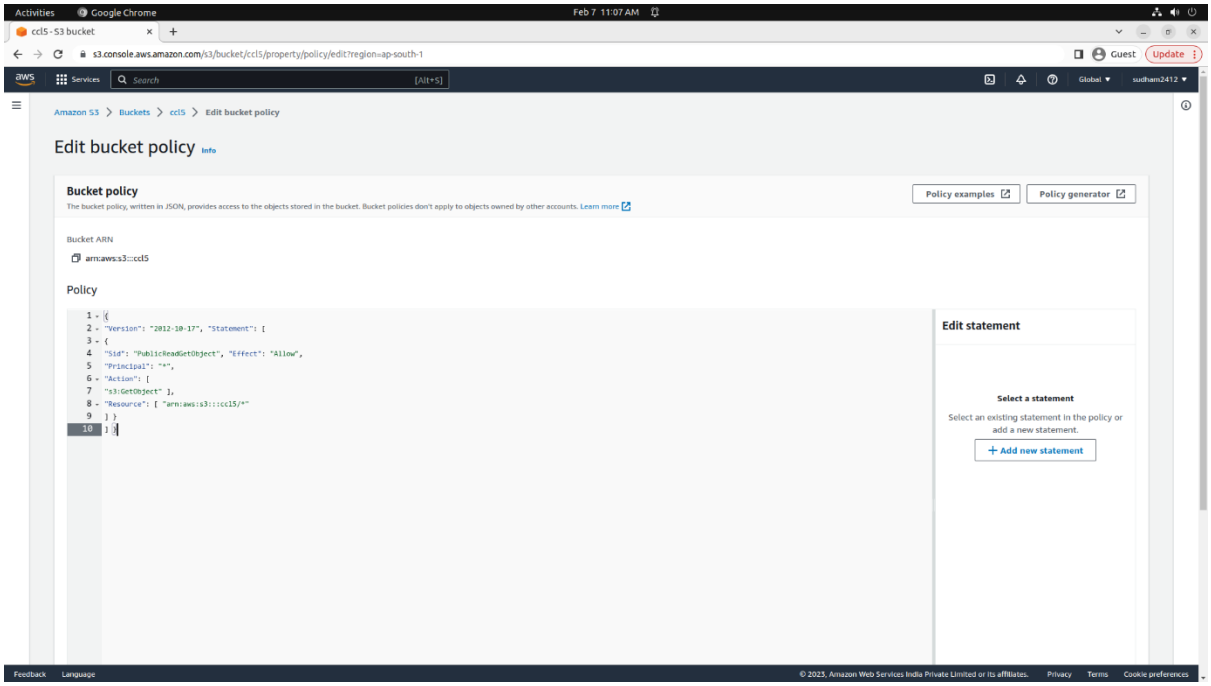


STEP 13: Click on **Permissions Tab**.

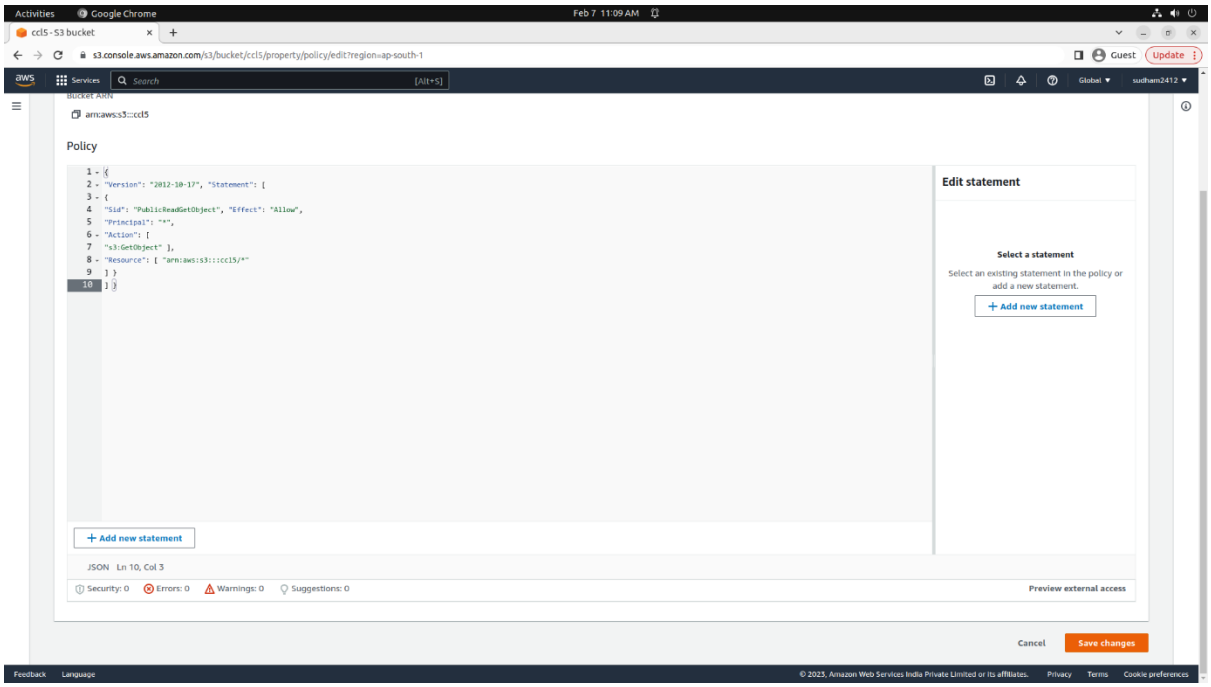
STEP 14: In **bucket policy** click on **Edit** option.

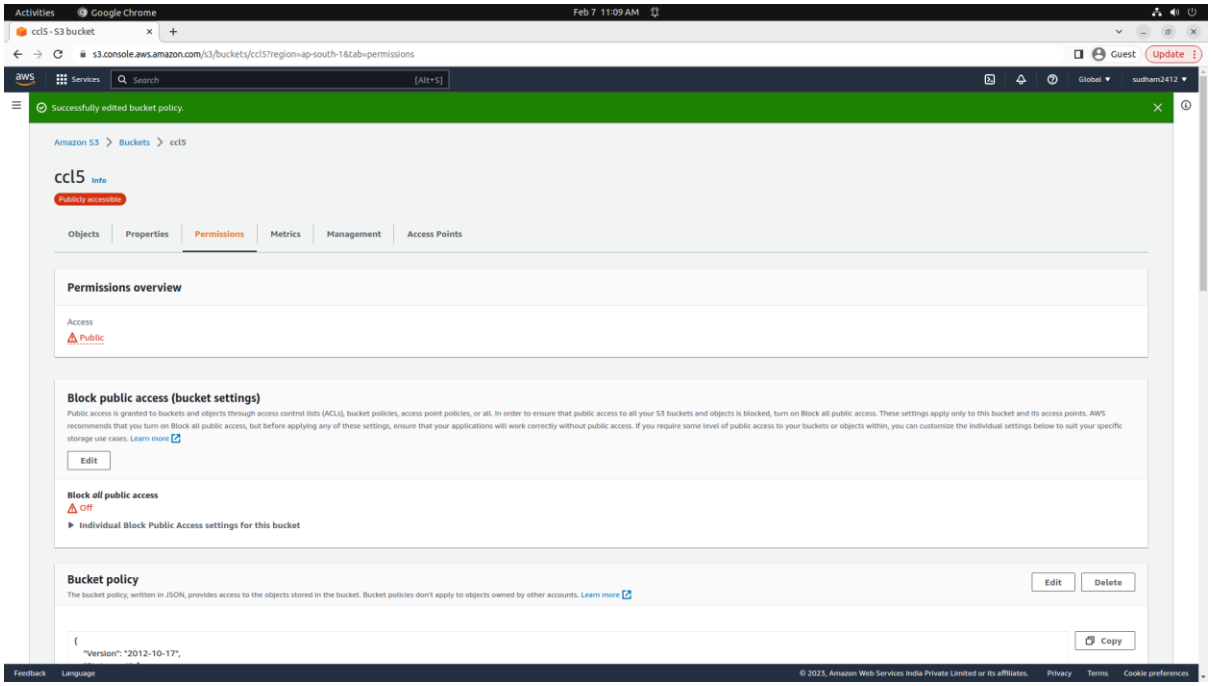


STEP 15: After clicking on edit button paste the following code in **bucket policy**.



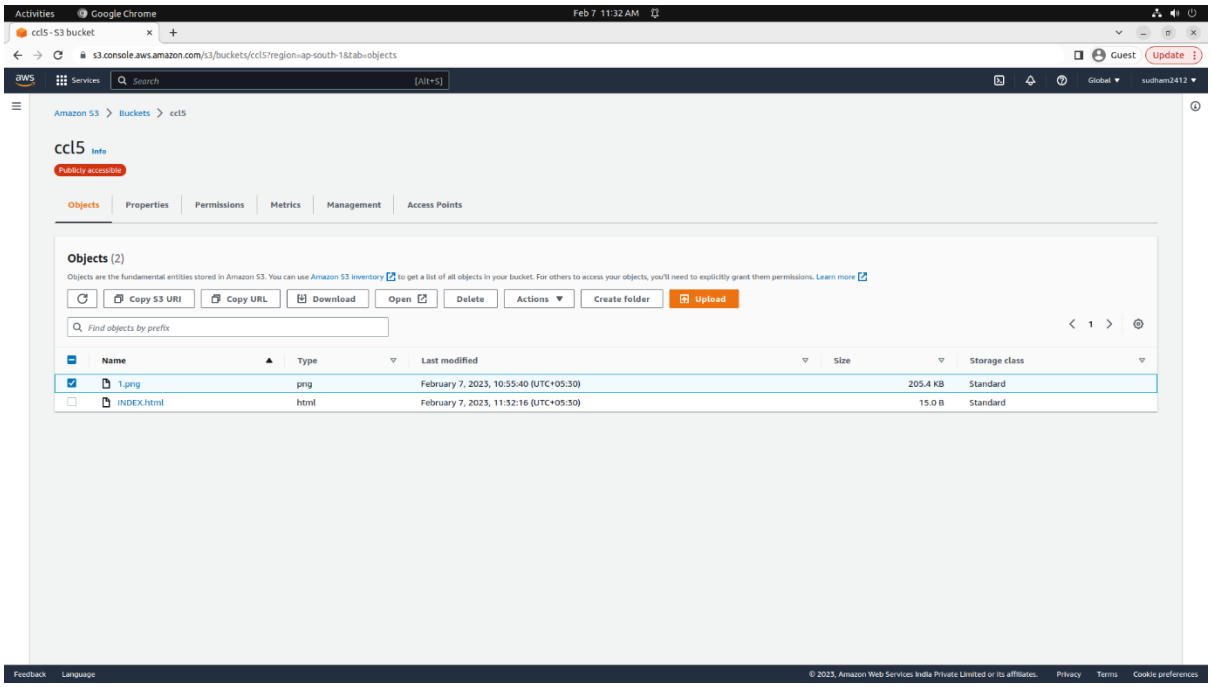
Scroll down and click on **Save Changes** button.



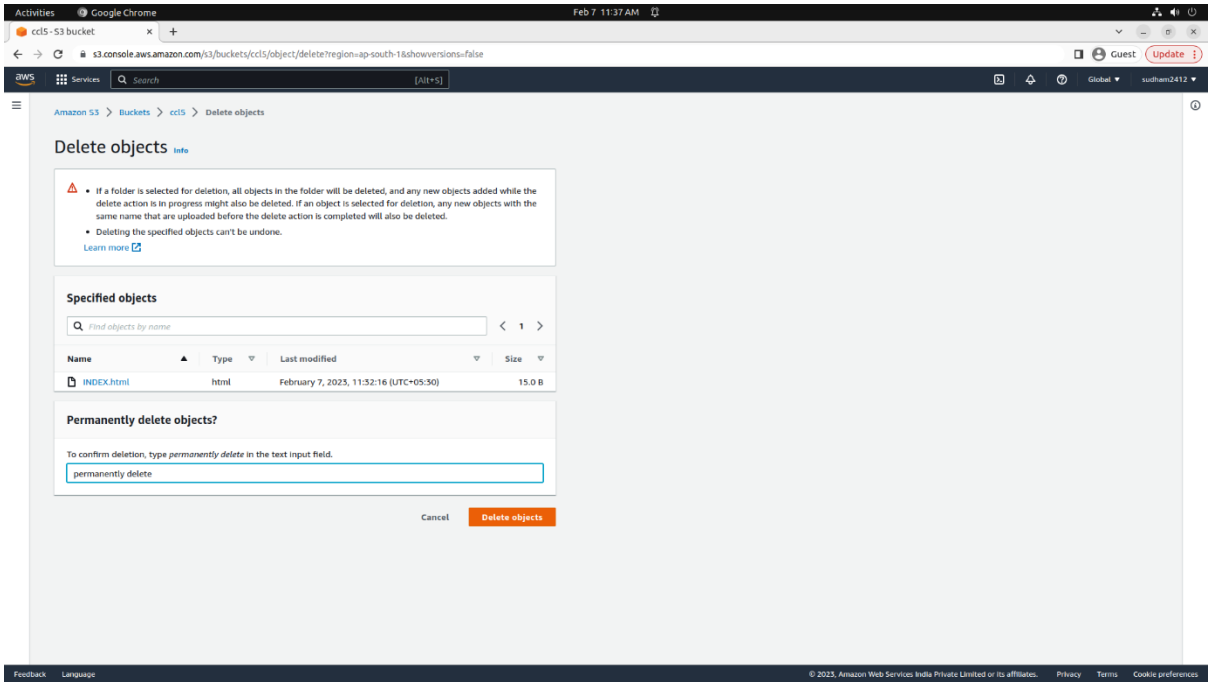


STEP 16: Open your html file and click on **Object URL**.

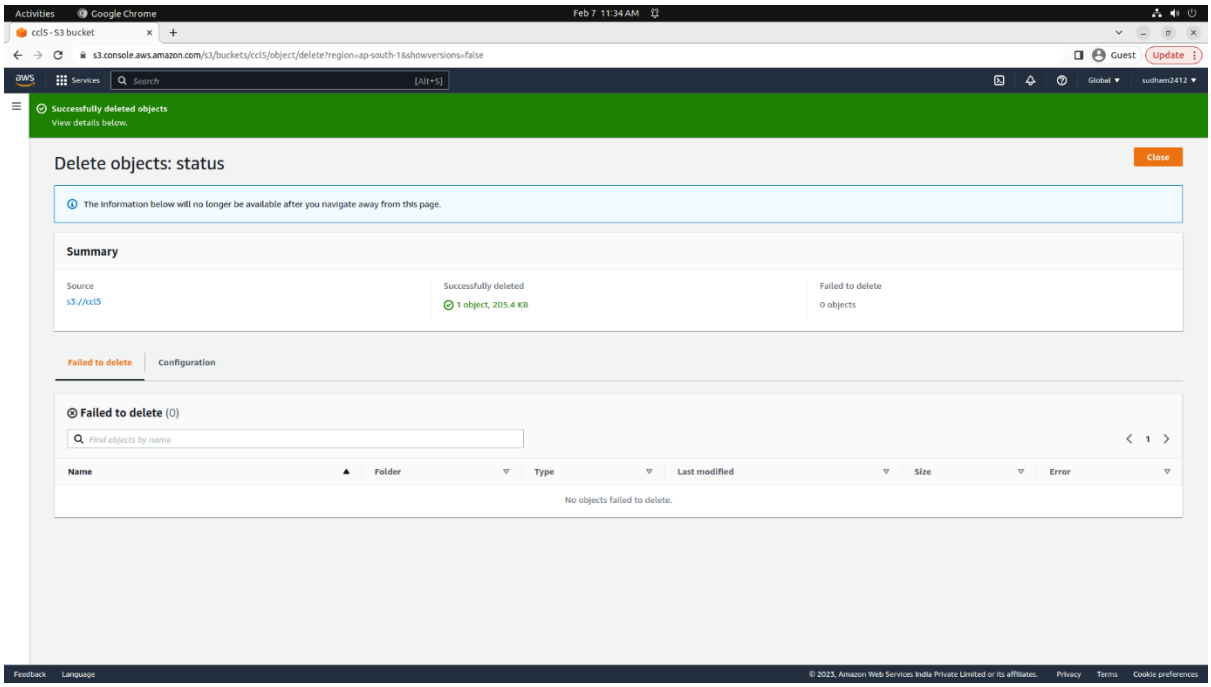
STEP 17: Now for delete files click on **checkbox** of your file and then click on **Delete** Button.



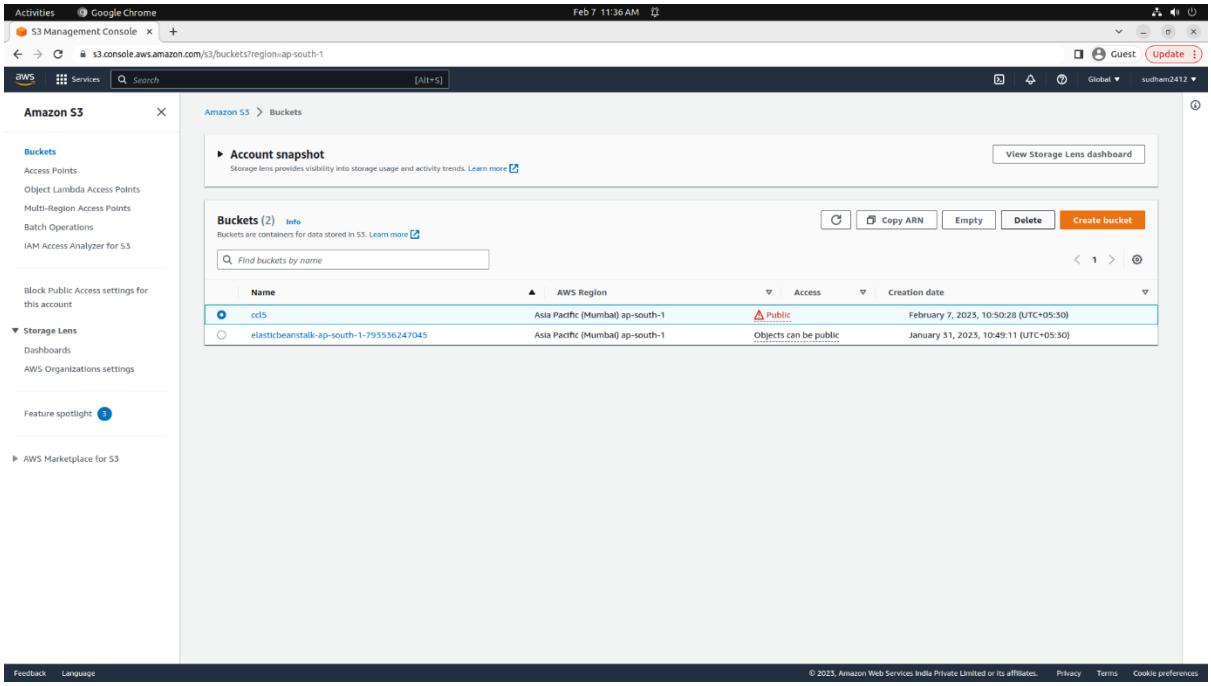
Write **permanently delete** and click on **delete** object button.

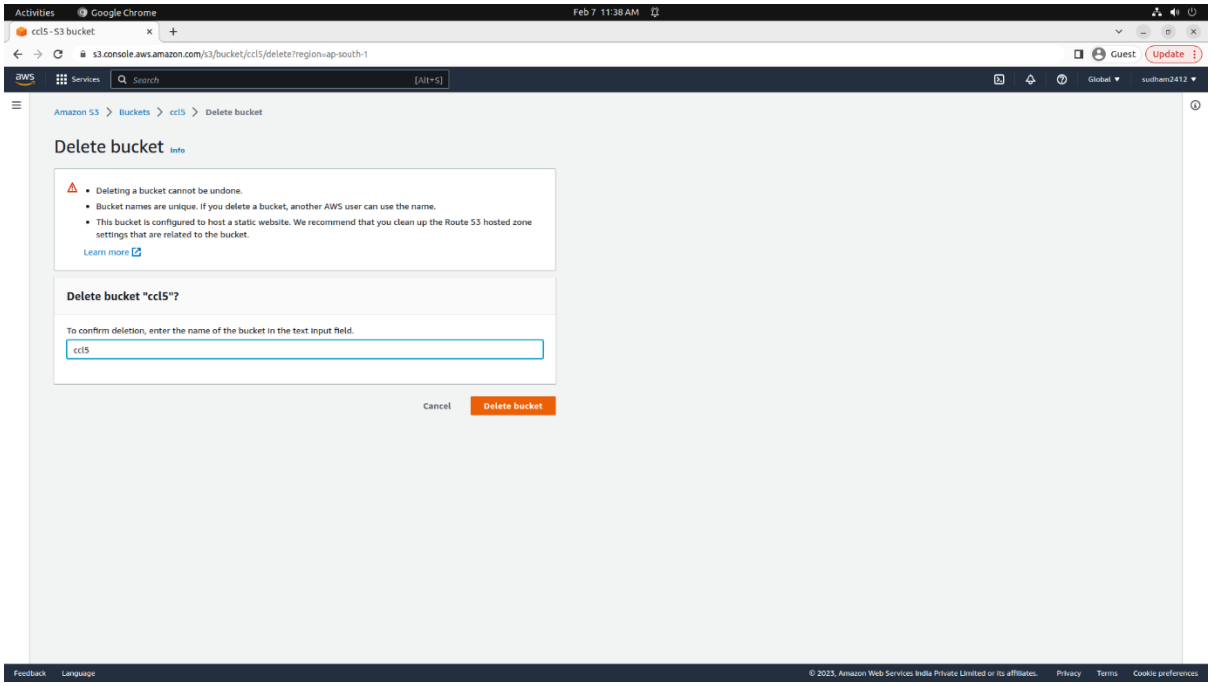


Now click on **close** button.

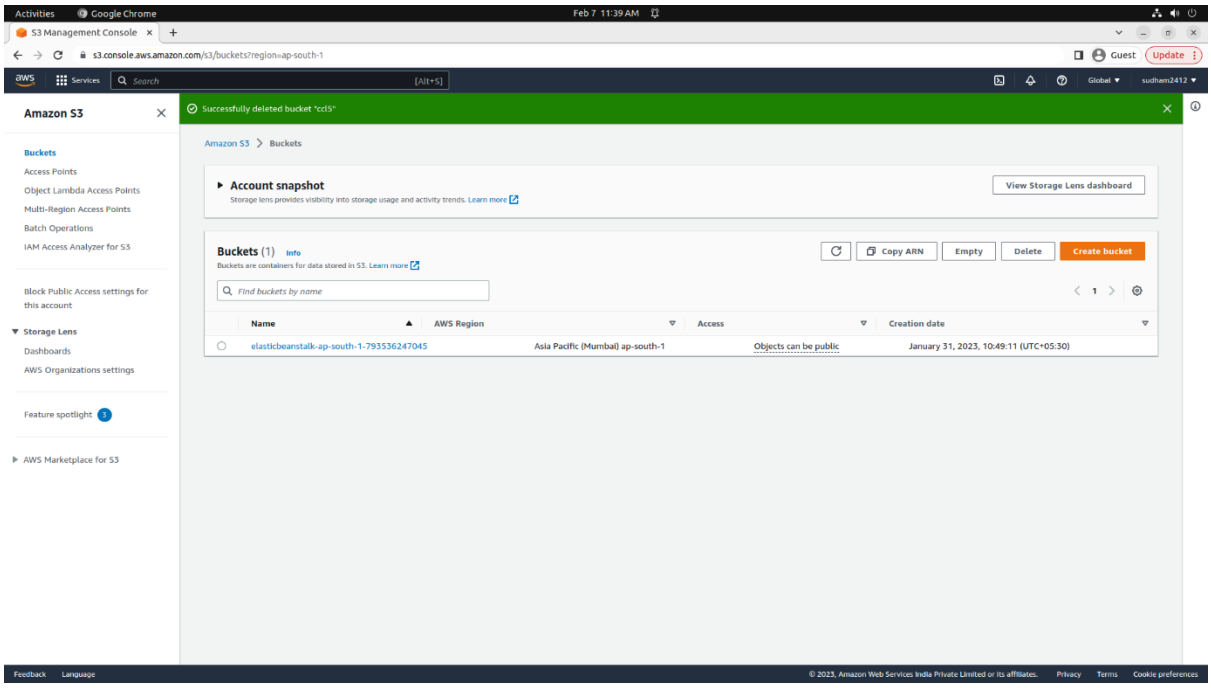


STEP 18: Now come to **Amazon S3** tab and select your **bucket** and then click on **delete** button.





You can see that the bucket is **deleted**.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

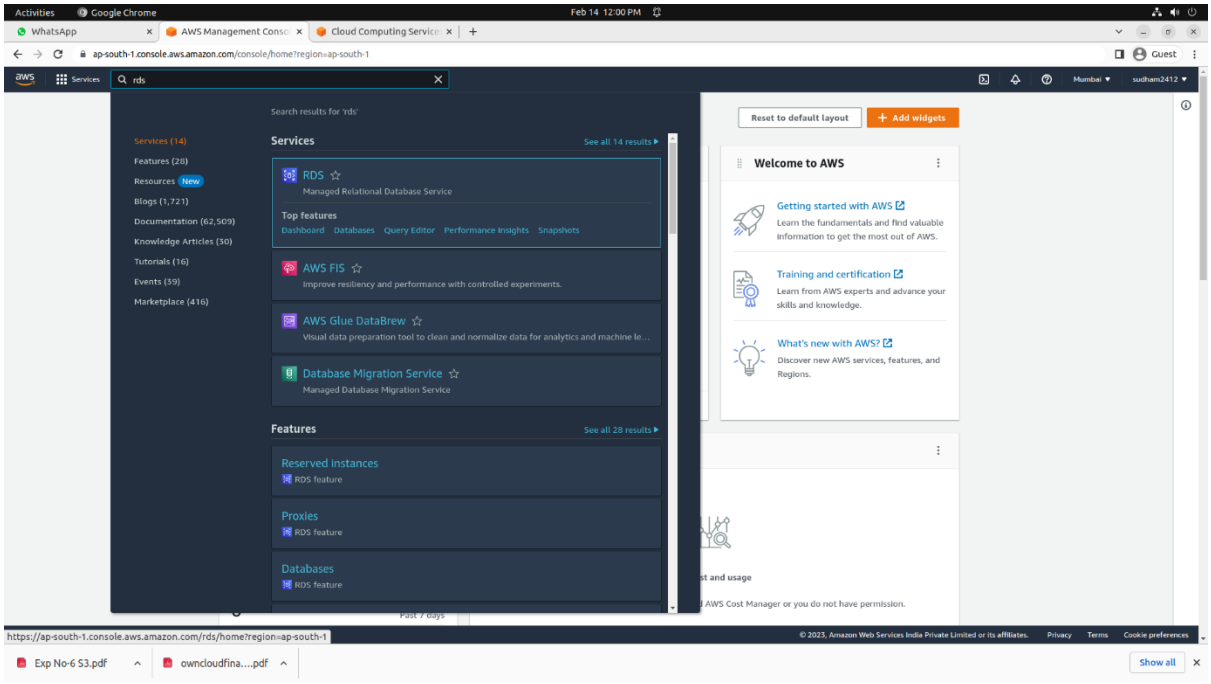
T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**CLOUD COMPUTING LAB**
COURSE CODE	**CSL605**
PRACTICAL NO.	**4**
DOP	
DOS	

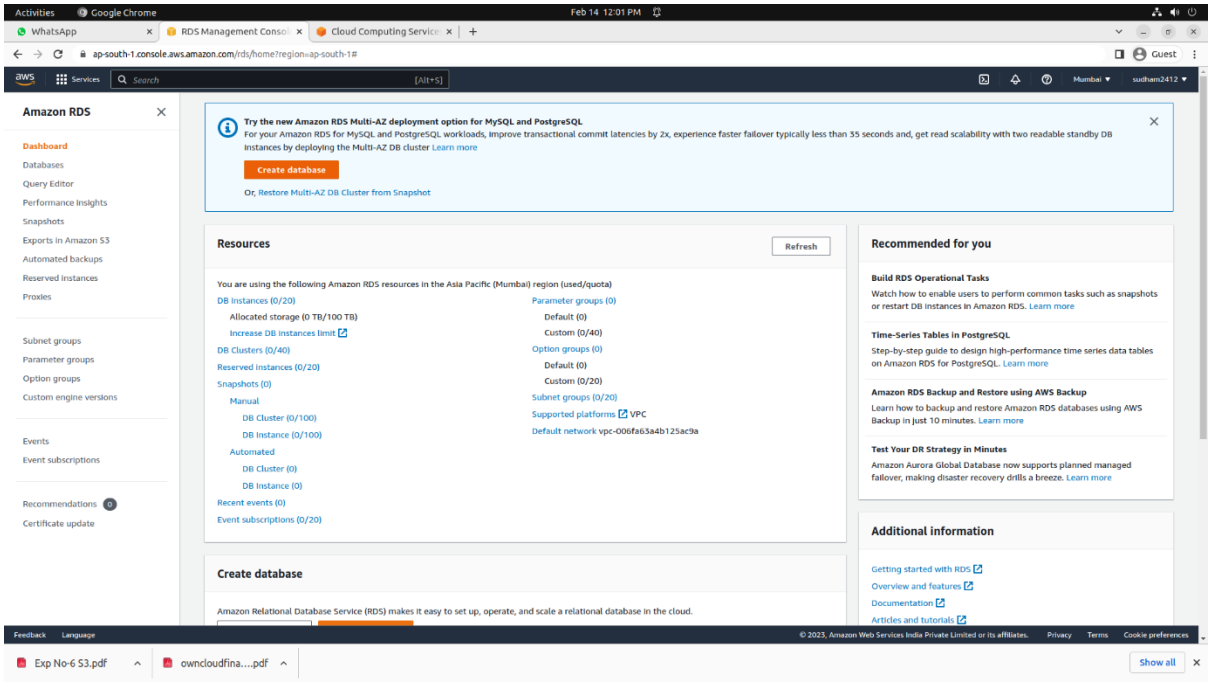
AMAZON RDS

OUTPUT:

STEP 1: Login to AWS console and search **RDS.**

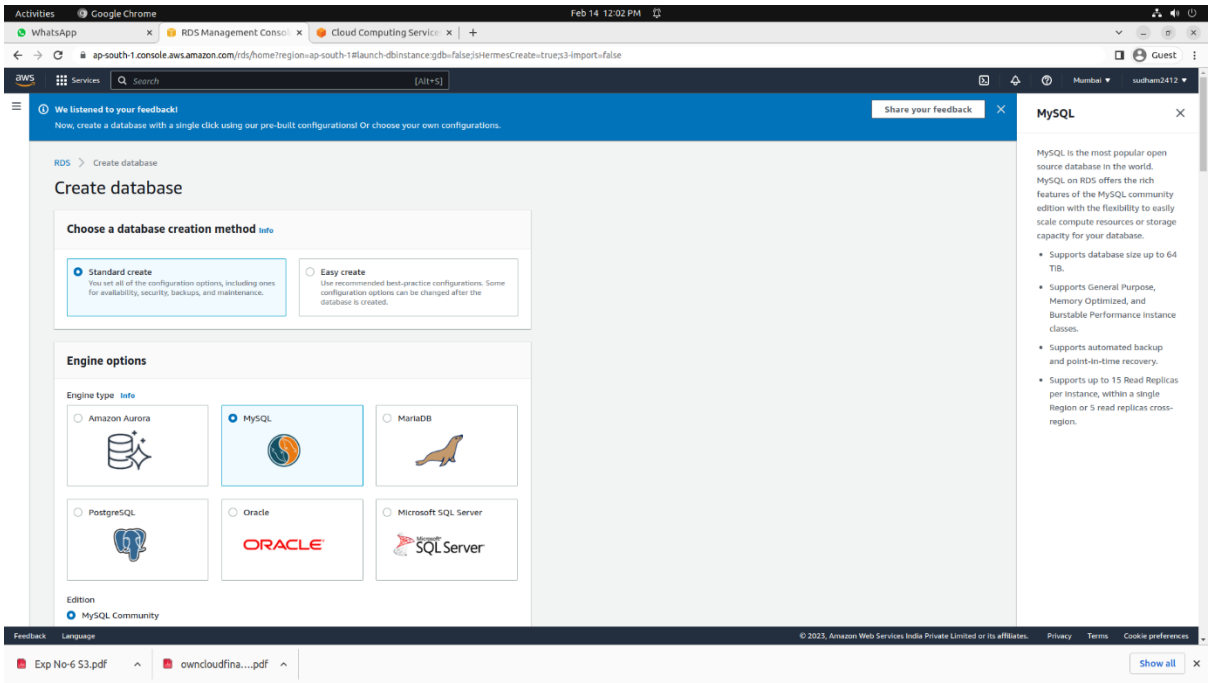


STEP 2: Click on to **RDS** and **Create database.**

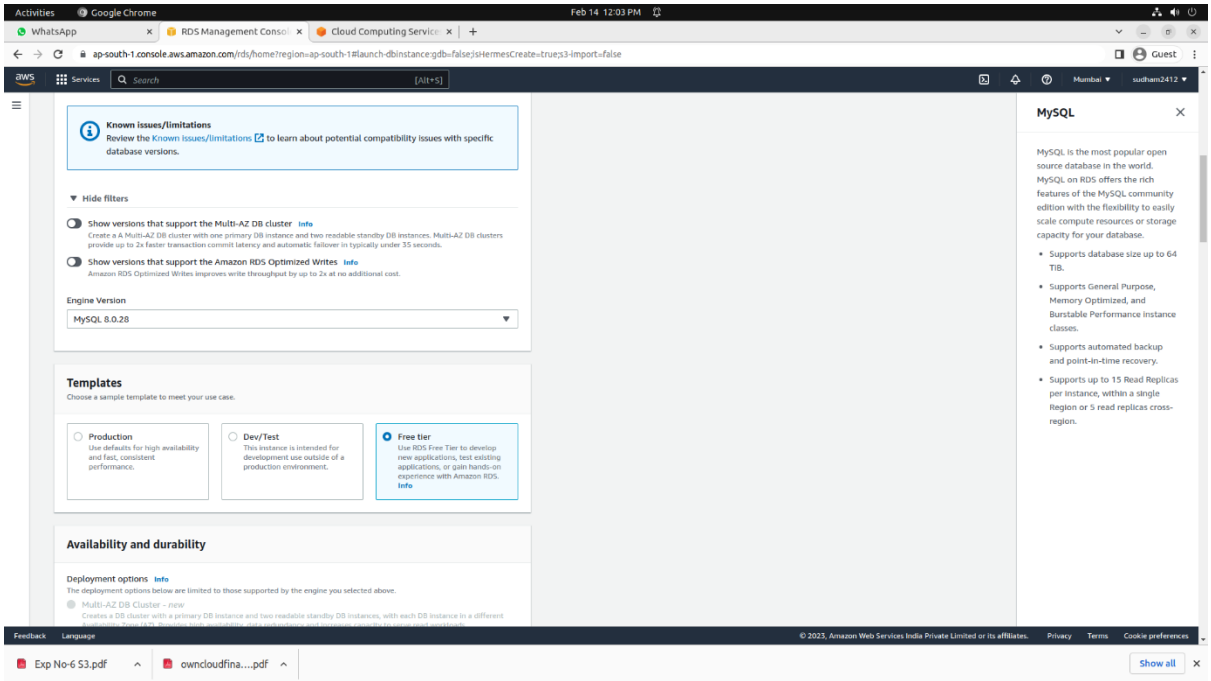


STEP 3: Select **standard database.**

STEP 4: Select **MySQL** and **MySQL Community** edition.

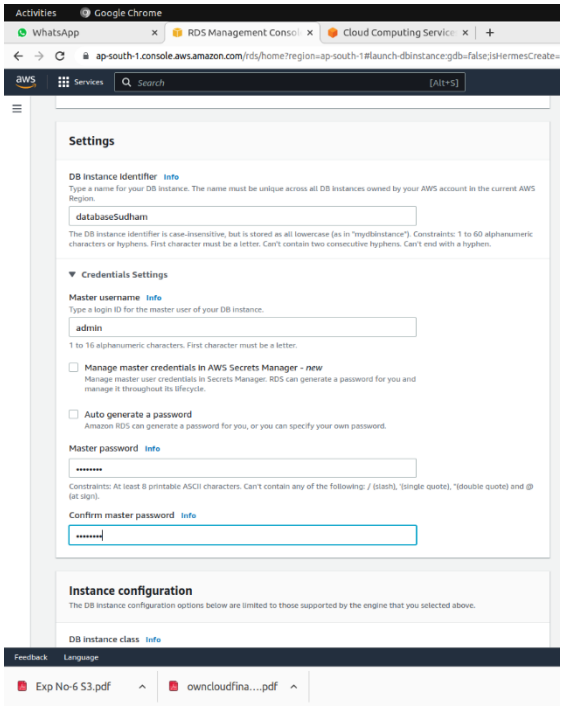


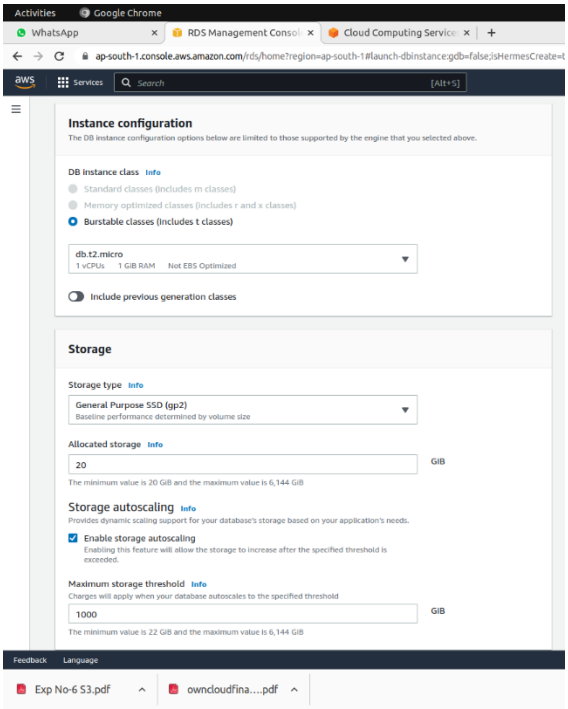
STEP 5: In Templates select **Free tier**.



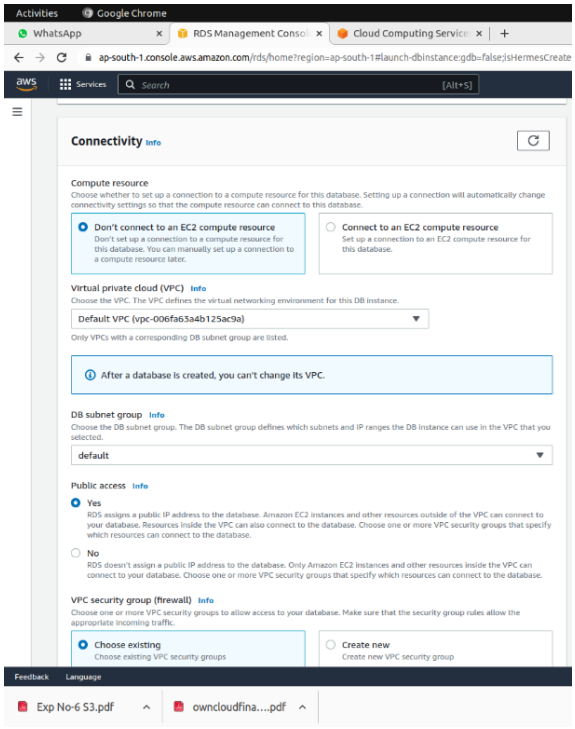
STEP 6: Mention **database name** (default is database1) and **username** and **password.**

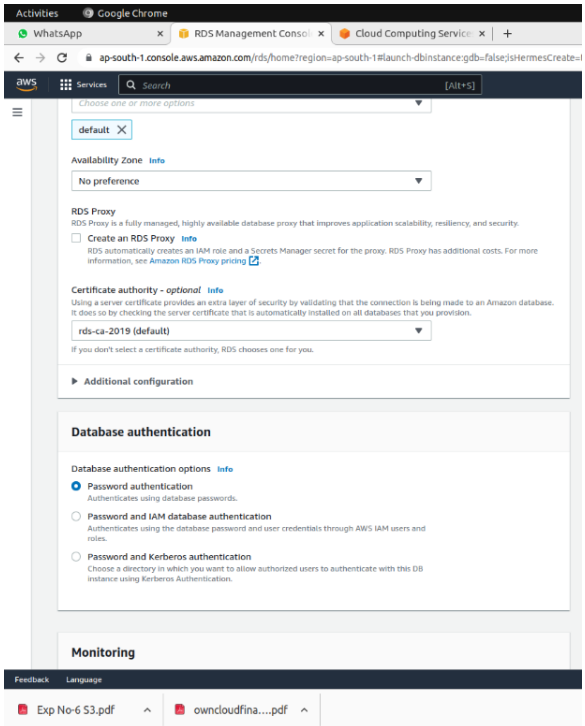
STEP 7: Instance is db.t2.micro .
Rest of things keep default



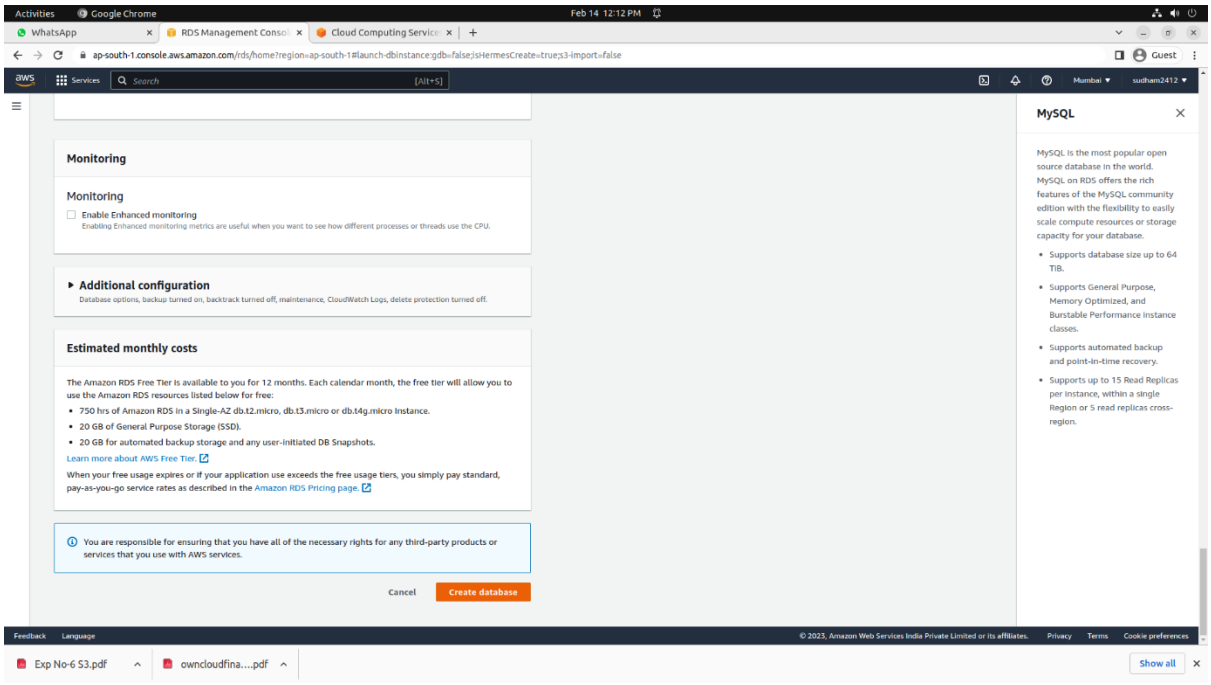


STEP 8: Select **Public Access -Yes**.

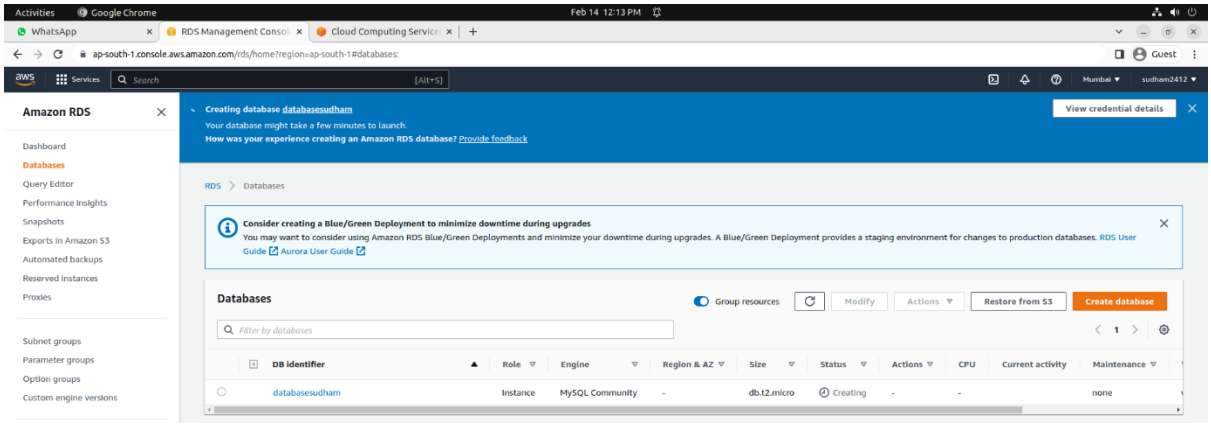




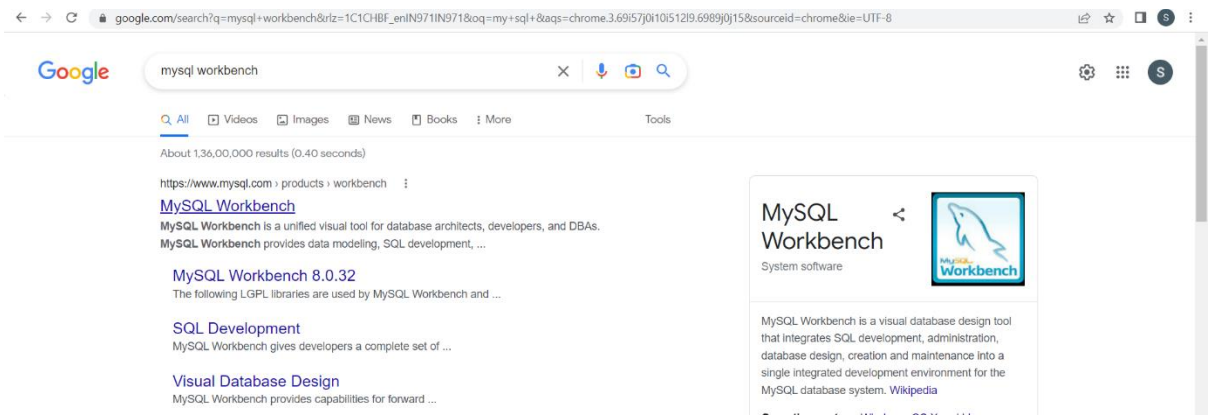
STEP 9: Click on to **Create Database.**



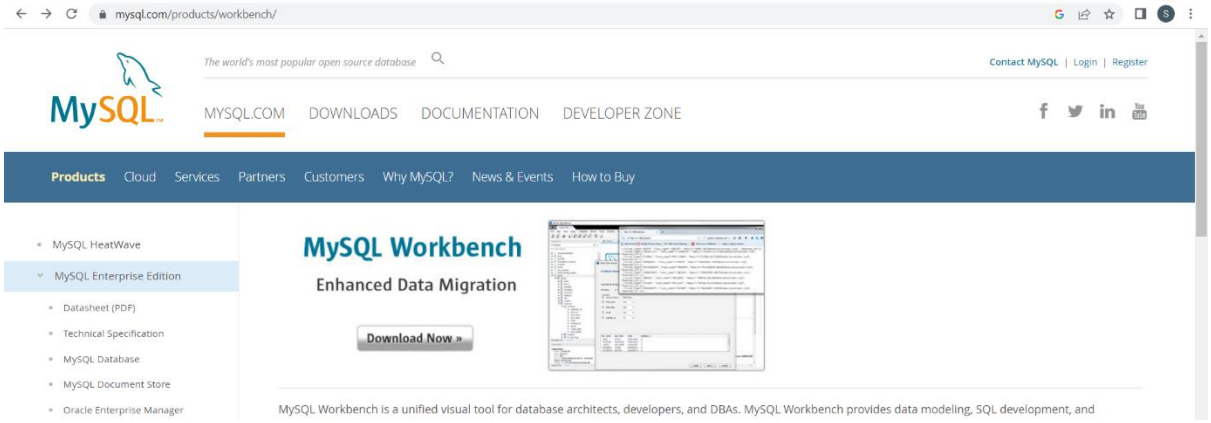
STEP 10: It will take some time.



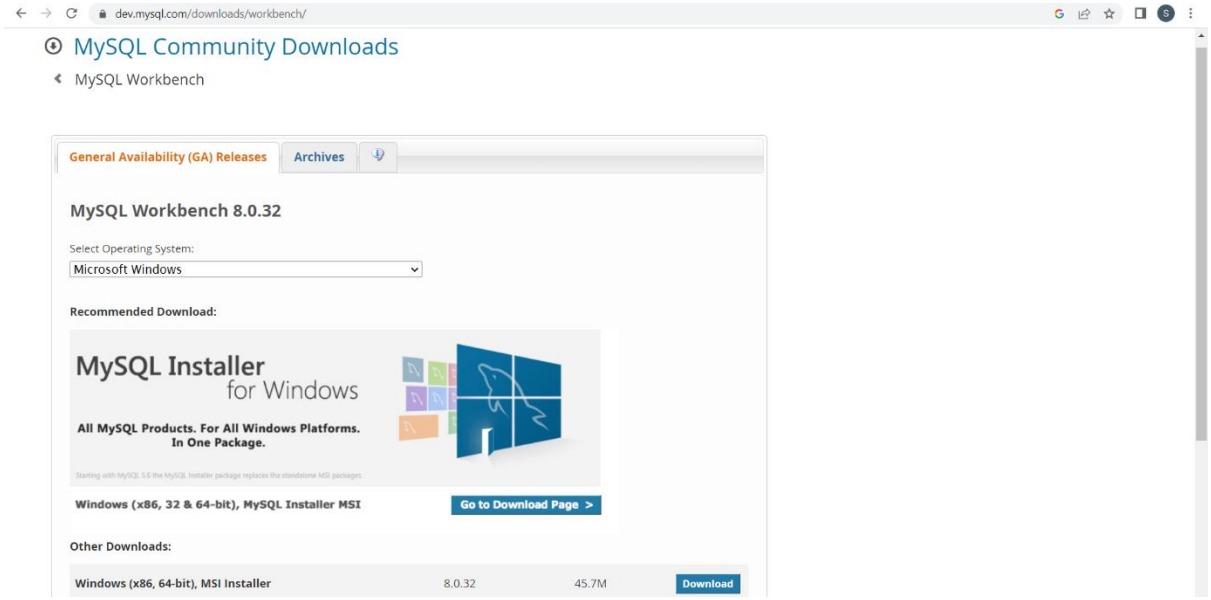
STEP 11: Go to google type **mysql workbench.**



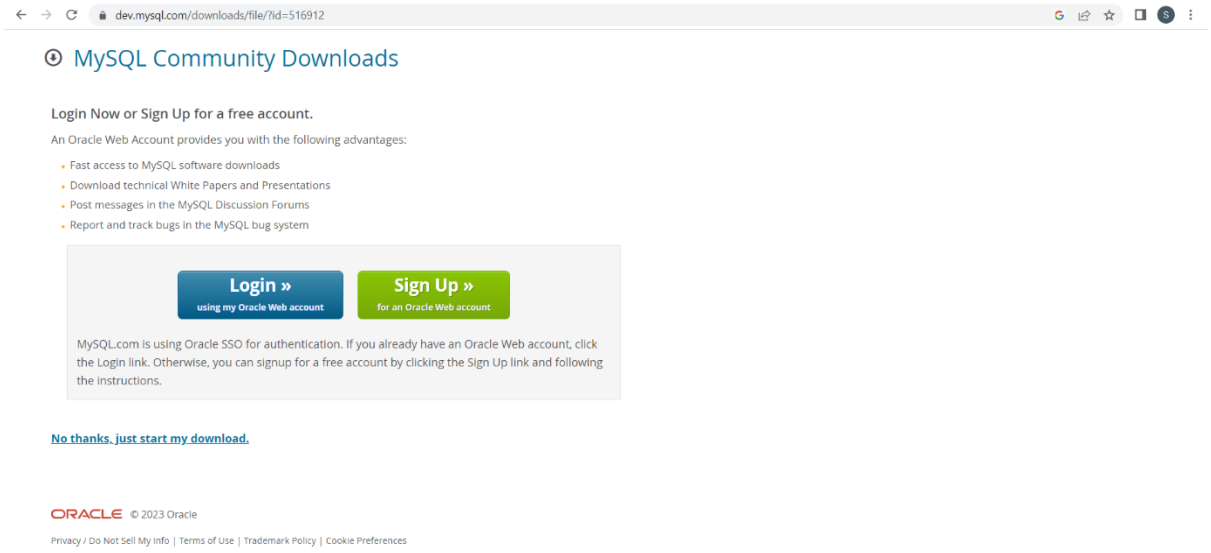
STEP 12: Click on to **download**.



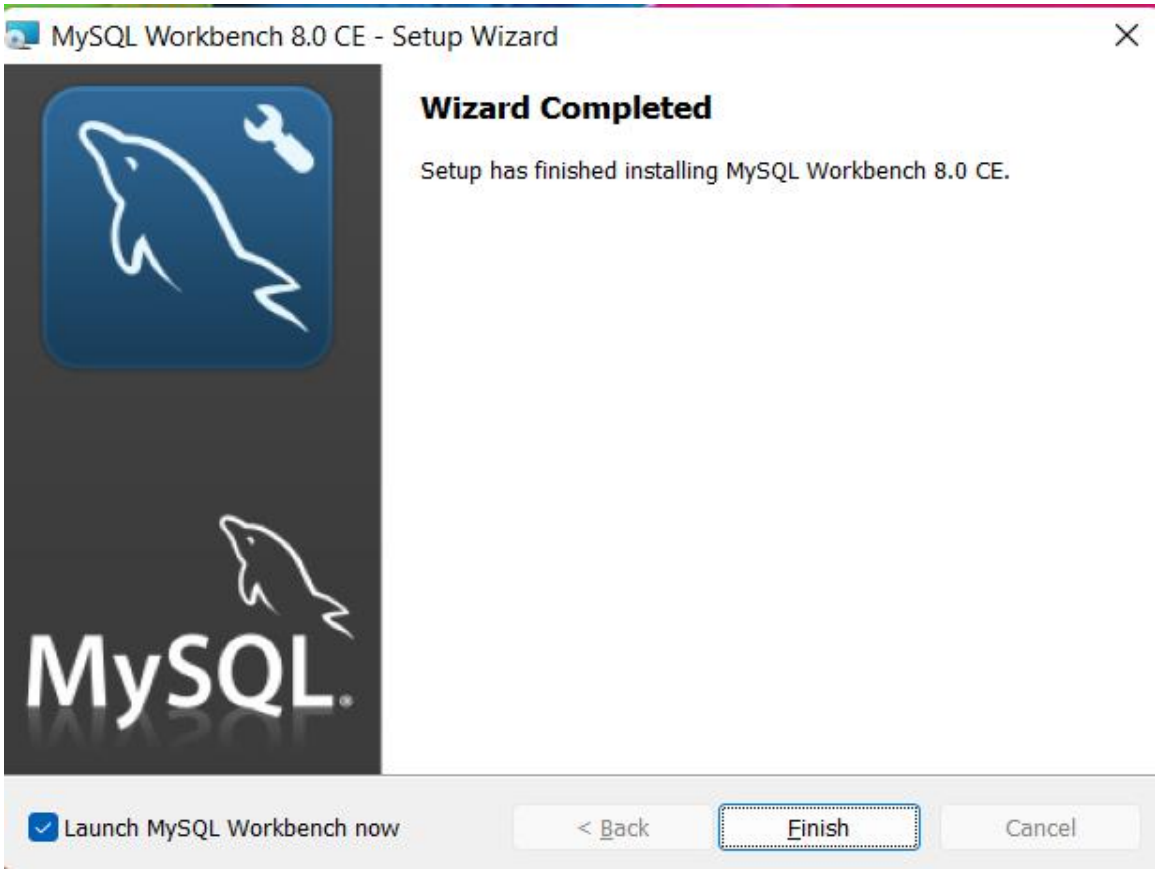
STEP 13: MySQL community download – **Microsoft Windows.**



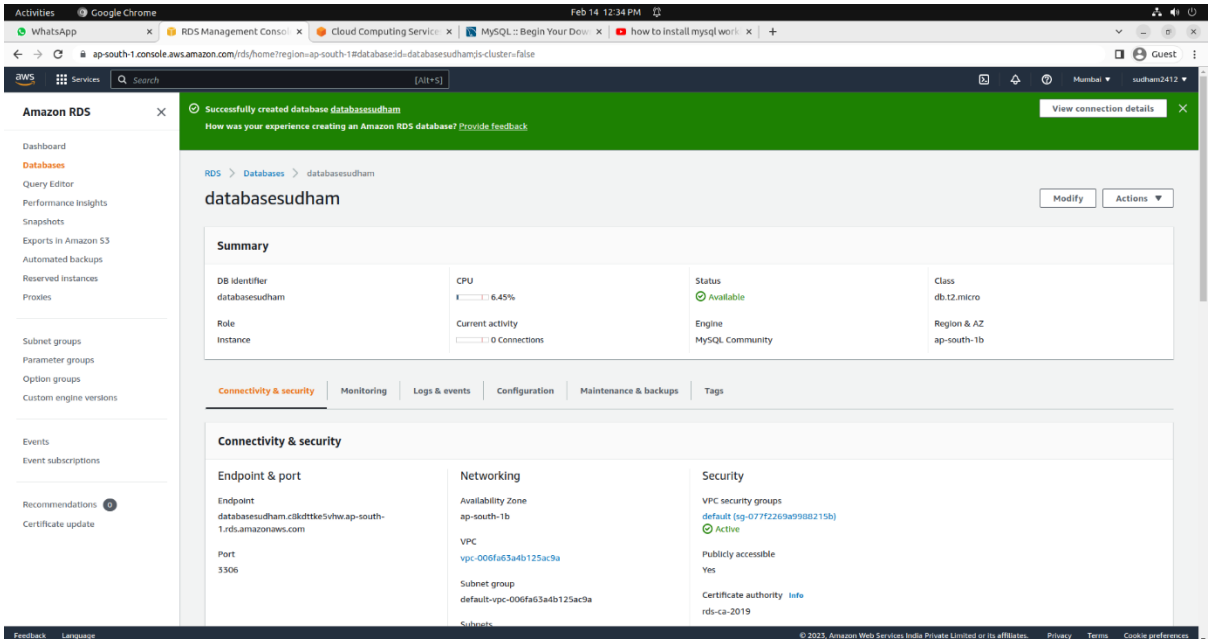
STEP 14: Click on to – **No thanks, just download**.



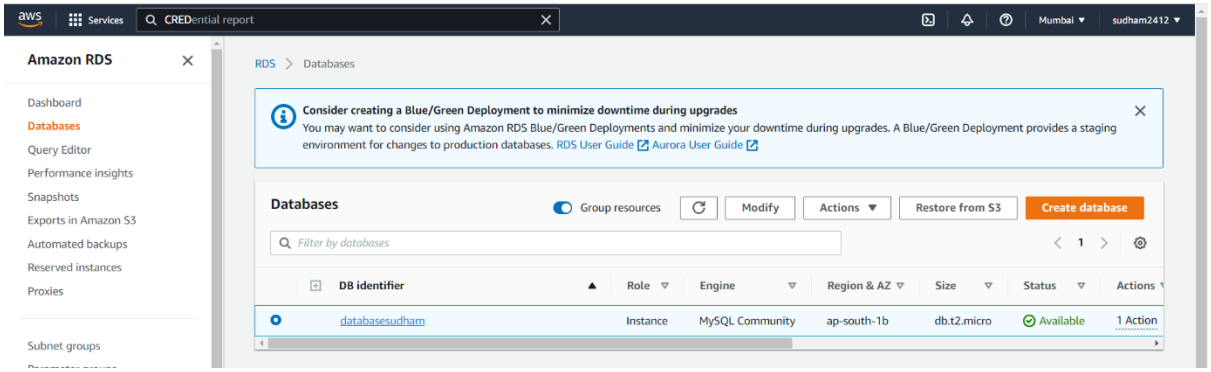
STEP 15: Go to downloads of your machine and **install** it with default settings.



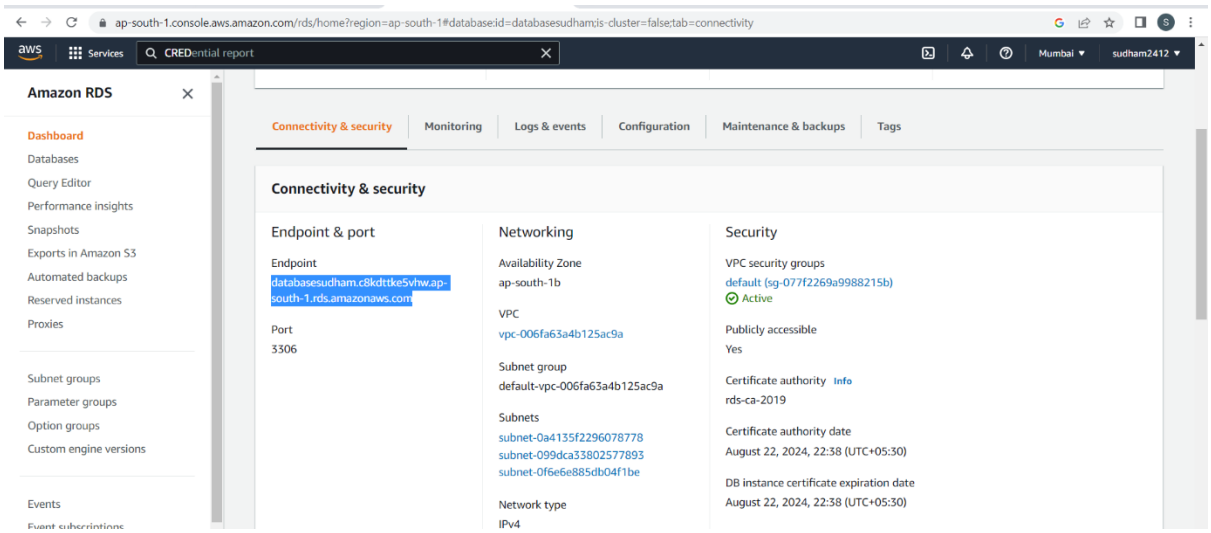
Check your database is created and status is available.



STEP 16: Click on to **view credential**.
STEP 17: Click on to **database**.

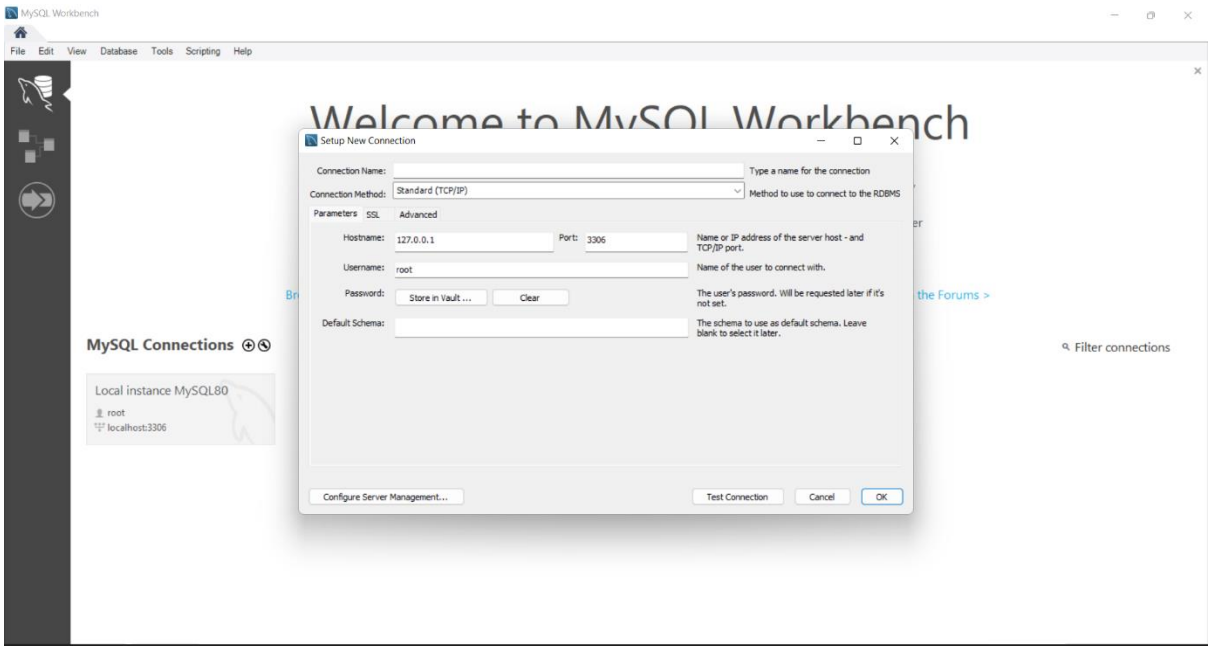


STEP 18: Copy **Endpoint.**

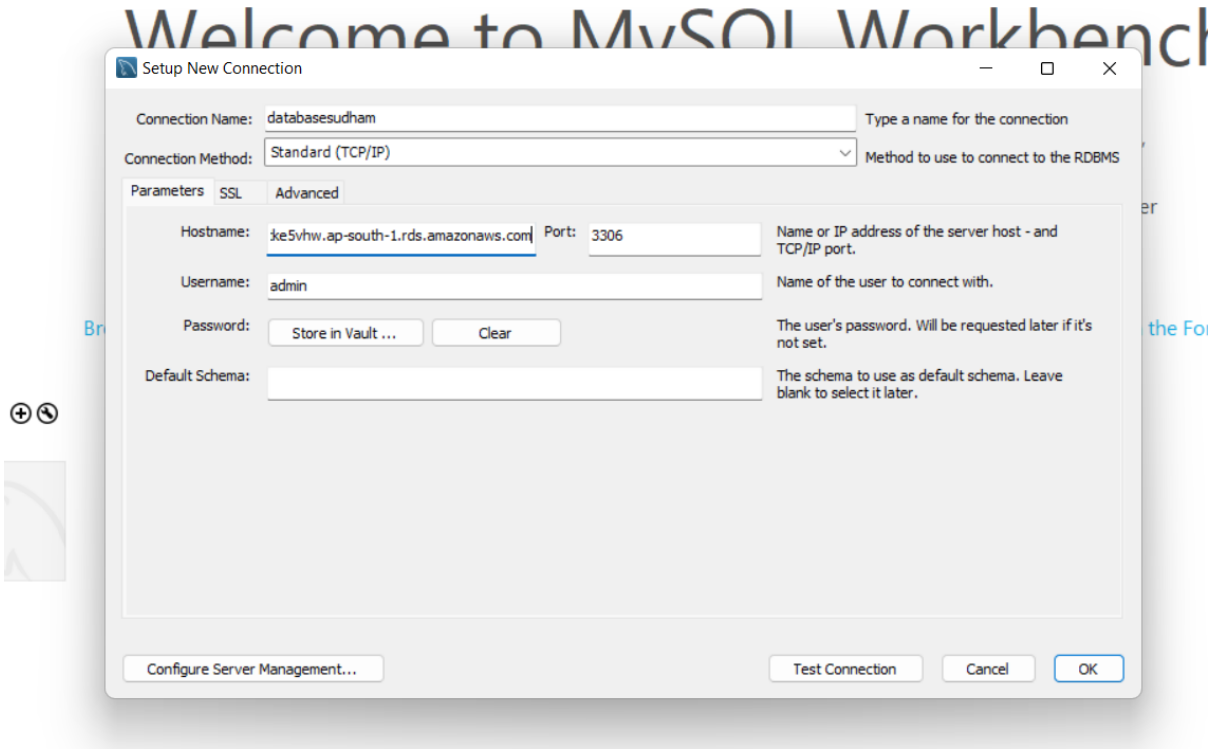


STEP 19: Go back to **workbench**.

STEP 20: Click on to mysql connection

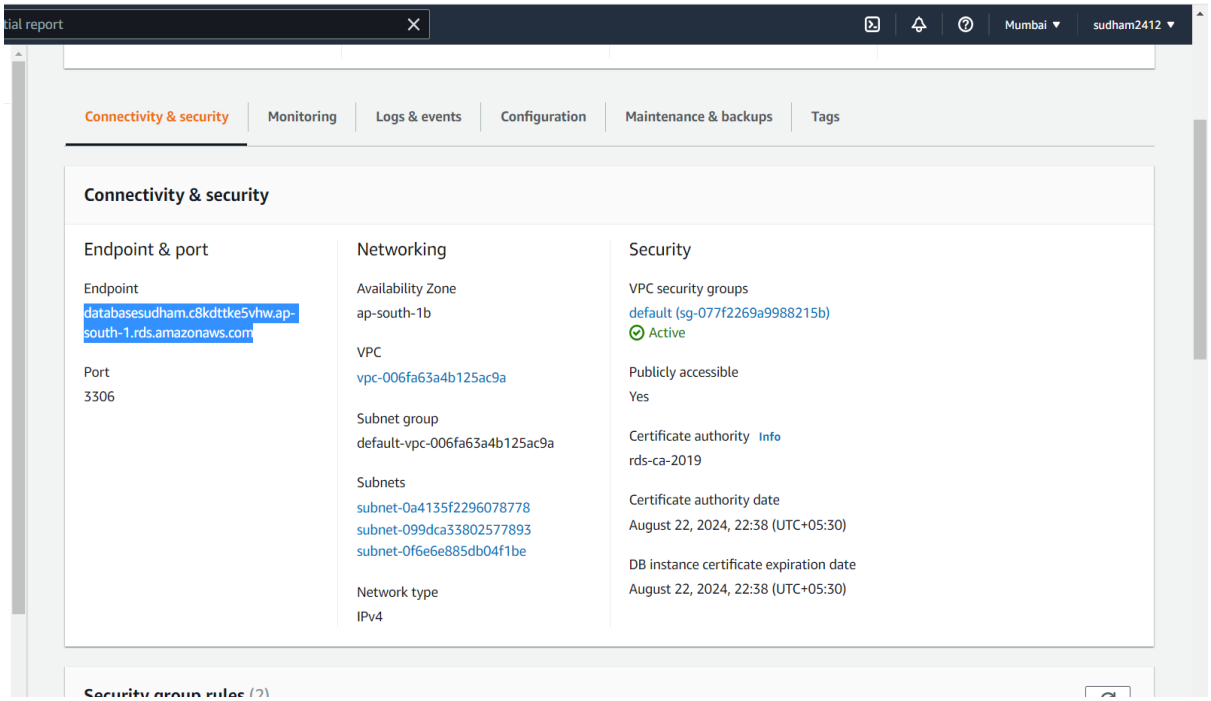


STEP 21: Paste copied endpoint in Hostname
Connection Name: databasesudham
Username: admin
Click on to **Test Connection**.

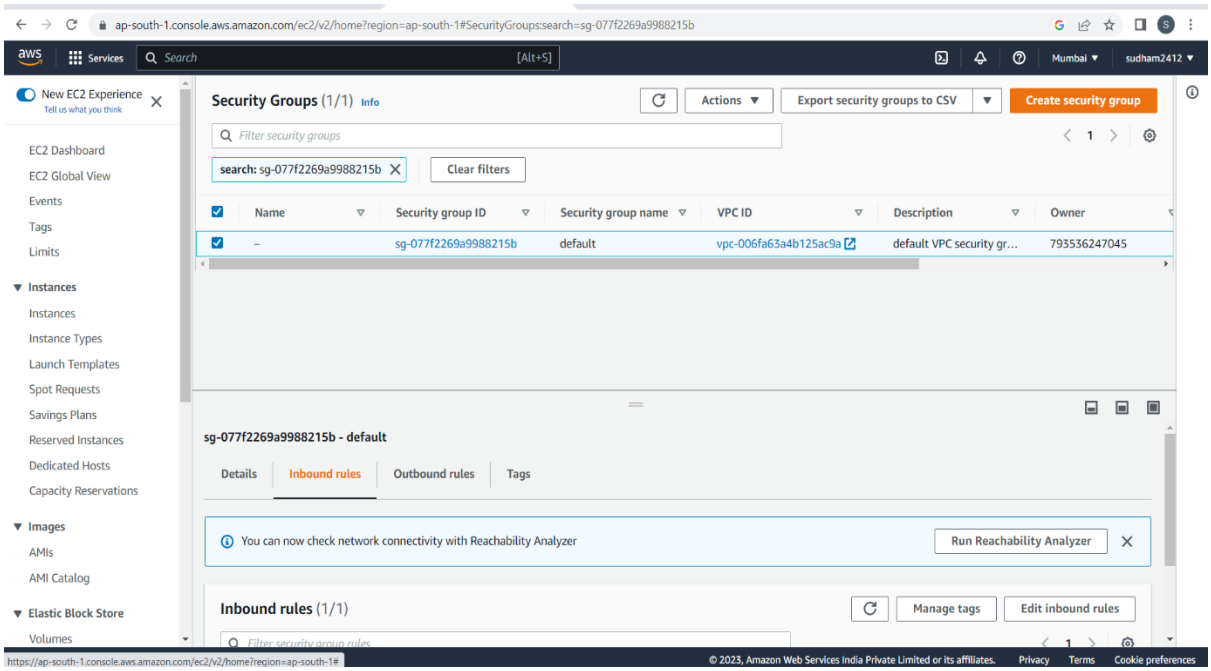


Enter **admin password**

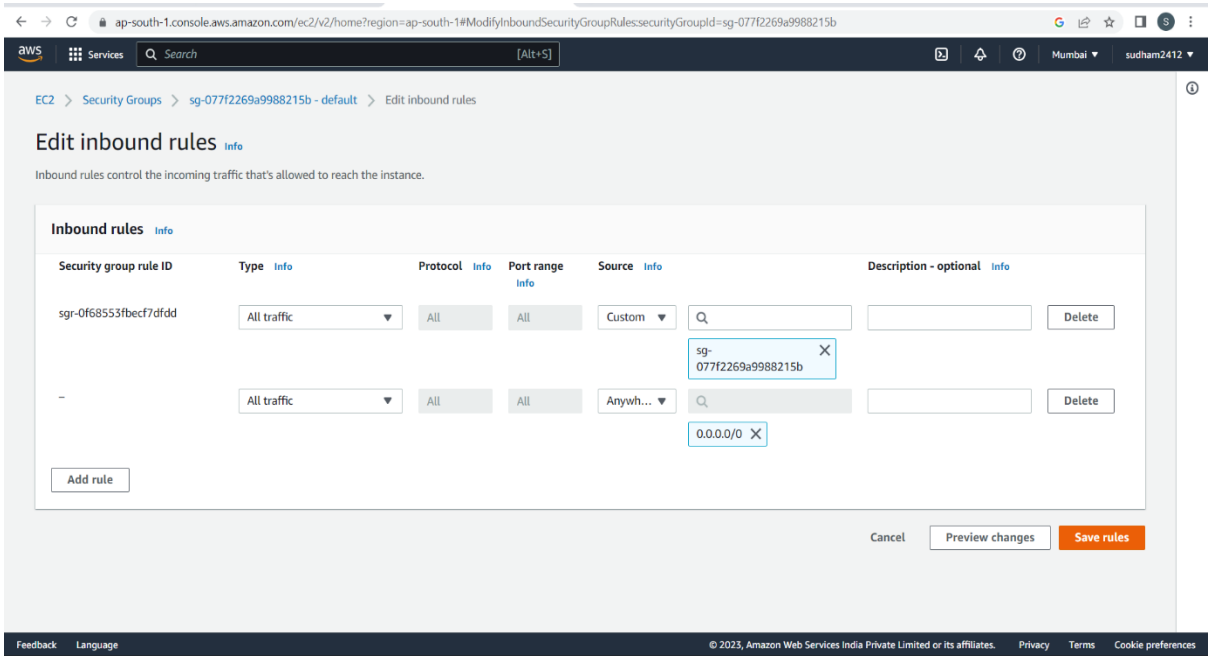
STEP 22: Go to **VPC security group**.

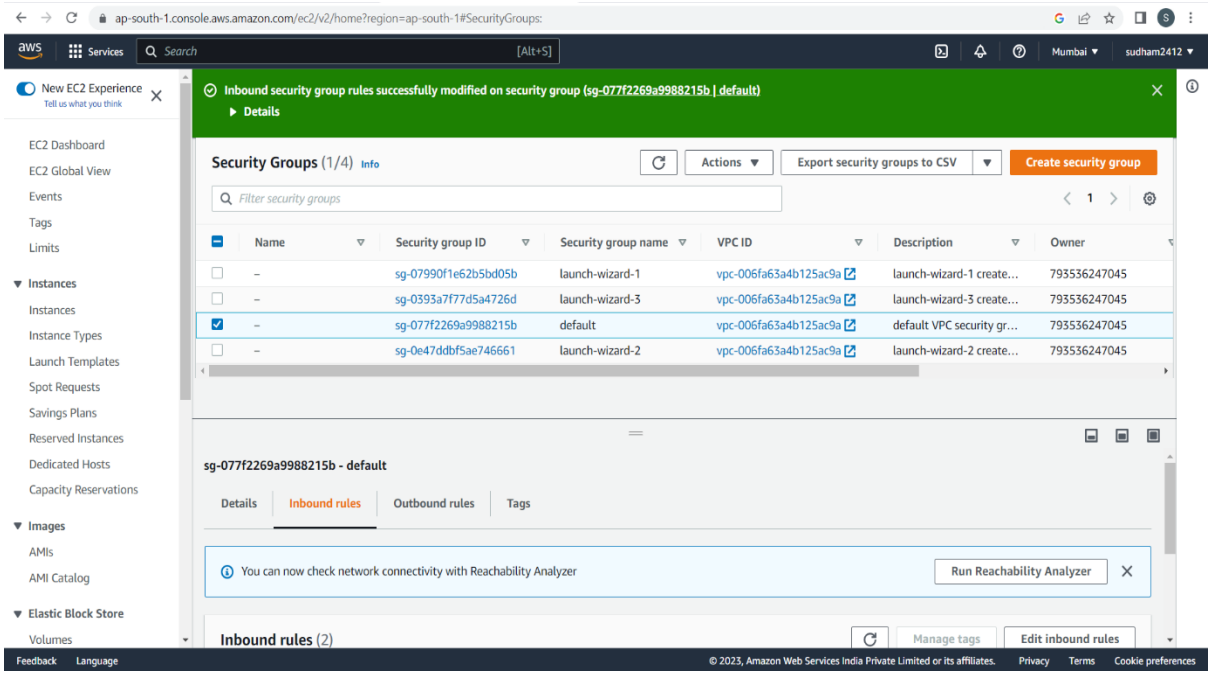


STEP 23: Click on to **Inbound rules**.

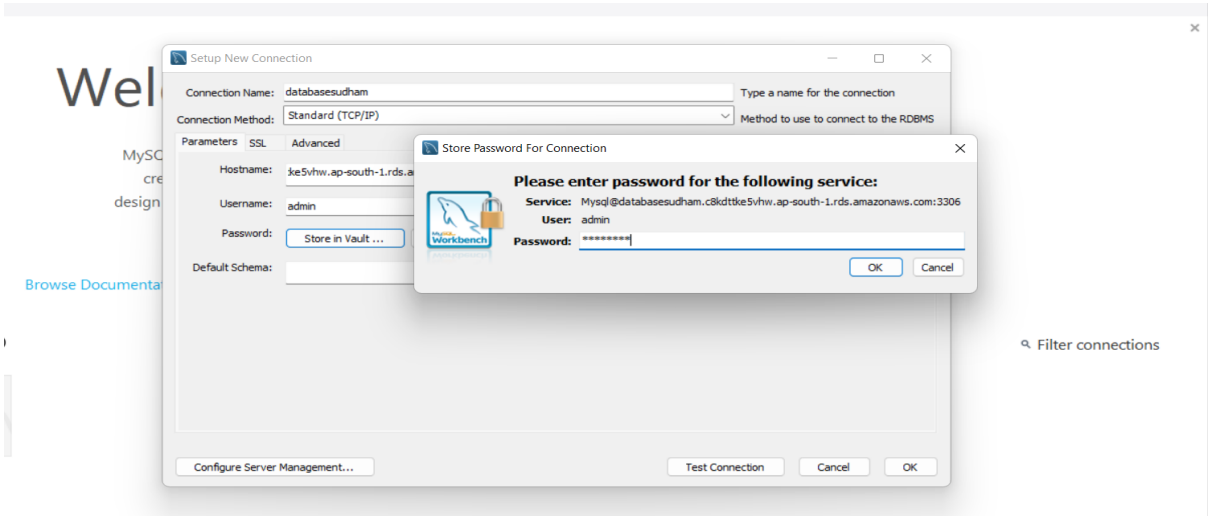


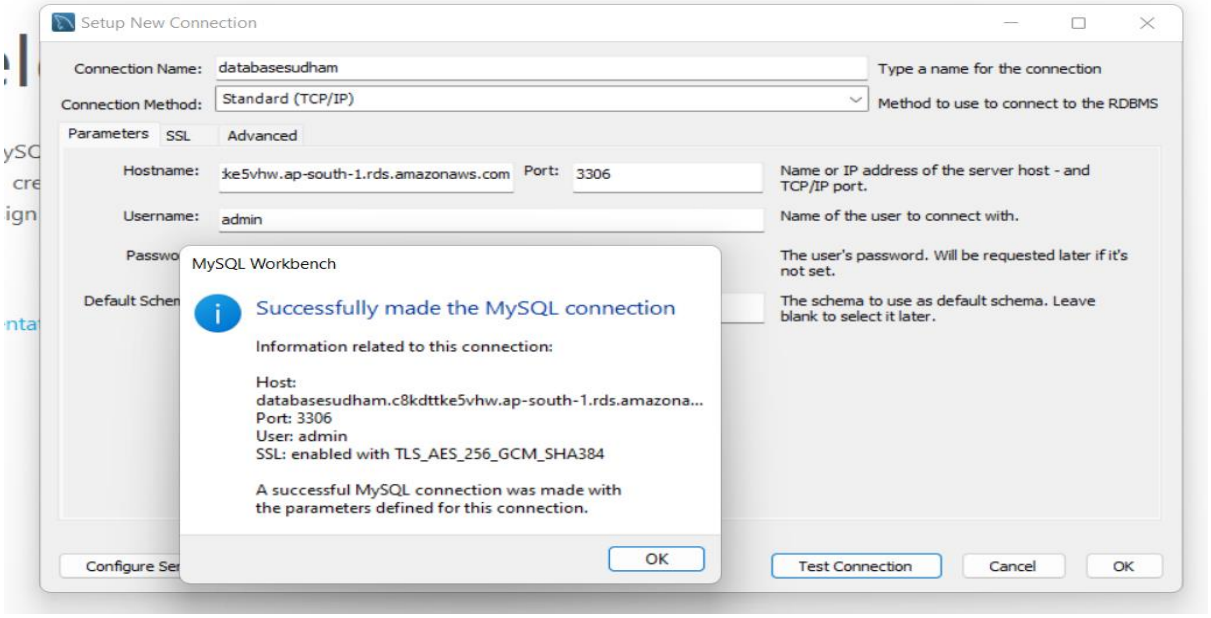
STEP 24: First select Click on to Edit inbound rule add rule select ipv4 --all traffic (add 0.0.0.0./0) and save Rules
(Important step to add inbound rule)





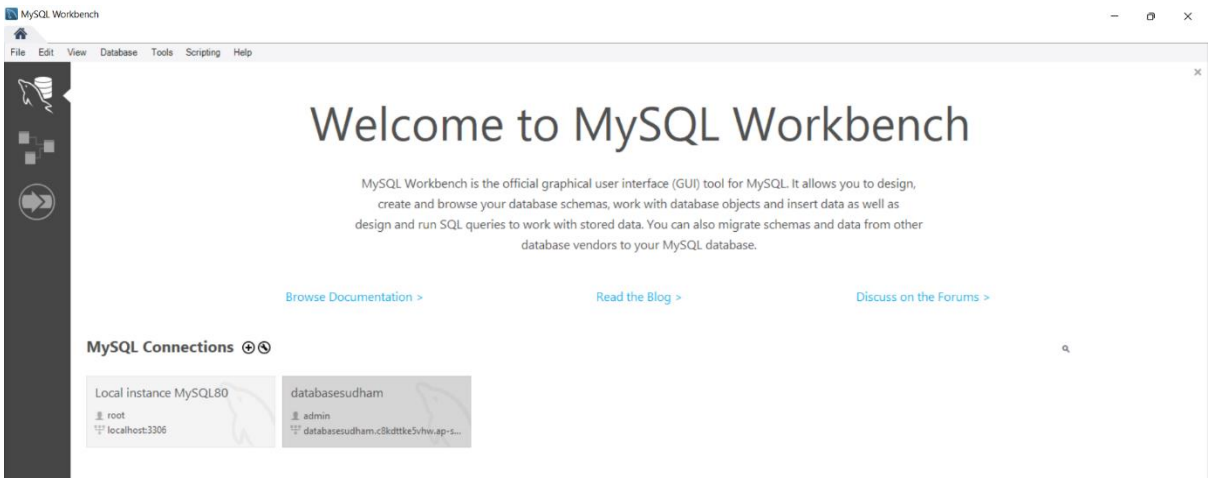
STEP 25: Go to **workbench** (after giving details click on to Test Connection).



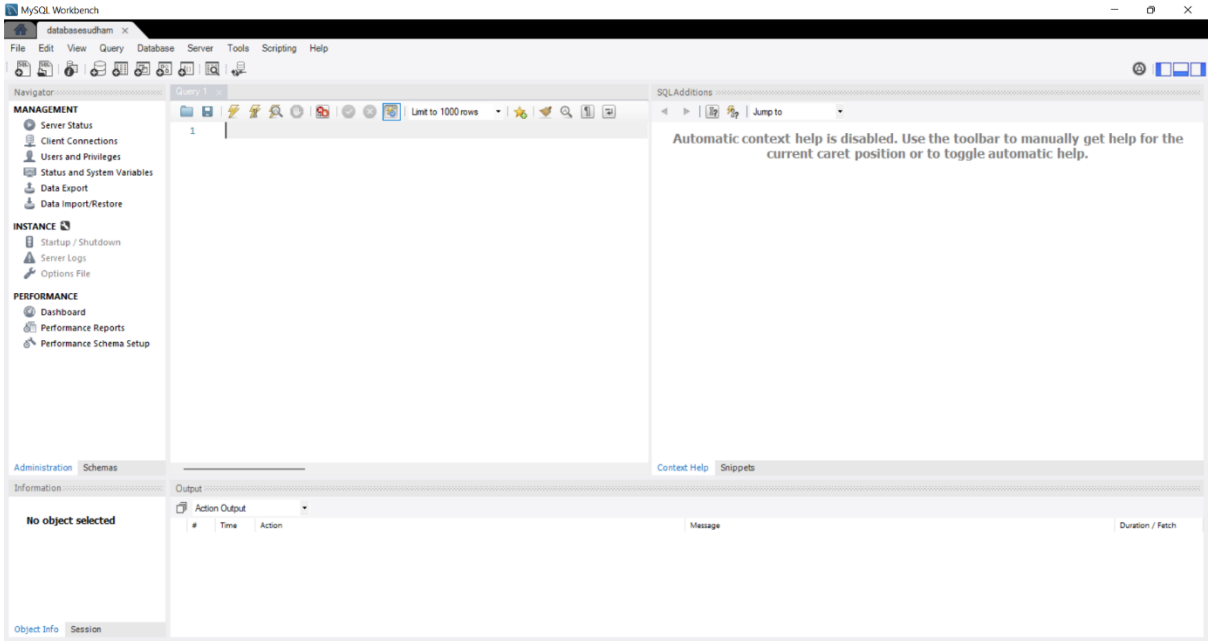


Click on **Ok** button

Go to workbench double click on connection (databasesudham)

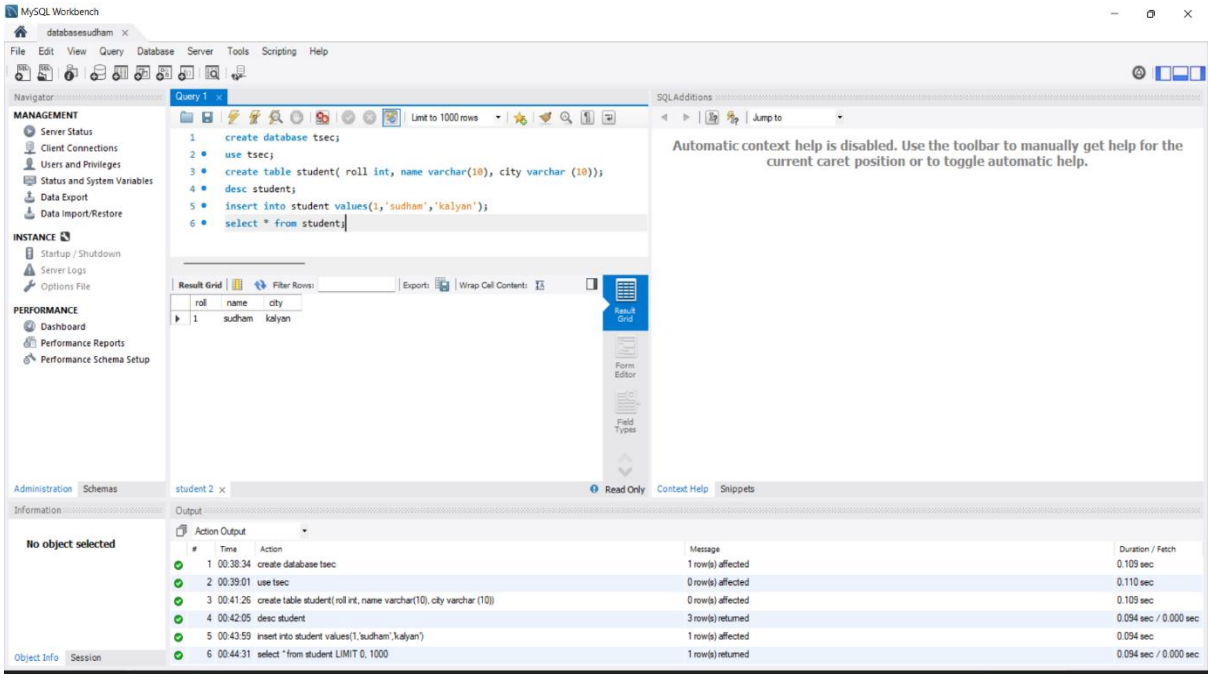


It will get opened.



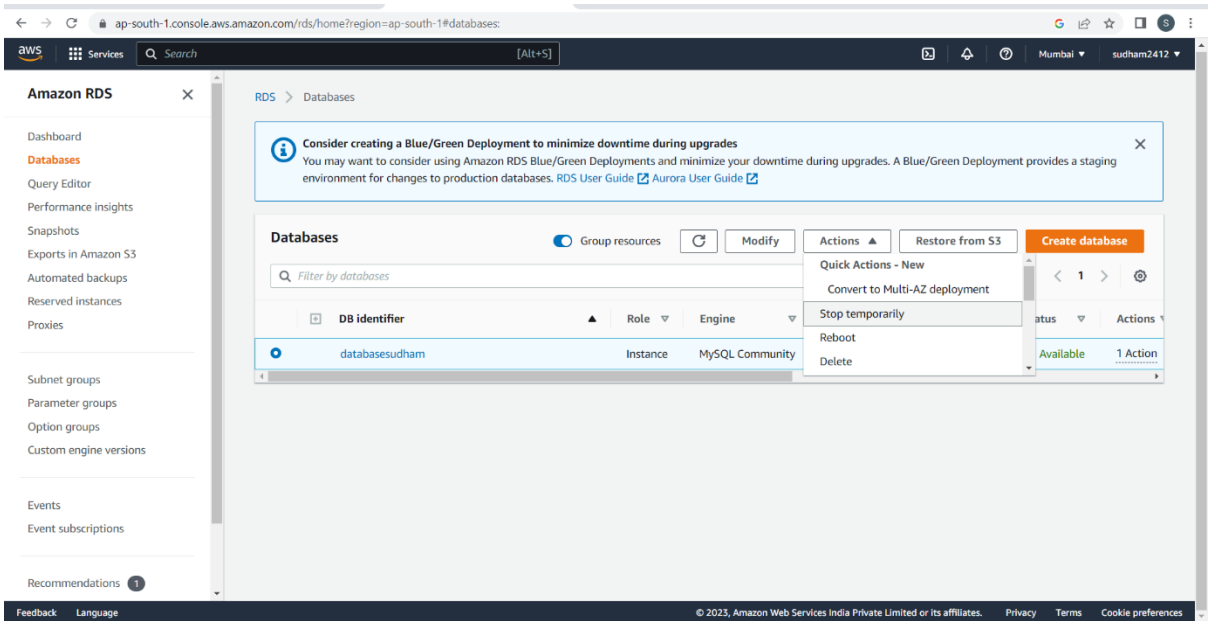
STEP 26: Write query and execute

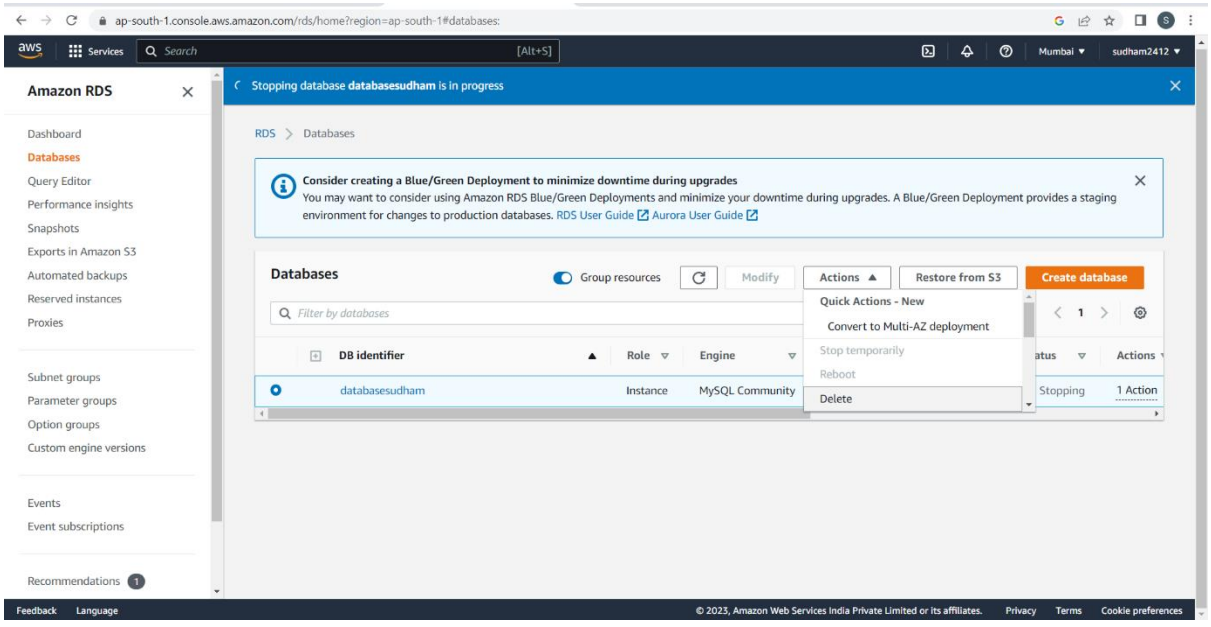
```
Create database tsec;
Use tsec;
Show tables;
create table student( roll int, name varchar(10), city varchar(10));
Desc student;
insert into student values(1,'sudham','kalyan');
```



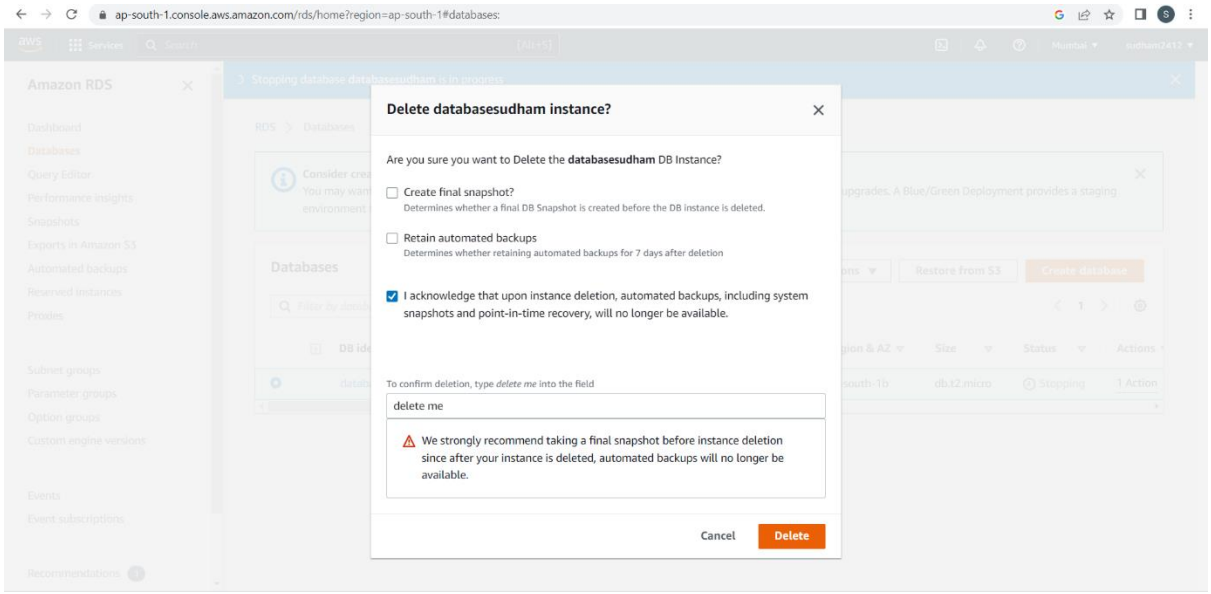
STEP 27: Now **delete the instance** (once you have done with it)

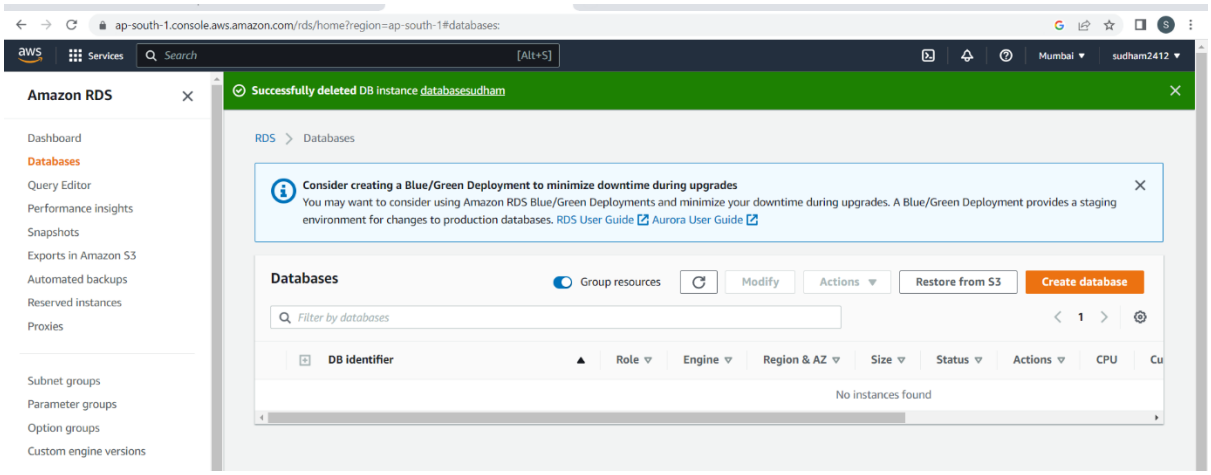
Select instance go to action **stop instance** and then **delete instance.**





Uncheck create final snapshot.





As you can see that DB instance is deleted successfully.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**CLOUD COMPUTING LAB**
COURSE CODE	**CSL605**
PRACTICAL NO.	**05**
DOP	**28/02/2023**
DOS	

SECURITY AS A SERVICE ON AWS

Aim : To study Security as a Service on AWS Security

Theory :

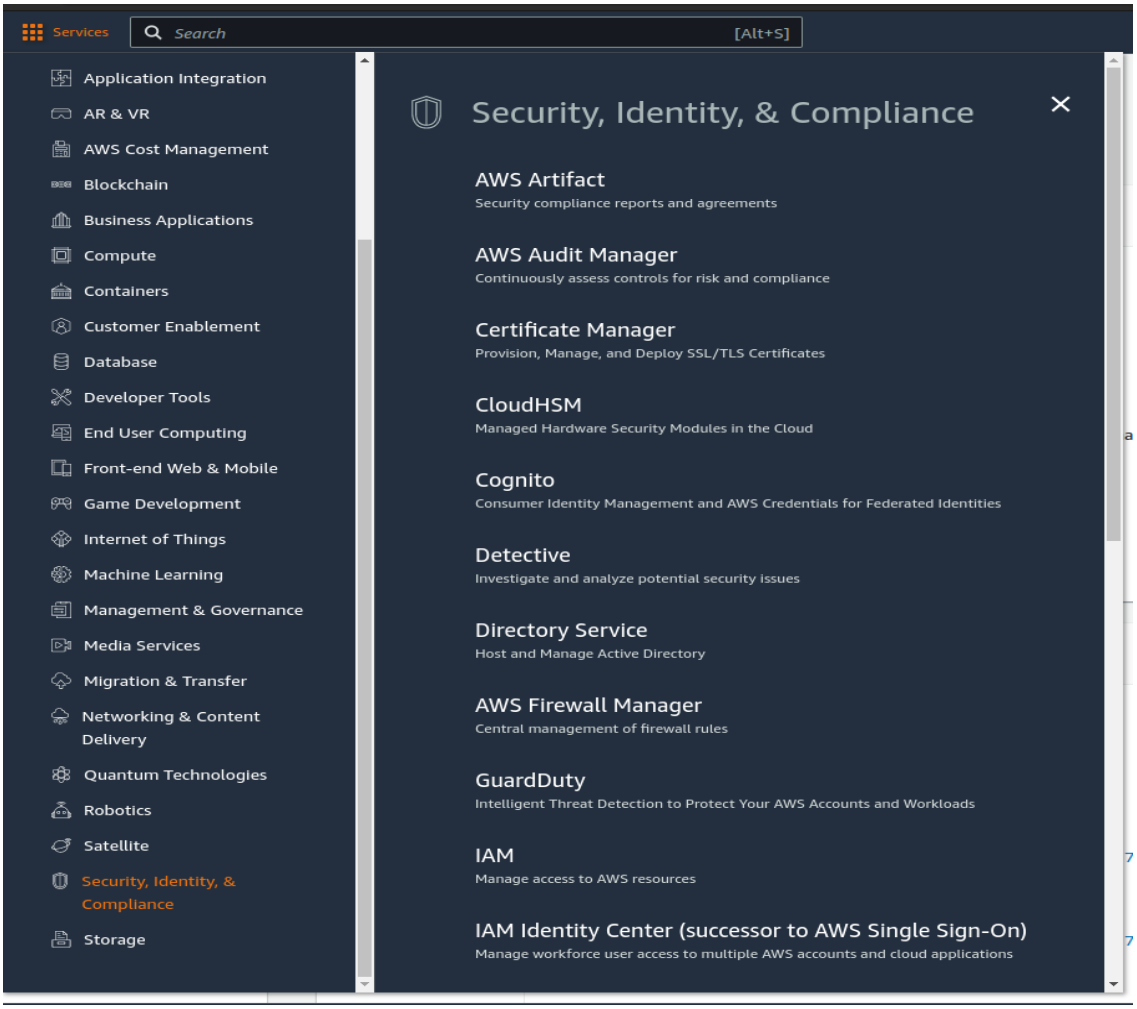
Cloud security at AWS is the highest priority. As organizations embrace the scalability and flexibility of the cloud, AWS is helping them evolve security, identity, and compliance into key business enablers. AWS builds security into the core of our cloud infrastructure, and offers foundational services to help organizations meet their unique security requirements in the cloud.

- ☐ As an AWS customer, you will benefit from a data center and network architecture built to meet the requirements of the most security-sensitive organizations. Security in the cloud is much like security in your on-premises data centers—only without the costs of maintaining facilities and hardware. In the cloud, you don't have to manage physical servers or storage devices. Instead, you use software-based security tools to monitor and protect the flow of information into and out of your cloud resources.
- ☐ An advantage of the AWS Cloud is that it allows you to scale and innovate, while maintaining a secure environment and paying only for the services you use. This means that you can have the security you need at a lower cost than in an on-premises environment.
- ☐ As an AWS customer you inherit all the best practices of AWS policies, architecture, and operational processes built to satisfy the requirements of our most security-sensitive customers. Get the flexibility and agility you need in security controls.
- ☐ The AWS Cloud enables a shared responsibility model. While AWS manages security of the cloud, you are responsible for security in the cloud. This means that you retain control of the security you choose to implement to protect your own content, platform, applications, systems, and networks no differently than you would in an on-site data center.
- ☐ AWS provides you with guidance and expertise through online resources, personnel, and partners. AWS provides you with advisories for current issues, plus you have the opportunity to work with AWS when you encounter security issues.
- ☐ You get access to hundreds of tools and features to help you to meet your security objectives. AWS provides security-specific tools and features across network security, configuration management, access control, and data encryption.
- ☐ Finally, AWS environments are continuously audited, with certifications from accreditation bodies across geographies and verticals. In the AWS environment, you can take advantage of automated tools for asset inventory and privileged access reporting.

Benefits of AWS security

- **Keep Your data safe —** The AWS infrastructure puts strong safeguards in place to help protect your privacy. All data is stored in highly secure AWS data centers.
- **Meet compliance requirements —** AWS manages dozens of compliance programs in its infrastructure. This means that segments of your compliance have already been completed.
- **Save money —**Cut costs by using AWS data centers. Maintain the highest standard of security without having to manage your own facility
- **Scale quickly —** Security scales with your AWS Cloud usage. No matter the size of your business, the AWS infrastructure is designed to keep your data safe.

Security as a Service on AWS refers to a set of cloud-based security services that are provided by Amazon Web Services (AWS) to help customers secure their applications and data in the cloud.



To study Security as a Service on AWS, you can follow the following steps:

- **Learn about the AWS Shared Responsibility Model:** The AWS Shared Responsibility Model is a critical concept to understand when it comes to security on AWS. AWS is responsible for securing the infrastructure that runs the cloud, while the customer is responsible for securing their data and applications in the cloud. Understanding the shared responsibility model is critical to ensuring that you are securing your applications and data appropriately on AWS.
- **Understand the various AWS Security Services:** AWS offers a wide range of security services that can help you secure your applications and data. These services include:
  1. Identity and Access Management (IAM): IAM allows you to manage access to AWS resources securely. You can create users and groups, assign permissions, and use IAM roles to grant temporary access to users or services.
  2. Amazon Inspector: Inspector is an automated security assessment service that helps you test the security of your applications and infrastructure.
  3. Amazon GuardDuty: GuardDuty is a threat detection service that continuously monitors for malicious activity in your AWS account.
  4. AWS WAF: The AWS Web Application Firewall (WAF) helps you protect your web applications from common web exploits and attacks.
  5. Amazon Macie: Macie is a fully managed data security and privacy service that uses machine learning and pattern matching to discover and protect sensitive data.
  6. AWS Certificate Manager: Certificate Manager is a service that lets you easily provision, manage, and deploy public and private Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services and your internal connected resources.

- **Study AWS Security Best Practices:** AWS publishes a set of security best practices for all of its services. These best practices include detailed guidance on how to secure your applications and data, and they cover topics such as access management, network security, data encryption, and logging.
- **Review AWS Compliance and Security Reports:** AWS regularly publishes compliance and security reports, such as SOC 2, ISO 27001, and PCI DSS reports. These reports provide independent verification of AWS's compliance with industry-standard security and compliance frameworks.
- **Practice with AWS Security Services:** AWS provides a free tier that allows you to experiment with many of its security services without incurring any costs. Use this opportunity to practice and experiment with AWS security services.
- **Get Certified:** AWS offers several security-related certifications, such as AWS Certified Security – Specialty. These certifications demonstrate your knowledge and expertise in securing applications and data on AWS. Consider getting certified to validate your skills.
- **Stay up-to-date:** AWS is constantly updating and adding new security features and services. Stay up-to-date with these changes by reading AWS blogs, attending webinars, and participating in AWS events.

By following these steps, you can gain a comprehensive understanding of Security as a Service on AWS and become proficient in securing applications and data in the cloud.

8 Securities Case Study

8 Securities uses the following products from AWS as part of its infrastructure:

- **Amazon EC2 Windows instances—** To run the portal, main website, and business intelligence tools
- **Elastic IP Addresses for Amazon EC2—**To bind to domain names, and start and stop instances on demand
- **Amazon EBS—** Used for data storage, for starting and stopping instances, and for having data always available
- **Amazon VPC—** Used for a production network. Roll notes, “As some of our applications are bound to the machine Media Access Control [MAC] address, the key benefit for us is to have static MAC addresses when instances are restarted.”
- **Elastic Load Balancing—** Used for the main website, to load balance across multiple instances

Top 6 AWS Account Security Tools

1. AWS Identity and Access Management (IAM)

AWS IAM is essential for controlling access to your AWS resources. It enables you to create users and roles with permissions to specific resources in your AWS environment. Always assigning least-privilege permissions to these users and roles minimizes the impact of a breach where an attacker has gained access. AWS IAM also has multi-factor authentication and supports single sign-on (SSO) access to further secure and centralize user access.

2. Amazon GuardDuty

Amazon GuardDuty uses machine learning to look for malicious activity in your AWS environments. It combines your CloudTrail event logs, VPC Flow Logs, S3 event logs, and DNS logs to continuously monitor and analyze all activity. GuardDuty identifies issues such as privilege escalation, exposed credentials, and communication with malicious IP addresses and domains. It can also detect when your EC2 instances are serving malware or mining bitcoin.

3. Amazon Macie

Amazon Macie discovers and protects your sensitive data stored in AWS S3 buckets. It first identifies sensitive data in your buckets, such as personally-identifiable information or personal health information, through discovery jobs. You can schedule these jobs to monitor new data added to your buckets. After it finds sensitive data, Macie continuously evaluates your buckets and alerts you when a bucket is unencrypted, is publicly accessible, or is shared with AWS accounts outside of your organization.

4. AWS Config

AWS Config records and continuously evaluates your AWS resource configuration. This includes keeping a historical record of all changes to your resources, which is useful for compliance with legal requirements and your organization’s policies. AWS Config evaluates new and existing resources against rules that validate certain configurations. For example, if all EC2 volumes must be encrypted, AWS Config can detect non-encrypted volumes and send a notification. In addition, it can also execute remediation actions such as encrypting the volume or deleting it.

5. AWS CloudTrail

AWS CloudTrail tracks all activity in your AWS environment. It records all actions a user executes in the AWS console and all API calls as events. You can view and search these events to identify unexpected or unusual requests in your AWS environment.

6. Security Hub

AWS Security Hub combines information from all the above services in a central, unified view. It collects data from all security services from multiple AWS accounts and regions, making it easier to get a complete view of your AWS security posture. In addition, Security Hub supports collecting data from third-party security products. Security Hub is essential to providing your security team with all the information they may need.

Top 4 AWS Application Security Tools

1. Amazon Inspector

Amazon Inspector is a security assessment service for applications deployed on EC2. These assessments include network access, common vulnerabilities and exposures (CVEs), Center for Internet Security (CIS) benchmarks, and common best practices such as disabling root login for SSH and validating system directory permissions on your EC2 instances.

2. AWS Shield

AWS Shield is a fully-managed distributed denial-of-service (DDoS) protection service. Shield is enabled by default as a free standard service with protection against common DDoS attacks against your AWS environment.

3. AWS Web Application Firewall

AWS Web Application Firewall (WAF) monitors and protects applications and APIs built on services such as CloudFront, API Gateway, and AppSync. You can block access to your endpoints based on different criteria such as the source IP address, the request's origin country, values in headers and bodies, and more (i.e, you can enable rate limiting, only allowing a certain number of requests per IP

4. AWS Secrets Manager

AWS Secrets Manager is a managed service where you can store and retrieve sensitive information such as database credentials, certificates, and tokens. Use fine-grained permissions to specify exact actions an entity can perform on the secrets, such as creating, updating, deleting, or retrieving secrets.

Conclusion : We had successfully studied Security as a Service on AWS Security.

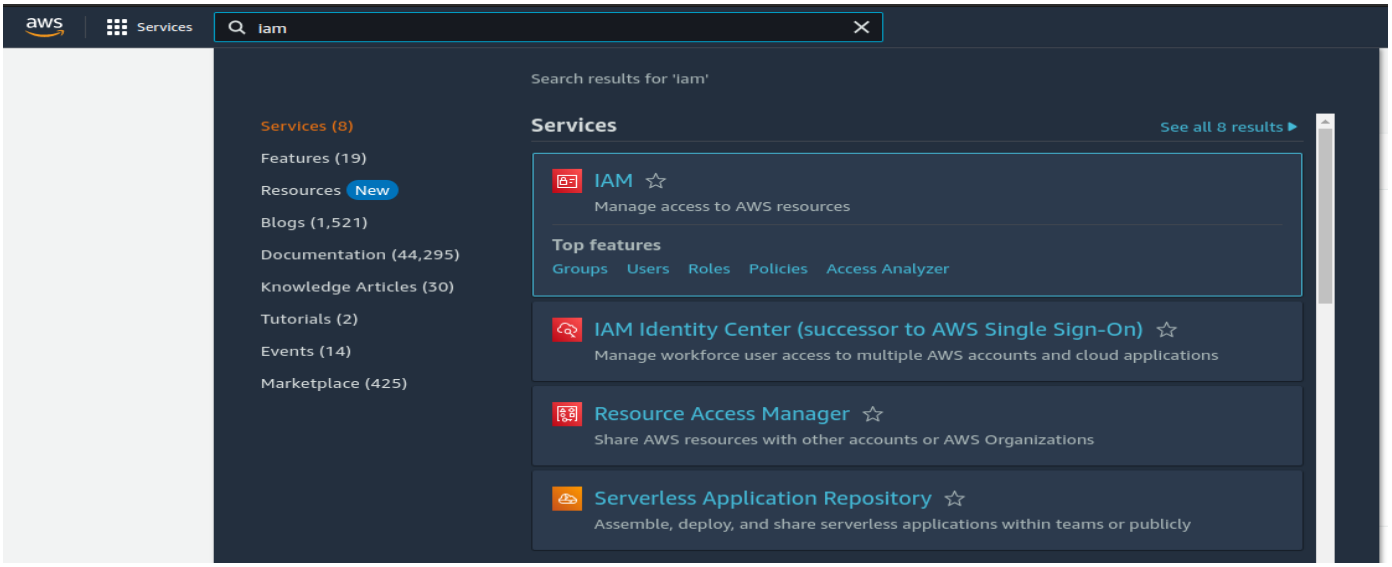
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

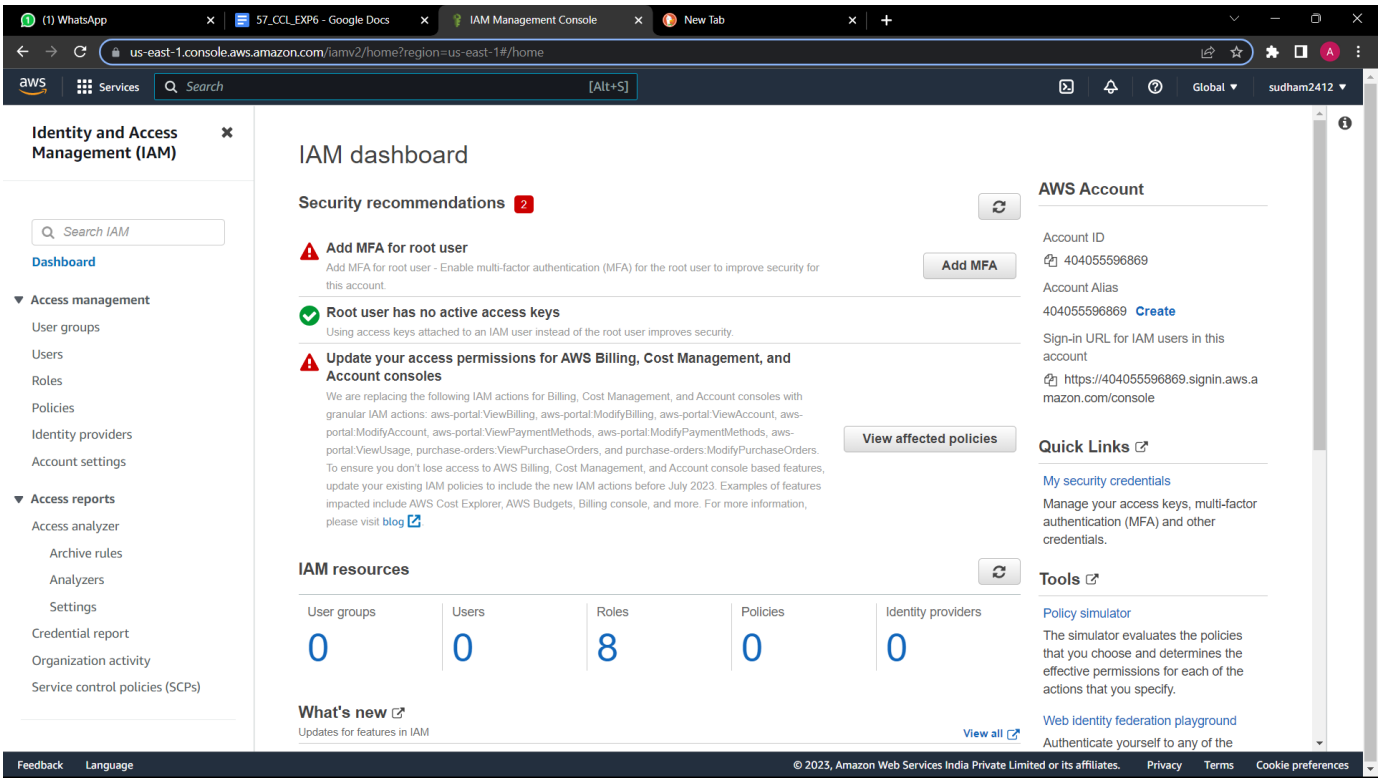
NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**CLOUD COMPUTING LAB**
COURSE CODE	**CSL605**
PRACTICAL NO.	**06**
DOP	
DOS	

IAM - (IDENTITY & ACCESS MANAGEMENT)

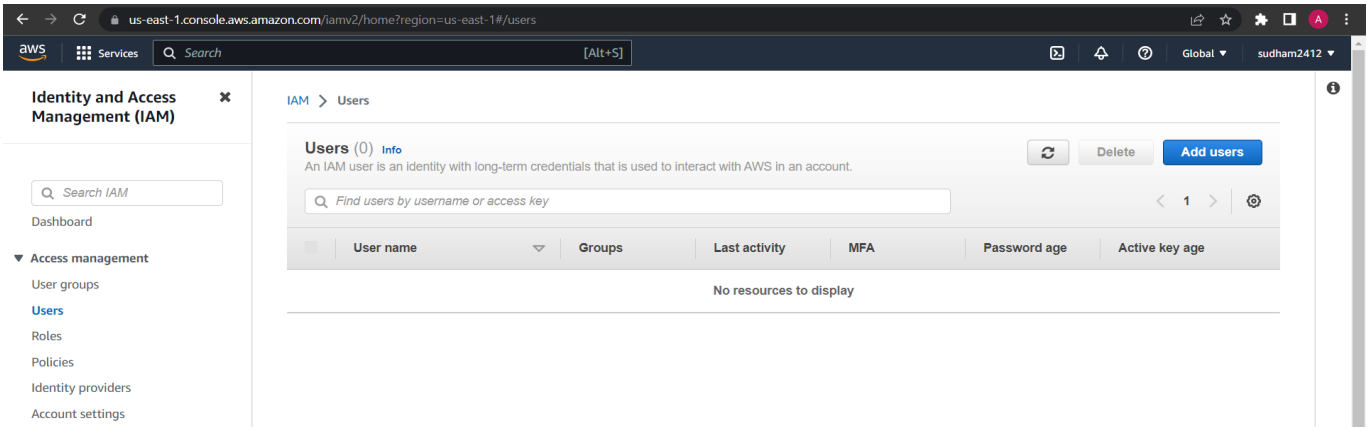
Step 1 - Sign in. Sign in as a root user. Provide username and password when prompted. Open the console and search for **IAM** and click on it.

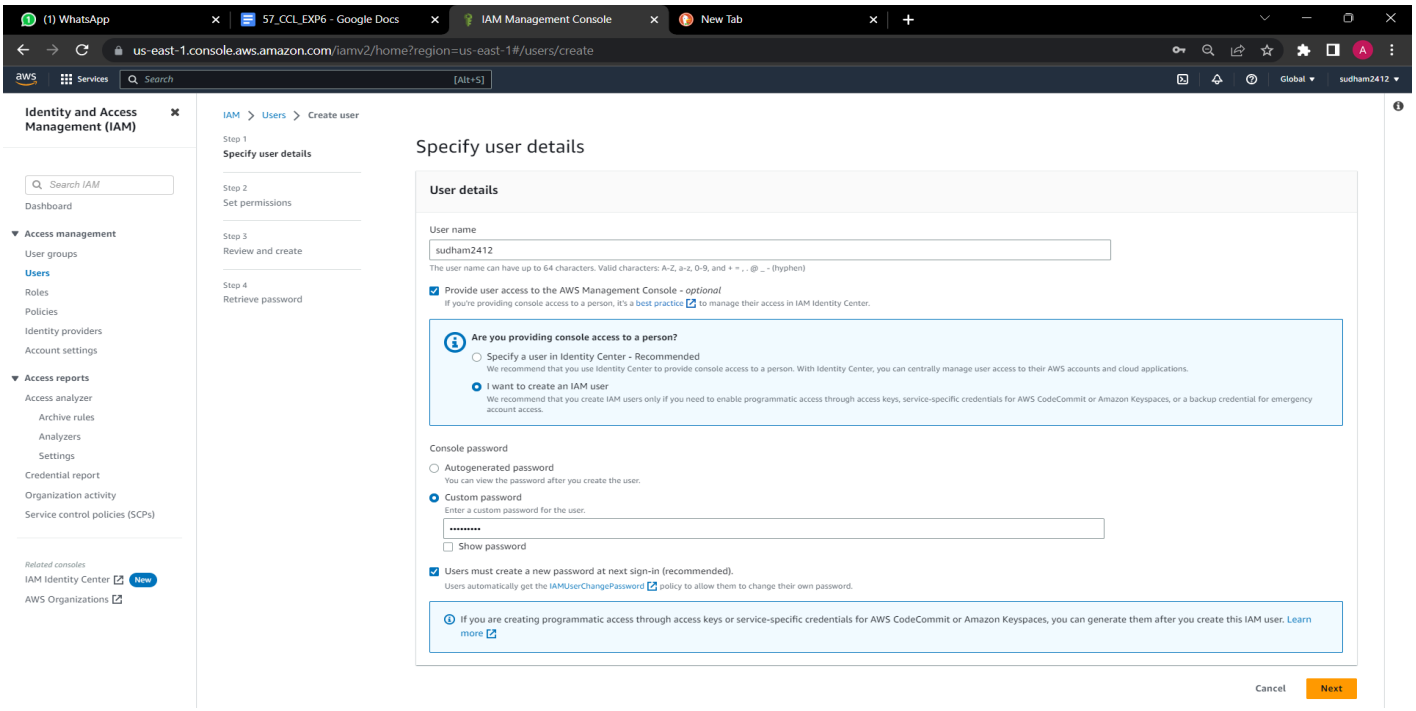


Step 2 - Select the Users menu. Navigate to the Users screen. You'll find it in the IAM dashboard, under **Identity and Access Management (IAM)** drop-down menu on the left side of the screen. Click on **Users.**



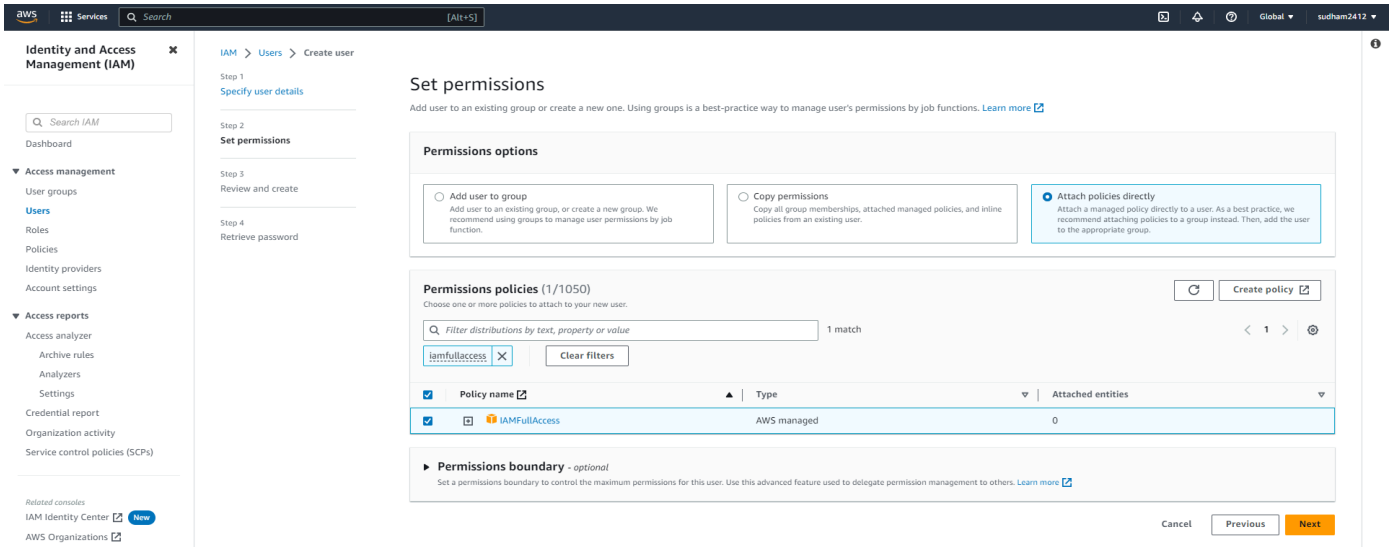
Step 3 - Add a user. Click on **Add User** to navigate to a user detail form. Provide all details, such as the username and access type. We use the name **cli-user**, and check the **Programmatic access** box under **Access type**. This option gives the user access to AWS development tools, such as the command line interface used later. Click on **Next: Permissions** to continue.



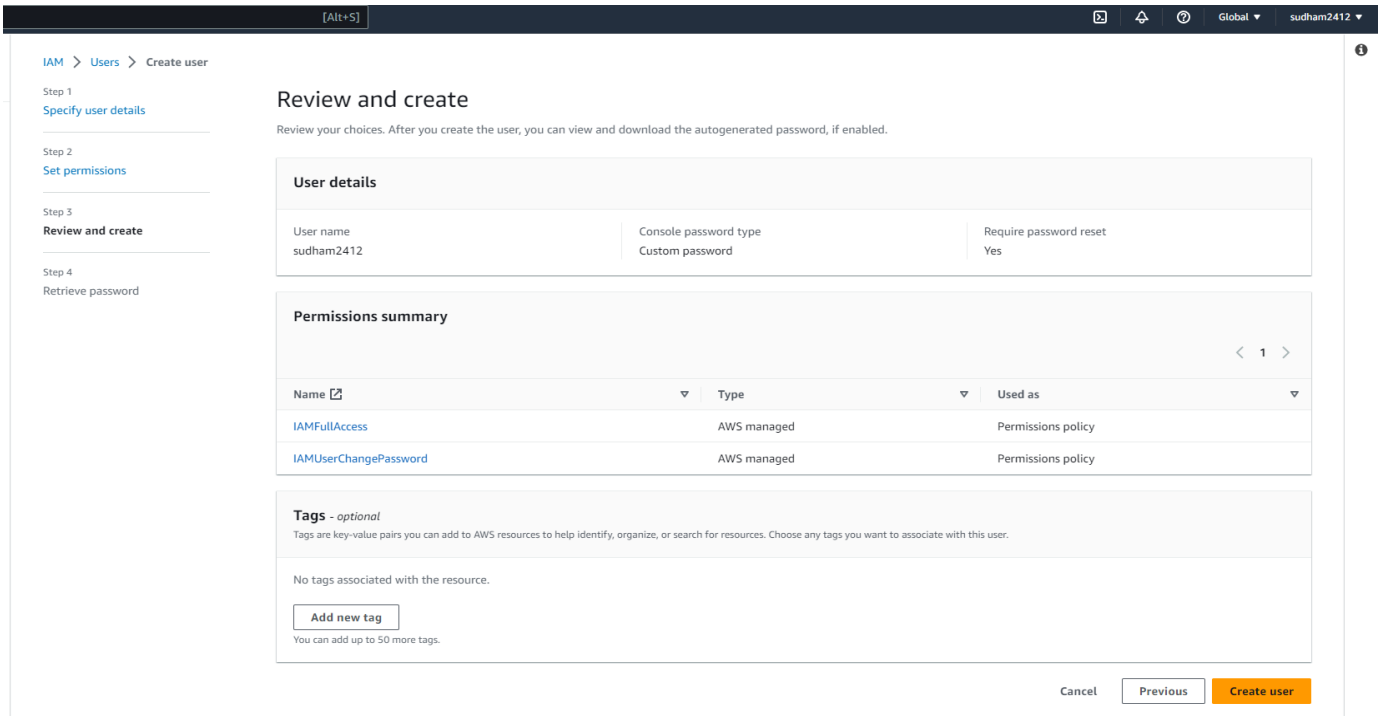


Step 4 - Set the user permissions. Click **Attach existing policies directly** and then filter the policies by keyword: IAM. For this user, select IAMFullAccess from the list of available policies.

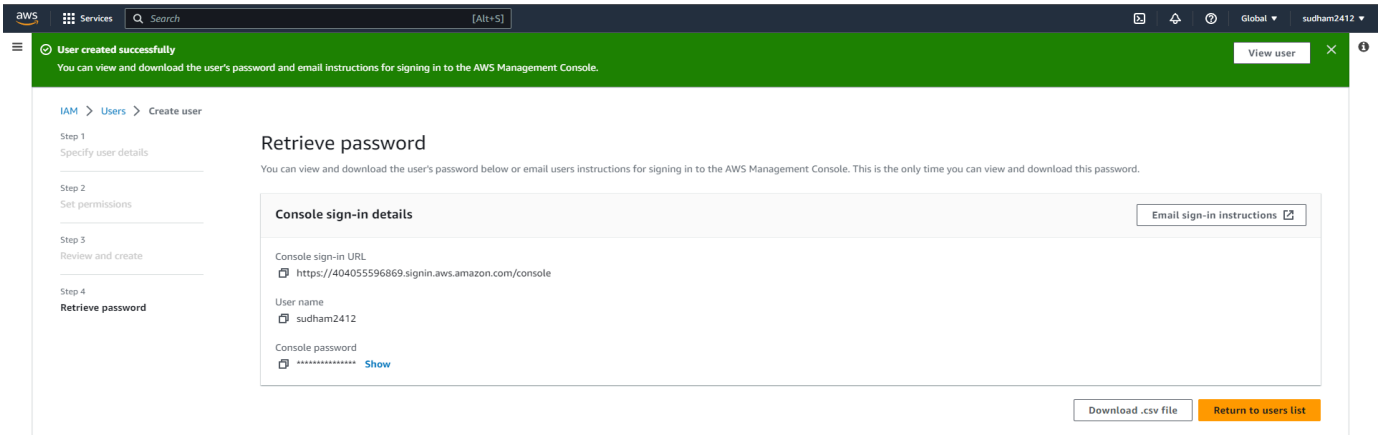
The IAMFullAccess policy enables this user to create and manage user permissions in AWS. Later, this user will perform AWS IAM operations.

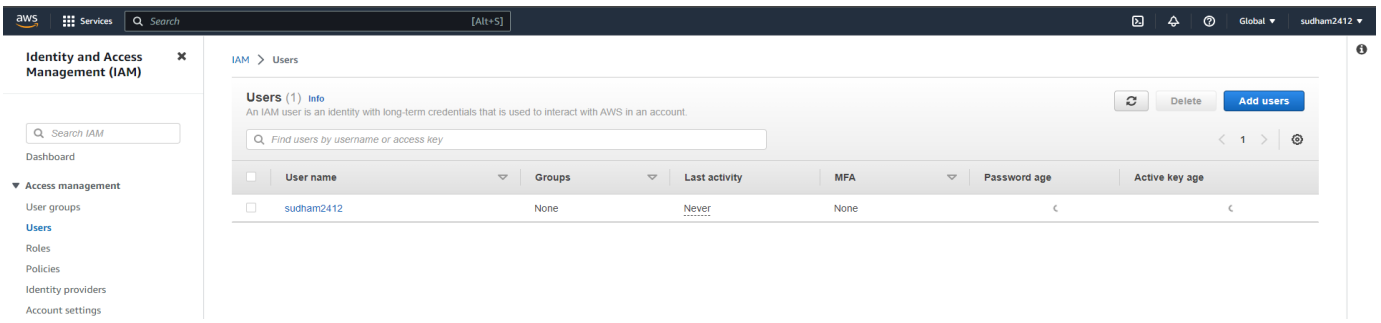


Step 5 - Finish the user setup. We will skip the tags section of user creation and go to the review page. Check details of the username, AWS access type & permissions. Then, click **Create user**.



Step 6 - At this point, the user cli-user exists, with the chosen policies applied to the account. AWS provides this user an access key ID and secret access key. Download or copy these keys to a secure place to use later.

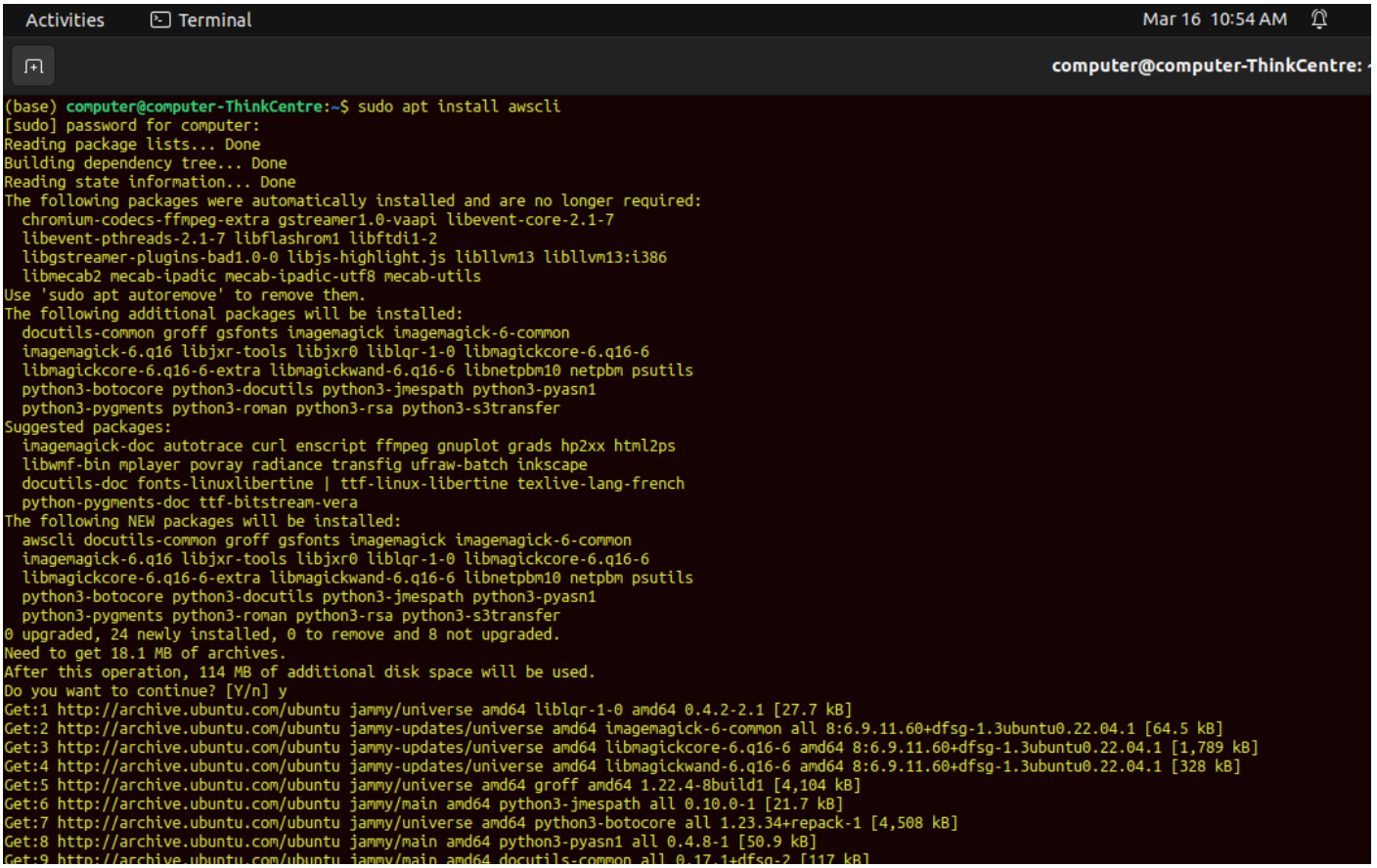


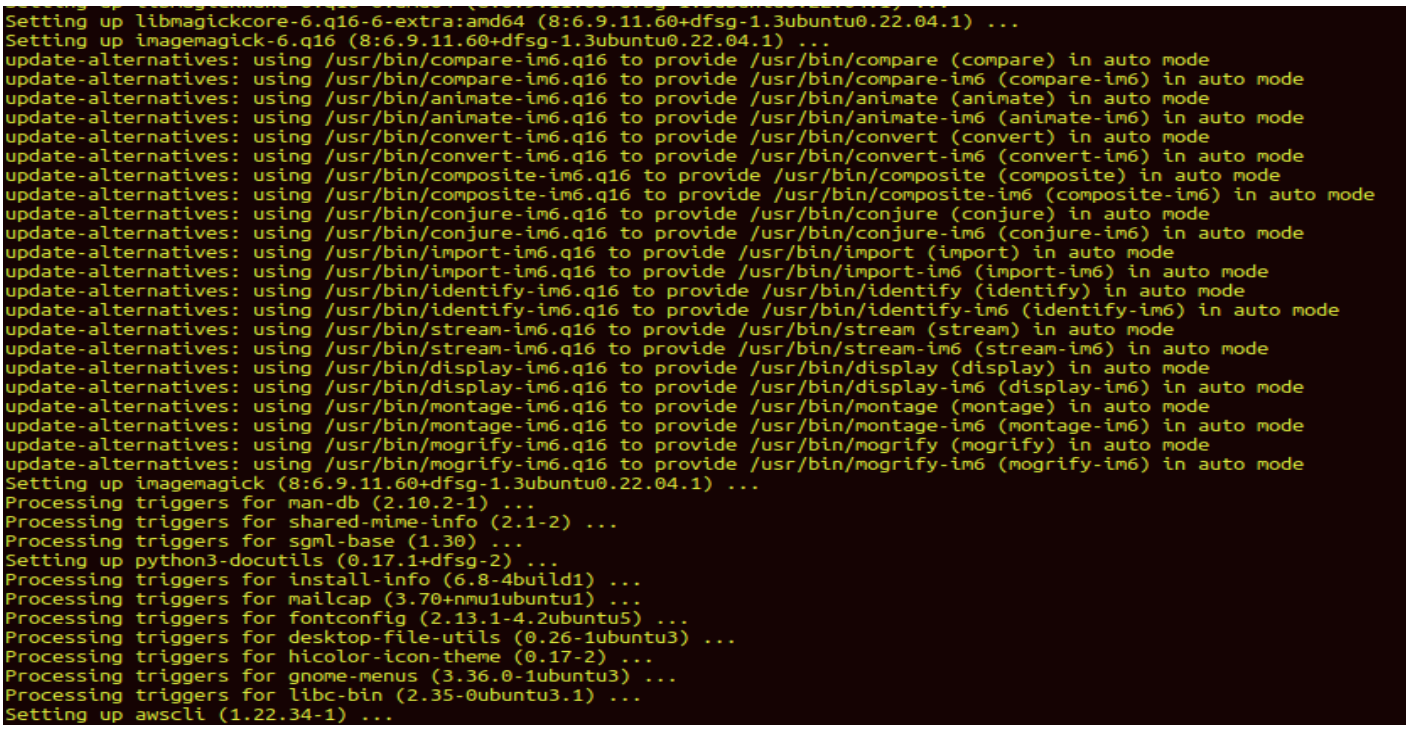


Set up AWS user credentials in the CLI

With minimal setup, AWS CLI enables an admin to use their favorite shell or CLI to interact with AWS services. You can choose any Linux distribution or shell. This demonstrates a Bash shell running on an Ubuntu Linux distribution.

1. Install the command line tool. First, install AWS CLI on your system using the following command in Bash terminal : **sudo apt install awscli**





Once the setup runs, verify the installation by checking the version : **aws –version**

```
(base) computer@computer-ThinkCentre:~$ aws --version
aws-cli/1.22.34 Python/3.10.6 Linux/5.19.0-35-generic botocore/1.23.34
(base) computer@computer-ThinkCentre:~$
```

2. Configure the user with the keys. Run the **aws configure** command in the shell to quickly set up the access key ID & secret access key obtained from AWS when you created a new user in IAM console.

```
(base) computer@computer-ThinkCentre:~$
(base) computer@computer-ThinkCentre:~$ aws configure
AWS Access Key ID [None]: AKIAXWRV!
AWS Secret Access Key [None]: sudham2412
Default region name [ap-south-1]:
Default output format [json]: json
(base) computer@computer-ThinkCentre:~$
```

This step saves your credentials in a local file at path: ~/.aws/credentials and region and output format configs at path: ~/.aws/config file.

Now that cli-user with programmatic access is set up, we can use that account to create other users and give them policy-based access through AWS CLI. The next two sections walk through these steps.

Create a user and assign permissions

To create a user using IAM, run the aws iam create-user command in AWS CLI with a username:

aws iam create-user --user-name sudham2412

It creates a new user and shows the user details in the bash console.

```
(base) computer@computer-ThinkCentre:~$ aws iam create-user --user-name sudham2412{
"User": {
"Path":
"UserName": "sudham2412",
"UserId": "AIDAXWRVBDHII7PCK47BG",
"Arn": "arn:aws:iam::529465940434:user/sudham2412",
"CreateDate": "2021-06-21T16:33:28Z"
          }
}
(base) computer@computer-ThinkCentre:~$
```

Suppose this user needs to manage EC2 services. To grant this new user EC2 admin rights, start by listing which EC2 policies we can grant. Use the command:

aws iam list-policies | grep EC2FullAccess

Identify the appropriate policy for the user's access level. In this case, it is AmazonEC2FullAccess. Pass the Amazon Resource Name (ARN) to the following command in --policy-arn parameter:

aws iam attach-user-policy --user-name sudham2412 --policy-arn "arn:aws:iam::aws:policy/AmazonEC2FullAccess"

```
(base) computer@computer-ThinkCentre:~$
(base) computer@computer-ThinkCentre:~$ aws iam list-policies | grep EC2FullAccess
 "PolicyName": "Amazon EC2FullAccess", "Arn": "arn:aws: iam: : aws: policy/Amazon EC2FullAccess",
(base) computer@computer-ThinkCentre:~$
(base)  computer@computer-ThinkCentre:~$  aws  iam  attach-user-policy  --user-name  sudham2412  --policy-arn
"arn:aws:iam::aws:policy/Amazon EC2FullAccess"
(base) computer@computer-ThinkCentre:~$
```

Check user details and list user permissions

Once you create the user and attach the appropriate user policy to them, verify that AWS assigned the appropriate policy by checking the user details.

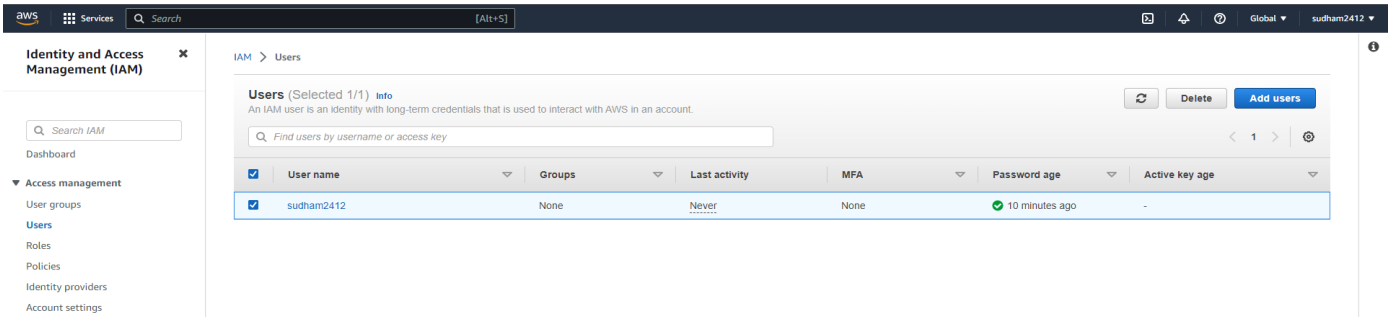
To check the list of IAM users, run: aws iam list-users

The following command tells AWS to list all attached policies for a user account:

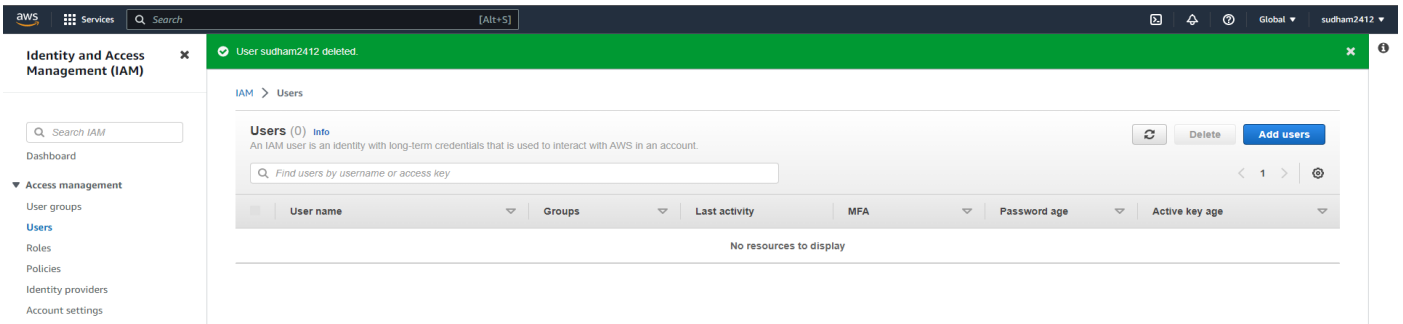
aws iam list-attached-user-policies --user-name sudham2412

```
(base) computer@computer-ThinkCentre:~$ ~
(base) computer@computer-ThinkCentre:~$
(base) computer@computer-ThinkCentre:~$ aws iam list-users
{
"Users": [
{
"Path": "/",
"UserName": "cli-user",
"UserId": "AIDAXWRVBDHJDOWXYJSI3",
"Arn": "arn:aws: iam::529465940434:user/cli-user",
"CreateDate": "2021-06-21T15:48:20Z"
},
{
"Path": "/",
"UserName": "sudham2412",
"UserId": "AIDAXWRVBDHJ17PCK47BG",
"Arn": "arn:aws:iam::529465940434: user/sudham2412", "CreateDate": "2021-06-21T16:33:28Z"
}
(base) computer@computer-ThinkCentre:~$ aws iam list-attached-user-policies
--user-name sudham2412{
"AttachedPolicies": [
{
"PolicyName": "Amazon EC2FullAccess",
"PolicyArn": "arn:aws:iam::aws:policy/Amazon EC2FullAccess"
}
(base) computer@computer-ThinkCentre:~$
```

- Now you can **delete** your users from **IAM** - **Access Management > Users >** select your username **>** click on **Delete**



- You can see the users you created deleted successfully.



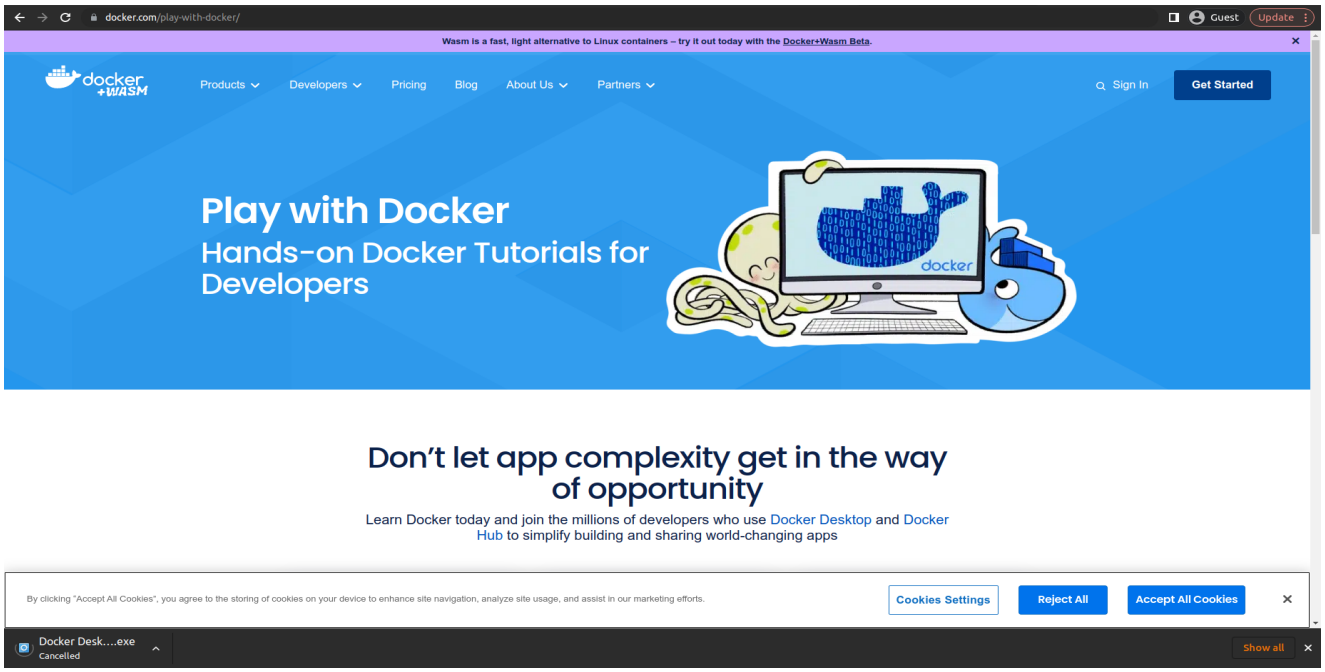
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**CLOUD COMPUTING LAB**
COURSE CODE	**CSL605**
PRACTICAL NO.	**07**
DOP	**21/03/2023**
DOS	

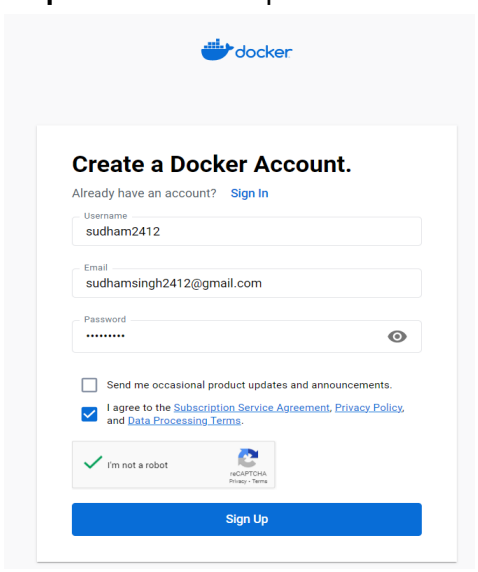
DOCKER

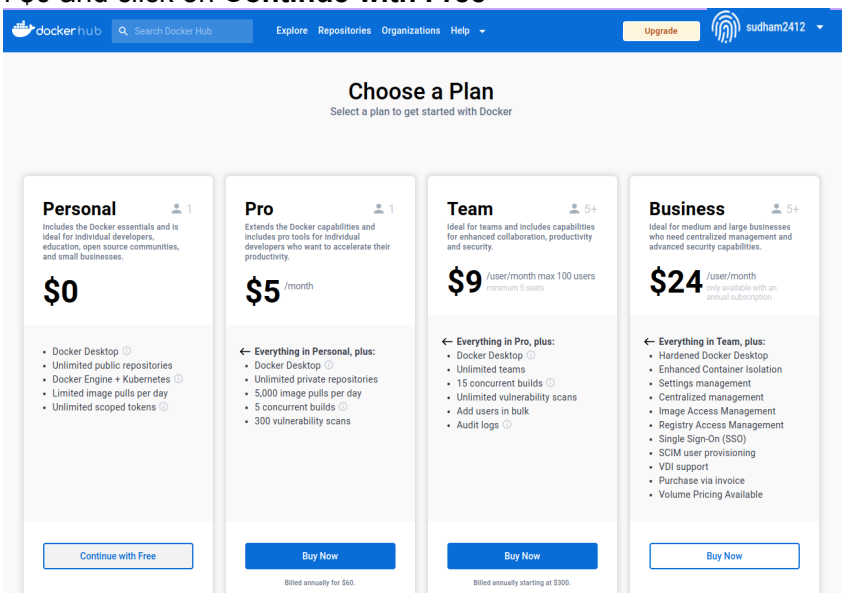
Step 1 - Go to official website of docker : https://hub.docker.com/



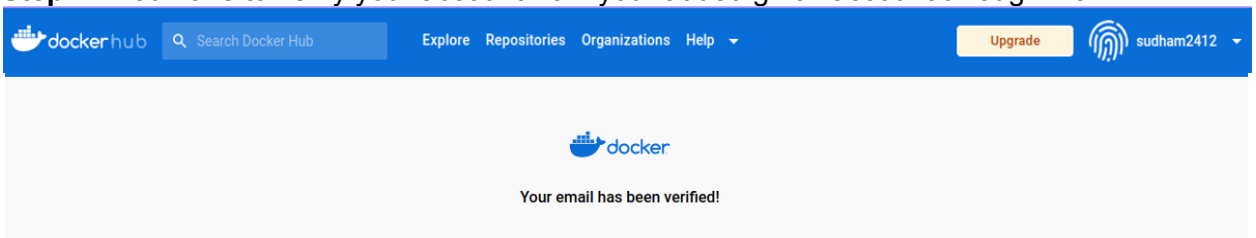
Step 2 - Create your docker account

Step 3 - Choose for personal use with $0 and click on **Continue with Free**



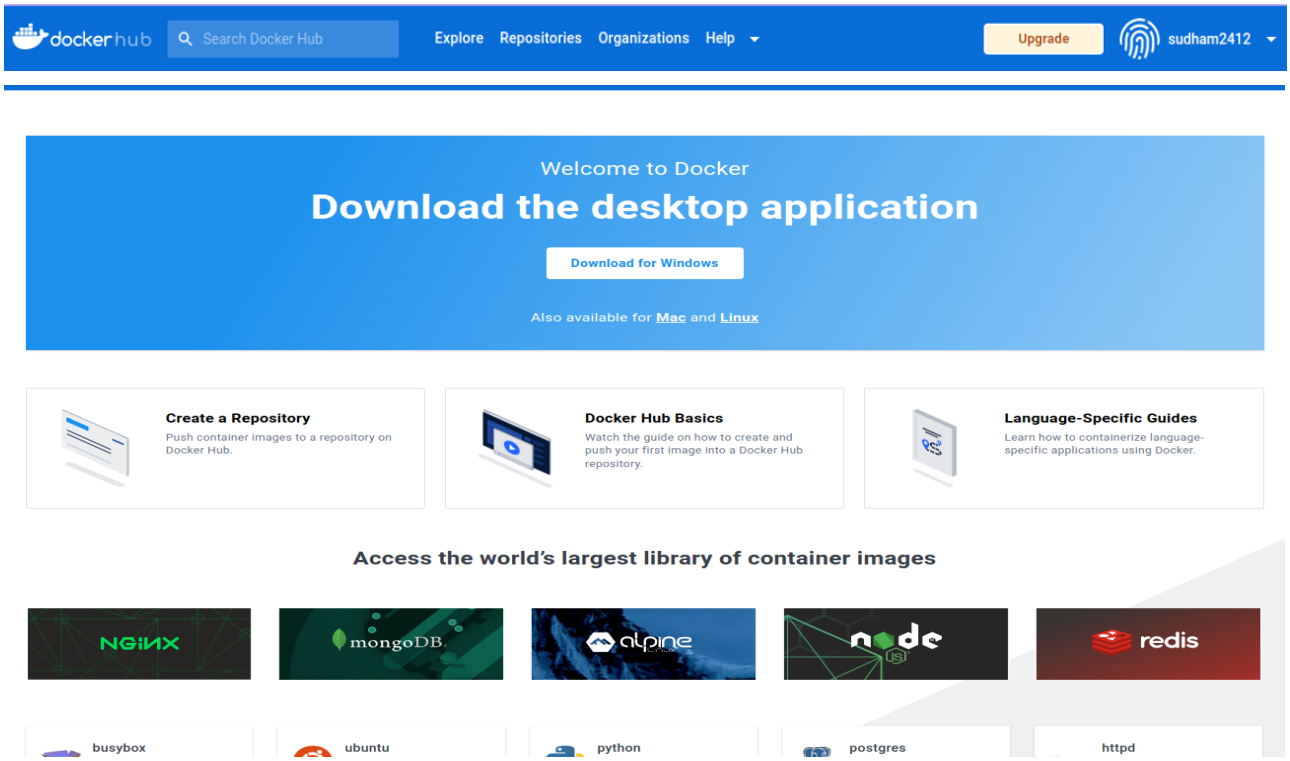


Step 4 - You have to verify your account from your added gmail account through mail



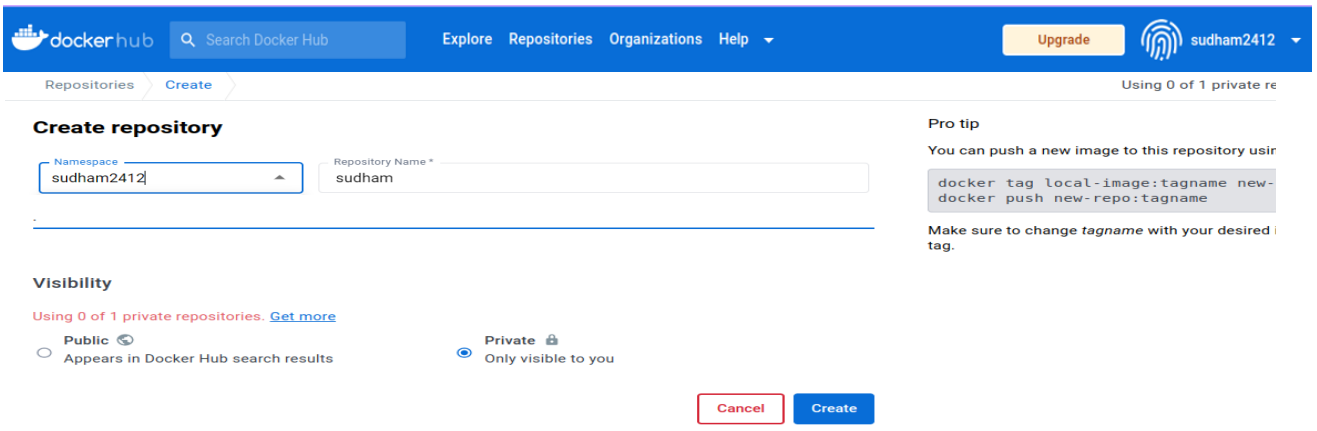
Step 5 - Creating first repository

Click on **Create a Repository**

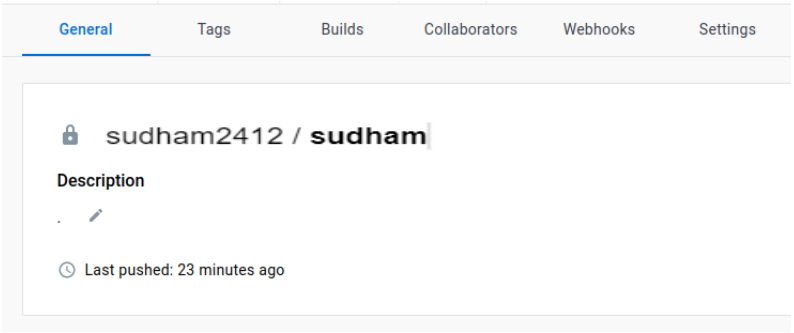


Name it <your-username>/sudham

Set the visibility to private

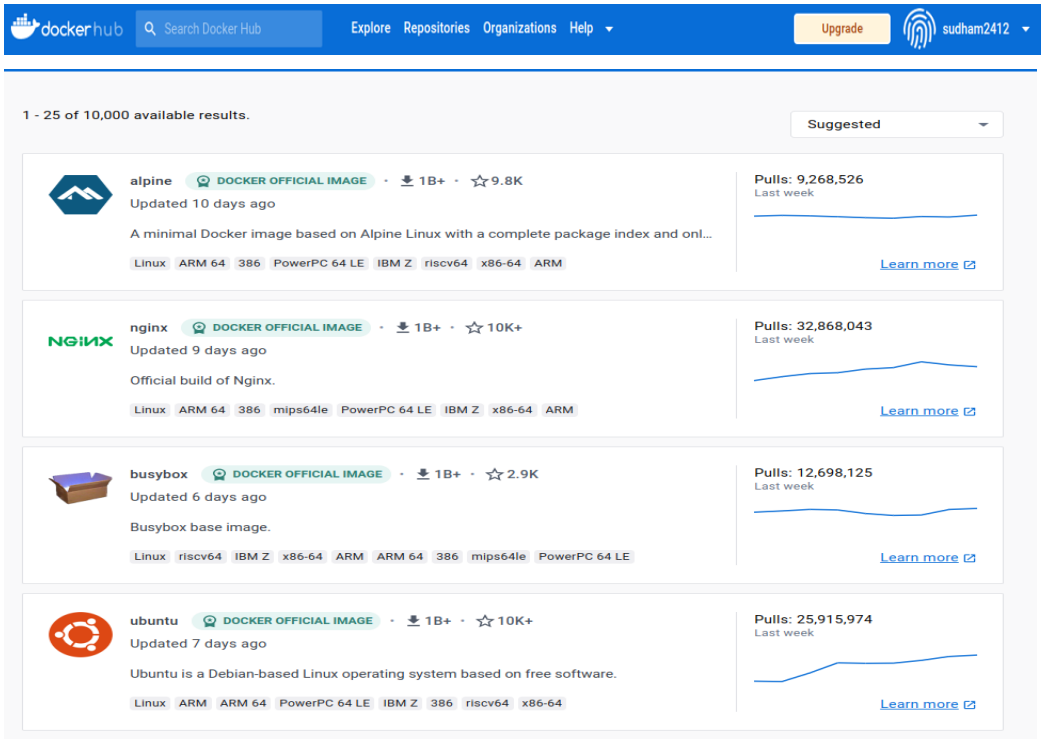


Click on **Create**

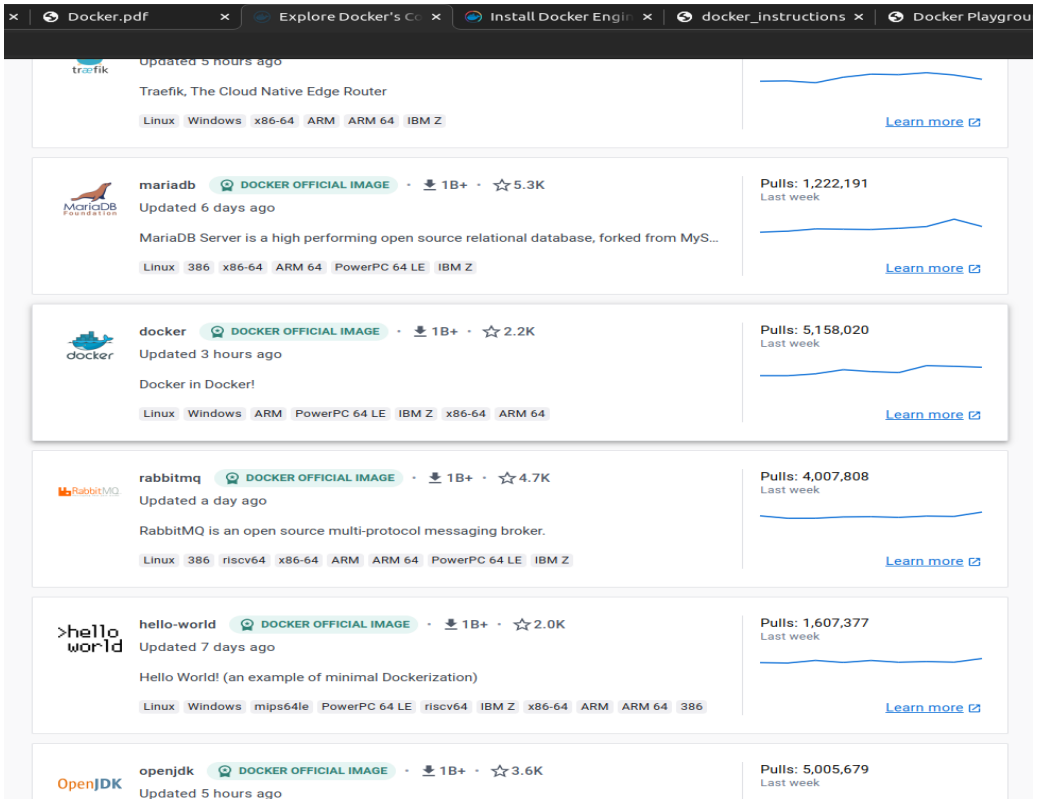


Now next,

Step 6 - Click on Explore tab to see official and publisher images



Here you can click on the docker tab



You can explore that tab in which various versions of docker's downloads are given.
You can select various platforms, architecture and operating systems

Installing using apt repository on ubuntu

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Set up the repository

Step 1 - Update the apt package index and install packages to allow apt to use a repository over HTTPS:

```
sudo apt-get update
sudo apt-get install \
  ca-certificates \
  curl \
  gnupg
```

Step 2 - Add Docker's official GPG key:

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Step 3 - Use the following command to set up the repository:

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
(base) computer@computer-ThinkCentre:~$ sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
Hit:1 https://deb.nodesource.com/node_16.x jammy InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:5 https://dl.google.com/linux/chrome/deb stable InRelease
Get:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Fetched 336 kB in 4s (79.7 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu4).
lsb-release set to manually installed.
ca-certificates is already the newest version (20211016ubuntu0.22.04.1).
ca-certificates set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi gyp javascript-common
  libflashrom1 libftdi1-2 libgstreamer-plugins-bad1.0-0 libjs-events
  libjs-highlight.js libjs-inherits libjs-is-typedarray libjs-psl
  libjs-source-map libjs-sprintf-js libjs-typedarray-to-buffer libnode-dev
  libnode72 libssl-dev libuv1-dev node-abbrev node-ansi-regex node-ansi-styles
  node-ansistyles node-are-we-there-yet node-arrify node-asap node-asynckit
  node-balanced-match node-brace-expansion node-chownr node-clean-yaml-object
  node-color-convert node-color-name node-commander node-core-util-is
  node-decompress-response node-delayed-stream node-delegates node-depd
  node-diff node-encoding node-end-of-stream node-err-code
  node-escape-string-regexp node-fancy-log node-foreground-child
  node-fs.realpath node-function-bind node-get-stream node-glob node-growl
  node-has-flag node-has-unicode node-hosted-git-info node-iconv-lite
  node-iferr node-imurmurhash node-indent-string node-inflight node-inherits
  node-ini node-ip node-ip-regex node-is-buffer node-is-plain-obj
  node-is-typedarray node-isarray node-isexe node-json-parse-better-errors
  node-jsonparse node-kind-of node-lodash-packages node-lowercase-keys
  node-lru-cache node-mimic-response node-minimatch node-minimist
  node-minipass node-mute-stream node-negotiator node-npm-bundled node-once
  node-osenv node-p-cancelable node-p-map node-path-is-absolute
  node-process-nextick-args node-promise-inflight node-promise-retry
  node-promzard node-pump node-quick-lru node-read node-readable-stream
  node-resolve node-retry node-safe-buffer node-set-blocking node-signal-exit
  node-slash node-slice-ansi node-source-map node-spdx-correct
  node-spdx-exceptions node-spdx-expression-parse node-spdx-license-ids
  node-sprintf-js node-stealthy-require node-string-decoder
  node-supports-color node-text-table node-time-stamp node-tmatch
  node-typedarray-to-buffer node-universalify node-util-deprecate
  node-validate-npm-package-license node-webidl-conversions node-whatwg-fetch
  node-wrappy node-yallist
Use 'sudo apt autoremove' to remove them.
```

Installing Docker Engine

Step 1 - Update the apt package index : **sudo apt-get update**

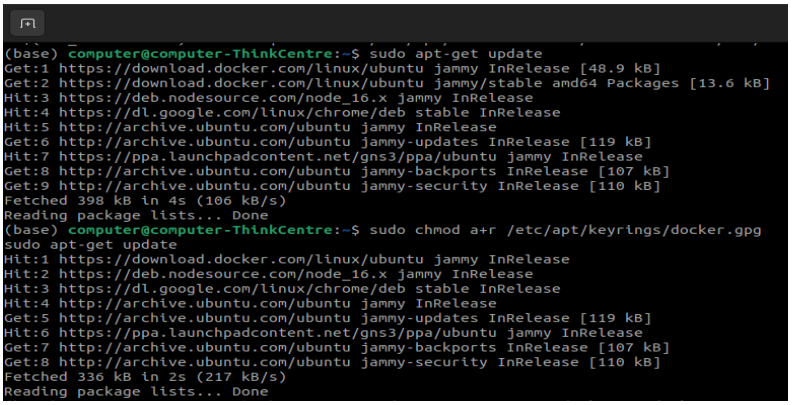
Receiving a GPG error when running apt-get update?

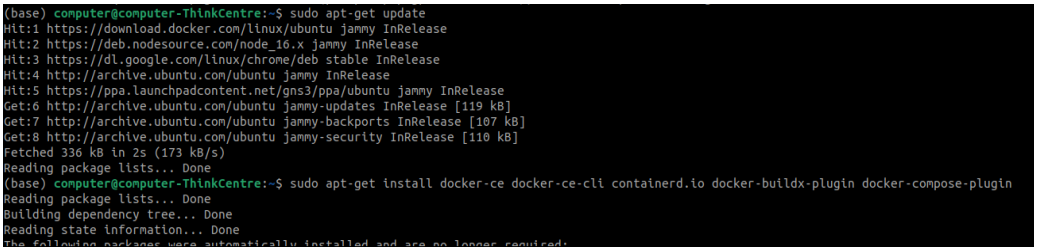
Your default umask may be incorrectly configured, preventing detection of the repository public key file. Try granting read permission for the Docker public key file before updating the package index : **sudo chmod a+r /etc/apt/keyrings/docker.gpg**

sudo apt-get update

Step 2 - Install Docker Engine, containerd, and Docker Compose.

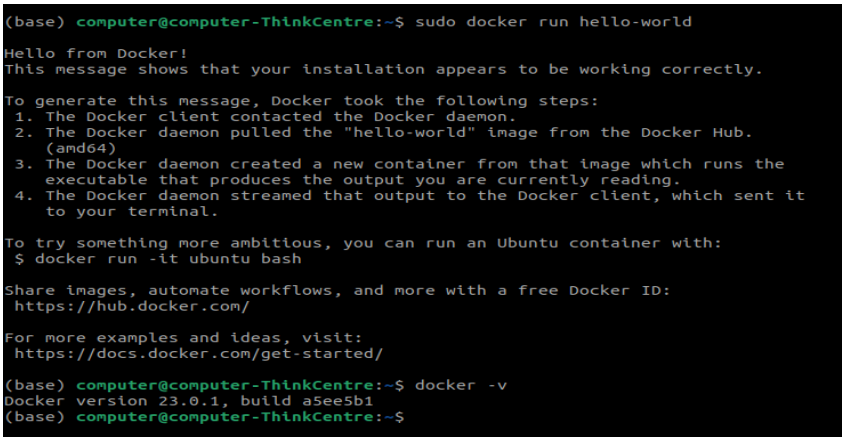
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin





Step 3 - Verify that the Docker Engine installation is successful by running the hello-world image : **sudo docker run hello-world**

Step 4 - Check docker version to ensure that its installed successfully : **docker -v**

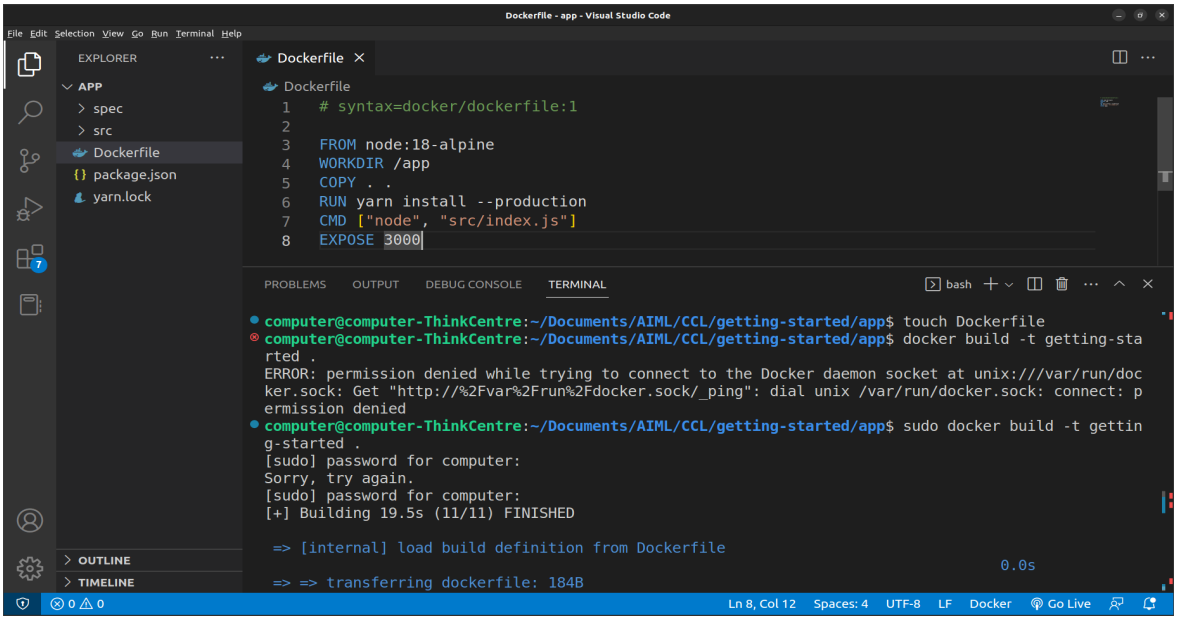


Creating containerised docker

Step 1 - Cloning : **git clone https://github.com/docker/Cloning into 'getting-started'...**
Step 2 - Open code : **code ./getting-started/app/**

```
(base) computer@computer-ThinkCentre:~/Documents/AIML/CCL$ git clone https://github.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 957, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 957 (delta 0), reused 1 (delta 0), pack-reused 952
Receiving objects: 100% (957/957), 5.24 MiB | 21.05 MiB/s, done.
Resolving deltas: 100% (541/541), done.
(base) computer@computer-ThinkCentre:~/Documents/AIML/CCL$ code ./getting-started/app/
(base) computer@computer-ThinkCentre:~/Documents/AIML/CCL$
```

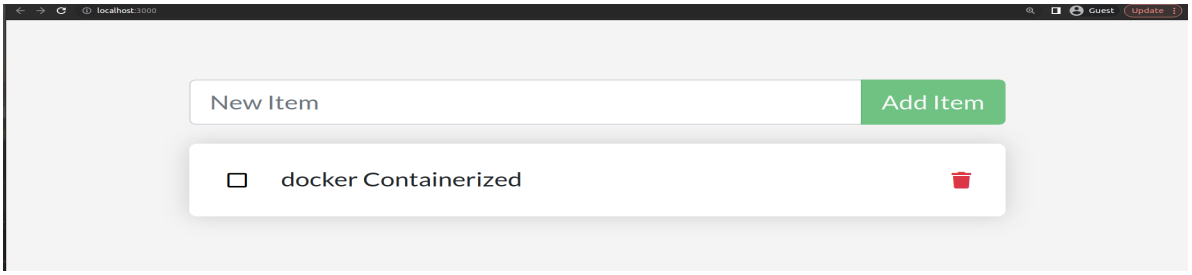
Step 3 - Create file : **Dockerfile** and locate localhost, then enter :
sudo docker build -t getting-started .
Type required password of your system



Step 4 - Getting-started : **sudo docker run -dp 3000:3000 getting-started**

```
computer@computer-ThinkCentre:~/Documents/AIML/CCL/getting-started/app$ sudo docker run -dp 3000:3000 getting-started
33762
computer@computer-ThinkCentre:~/Documents/AIML/CCL/getting-started/app$
```

You can see that containerized docker created successfully on your localhost



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM VI/CBCGS/AIML
Academic Year: 2022-23

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**57**
SUBJECT	**CLOUD COMPUTING LAB**
COURSE CODE	**CSL605**
PRACTICAL NO.	**08**
DOP	
DOS	

KUBERNETES

Aim : To study container orchestration using Kubernetes

Theory :

Kubernetes (also known as "K8s") is an open-source container orchestration tool that helps manage and automate the deployment, scaling, and management of containerized applications. Kubernetes was originally developed by Google, and is now maintained by the Cloud Native Computing Foundation (CNCF).

At a high level, Kubernetes works by managing a cluster of nodes (physical or virtual machines) that run containerized applications. A Kubernetes cluster consists of one or more master nodes, which control the cluster and make decisions about scheduling and scaling containers, and one or more worker nodes, which actually run the containers.

Benefits of Kubernetes

Kubernetes is a powerful container orchestration tool that offers a number of benefits to organizations deploying containerized applications:

- Scalability: Kubernetes enables automatic scaling of containers to meet the changing demands of an application. It can automatically add or remove containers based on metrics like CPU and memory utilization.

- High availability: Kubernetes provides a highly available infrastructure for containerized applications by automatically restarting containers that fail, and distributing containers across multiple nodes in a cluster.

- Portability: Kubernetes enables easy deployment of containerized applications across different environments, including on-premises data centers, public clouds, and hybrid environments.

- Resource efficiency: Kubernetes helps optimize the use of computing resources by packing containers efficiently onto nodes in the cluster, and dynamically allocating resources as needed.

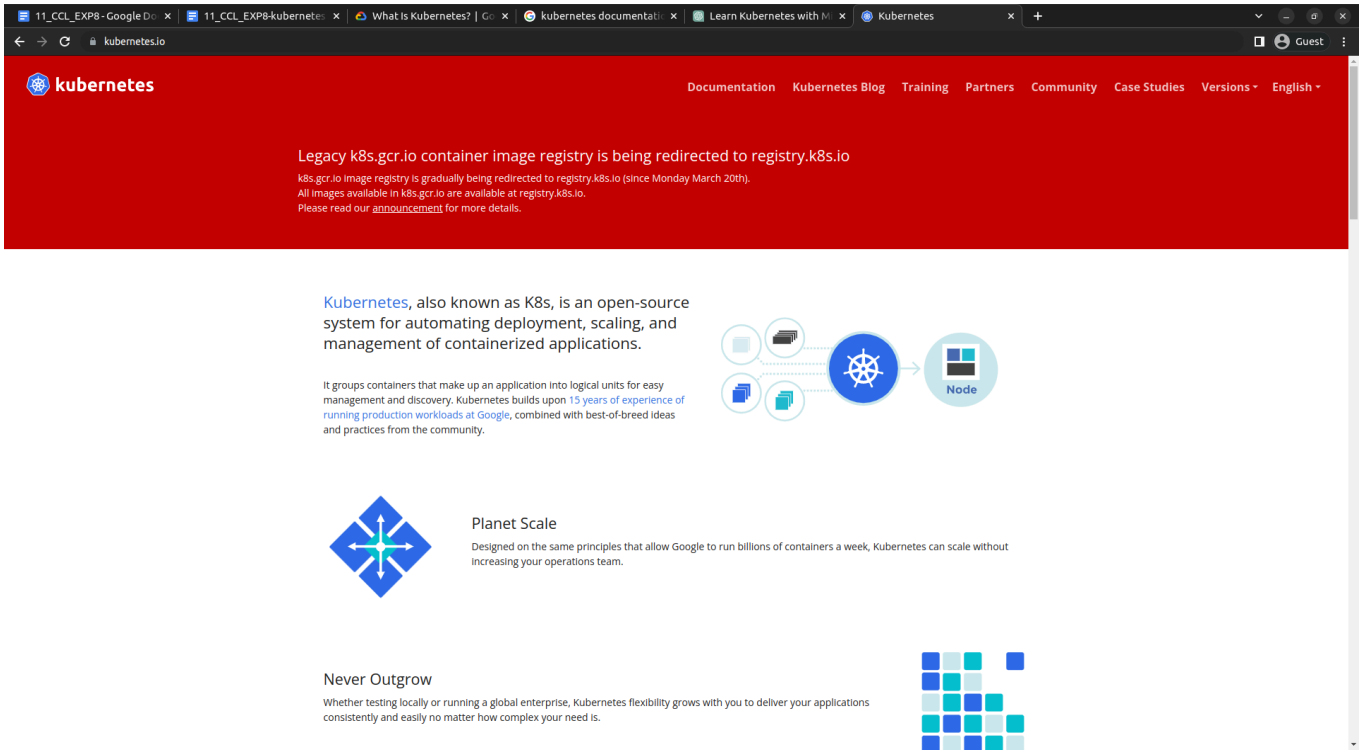
- Simplified management: Kubernetes provides a declarative API for defining the desired state of an application, and automatically reconciles the actual state of the application with the desired state. This simplifies management and reduces the risk of configuration errors.

- Ecosystem support: Kubernetes has a large and growing ecosystem of tools and add-ons, such as Helm charts and Operators, which help automate the deployment and management of complex applications on Kubernetes.

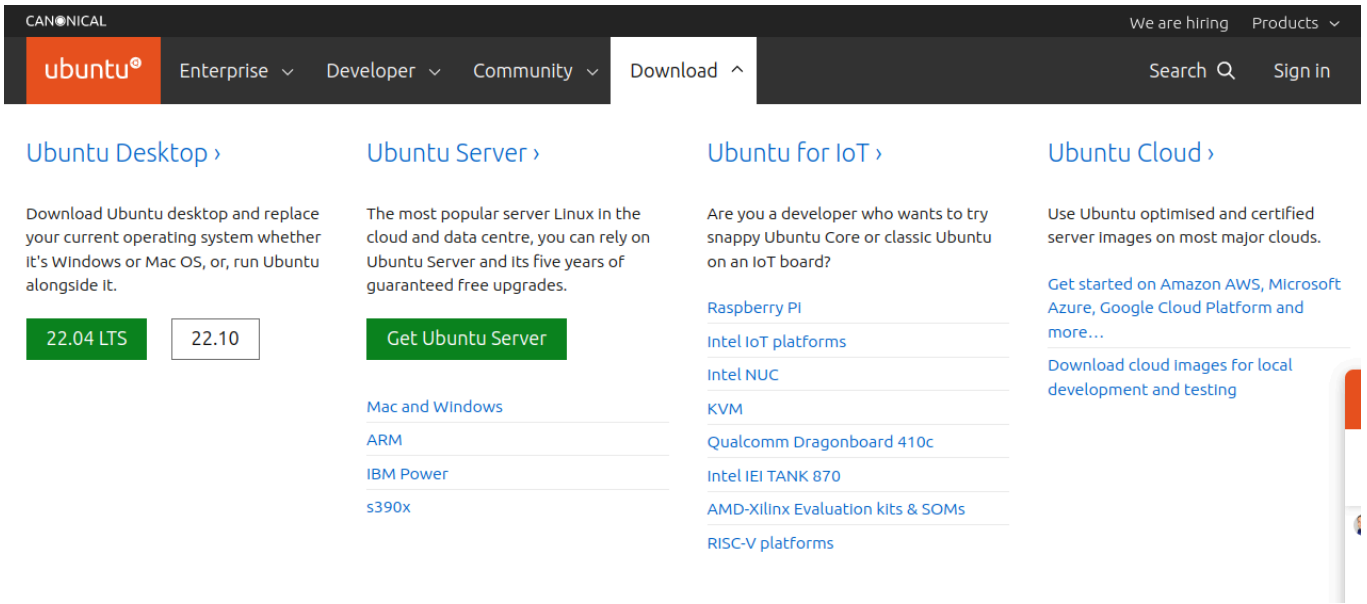
Overall, Kubernetes provides a powerful platform for managing containerized applications, with features that help simplify management, improve scalability and availability, and optimize resource utilization.

Kubernetes is a powerful tool for managing containerized applications, with a rich set of features that help automate the deployment, scaling, and management of applications at scale. While Kubernetes can be complex to set up and manage, it has become a de facto standard for container orchestration in the cloud-native ecosystem, and is widely used by organizations of all sizes.

Official website : https://kubernetes.io/



For installation in ubuntu visit : https://ubuntu.com/kubernetes/install#download



Here are some steps you can take to get started:

- Learn the basics of containers: Before diving into Kubernetes, it's important to understand the basics of containers. You can start by reading about Docker, which is one of the most popular containerization tools.

- Set up a Kubernetes cluster: You can set up a Kubernetes cluster locally using tools like Minikube or Kind, or you can use a cloud provider like Google Cloud, Amazon Web Services, or Microsoft Azure to create a Kubernetes cluster.

- Learn the Kubernetes architecture: Kubernetes has a complex architecture, so it's important to understand how all the components fit together. The Kubernetes documentation is a good place to start.

- Practice deploying applications: Once you have a Kubernetes cluster set up, you can practice deploying applications using Kubernetes. Start with a simple application and gradually work your way up to more complex deployments.

- Learn about Kubernetes networking: Kubernetes has its own networking model, so it's important to understand how it works. You can start by reading about Kubernetes Services and Ingress.

- Learn about Kubernetes storage: Kubernetes also has its own storage model, so it's important to understand how to manage storage for your applications. You can start by reading about Kubernetes Volumes.

- Explore Kubernetes ecosystem: Kubernetes has a large ecosystem of tools and add-ons that can help you manage and extend your cluster. Some popular tools include Helm, Prometheus, and Istio.

Overall, learning Kubernetes takes time and practice, but it can be a very rewarding skill to have in today's world of cloud-native applications.

Container orchestration using Kubernetes involves managing a cluster of nodes (physical or virtual machines) that run containerized applications using Kubernetes.

Kubernetes is a popular open-source container orchestration tool that has become a de facto standard for managing containerized applications in cloud computing environments. Kubernetes provides a powerful platform for automating the deployment, scaling, and management of containerized applications, and can be used with a wide range of cloud computing services and platforms.

In cloud computing environments, Kubernetes can be used to manage containerized applications deployed on a range of cloud platforms, including public clouds like Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, as well as private clouds and hybrid clouds. Kubernetes can also be used to manage containers on on-premises data centers.

Applications

Kubernetes provides a number of key features that help automate the management of containerized applications:

- Container orchestration: Kubernetes can manage containers at scale, orchestrating the deployment, scaling, and management of containerized applications across multiple nodes in a cluster.

- Service discovery and load balancing: Kubernetes provides a built-in service discovery and load balancing mechanism, enabling containers to find and communicate with other containers in the same application.

- Automatic scaling: Kubernetes can automatically scale containers based on metrics like CPU and memory usage, enabling applications to handle traffic spikes and sudden increases in demand.

- Self-healing: Kubernetes can automatically restart containers that fail, and can detect and replace nodes that become unavailable.

- Rollouts and rollbacks: Kubernetes enables controlled rollouts and rollbacks of containerized applications, allowing organizations to test new versions of applications before deploying them to production.

- Configuration management: Kubernetes provides a declarative API for defining the desired state of an application, and can automatically reconcile the actual state of the application with the desired state. This helps simplify configuration management and reduces the risk of configuration errors.

Conclusion : We had successfully studied container orchestration using Kubernetes