KIT- KALAIGNAR KARUNANIDHI INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
(Accredited by NAAC & NBA with 'A' Grade)
Kannampalayam Post, Coimbatore -641 402

Department of Computer Science and Business Systems

B19CBP702 – DEVOPS -CLOUD COMPUTING LABORATORY

Name	
Batch	Reg. No. ....
Branch	Year ....

KIT-Kalaignarkarunanidhi Institute of Technology

(An Autonomous Institution, Approved by AICTE & Affiliated to Anna University, Chennai)

Coimbatore – 641 402

Department of

……………………………………………………………………………………………

Record Work of

………………………………………………………………………………...Laboratory

Certified that this record is the bona fide work done by Name:

…………………………………………………………………………………………

Class:…………………………………..   Roll No:…………………………………

Branch………………………………………………………………………………………..

Place: KIT, CBE   Faculty In-Charge   HOD

Date:

University Register No……………………………………………

Submitted for the University Practical Examination held on

…………………………………………………

Internal Examiner   External Examiner

Instructions for Laboratory Classes

1. Enter the lab with record workbook & necessary things.
2. Enter the lab without bags and footwear.
3. Footwear should be kept in the outside shoe rack neatly.
4. Maintain silence during the Lab Hours.
5. Read and follow the work instructions inside the laboratory.
6. Handle the computer systems with care.
7. Shutdown the Computer properly and arrange chairs in order before leaving the lab.
8. The program should be written on the left side pages of the record work book.
9. The record workbook should be completed in all aspects and submitted in the next class itself.
10. Experiment number with date should be written at the top left-hand corner of the record work book page.
11. Strictly follow the uniform dress code for Laboratory classes.
12. Maintain punctuality for lab classes.
13. Avoid eatables inside and maintain the cleanliness of the lab.

VISION

To transform learners into competent industry ready innovative computer science professionals with managerial people skills and social values to contribute to the society.

MISSION

- To induce and develop student ability to compete globally through excellence in education.
- To impart technical knowledge through innovative teaching, research, and consultancy.
- To create healthy environment for developing innovative ideas.
- To facilitate students' development through academic-industry interaction.
- To enhance quality professionals for development of the socio-economic structure.

PROGRAMME OUTCOMES (POs)

Students graduating from Computer Science and Business Systems should be able to:

PO1 Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of Computer Science and Business Systems problems.

PO2 Problem Analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and Computer Science and Business Systems.

PO3 Design/Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations in the field of Computer Science and Business Systems.

PO4 Conduct Investigations of Complex Problems: Using research-based knowledge and Computer Science and Business Systems oriented research methodologies including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex Computer Science and Business Systems Engineering activities with an understanding of the limitations.

PO6 The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7 Environment and Sustainability: Understand the impact of the professional Computer Science and Business Systems Engineering solutions in societal and environmental contexts, and demonstrate the knowledge, and need for the sustainable development.

PO8 Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

PO10 Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 Project Management and Finance: Demonstrate knowledge and understanding of the Computer Science and Business Systems engineering and management principles and apply these to one's own work, as a member and leader in a team and, to manage projects in multidisciplinary environments.

PO12 Life-long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: Graduates will perform well in their professional career by acquiring enough knowledge in mathematics, computer science and business systems to concord the industry engrossment.

PEO 2: Graduates will improve communication skills, business management skills, as well as educate on the service orientation principles for various business disciplines.

PEO 3: Graduates will demonstrate ethical and moral values, also involve in team work in their profession.

PROGRAM SPECIFIC OUTCOME (PSOs)

Graduates of Computer Science and Business Systems Programmed should be able to:

PSO1: Ability to solve state-of-the-art problems in Computer Science by applying management principles according to the environmental needs.

PSO2: Ability to create an industry demand professional in the evolving discipline of Computer Science and Business Systems to provide solutions to complex engineering and business-related problems.

COURSE OUTCOMES

At the end of this course, the student will be able to:

Course Outcomes	**Knowledge Level**
CO1: Understand about AWS account creation and setup.	K2
CO2: Apply the concept of Cloud on DevOps using cloud computing models.	K3
CO3: Demonstrate about DevOps Development Tools.	K3
CO4: Interpret various operational Tools on DevOps.	K3
CO5: Illustrate about Cloud Computing Services.	K3

CO/PO & PSO		**PO1 (K3)**	**PO2 (K4)**	**PO3 (K5)**	**PO4 (K5)**	**PO5 (K6)**	**PO6 (K3) (A3)**	**PO7 (K2) (A3)**	**PO8 (K3) (A3)**	**PO9 (A3)**	**PO10 (A3)**	**PO11 (K3) (A3)**	**PO12 (A3)**	**PSO1 (K3 A3)**	**PSO2 (K3 A3)**
CO1	K2	2	2	2	1	1	-	-	-	-	-	-	3	-	2
CO2	K3	2	3	3	3	3	-	-	-	-	-	-	3	-	2
CO3	K3	3	3	3	3	3	-	-	-	-	-	-	3	-	2
CO4	K3	3	1	3	2	2	-	-	-	-	-	-	3	-	3
CO5	K3	3	1	2	1	1	-	-	-	-	-	-	3	-	3
Weighted average		3	3	3	3	2	-	-	-	-	-	-	3	-	2

SYLLABUS

LIST OF EXPERIMENTS

1 AWS- Create and setup AWS account

2 AWS-Create a VPC and subnets.

3 AWS-Create EC2 instance & volume.

4 AWS-Create a S3 bucket.

5 AWS-Create a RDS instance with private access only.

6 AWS-Deploy a server with EC2 instance, 83 bucket and RDS instances.

7 Configure Jenkins with git hub and deploy it on a server/computer.

8 Create a configuration with Jenkins with git push auto deploy functionality.

9 Create a configuration with Jenkins with manual deploy functionality.

10 Run MySQL in cocker and configure query login to host disk (system memory rather than docker memory) and check the health of this container every 10 seconds.

11 Run nginx in docker and configure any web location/url to redirect to an url and limit only 5 connection per user each minute.

TOTAL PRACTICAL HOURS 45

CONTENT

Practical Record Book Index Page

Sl. No.	Date	Name of the Experiment	Page Number	Aim & Algorithm (20 Marks)	Program (25 Marks)	Output & Inference (10 Marks)	Viva-Voce(20 Marks)	Total (75Marks)	Signature of the Faculty Member

Model Exam Marks (25):________Total (100):________ Signature of the Faculty Member

Practical Record Book Index Page

Sl. No.	Date	Name of the Experiment	Page Number	Aim &Algorithm (20 Marks)	Program (25 Marks)	Output & Inference (10 Marks)	Viva-Voce (20 Marks)	Total (75 Marks)	Signature of the Faculty Member

Model Exam Marks (25):________Total (100):________ **Signature of the Faculty Member**

Ex.No.	1	AWS- Create and setup AWS account
Date		

AIM:

To Create and setup AWS account.

ALGORITHM:

Step 1: Create an AWS Account

- Go to the AWS website (https://aws.amazon.com/)
- Click "Create an AWS Account" at the top-right corner
- Enter email address, password, and account name
- Click "Continue"

Step 2: Contact Information

- Choose whether you're creating a Personal or Professional account
- Enter full name, address, phone number, and organization (if applicable)
- Agree to the AWS Customer Agreement by checking the box
- Click "Continue"

Step 3: Payment Information

- Enter credit/debit card details (required for free-tier services)
- Provide CVV for verification (if required)
- Click "Verify and Add"

Step 4: Identity Verification

- Enter phone number and choose text message or voice call for verification
- Enter the verification code received

Step 5: Choose a Support Plan

- Select between Basic (free), Developer, or Business support plans
- Choose Basic if you don't need paid support services

Step 6: Sign in to the AWS Console

- After completing registration, receive an email confirmation
- Sign in to the AWS Management Console using root account email and password

Step 7: Secure Your Root Account

- Set up Multi-Factor Authentication (MFA) for added security
- Configure MFA using a mobile authenticator app (e.g., Google Authenticator)

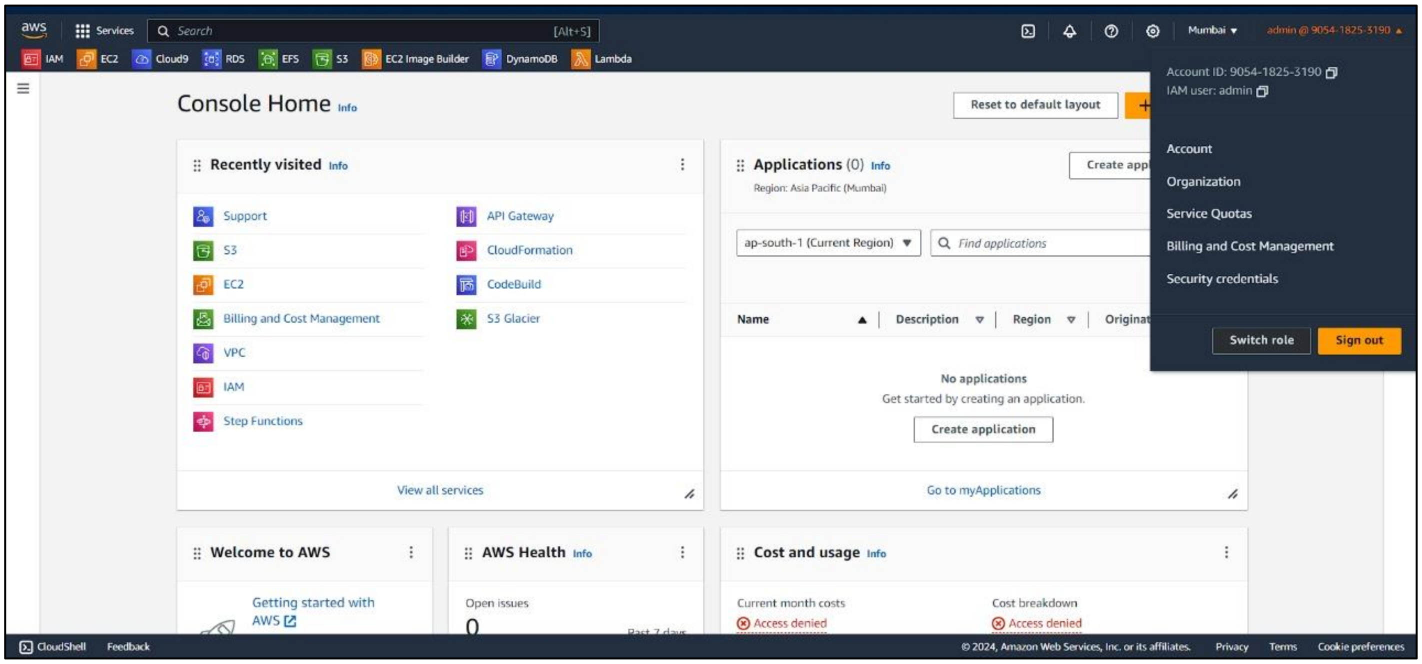
Step 8: Create an IAM User for Day-to-Day Use

- Create an IAM user for daily activities (instead of using the root account)
- Assign a group with appropriate permissions for the user
- Complete setup and download credentials

Step 9: Set Billing Alerts (Optional)

- Set up billing alerts to monitor AWS costs
- Enable "Receive Billing Alerts" under "Billing Preferences" and configure alerts

OUTPUT:



RESULT:

Thus the AWS Account created successfully.

Ex.No.	2	AWS-Create a VPC and subnets
Date		

AIM:

To Create a VPC and subnets.

ALGORITHM:

Step 1: Create a VPC

- Navigate to the VPC service in the AWS Management Console
- Click on "Create VPC" and configure the settings:
- Name tag: myVPC
- IPv4 CIDR block: 10.0.0.0/16
- Click "Create VPC" and then edit DNS hostnames to enable DNS support

Step 2: Create Subnets

- Go back to the VPC dashboard and click on "Subnets"
- Click on "Create Subnet" and configure the settings:
- Subnet Name: myPublicSN
- Availability Zone: ap-south-1a (or any region you prefer)
- IPv4 CIDR block: 10.0.0.0/24
- Repeat the process to create a private subnet:
- Subnet Name: myPrivateSN
- Availability Zone: ap-south-1b (or any region you prefer)
- IPv4 CIDR block: 10.0.1.0/24

Step 3: Create an Internet Gateway

- Go to the Internet Gateways section and click on "Create Internet Gateway"
- Configure the settings:
- Name tag: myIGW
- Attach the internet gateway to the VPC

Step 4: Create Route Tables

- Go to the Route Tables section and click on "Create route table"
- Configure the settings:
- Name Tag: PublicRT
- VPC: myVPC
- Repeat the process to create a private route table:
- Name Tag: PrivateRT
- VPC: myVPC
- Associate the route tables with the subnets and configure the routes

Step 5: Create Security Groups

- Go to the Security Groups section and click on "Create Security Group"
- Configure the settings:
- Security Group Name: MyWebServerSG
- Description: Security Group for EC2 Web Server in custom VPC
- VPC: myVPC
- Add inbound rules for HTTP, SSH, and other protocols as needed
- Repeat the process to create a security group for the database:
- Security Group Name: MyDatabaseSG
- Description: Security Group for RDS Database in custom VPC
- VPC: myVPC

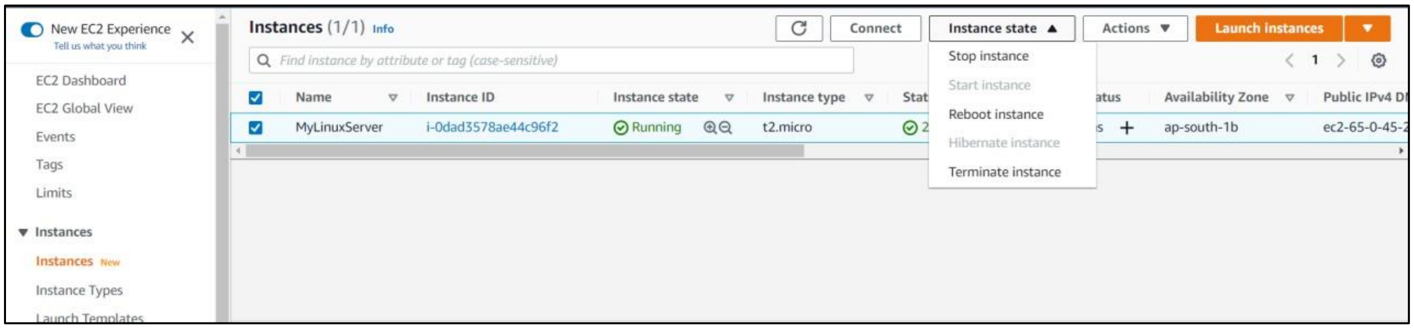
Step 6: Create a NAT Gateway

- Go to the NAT Gateways section and click on "Create NAT Gateway"
- Configure the settings:
- Name: myNATGW
- Subnet: myPublicSN
- Elastic IP allocation ID: Select Allocate Elastic IP

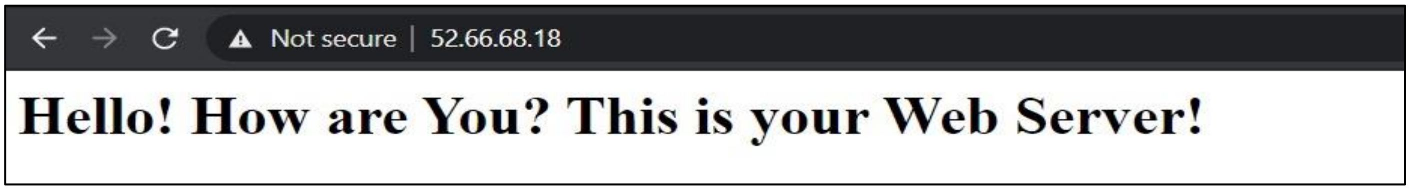
Step 7: Launch EC2 Instance

- Go to the EC2 service and click on "Launch Instance"
- Select the instance type and configure the settings:
- Network: myVPC
- Subnet: myPublicSN
- User data: paste the script to install and configure the web server
- Add tags and select the security group
- Launch the instance and test the web server by accessing the public IPv4 address in a browser

OUTPUT:



Name	Instance ID	Instance state	Instance type	Availability Zone	Public IPv4 DNS
MyLinuxServer	i-0dad3578ae44c96f2	Running	t2.micro	ap-south-1b	ec2-65-0-45-2



Hello! How are You? This is your Web Server!

RESULT:

Thus the VPC and subnets create successfully.

Ex.No.	**3**	**AWS-Create EC2 instance & volume.**
Date		

AIM:

To Create a EC2 instance & volume.

ALGORITHM:

Step 1: Log in to AWS Console and Navigate to EC2

- Log in to your AWS account.
- In the AWS Management Console, search for EC2 and click on EC2 services.
- On the EC2 dashboard, click on Launch Instance.

Step 2: Name and Tags

- In the Name and tags section, type windows-server in the text field labeled Name.
- Step 3: Select Microsoft Windows Server AMI
- Under the Quick Start section, select Windows to choose a Microsoft Windows Server AMI.

Step 4: Choose Instance Type

- In the Instance Type section, select t2.micro.

Step 5: Set Up Key Pair (Login)

- In the Key pair (login) section, click on the dropdown to select an existing Key Pair.
- If you don’t have an existing key pair, follow these steps:
- Click on Create a new key pair.
- Type a Key Pair Name (e.g., MyKeyPair).
- Click on Create key pair. This will download a .pem file.
- Save the .pem file securely for future use.
- Click Launch Instance to proceed.

Step 6: Configure Network Settings

- In the Network settings section, click Edit to modify the following:
- Set network subnet: Select a Subnet from the dropdown menu.
- Select security group: If you don't have an existing security group, follow these steps:
- Select Create security group.
- Type Security Group Name (e.g., Windows-SG).
- Type a Description (e.g., Security Group for Windows Server).
- Keep the default RDP (Remote Desktop Protocol) rule as is.

Step 7: Launch the Instance

- Review the settings in the Summary section on the right.
- Click Launch instance to launch your Windows Server instance.

Step 8: Check Instance Status and Connect

- After launching, click Instances at the top left of the page to go to the Instances dashboard.
- Ensure your Windows instance is in the running state.
- Select the instance checkbox and click Connect.
- Switch to the RDP Client tab and click Download Remote Desktop File.

Step 9: Retrieve and Decrypt Password

- Click on Get Password.
- In the Get Password window, upload the private key file (.pem) that was downloaded earlier.
- Click on Decrypt Password. AWS will provide the password needed for logging into the Windows instance.
- Copy the decrypted password and save it for further use.

Step 10: Open Remote Desktop File

- Navigate to the location where your Remote Desktop File was downloaded.
  Double-click the RDP file to open it.

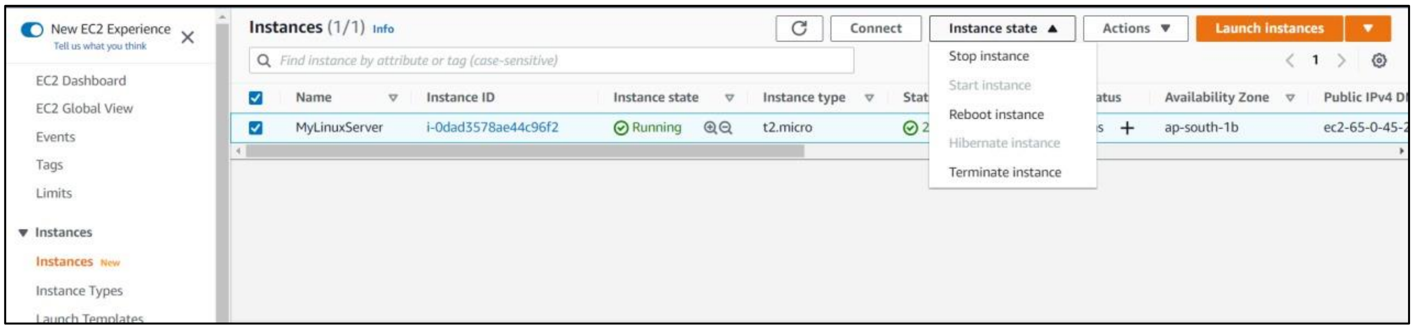
Step 11: Connect to the Instance

- In the pop-up window, click on Connect.
- Enter the password you saved in Step 9.
- Follow the on-screen instructions to log in.

Step 12: Instance Management

- You have successfully logged in and started operating your Windows instance.

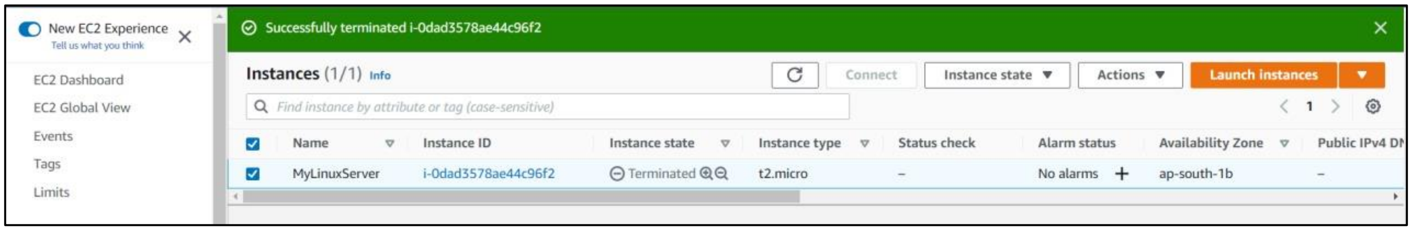
OUTPUT:



Instances (1/1) Info

Find instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Stat	atus	Availability Zone	Public IPv4 D
MyLinuxServer	i-0dad3578ae44c96f2	Running	t2.micro	2	s	ap-south-1b	ec2-65-0-45-2

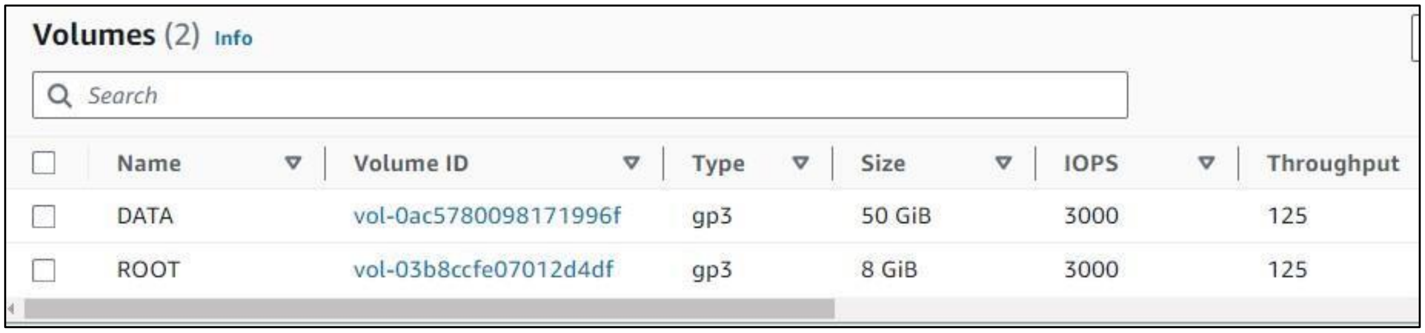


Successfully terminated i-0dad3578ae44c96f2

Instances (1/1) Info

Find instance by attribute or tag (case-sensitive)

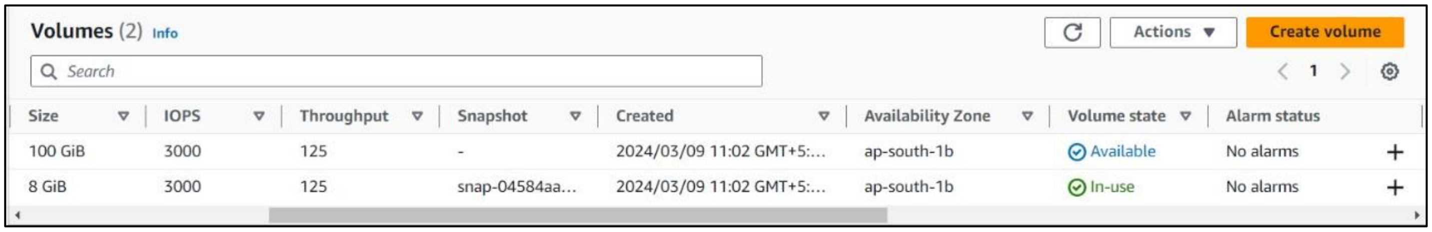
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
MyLinuxServer	i-0dad3578ae44c96f2	Terminated	t2.micro	–	No alarms	ap-south-1b	–



Volumes (2) Info

Search

Name	Volume ID	Type	Size	IOPS	Throughput
DATA	vol-0ac5780098171996f	gp3	50 GiB	3000	125
ROOT	vol-03b8ccfe07012d4df	gp3	8 GiB	3000	125



Volumes (2) Info

Search

Size	IOPS	Throughput	Snapshot	Created	Availability Zone	Volume state	Alarm status
100 GiB	3000	125	-	2024/03/09 11:02 GMT+5:...	ap-south-1b	Available	No alarms
8 GiB	3000	125	snap-04584aa...	2024/03/09 11:02 GMT+5:...	ap-south-1b	In-use	No alarms

RESULT:

Thus the EC2 instance & volume created successfully.

Ex. No.	**4**	**AWS-CREATE A S3 BUCKET.**
Date		

AIM:

To Create a S3 bucket.

ALGORITHM:

1. Sign in to the AWS Management Console

- Go to AWS Console, and log in using your credentials.

2. Navigate to the S3 Service

- In the AWS Management Console, type “S3” in the search bar located at the top of the page.
- Click on the *S3* service from the search results to open the Amazon S3 dashboard.

3. Create a Bucket

- On the S3 dashboard, click on the orange *Create bucket* button.

4. Name the Bucket

- In the "Bucket name" field, enter a globally unique name for your bucket. The name can include letters, numbers, periods, and hyphens, but it must not contain spaces and should be DNS-compliant (no uppercase letters or underscores).

5. Choose an AWS Region

- Under the "AWS Region" section, select the region where you want to create your bucket. The region you choose determines where your bucket will be physically stored, and it can impact latency and costs.

6. Configure Bucket Settings (Optional)

- *Block Public Access:* This is enabled by default for security. If you want your bucket to be accessible to the public (not recommended for sensitive data), uncheck the "Block all public access" option.
- *Bucket Versioning:* If you want to keep multiple versions of an object in your bucket, enable versioning.
- *Encryption:* You can choose to enable default encryption for all objects uploaded to the bucket.
- *Tags:* You can add tags to help manage and categorize the bucket (optional).
- *Object Lock:* If needed, you can enable Object Lock for data protection and retention.

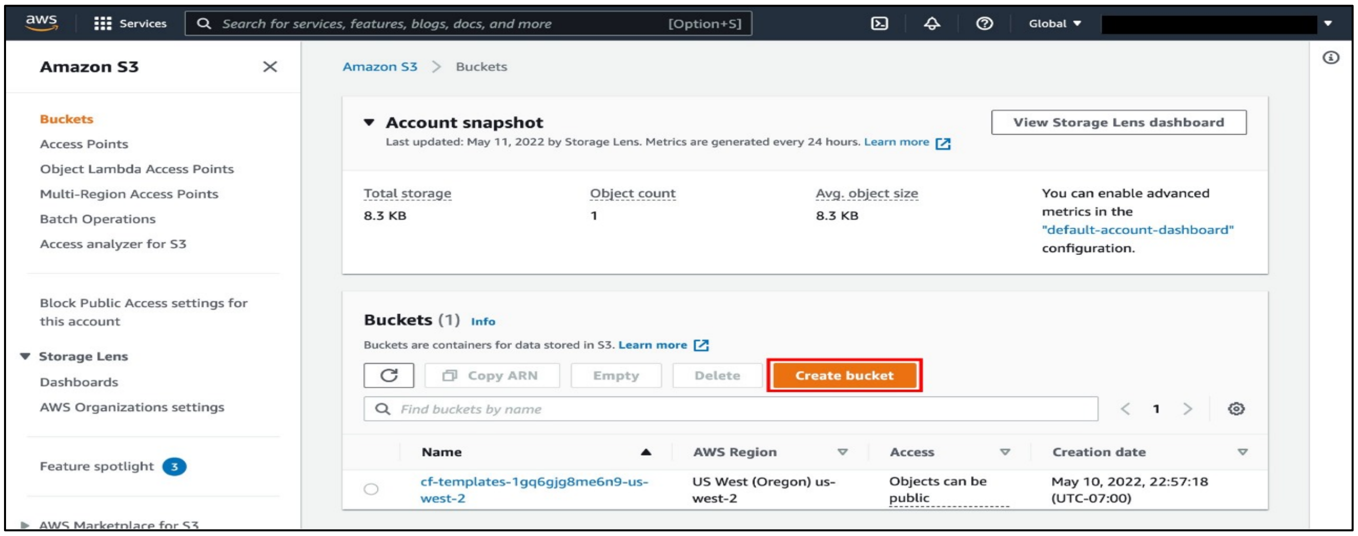
7. Review the Bucket Settings

- Review all the settings to ensure they meet your requirements.

8. Create the Bucket

- Click on the *Create bucket* button at the bottom of the page to finalize the process.

OUTPUT:


aws
Services
Search for services, features, blogs, docs, and more [Option+S]
Global
[redacted]
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight 3
AWS Marketplace for S3
Amazon S3 > Buckets

Account snapshot

Last updated: May 11, 2022 by Storage Lens. Metrics are generated every 24 hours. Learn more

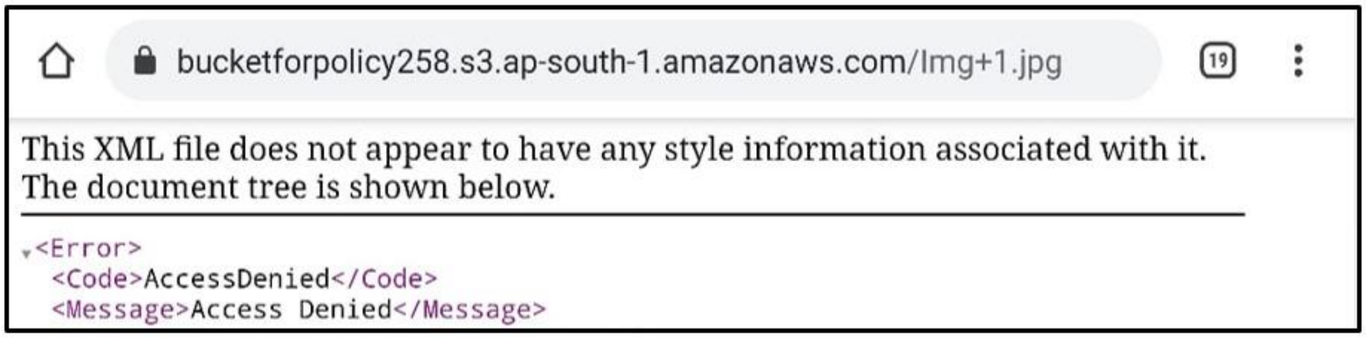
View Storage Lens dashboard

Total storage	Object count	Avg. object size	You can enable advanced metrics in the "default-account-dashboard" configuration.
8.3 KB	1	8.3 KB	

Buckets (1) Info

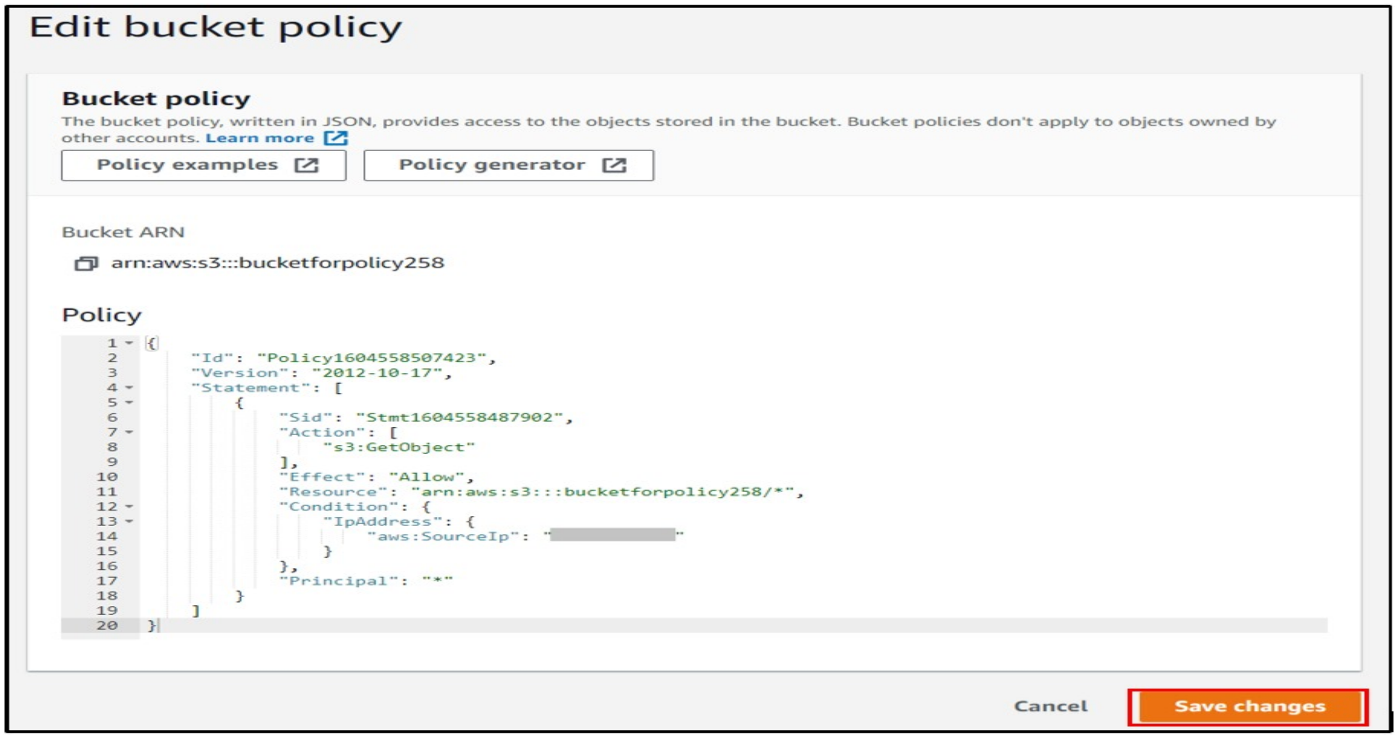
Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
cf-templates-1gq6gjg8me6n9-us-west-2	US West (Oregon) us-west-2	Objects can be public	May 10, 2022, 22:57:18 (UTC-07:00)



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
```



Edit bucket policy

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

Bucket ARN

arn:aws:s3:::bucketforpolicy258

Policy

```
{
    "Id": "Policy1604558507423",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1604558487902",
            "Action": [
                "s3:GetObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::bucketforpolicy258/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "[redacted]"
                }
            },
            "Principal": "*"
        }
    ]
}
```

RESULT:

Thus, the S3 bucket create successfully.

Ex.No.	5	AWS-CREATE A RDS INSTANCE WITH PRIVATE ACCESS ONLY.
Date		

AIM:

To create a rds instance with private access only.

ALGORITHM:

Step 1: Log in to AWS Console and Navigate to RDS

- Log in to your AWS account.
- In the AWS Management Console, search for RDS and click on RDS service.
- On the RDS dashboard, click Create Database.

Step 2: Choose a Database Creation Method

- Select Standard Create under the Choose a database creation method section.

Step 3: Select Database Engine

- Choose your preferred database engine (e.g., Amazon Aurora, MySQL, PostgreSQL, etc.).
- Choose the version of the engine as per your requirement.

Step 4: Configure Settings

- In the Settings section, provide:
- DB instance identifier: Provide a name for your database (e.g., myPrivateRDS).
- Master username: Choose a username.
- Master password: Set a strong password or let AWS generate one for you.

Step 5: Choose an Instance Class

- In the DB instance class section, choose the instance size (e.g., db.t3.micro for a small setup).
- For Storage, choose the type and size as per your requirements.

Step 6: Configure Connectivity for Private Access

- In the Connectivity section, set the following:
- Virtual Private Cloud (VPC): Choose the VPC where you want the RDS to reside. Ensure it is not the default VPC.
- Subnet group: Choose an existing DB subnet group or create a new one with only private subnets.
- Public access: Select No to restrict public access to your RDS instance.
- VPC security group(s): Either select an existing security group or create a new one that only allows access from your application (e.g., EC2 in the private subnet).

Step 7: Configure Additional Settings

- In the Additional configuration section:
- Initial database name: Provide a name if you want to create a database when the instance is launched.
- Backup retention period: Choose how long backups should be retained.
- Enable automatic backups: Keep enabled for data recovery purposes.

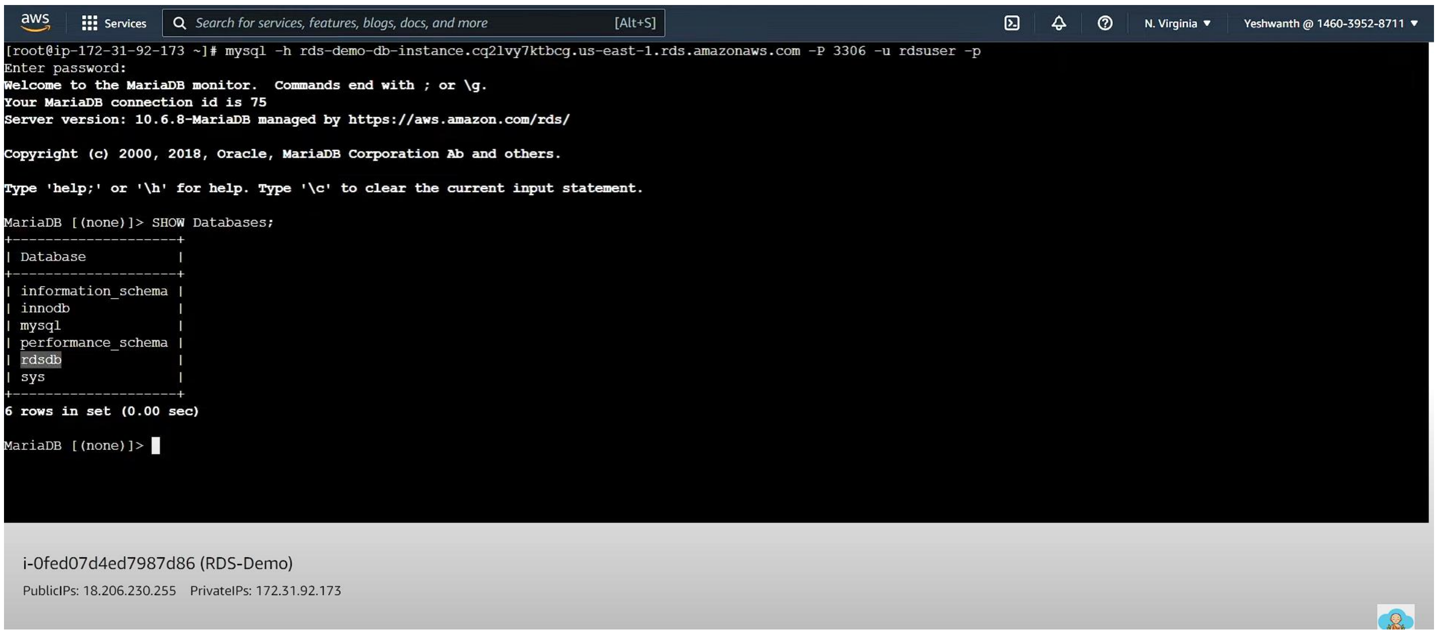
Step 8: Finalize and Create

- Review your configuration in the Summary section.
- Click Create Database.

Step 9: Verify the RDS Instance

- Once the RDS instance is created, go back to the RDS Dashboard.
- Verify the Status of your database instance (it will show "available" once ready).
- Ensure that the RDS instance has no public IP address and is accessible only within the private VPC subnets.

OUTPUT:


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
Yeshwanth @ 1460-3952-8711

```
[root@ip-172-31-92-173 ~]# mysql -h rds-demo-db-instance.cq2lvy7ktbcg.us-east-1.rds.amazonaws.com -P 3306 -u rdsuser -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 75
Server version: 10.6.8-MariaDB managed by https://aws.amazon.com/rds/

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SHOW Databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| innodb             |
| mysql              |
| performance_schema |
| rdsdb              |
| sys                |
+--------------------+
6 rows in set (0.00 sec)

MariaDB [(none)]>
```

i-0fed07d4ed7987d86 (RDS-Demo)

PublicIPs: 18.206.230.255 PrivateIPs: 172.31.92.173

RESULT:

Thus the RDS instance with private access only was created successfully.

Ex.No.	6	AWS-DEPLOY A SERVER WITH EC2 INSTANCE, s3 BUCKET AND RDS INSTANCES.
Date		

AIM:

To Deploy a server with EC2 instance, 83 bucket and RDS instances.

ALGORITHM:

Step 1: Install and Configure Jenkins

- Install Jenkins on your server or use an existing instance.
- Start Jenkins by accessing http://your-server-ip:8080/.
- Install any required plugins (e.g., **Git** and **SSH** plugins) through the Jenkins plugin manager.

Step 2: Set Up Git Repository

- Initialize a Git repository in your project directory or use an existing repository.
- Push your code to a remote Git platform (e.g., GitHub, GitLab, Bitbucket).
- Ensure your Jenkins server has access to your Git repository by generating and adding SSH keys or OAuth tokens to Jenkins.

Step 3: Create a Jenkins Job

- Go to the Jenkins dashboard and click New Item.
- Enter a name for your job (e.g., AutoDeployJob) and choose Freestyle project as the project type.
- Click OK to create the job.

Step 4: Configure Git Repository in Jenkins

- Under the Source Code Management section, select Git.
- In the **Repository URL**, provide the URL of your Git repository.
- Add credentials (if necessary) by selecting **Add** under **Credentials** and inputting the required access keys or tokens.
- Specify the branch to track (e.g., `main` or `master`).

Step 5: Add Build Trigger for Auto-Deploy

- Scroll down to the Build Triggers section.
- Select **GitHub hook trigger for GITScm polling** or **Poll SCM**.
- For **GitHub hook trigger**: Configure a webhook in your Git repository settings that points to your Jenkins server.
- For **Poll SCM**: Set a schedule (e.g., H/5 * * * * for polling every 5 minutes).

Step 6: Add Build Steps for Deployment

- Under the Build section, click Add build step and select Execute shell.
- In the shell script area, write the commands needed to deploy your application. For example, commands to stop the current application, pull the latest changes from Git, and restart the application:

```
cd /path/to/your/project
git pull origin main
    ./deploy.sh  # Custom deployment script
```

Step 7: Save and Test the Jenkins Job

- After setting up the build steps, click Save.
- Manually trigger the build by clicking **Build Now** on the job page to ensure it works correctly.

Step 8: Configure Git Webhook (for Auto Trigger)

- Go to your Git repository (GitHub, GitLab, etc.).
- Navigate to **Settings** -> **Webhooks** -> **Add Webhook**.
- Add your Jenkins server URL with /github-webhook/ at the end, e.g., http://your-server-ip:8080/github-webhook/
- Set the webhook to trigger on **Push events**.

Step 9: Verify Auto-Deploy Functionality

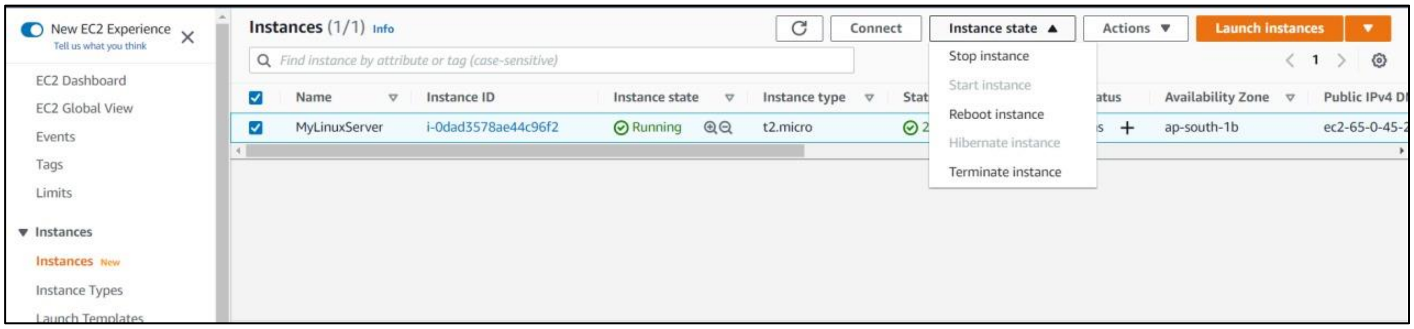
- Push a change to your Git repository.
- Jenkins should automatically trigger the job based on the Git webhook.
- Monitor the Jenkins console output to ensure that the deployment script runs successfully.

Step 10: Verify Successful Deployment

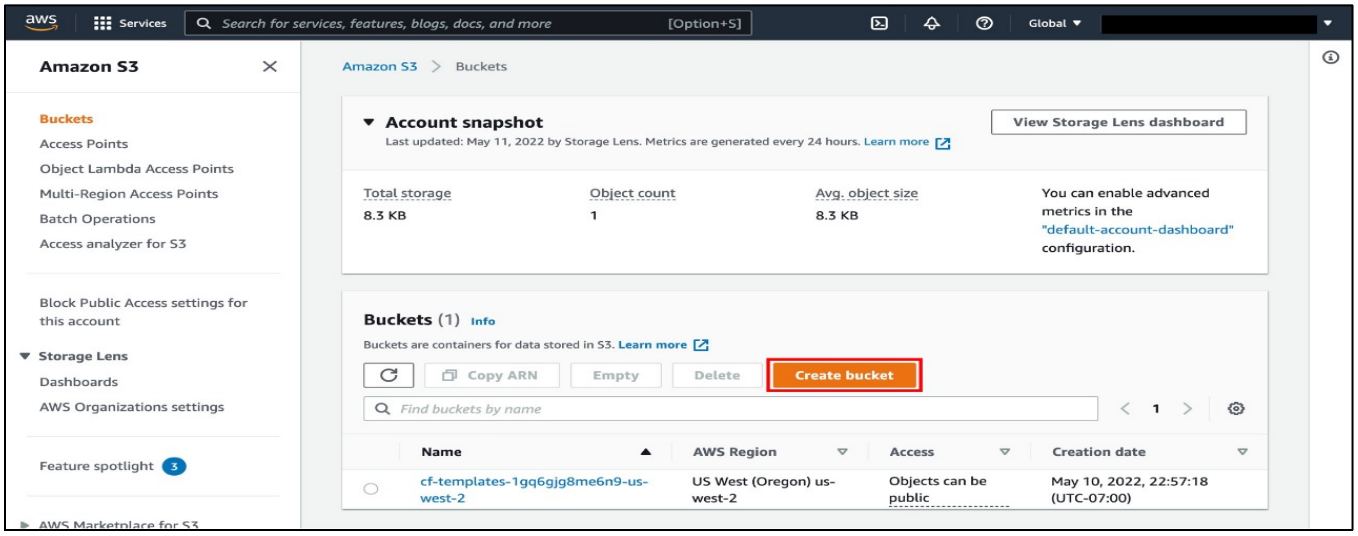
- Access your application to verify that the latest changes have been deployed**.**

Confirm that Jenkins has triggered the deployment correctly by reviewing the build logs**.**

OUTPUT:



Name	Instance ID	Instance state	Instance type	Stat…	…atus	Availability Zone	Public IPv4 D…
MyLinuxServer	i-0dad3578ae44c96f2	Running	t2.micro	2…		ap-south-1b	ec2-65-0-45-2…



[redacted]

Account snapshot

Last updated: May 11, 2022 by Storage Lens. Metrics are generated every 24 hours. Learn more

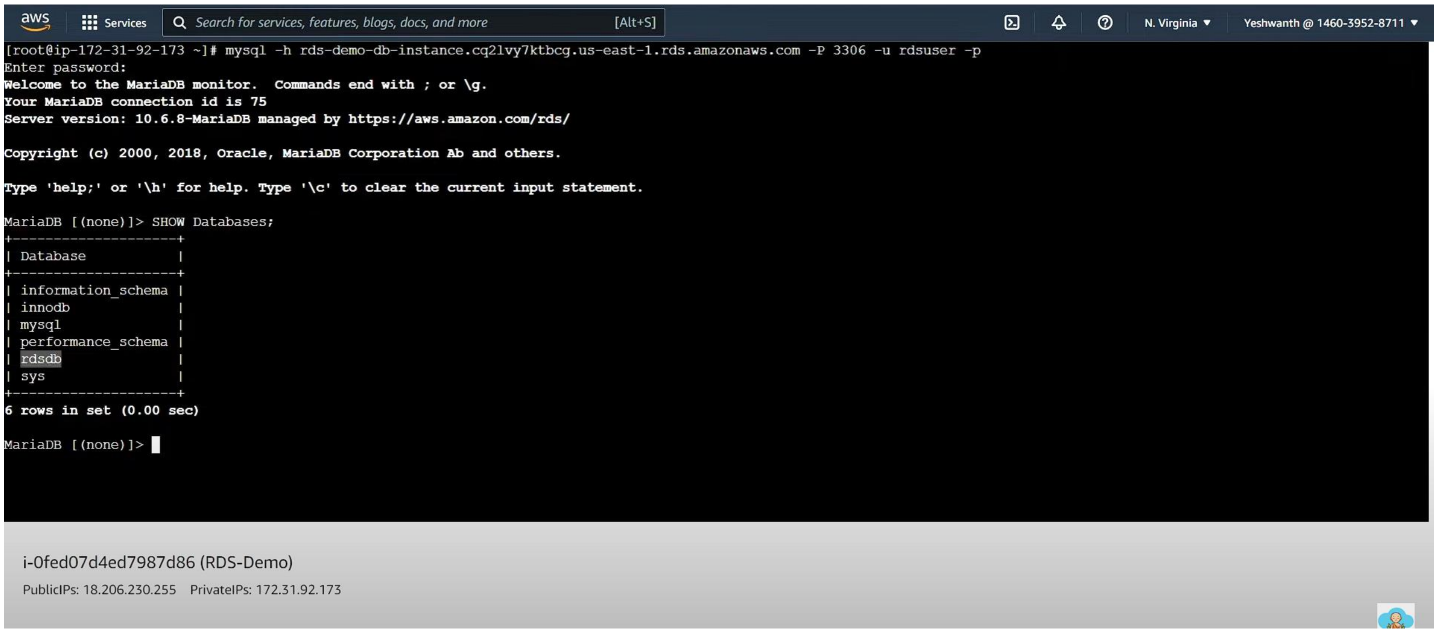
Total storage	Object count	Avg. object size
8.3 KB	1	8.3 KB

You can enable advanced metrics in the "default-account-dashboard" configuration.

Buckets (1) Info

Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
cf-templates-1gq6gjg8me6n9-us-west-2	US West (Oregon) us-west-2	Objects can be public	May 10, 2022, 22:57:18 (UTC-07:00)


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
Yeshwanth @ 1460-3952-8711

```
[root@ip-172-31-92-173 ~]# mysql -h rds-demo-db-instance.cq2lvy7ktbcg.us-east-1.rds.amazonaws.com -P 3306 -u rdsuser -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 75
Server version: 10.6.8-MariaDB managed by https://aws.amazon.com/rds/

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SHOW Databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| innodb             |
| mysql              |
| performance_schema |
| rdsdb              |
| sys                |
+--------------------+
6 rows in set (0.00 sec)

MariaDB [(none)]>
```

i-0fed07d4ed7987d86 (RDS-Demo)
PublicIPs: 18.206.230.255 PrivateIPs: 172.31.92.173

RESULT:

Thus, the Deploy a server with EC2 instance, 83 bucket and RDS instances was created successfully.

Ex.No.	7	CONFIGURE JENKINS WITH GIT HUB AND DEPLOY IT ON A SERVER/COMPUTER.
Date		

AIM:

To Configure Jenkins with git hub and deploy it on a server/computer.

ALGORITHM:

Step 1: Install Jenkins on Your Server

1. **Update your system:** Open a terminal and update the system's package list:
```
sudo apt update
sudo apt upgrade
```
2. **Install Java:** Jenkins requires Java to run, so install Java:
```
sudo apt install openjdk-11-jdk
```
3. **Add the Jenkins repository:**
```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
```
4. **Install Jenkins:** Update your package list and install Jenkins:
```
sudo apt update
sudo apt install jenkins
```
5. **Start Jenkins:**
```
sudo systemctl start jenkins
sudo systemctl enable jenkins
```
6. **Access Jenkins:**
   - Open a browser and go to http://<your-server-ip>:8080
   - Unlock Jenkins by entering the password located in the following file:
   - sudo cat /var/lib/jenkins/secrets/initialAdminPassword
   - Set up your admin user and complete the setup wizard.

Step 2: Install Git Plugin in Jenkins

1. **Install Git:**
```
sudo apt install git
```
2. **Install Git plugin in Jenkins:**
   - In Jenkins, go to Manage Jenkins > Manage Plugins > Available.
   - Search for **Git Plugin** and install it.
3. **Install GitHub plugin:** Similarly, search for **GitHub Integration Plugin** and install it.

Step 3: Configure Jenkins with GitHub

1. **Create a GitHub Personal Access Token:**
   - Go to GitHub > Profile > Settings > Developer Settings > Personal Access Tokens.

Generate a new token with repo and admin permissions.

2. **Add GitHub credentials to Jenkins:**
   - In Jenkins, go to Manage Jenkins > Manage Credentials > System > Global credentials > Add Credentials.
   - Choose **Username with password**, and use your GitHub username and the token generated earlier.

3. **Create a New Job in Jenkins:**
   - Go to Jenkins Dashboard, click **New Item**, and select **Freestyle project**.
   - Enter a project name and click **OK**.

4. **Configure GitHub Repository:**
   - In your project settings, go to **Source Code Management** and select **Git**.
   - Enter your GitHub repository URL.
   - Under **Credentials**, select the GitHub token credentials added earlier.

5. **Set up Build Triggers:**
   - Go to **Build Triggers** and select **GitHub hook trigger for GITScm polling**.
   - This will automatically trigger Jenkins builds when there are changes in the GitHub repository.

Step 4: Set Up Webhooks in GitHub

1. **Go to your GitHub repository** and navigate to **Settings > Webhooks**.
2. **Add a new Webhook:**
   - **Payload URL**: http://<your-server-ip>:8080/github-webhook/
   - **Content Type**: application/json
   - Check **Just the push event**.
3. **Save the webhook**.

Step 5: Configure Build Steps in Jenkins

1. In your Jenkins project, go to the **Build** section.
2. **Add build steps** as per your project needs. For example:
   - If you're using Maven, add **Invoke top-level Maven targets**.
   - If you're deploying to a server, you might use **Execute shell** to run deployment scripts.
2. Example shell script for deploying to a server:

```
git pull origin main
```

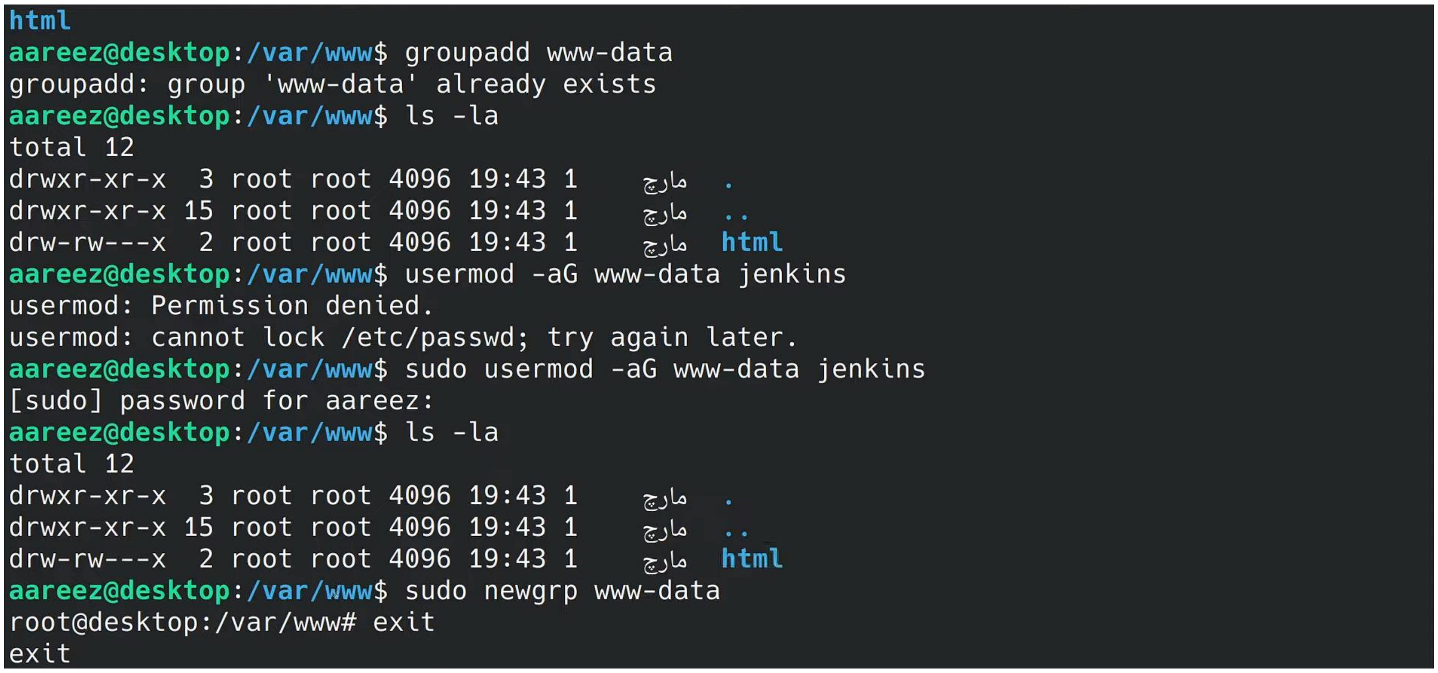
Step 6: Test the Setup

1. **Make a commit** to your GitHub repository.
2. Jenkins will trigger the build automatically based on the webhook.

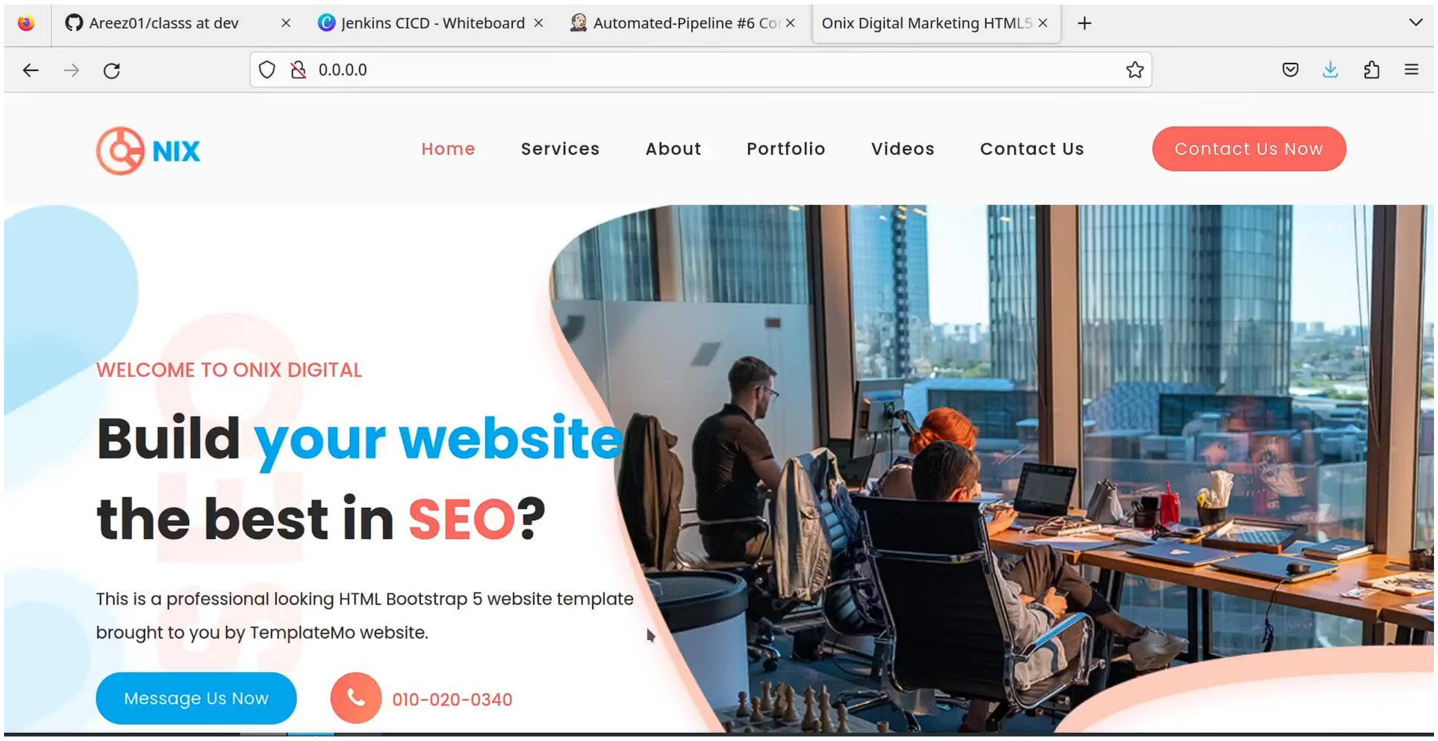
Step 7: Monitor Jenkins

1. You can view the build status in **Jenkins Dashboard > Build History**.
2. Jenkins will fetch the latest changes from GitHub and execute the build steps as configured.

OUTPUT:



```
html
aareez@desktop:/var/www$ groupadd www-data
groupadd: group 'www-data' already exists
aareez@desktop:/var/www$ ls -la
total 12
drwxr-xr-x  3 root root 4096 19:43 1    مارچ  .
drwxr-xr-x 15 root root 4096 19:43 1    مارچ  ..
drw-rw---x  2 root root 4096 19:43 1    مارچ  html
aareez@desktop:/var/www$ usermod -aG www-data jenkins
usermod: Permission denied.
usermod: cannot lock /etc/passwd; try again later.
aareez@desktop:/var/www$ sudo usermod -aG www-data jenkins
[sudo] password for aareez:
aareez@desktop:/var/www$ ls -la
total 12
drwxr-xr-x  3 root root 4096 19:43 1    مارچ  .
drwxr-xr-x 15 root root 4096 19:43 1    مارچ  ..
drw-rw---x  2 root root 4096 19:43 1    مارچ  html
aareez@desktop:/var/www$ sudo newgrp www-data
root@desktop:/var/www# exit
exit
```


Areez01/classs at dev
Jenkins CICD - Whiteboard
Automated-Pipeline #6 Co
Onix Digital Marketing HTML5
0.0.0.0
NIX
Home
Services
About
Portfolio
Videos
Contact Us
Contact Us Now
WELCOME TO ONIX DIGITAL
Build your website the best in SEO?
This is a professional looking HTML Bootstrap 5 website template brought to you by TemplateMo website.
Message Us Now
010-020-0340

RESULT:

Thus the Configure of Jenkins with git hub and deploy it on a server/computer was successfully completed.

Ex.No.	8	CREATE A CONFIGURATION WITH JENKINS WITH GIT PUSH AUTO DEPLOY FUNCTIONALITY.
Date		

AIM:

To Create a configuration with Jenkins with git push auto deploy functionality.

ALGORITHM:

1. **Install Required Plugins:**
    - Login to your Jenkins server.
    - Go to **Manage Jenkins** > **Manage Plugins** > **Available**.
    - Search and install the following plugins:
        - Git Plugin
        - GitHub Integration Plugin
        - SSH Plugin (Optional, for deploying to a remote server)
2. **Create a New Jenkins Job:**
    - Go to the Jenkins dashboard.
    - Click on **New Item**.
    - Name the job (e.g., AutoDeploy) and select **Freestyle project**.
    - Click **OK** to create the job.
3. **Configure Source Code Management:**
    - In the job configuration, navigate to **Source Code Management**.
    - Select **Git** and enter your Git repository URL.
    - Under **Credentials**, click **Add** and select **Jenkins > Username with password**.
    - Enter your GitHub username and personal access token (generated from GitHub settings).
    - Specify the branch you want to build and deploy from (e.g., main or master).
4. **Set Up Build Triggers (Git Push Trigger):**
    - In the job configuration, go to **Build Triggers**.
    - Select **GitHub hook trigger for GITScm polling**.
5. **Configure Webhooks in GitHub:**
    - Go to your GitHub repository.
    - Navigate to **Settings** > **Webhooks**.

- Click **Add webhook**.
- Set **Payload URL** to `http://<jenkins-server-ip>:8080/github-webhook/` (replace `<jenkins-server-ip>` with your server's IP).
- Set **Content Type** to `application/json`.
- Select trigger for **Just the push event**.
- Click **Save webhook**.

6. **Add Build Steps for Deployment:**
   - In the job configuration, go to **Build**.
   - Add a build step: **Execute Shell**.
   - Paste the following script, replacing the placeholders:

7. **`ssh user@remote-server "cd /path/to/deployment && git pull origin main && ./deploy.sh"`**
   - Replace:
     - `user@remote-server`: Your server's username and login credentials.
     - `/path/to/deployment`: The directory where your application resides on the server.
     - `./deploy.sh`: Your deployment script (optional, explained in step 8).

8. **Enable SSH Access (Optional):**
   - If deploying to a remote server, set up SSH key access for Jenkins:
     - Generate an SSH key pair on the Jenkins server: `ssh-keygen -t rsa -b 4096`
     - Copy the public key to your remote server using `ssh-copy-id user@remote-server`.
     - Configure SSH credentials in Jenkins: Go to **Manage Jenkins** > **Manage Credentials** and add your SSH private key credentials.

9. **Deployment Script (Optional):**

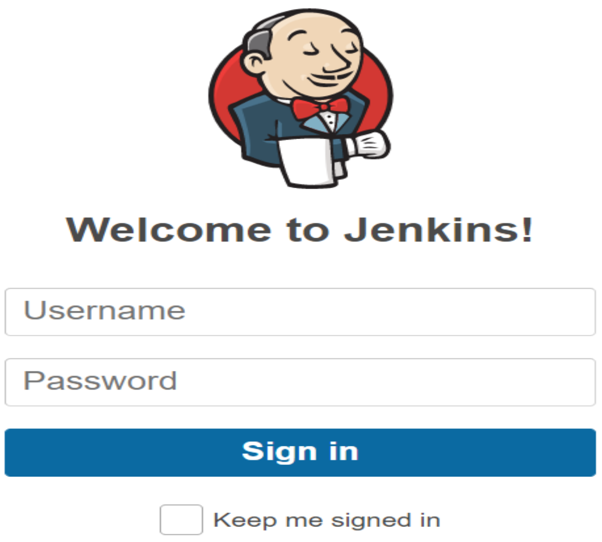
   For a more automated deployment, create a script (`deploy.sh`) on your server:

```
cd /path/to/your/project
git pull origin main
pm2 restart app-name
```

   Make the script executable using `chmod +x deploy.sh`.

10. **Test Auto-Deployment:**
    - Make a change in your codebase and push it to GitHub.
    - Jenkins will automatically trigger a build and deploy the updated code if everything is configured correctly.

OUTPUT:

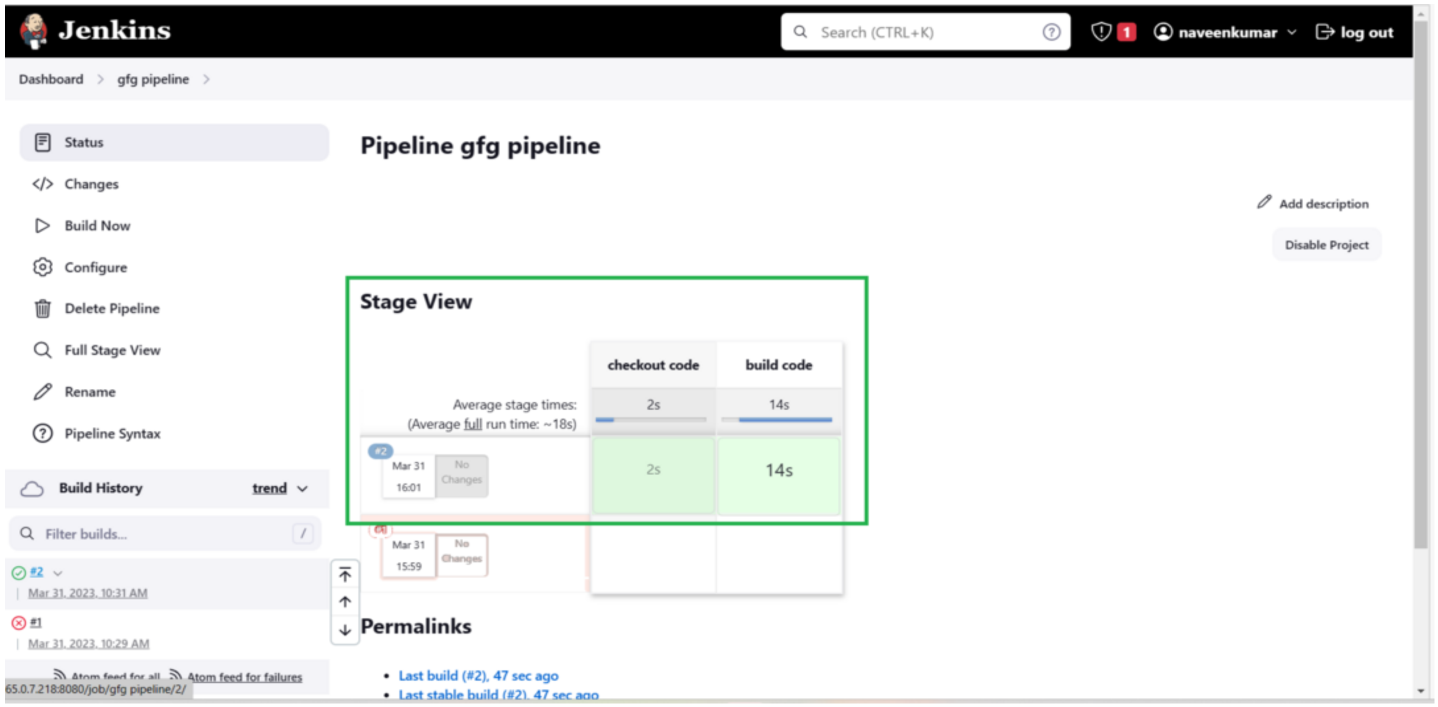

Welcome to Jenkins!

Username

Password

Sign in

Keep me signed in


Jenkins

Search (CTRL+K)

naveenkumar

log out

Dashboard > gfg pipeline >

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Build History trend

Filter builds...

#2 Mar 31, 2023, 10:31 AM

#1 Mar 31, 2023, 10:29 AM

Atom feed for all Atom feed for failures

65.0.7.218:8080/job/gfg pipeline/2/

Pipeline gfg pipeline

Add description

Disable Project

Stage View

	checkout code	build code
Average stage times: (Average full run time: ~18s)	2s	14s
#2 Mar 31 16:01 No Changes	2s	14s
#1 Mar 31 15:59 No Changes		

Permalinks

- Last build (#2), 47 sec ago
- Last stable build (#2), 47 sec ago

RESULT:

Thus the configuration with Jenkins with git push auto deploy functionality completed successfully.

Ex.No.	9	CREATE A CONFIGURATION WITH JENKINS WITH MANUAL DEPLOY FUNCTIONALITY.
Date		

AIM:

To Create a configuration with Jenkins with manual deploy functionality.

ALGORITHM:

1. Install Required Plugins:

- Log in to your Jenkins server.
- Go to **Manage Jenkins** > **Manage Plugins**.
- Search for and install the following plugins:
    - **Git Plugin**
    - **GitHub Integration Plugin** (if using GitHub)
    - **SSH Plugin** (if deploying to a remote server)

2. Create a New Jenkins Job:

- Go to the Jenkins dashboard.
- Click **New Item**.
- Name the job (e.g., "Manual Deployment") and select **Freestyle project**.
- Click **OK** to create the job.

3. Configure Source Code Management:

- In the job configuration, navigate to **Source Code Management**.
- Select **Git** and enter the URL of your Git repository.
- Under **Credentials**, add your Git credentials (e.g., GitHub username and personal access token).
- Specify the branch you want to build from (e.g., main or master).

4. Set Up Build Triggers (Manual Build):

- In the job configuration, go to **Build Triggers**.
- Select **Build periodically**.
- Leave the schedule field blank or set it to a specific schedule if you want to trigger builds automatically at certain intervals.

5. Add Build Steps:

- In the job configuration, go to **Build**.
- Add a build step: **Execute Shell**.
- Paste the following script, replacing the placeholders:

Bash
```
git pull origin main
```

```
# Your build commands here (e.g., compile, build artifacts)
```
Use code with caution.

- Replace the placeholder with your actual build commands.

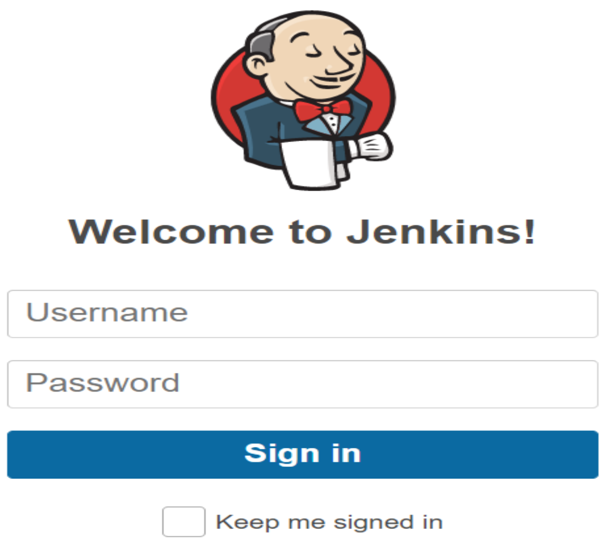
6. Add Post-Build Actions (Optional):

- If you want to perform actions after the build completes (e.g., archive artifacts, send notifications), go to **Post-build Actions** and add the desired actions.

7. Save and Test:

- Save the job configuration.
- To manually trigger a build, go to the job's page and click **Build Now**.
- Jenkins will execute the build steps and perform any configured post-build actions.

OUTPUT:



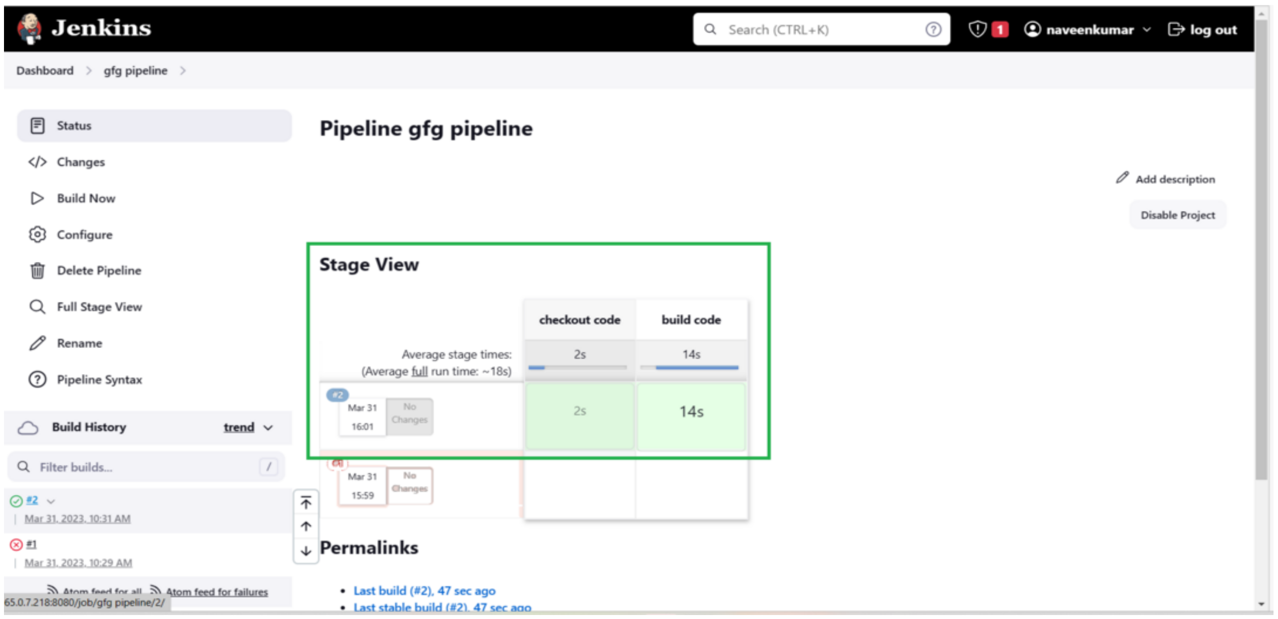
Welcome to Jenkins!

Username

Password

Sign in

Keep me signed in



Jenkins — Search (CTRL+K) — naveenkumar — log out

Dashboard > gfg pipeline >

Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax

Build History trend

Filter builds...

#2 Mar 31, 2023, 10:31 AM

#1 Mar 31, 2023, 10:29 AM

Atom feed for all Atom feed for failures

65.0.7.218:8080/job/gfg pipeline/2/

Pipeline gfg pipeline

Add description

Disable Project

Stage View

	checkout code	build code
Average stage times: (Average full run time: ~18s)	2s	14s
#2 Mar 31 16:01 No Changes	2s	14s
#1 Mar 31 15:59 No Changes		

Permalinks

- Last build (#2), 47 sec ago
- Last stable build (#2), 47 sec ago

RESULT:

Thus the configuration with Jenkins with manual deploy functionality completed successfully.

Ex.No.	10	RUN MYSQL IN DOCKER AND CONFIGURE QUERY LOGIN TO HOST DISK (SYSTEM MEMORY RATHER THAN DOCKER MEMORY) AND CHECK THE HEALTH OF THIS CONTAINER EVERY 10 SECONDS.
Date		

AIM:

To Run MySQL in cocker and configure query login to host disk (system memory rather than docker memory) and check the health of this container every 10 seconds.

ALGORITHM:

Steps:

1. **Create a Directory on Host:**
   - Open File Explorer.
   - Create a new directory to store your MySQL data (e.g., C:\mysql-data).
2. **Run the MySQL Container:**
   - Open PowerShell or Command Prompt.
   - Execute the following command, replacing placeholders with your desired values:
3. docker run -d \
4. --name mysql-container \
5. -e MYSQL_ROOT_PASSWORD=your_root_password \
6. -v C:\mysql-data:/var/lib/mysql \
7. --health-cmd="mysqladmin ping -h localhost" \
8. --health-interval=10s \
9. --health-timeout=5s \
10. --health-retries=3 \
11. -p 3306:3306 \
12. mysql:latest

Explanation of the Command:

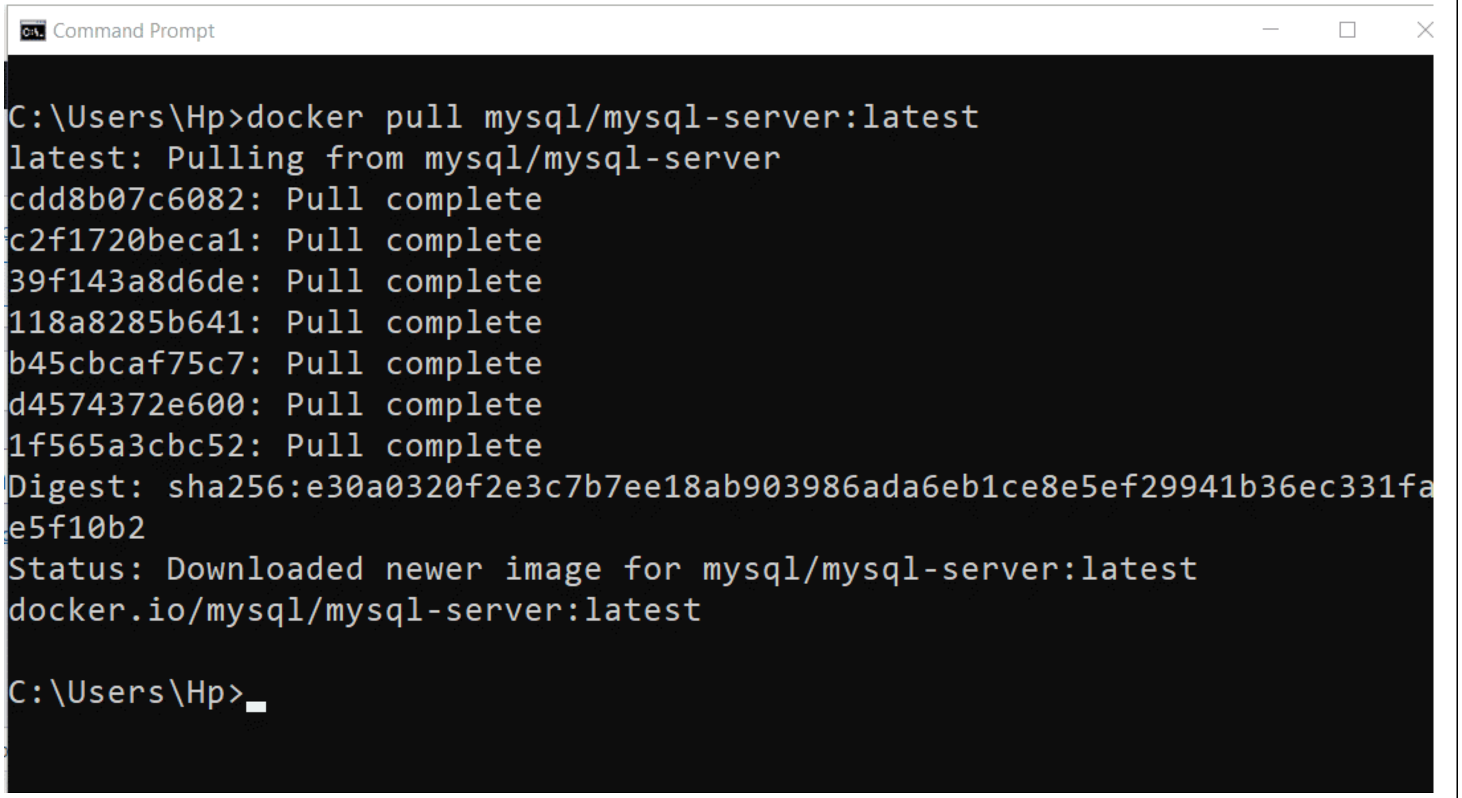
- -d: Runs the container in detached mode (background).
- --name mysql-container: Assigns a name to the container ("mysql-container").
- -e MYSQL_ROOT_PASSWORD=your_root_password: Sets the MySQL root password (replace with your desired password).
- -v C:\mysql-data:/var/lib/mysql: Mounts the host directory (C:\mysql-data) to the container's data directory (/var/lib/mysql), ensuring data persistence on the host.

- --health-cmd="mysqladmin ping -h localhost": Defines the health check command to ping the MySQL server locally.
- --health-interval=10s: Sets the health check interval to 10 seconds.
- --health-timeout=5s: Sets the timeout for the health check command to 5 seconds.
- --health-retries=3: Sets the number of retries for the health check before marking the container unhealthy.
- -p 3306:3306: Maps port 3306 of the container to port 3306 of the host, allowing access to the MySQL service.
- MySQL: latest: Specifies the MySQL image to use (latest version).

13. **Verify Container Health (Optional):**
    - Use the following command to check the health status of your container:
14. docker inspect --format='{{json .State.Health}}' mysql-container
    - This will display a JSON output indicating the container's health status (healthy or unhealthy).
15. **Connecting to MySQL (Optional):**
    - To connect to the MySQL database from the command line:
16. docker exec -it mysql-container mysql -u root -p
    - Enter the root password you set earlier (from step 2) when prompted.
17. **Stopping and Removing the Container (Optional):**
    - To stop the running container:
18. docker stop mysql-container
    - To remove the container completely:

docker rm mysql-container

OUTPUT:



```
C:\Users\Hp>docker pull mysql/mysql-server:latest
latest: Pulling from mysql/mysql-server
cdd8b07c6082: Pull complete
c2f1720beca1: Pull complete
39f143a8d6de: Pull complete
118a8285b641: Pull complete
b45cbcaf75c7: Pull complete
d4574372e600: Pull complete
1f565a3cbc52: Pull complete
Digest: sha256:e30a0320f2e3c7b7ee18ab903986ada6eb1ce8e5ef29941b36ec331fa
e5f10b2
Status: Downloaded newer image for mysql/mysql-server:latest
docker.io/mysql/mysql-server:latest

C:\Users\Hp>_
```

RESULT:

Thus, the MySQL in docker and configure query login to host was completed successful

Ex.No.	11	RUN NGINX IN DOCKER AND CONFIGURE ANY WEB LOCATION/URL TO REDIRECT TO AN URL AND LIMIT ONLY 5 CONNECTION PER USER EACH MINUTE.
Date		

AIM:

To Run nginx in docker and configure any web location/url to redirect to an url and limit only 5 connection per user each minute.

ALGORITHM:

Step 1: Create a Docker Compose File

Create a new file named docker-compose.yml and paste the following configuration:

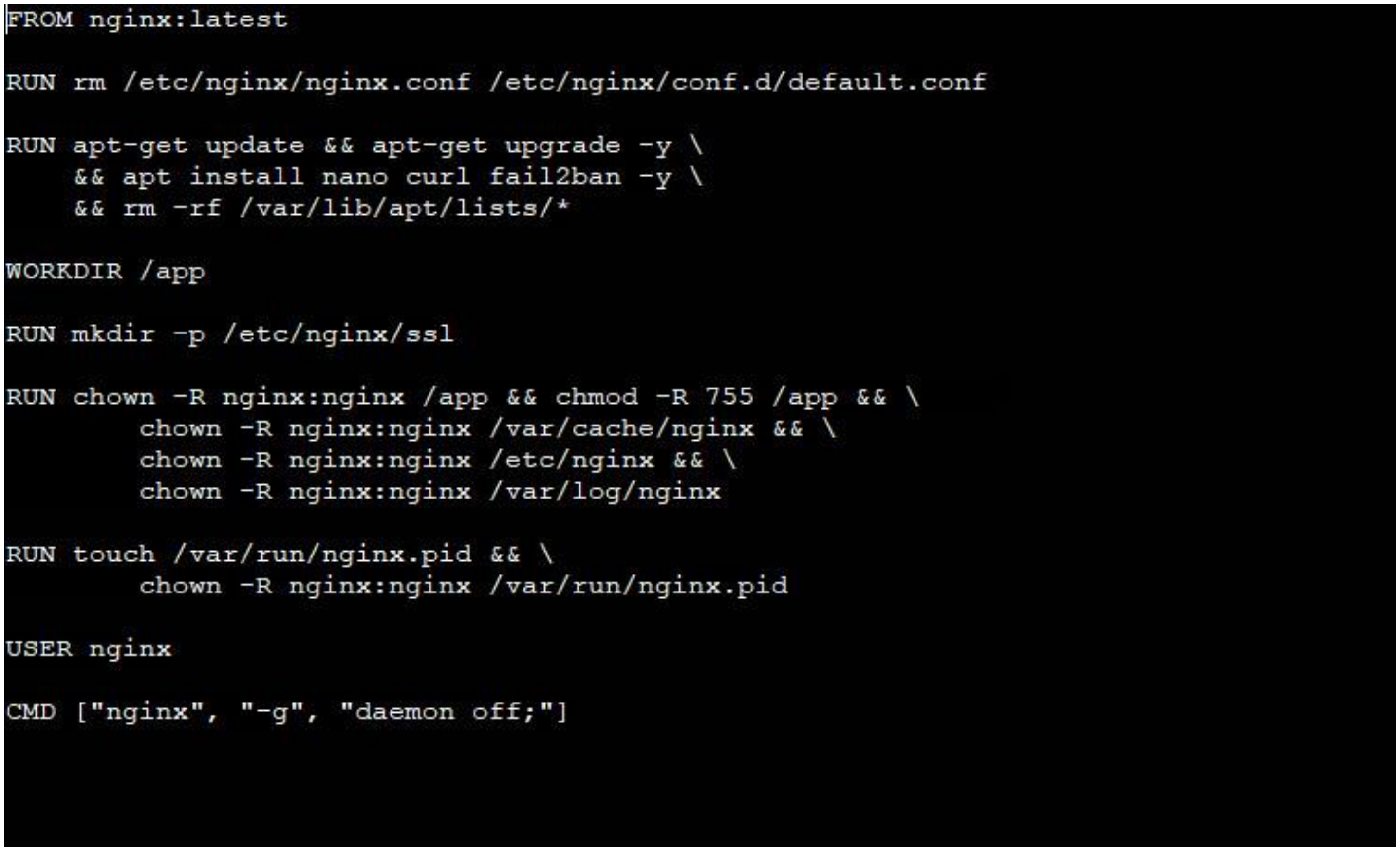
```
YAML
version: '3.7'

services:
  nginx:
    image: nginx:latest
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
    ports:
      - 80:80
    command: nginx -g 'daemon off;'

volumes:
  nginx.conf:
    driver: local
```

Step 2: Create the Nginx Configuration File

Create a file named nginx.conf in the same directory as your docker-compose.yml file. Paste the following configuration into it:

```
Nginx
user www-data;
worker_processes auto;
error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include
mime.types;
   default_type application/octet-stream;

   log_format main '$remote_addr - $remote_user [$time_local] "$request" $status
$bytes_sent "$http_referer" "$http_user_agent"';
   access_log /var/log/nginx/access.log main;


   server {
       listen 80;
       server_name your_domain_name;

       location / {
           # Redirect to your desired URL
           rewrite ^ /your_desired_url permanent;

           # Limit connections to 5 per user per minute
           limit_conn_zone $binary_remote_addr zone=per_user:10m;
           limit_conn per_user 5;
       }
   }
}
```

OUTPUT:



```
FROM nginx:latest

RUN rm /etc/nginx/nginx.conf /etc/nginx/conf.d/default.conf

RUN apt-get update && apt-get upgrade -y \
    && apt install nano curl fail2ban -y \
    && rm -rf /var/lib/apt/lists/*

WORKDIR /app

RUN mkdir -p /etc/nginx/ssl

RUN chown -R nginx:nginx /app && chmod -R 755 /app && \
        chown -R nginx:nginx /var/cache/nginx && \
        chown -R nginx:nginx /etc/nginx && \
        chown -R nginx:nginx /var/log/nginx

RUN touch /var/run/nginx.pid && \
        chown -R nginx:nginx /var/run/nginx.pid

USER nginx

CMD ["nginx", "-g", "daemon off;"]
```

RESULT:

Thus, the nginx in docker and configure any web location was completed successfully