

Creating an EC2 Instance and Deploying a Sample Web Application

Aim:

The aim of this exercise is to demonstrate the process of creating an EC2 instance on Amazon Web Services (AWS) and deploying a sample web application on the instance.

Procedure:

Step 1: Log in to your AWS Management Console.



Sign in

Root user

Account owner that performs tasks requiring unrestricted access. [Learn more](#)

IAM user

User within an account that performs daily tasks.
[Learn more](#)

Root user email address

22pds007@loyolacollege.edu

Next

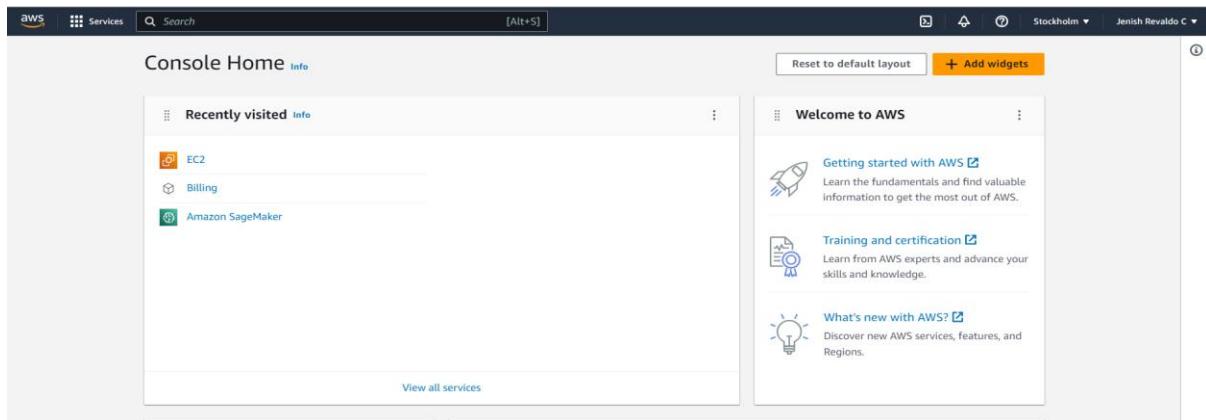
Email: 22pds007@loyolacollege.edu

Password

[Forgot password?](#)

.....

Sign in



Step 2: Navigate to the EC2 Dashboard.

A screenshot of the AWS search interface. The search term 'ec2' is entered in the search bar. The results are categorized under 'Services (13)' and 'Features (52)'. Under 'Services', the 'EC2' service is listed first, followed by 'EC2 Image Builder', 'Recycle Bin', and 'Amazon Inspector'. The 'Instances' tab under EC2 is highlighted with a yellow box. Under 'Features', there's a 'Dashboard' link. A 'See all 13 results' link is also visible.

#EC2 Dashboard

A screenshot of the EC2 Instances dashboard. The left sidebar shows navigation links like 'EC2 Dashboard', 'EC2 Global View', 'Events', 'Instances' (selected), 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Scheduled Instances', and 'Capacity Reservations'. The main area is titled 'Instances Info' and shows a table with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Avail. A message says 'No matching instances found'. A modal window titled 'Select an instance' is open at the bottom.

Step 3: Click on "Launch Instance" to start the EC2 instance creation wizard.

The screenshot shows the AWS EC2 Instances page. At the top, there are several buttons: Instances Info, Connect, Instance state, Actions, and Launch instances. The Launch instances button is highlighted with a yellow box. Below the buttons is a search bar with the placeholder text "Find instance by attribute or tag (case-sensitive)". Underneath the search bar is a table header with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Avail. A message "No matching instances found" is displayed below the table. At the bottom right of the page is a small gear icon.

Step 4:

- Add Name to the Instance

The screenshot shows the "Launch an instance" step. At the top, it says "Launch an instance" with an "Info" link. Below that, a message says "Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below." The main section is titled "Name and tags" with an "Info" link. It has a "Name" field containing "Sample Web Application". To the right of the name field is a link "Add additional tags".

- Choose an Amazon Machine Image (AMI) for your instance (e.g., Amazon Linux 2).

The screenshot shows the "Application and OS Images (Amazon Machine Image)" section. At the top, it says "▼ Application and OS Images (Amazon Machine Image)" with an "Info" link. Below that, a message says "An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below". There is a search bar with the placeholder text "Search our full catalog including 1000s of application and OS images". The "Quick Start" section lists several AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. To the right of the quick start section is a search icon and a link "Browse more AMIs" which includes a note "Including AMIs from AWS, Marketplace and the Community". Below the quick start section, a specific AMI is selected: "Amazon Linux 2023 AMI". The details for this AMI are shown: "ami-040d60c831d02d41c (64-bit (x86)) / ami-06c326063d3b494f1 (64-bit (Arm))", "Virtualization: hvm", "ENA enabled: true", "Root device type: ebs", and "Free tier eligible". Below the AMI details, there is a "Description" section with the text "Amazon Linux 2023 AMI 2023.1.20230725.0 x86_64 HVM kernel-6.1", an "Architecture" dropdown set to "64-bit (x86)", an "AMI ID" field with the value "ami-040d60c831d02d41c", and a "Verified provider" badge.

- Select an instance type (e.g., t2.micro) based on your requirements.

▼ Instance type [Info](#)

Instance type

t3.micro	Free tier eligible
Family: t3 2 vCPU 1 GiB Memory Current generation: true	
On-Demand RHEL pricing: 0.0708 USD per Hour	
On-Demand SUSE pricing: 0.0108 USD per Hour	
On-Demand Linux pricing: 0.0108 USD per Hour	
On-Demand Windows pricing: 0.02 USD per Hour	

All generations

[Compare instance types](#)

- Create a new key pair or use an existing one to securely connect to the instance later.

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

#Give name and create key pair

Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

<input checked="" type="radio"/> RSA RSA encrypted private and public key pair	<input type="radio"/> ED25519 ED25519 encrypted private and public key pair
---	--

Private key file format

<input checked="" type="radio"/> .pem For use with OpenSSH	<input type="radio"/> .ppk For use with PuTTY
---	--

⚠ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

[Create new key pair](#)

- Configure the instance details (e.g., network settings, security groups).

▼ Network settings [Info](#)

[Edit](#)

Network [Info](#)

vpc-09f63052e479c3681

Subnet [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)

Enable

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

 Create security group Select existing security group

We'll create a new security group called '**launch-wizard-9**' with the following rules:

Allow SSH traffic from

Helps you connect to your instance

Anywhere

▼

0.0.0.0/0

Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

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

X

- Add storage if needed.

▼ Configure storage [Info](#)

[Advanced](#)

1x GiB Root volume (Not encrypted)

ⓘ Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

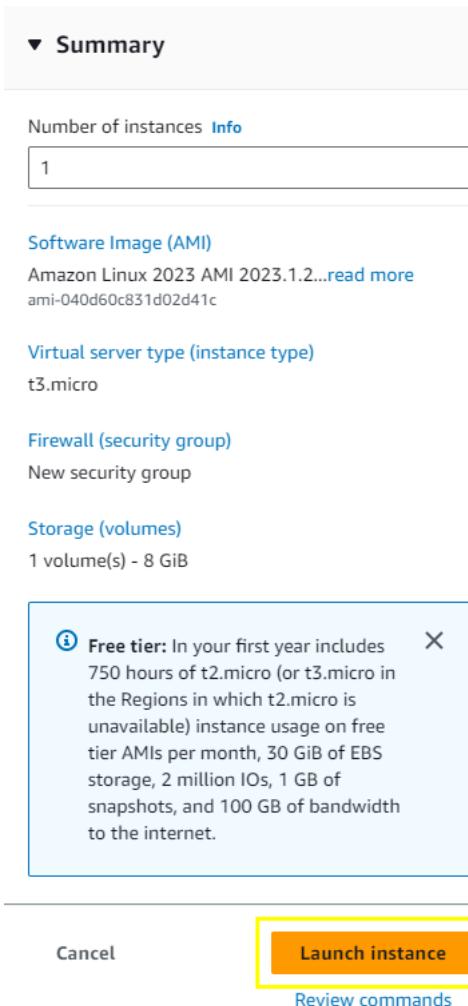
X

[Add new volume](#)

0 x File systems

[Edit](#)

- Review your settings and click on "Launch."



#After Launching



Step 5: Connecting the Instance

#Move to EC2 Dashboard, Select the newly created Instance and click Connect.

Instances (1/1) Info										
Find instance by attribute or tag (case-sensitive) Launch instances										
<input checked="" type="checkbox"/> Instance state = running X Clear filters										
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	
Sample Web A...	i-0b9f3dc6aaf598f79	Running @Q	t3.micro	2/2 checks passed No alarms	+	eu-north-1a	ec2-13-49-18-49.eu-no...	13.49.18.49	-	

#Connect to instance using EC2 Instance Connect

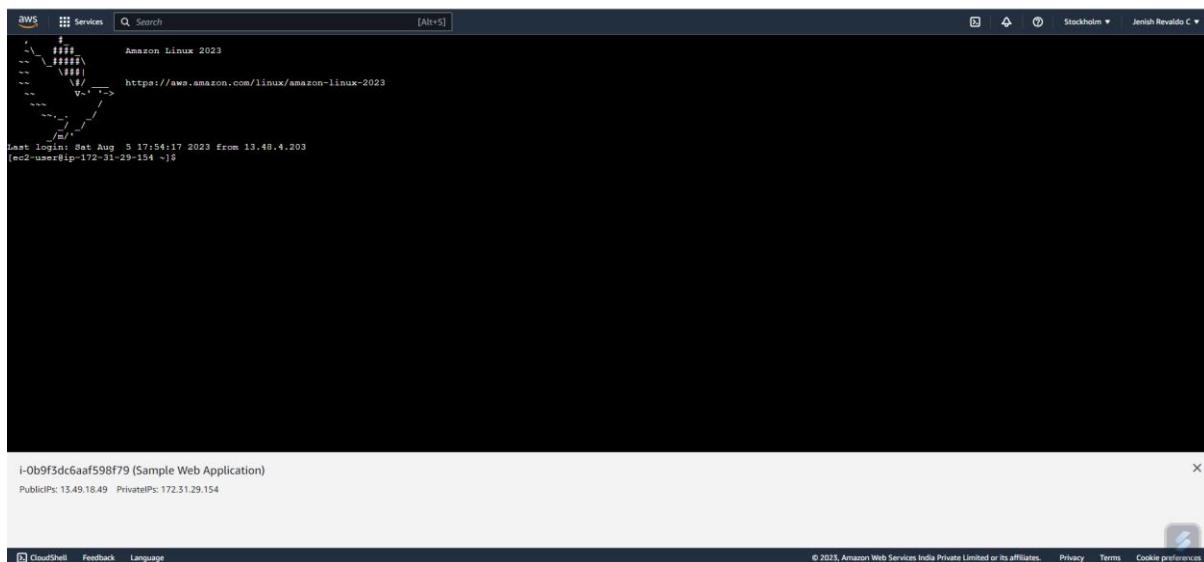
The screenshot shows the 'Connect to instance' dialog box. At the top, there's a breadcrumb navigation: EC2 > Instances > i-0b9f3dc6aaf598f79 > Connect to instance. Below the title 'Connect to instance' is a link 'Info'. A sub-header says 'Connect to your instance i-0b9f3dc6aaf598f79 (Sample Web Application) using any of these options'. There are four tabs at the top: 'EC2 Instance Connect' (selected), 'Session Manager', 'SSH client', and 'EC2 serial console'.
Instance ID: i-0b9f3dc6aaf598f79 (Sample Web Application)
Connection Type: Two options are shown:

- Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.
- Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address: 13.49.18.49
User name: ec2-user
Note: In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

At the bottom right are 'Cancel' and 'Connect' buttons, with 'Connect' being highlighted by a yellow box.

#After establishing connection



Step 6: Switch to the superuser (root) account in Linux systems

```
[ec2-user@ip-172-31-29-154 ~]$ sudo su -
```

Step 7: Update the package manager and install Apache HTTP server.

```
[root@ip-172-31-29-154 ~]# yum update -y
```

```
[root@ip-172-31-29-154 ~]# yum install -y httpd
```

Step 8: Check the status of the Apache HTTP server

```
[root@ip-172-31-29-154 ~]# systemctl status httpd
```

○ *httpd.service - The Apache HTTP Server*

Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)

Active: inactive (dead)

Docs: man:httpd.service(8)

Step 9: Make a directory temp1 and change to that directory

```
[root@ip-172-31-29-154 ~]# mkdir temp1
```

```
[root@ip-172-31-29-154 ~]# cd temp1
```

Step 10: Download the template from free-css.com using wget command

```
[root@ip-172-31-38-20 temp1]# wget https://www.free-css.com/assets/files/free-css-templates/download/page293/fitapp.zip
```

Step 11: Unzip the file

```
[root@ip-172-31-38-20 temp1]# unzip fitapp.zip
```

Step 12: List files and directories in the directory in long format.

```
[root@ip-172-31-29-154 temp1]# ls -ltr
```

total 552
-rw-r--r--. 1 root root 564102 Jan 7 2022 fitapp.zip
drwxr-xr-x. 7 root root 153 Aug 5 18:00 mobile-app-html-template

Step 13: Change directory to mobile-app-html-template

```
[root@ip-172-31-29-154 temp1]# cd mobile-app-html-template
```

Step 14: Move all the files to the Apache web server's document root.

```
[root@ip-172-31-29-154 mobile-app-html-template]# mv * /var/www/html
```

Step 15: Start the Apache web server

```
[root@ip-172-31-29-154 mobile-app-html-template]# service httpd start
```

Step 16: Check the status of the Apache HTTP server

```
[root@ip-172-31-29-154 mobile-app-html-template]# systemctl status httpd
```

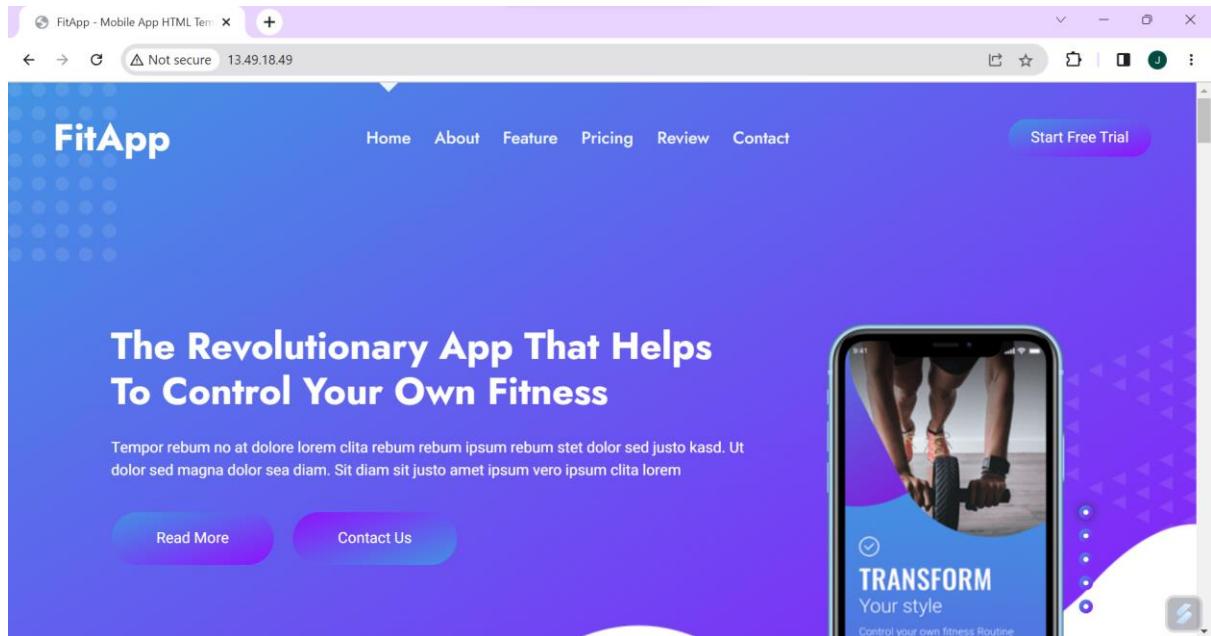
- *httpd.service - The Apache HTTP Server*

Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)

Active: active (running) since Sat 2023-08-05 18:02:26 UTC; 16s ago

Docs: man:httpd.service(8)

Output:



Result:

The steps for creating an Amazon EC2 instance, installing the Apache HTTP Server, and deploying a sample web application (FitApp) have been successfully performed, resulting in a fully operational web server hosting the web application on the EC2 instance.

Deployment of a basic Python web application using the Flask framework

Aim:

To demonstrate the deployment of a basic Python web application using the Flask framework on an Ubuntu EC2 instance.

Procedure:

Step 1: Create an EC2 instance with Ubuntu in Amazon Machine Image allowing HTTP and HTTPS traffic.

Step 2: Update package information on the Ubuntu EC2 instance.

```
ubuntu@ip-172-31-46-36:~$ sudo apt update
```

Step 3: Install Python3 package manager (pip) on the instance.

```
ubuntu@ip-172-31-46-36:~$ sudo apt install python3-pip
```

Step 4: Check the version of pip to verify installation.

```
ubuntu@ip-172-31-46-36:~$ pip --version
```

pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)

Step 5: Check the version of Git to verify installation.

```
ubuntu@ip-172-31-46-36:~$ git --version
```

git version 2.34.1

Step 6: Clone the GitHub repository containing the Flask web application code.

```
ubuntu@ip-172-31-46-36:~$ git clone https://github.com/yeshwanthlm/docker-flask-demo
```

Step 7: List files and directories in the directory in long format.

```
ubuntu@ip-172-31-46-36:~$ ls -ltr
total 4
drwxrwxr-x 4 ubuntu ubuntu 4096 Jul 15 16:13 docker-flask-demo
```

Step 8: Navigate to the cloned directory.

```
ubuntu@ip-172-31-46-36:~$ cd docker-flask-demo
```

Step 9: List the contents of the directory to see the files.

```
ubuntu@ip-172-31-46-36:~/docker-flask-demo$ ls -ltr
total 20
drwxrwxr-x 2 ubuntu ubuntu 4096 Jul 15 16:13 templates
-rw-rw-r-- 1 ubuntu ubuntu 5 Jul 15 16:13 requirements.txt
-rw-rw-r-- 1 ubuntu ubuntu 2258 Jul 15 16:13 app.py
-rw-rw-r-- 1 ubuntu ubuntu 687 Jul 15 16:13 Jenkinsfile
-rw-rw-r-- 1 ubuntu ubuntu 112 Jul 15 16:13 Dockerfile
```

Step 10: Install the Flask package using pip3.

```
ubuntu@ip-172-31-46-36:~/docker-flask-demo$ pip3 install flask
```

Step 11: Run the Flask web application.

```
ubuntu@ip-172-31-46-36:~/docker-flask-demo$ python3 app.py
```

*This is a sample web application that displays a colored background.
A color can be specified in two ways.*

1. As a command line argument with --color as the argument. Accepts one of red,green,blue,blue2,pink,darkblue
2. As an Environment variable APP_COLOR. Accepts one of red,green,blue,blue2,pink,darkblue
3. If none of the above then a random color is picked from the above list.
Note: Command line argument precedes over environment variable.

```

No command line argument or environment variable. Picking a Random Color
=blue2
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8080
* Running on http://172.31.46.36:8080
Press CTRL+C to quit

```

Step 12: Allow Custom TCP port 8080 in Inbound rules.

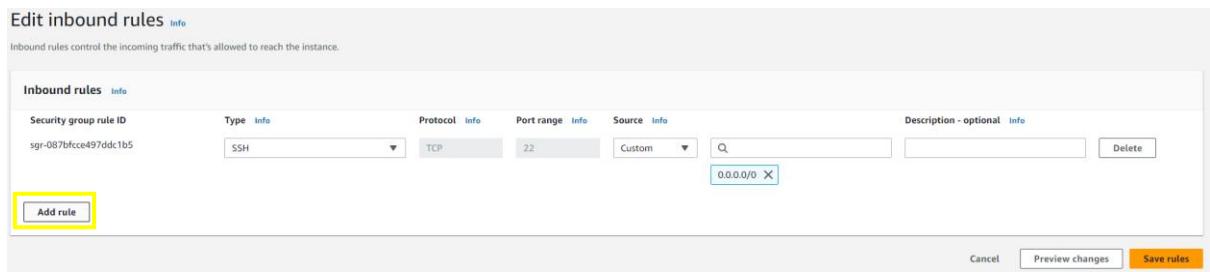
#Go to EC2 Dashboard, Select Instance – Security – Security groups

The screenshot shows the AWS EC2 Instances dashboard. A single instance, 'python app' (ID: i-0146f5973a8490eb4), is listed as 'Running'. The 'Security' tab is selected in the instance details panel, showing the security group 'sg-04d671423ae6b70f6 (launch-wizard-4)'.

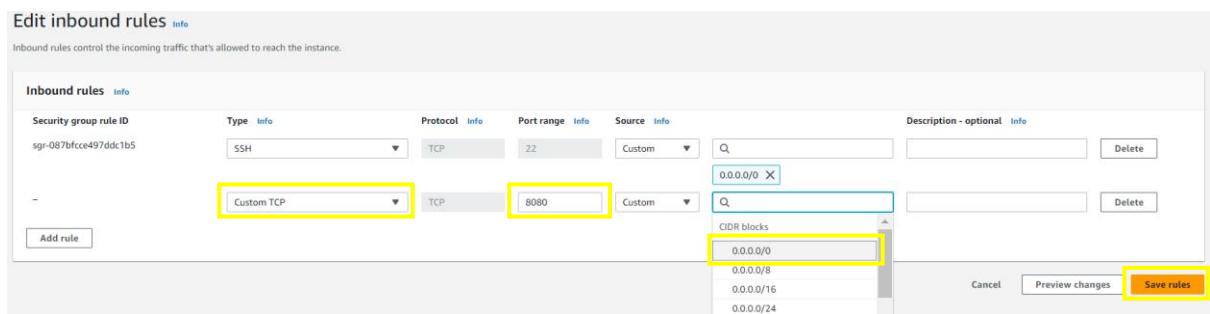
#Select Edit inbound rules

The screenshot shows the AWS Security Groups details page for the security group 'sg-04d671423ae6b70f6 - launch-wizard-4'. The 'Inbound rules' tab is selected, showing one rule: 'sgr-087bfccce497ddc1b5' (Protocol: TCP, Port range: 22, Source: 0.0.0.0/0). The 'Edit inbound rules' button is highlighted with a yellow box.

#Select Add rule



#Set port range 8080, choose Anywhere-IPv4 and Click save rules



Step 13: Now open <Public_IP_Address_of_EC2_Instance>:8080 in Browser

Output:



Result:

Thus, successfully deployed the basic Python web application using Flask on the Ubuntu EC2 instance, and the application was accessible through the EC2 instance's public IP address.

Installing MySQL in Ubuntu Instance and Performing CRUD Operations

Aim:

To demonstrate how to install MySQL on an Ubuntu EC2 instance and perform basic CRUD (Create, Read, Update, Delete) operations on a database using the MySQL command-line interface.

Procedure:

Step 1: Create an EC2 instance with Ubuntu in Amazon Machine Image

Step 2: Update package information on the Ubuntu EC2 instance.

```
ubuntu@ip-172-31-45-247:~$ sudo apt update
```

Step 3: Install MySQL server on the instance.

```
ubuntu@ip-172-31-45-247:~$ sudo apt install mysql-server
```

Step 4: Check the status of the MySQL service to ensure it's running.

```
ubuntu@ip-172-31-45-247:~$ sudo systemctl status mysql
```

```
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2023-06-28 09:42:45 UTC; 8min ago
       Process: 2884 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
      Main PID: 2892 (mysqld)
        Status: "Server is operational"
         Tasks: 37 (limit: 1111)
        Memory: 354.1M
          CPU: 3.015s
        CGroup: /system.slice/mysql.service
                  └─2892 /usr/sbin/mysqld

Jun 28 09:42:45 ip-172-31-45-247 systemd[1]: Starting MySQL Community Server...
Jun 28 09:42:45 ip-172-31-45-247 systemd[1]: Started MySQL Community Server.
```

Step 5: Access the MySQL command-line interface (CLI) as the root user.

```
ubuntu@ip-172-31-45-247:~$ sudo mysql
```

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.33-Ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

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

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

Now you can run the SQL commands

Step 6: Create a new database named 'student'.

```
mysql> create database student;
```

Query OK, 1 row affected (0.02 sec)

Step 7: Show the list of databases to verify the creation of the 'student' database.

```
mysql> show databases;
```

Database
information_schema
mysql
performance_schema
student
sys

5 rows in set (0.20 sec)

Step 8: Switch to the 'student' database for further operations.

```
mysql> use student;
```

Database changed

Step 9: Create a table named 'name_details' with two columns 'id' (integer) and 'name' (varchar).

```
mysql> create table name_details(id int,name varchar(45));
```

Query OK, 0 rows affected (0.04 sec)

Step 10: Insert three rows of data into the 'name_details' table.

```
mysql> insert into name_details values(1,'Jenish'),(2,'Revaldo'),(3,'mery');
```

Query OK, 3 rows affected (0.01 sec)

Records: 3 Duplicates: 0 Warnings: 0

Step 11: Retrieve and display all records from the 'name_details' table.

```
mysql> select * from name_details;
```

id	name
1	Jenish
2	Revaldo
3	mery

3 rows in set (0.00 sec)

Step 12: Update a record from the 'name_details' table.

```
mysql> UPDATE name_details SET name = 'Mahe' WHERE id = 3;
```

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

Step 13: Delete a record from the 'name_details' table.

```
mysql> DELETE FROM name_details WHERE id = 2;
```

Query OK, 1 row affected (0.00 sec)

Step 14: Retrieve and display all records from the 'name_details' table.

```
mysql> select * from name_details;
```

id	name
1	Jenish
3	Mahe

2 rows in set (0.00 sec)

Step 15: Exit the MySQL CLI.

```
mysql> exit
```

Bye

Result:

Thus, successfully installed MySQL, created a database, and performed CRUD operations to manage data efficiently on the Ubuntu instance.

Deployment of Flask Web Application with Docker

Aim:

To deploy a Flask web application using Docker on an Ubuntu instance with seamless containerization and port mapping.

Procedure:

Step 1: Create an EC2 instance with Ubuntu in Amazon Machine Image.

Step 2: Update package information on the Ubuntu EC2 instance.

```
ubuntu@ip-172-31-42-183:~$ sudo apt update
```

Step 3: Install Docker on the Ubuntu instance by downloading the installation script using ‘curl’ and then executing the script using the ‘sh’ shell to perform the installation

```
ubuntu@ip-172-31-42-183:~$ curl -fsSL https://get.docker.com -o get-docker.sh
```

```
ubuntu@ip-172-31-42-183:~$ sh get-docker.sh
```

Step 4: Check the version of docker to verify installation.

```
ubuntu@ip-172-31-42-183:~$ docker --version
```

Docker version 24.0.4, build 3713ee1

Step 5: Clone the GitHub repository containing the Flask web application and docker file.

```
ubuntu@ip-172-31-42-183:~$ git clone https://github.com/yeshwanthlm/docker-flask-demo
```

Step 6: List files and directories in the directory in long format.

```
ubuntu@ip-172-31-42-183:~$ ls -ltr
total 28
-rw-rw-r-- 1 ubuntu ubuntu 21926 Jul 17 08:22 get-docker.sh
drwxrwxr-x 4 ubuntu ubuntu 4096 Jul 17 08:26 docker-flask-demo
```

Step 7: Navigate to the cloned directory.

```
ubuntu@ip-172-31-42-183:~$ cd docker-flask-demo
```

Step 8: List the contents of the directory to see the files.

```
ubuntu@ip-172-31-42-183:~/docker-flask-demo$ ls -ltr
total 20
drwxrwxr-x 2 ubuntu ubuntu 4096 Jul 15 16:13 templates
-rw-rw-r-- 1 ubuntu ubuntu 5 Jul 15 16:13 requirements.txt
-rw-rw-r-- 1 ubuntu ubuntu 2258 Jul 15 16:13 app.py
-rw-rw-r-- 1 ubuntu ubuntu 687 Jul 15 16:13 Jenkinsfile
-rw-rw-r-- 1 ubuntu ubuntu 112 Jul 15 16:13 Dockerfile
```

Step 9: Display the contents of the Dockerfile in the terminal.

```
ubuntu@ip-172-31-42-183:~/docker-flask-demo$ cat Dockerfile
FROM python:3.6
RUN pip install flask
COPY . /opt/
EXPOSE 8080
WORKDIR /opt
```

Step 10: Build the Docker image.

```
ubuntu@ip-172-31-42-183:~/docker-flask-demo$ sudo docker build -t amc/flaskapp .
```

Step 11: List all the Docker images available on the system.

```
ubuntu@ip-172-31-42-183:~/docker-flask-demo$ sudo docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
amc/flaskapp    latest   e85ea9eeabdb  5 minutes ago  913MB
```

Step 12: Run the Flask web application as a Docker container and map port 8080 to port 80 on the host.

```
ubuntu@ip-172-31-42-183:~/docker-flask-demo$ sudo docker run -d -p 80:8080
e85ea9eeabdb
```

Step 13: List all the running containers on the system.

```
ubuntu@ip-172-31-42-183:~/docker-flask-demo$ sudo docker ps
CONTAINER ID  IMAGE      COMMAND      CREATED      STATUS
77904b80ac1d  e85ea9eeabdb  "python app.py"  19 seconds ago  Up 18 seconds
PORTS          NAMES
0.0.0.0:80->8080/tcp, :::80->8080/tcp  reverent_williamson
```

Step 14: Allow Custom TCP port 80 in Inbound rules.

Step 15: Now open <Public_IP_Address_of_EC2_Instance>:80 in Browser

Output:



Result:

The Flask web application is deployed using Docker, and the container is running, accessible on port 80 of the host machine.

Setting up simple pipeline using Git, Jenkins, and Docker in local mode on an Ubuntu instance

Aim:

To set up a simple pipeline using Git, Jenkins, and Docker in local mode on an Ubuntu instance.

Procedure:

Step 1: Create an EC2 instance with Ubuntu in Amazon Machine Image.

Step 2: Update package information on the Ubuntu EC2 instance.

```
ubuntu@ip-172-31-47-227:~$ sudo apt-get update
```

```
ubuntu@ip-172-31-47-227:~$ sudo apt update
```

Step 3: Install Docker on the Ubuntu instance by downloading the installation script using ‘curl’ and then executing the script using the ‘sh’ shell to perform the installation

```
ubuntu@ip-172-31-47-227:~$ curl -fsSL https://get.docker.com -o get-docker.sh
```

```
ubuntu@ip-172-31-47-227:~$ sh get-docker.sh
```

Step 4: Check the version of docker to verify installation.

```
ubuntu@ip-172-31-47-227:~$ docker --version
```

Docker version 24.0.4, build 3713ee1

Step 5: Create a docker hub account in DockerHub website

Step 6: Login docker

```
ubuntu@ip-172-31-47-227:~$ sudo docker login -u <username> -p <password>
```

Replace <username> and <password> with DockerHub username and password.

Step 7: Install OpenJDK 17 (Java Runtime Environment).

```
ubuntu@ip-172-31-47-227:~$ sudo apt install openjdk-17-jre
```

Step 8: Check the Java version to ensure it is installed correctly.

```
ubuntu@ip-172-31-47-227:~$ java -version
```

openjdk version "17.0.7" 2023-04-18

OpenJDK Runtime Environment (build 17.0.7+7-Ubuntu-0ubuntu122.04.2)

OpenJDK 64-Bit Server VM (build 17.0.7+7-Ubuntu-0ubuntu122.04.2, mixed mode, sharing)

Step 9: Import Jenkins Repository Key.

```
ubuntu@ip-172-31-47-227:~$ curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key |  
sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

Step 10: Add Jenkins Repository to Sources List.

```
ubuntu@ip-172-31-47-227:~$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]  
https://pkg.jenkins.io/debian binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
```

Step 11: Update the package lists from all repositories, including the newly added Jenkins repository.

```
ubuntu@ip-172-31-47-227:~$ sudo apt-get update
```

Step 12: Install Jenkins.

```
ubuntu@ip-172-31-47-227:~$ sudo apt-get install jenkins
```

Step 13: Start Jenkins service and check its status.

```
ubuntu@ip-172-31-47-227:~$ sudo systemctl start jenkins.service
```

```
ubuntu@ip-172-31-47-227:~$ sudo systemctl status jenkins
```

• *jenkins.service - Jenkins Continuous Integration Server*

Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)

Active: active (running) since Fri 2023-07-21 12:50:18 UTC; 1min 57s ago

Main PID: 6680 (java)

Tasks: 38 (limit: 1111)

Memory: 292.6M

CPU: 1min 17.623s

CGroup: /system.slice/jenkins.service

└─6680 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Jul 21 12:49:30 ip-172-31-47-227 jenkins[6680]: c84c43b9ad14d60873f4185427328ba

Step 14: Configure firewall to allow traffic on port 8080 for Jenkins, SSH connections for remote access, port 8000 for container communication.

```
ubuntu@ip-172-31-47-227:~$ sudo ufw allow 8080
```

```
ubuntu@ip-172-31-47-227:~$ sudo ufw allow OpenSSH
```

```
ubuntu@ip-172-31-47-227:~$ sudo ufw allow 8000
```

Step 15: Enable the firewall.

```
ubuntu@ip-172-31-47-227:~$ sudo ufw enable
```

Step 16: Check the firewall status.

```
ubuntu@ip-172-31-47-227:~$ sudo ufw status
```

```
Status: active

To          Action    From
--          ----     ---
8080        ALLOW     Anywhere
OpenSSH      ALLOW     Anywhere
8000        ALLOW     Anywhere
8080 (v6)   ALLOW     Anywhere (v6)
OpenSSH (v6) ALLOW     Anywhere (v6)
8000 (v6)   ALLOW     Anywhere (v6)
```

Step 17: Retrieve the initialAdminPassword for Jenkins setup.

```
ubuntu@ip-172-31-47-227:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
c84c43b9adf14d60873f4185427328ba
```

Step 18: Add the Jenkins user to the docker group for Docker access.

```
ubuntu@ip-172-31-47-227:~$ sudo usermod -aG docker jenkins
```

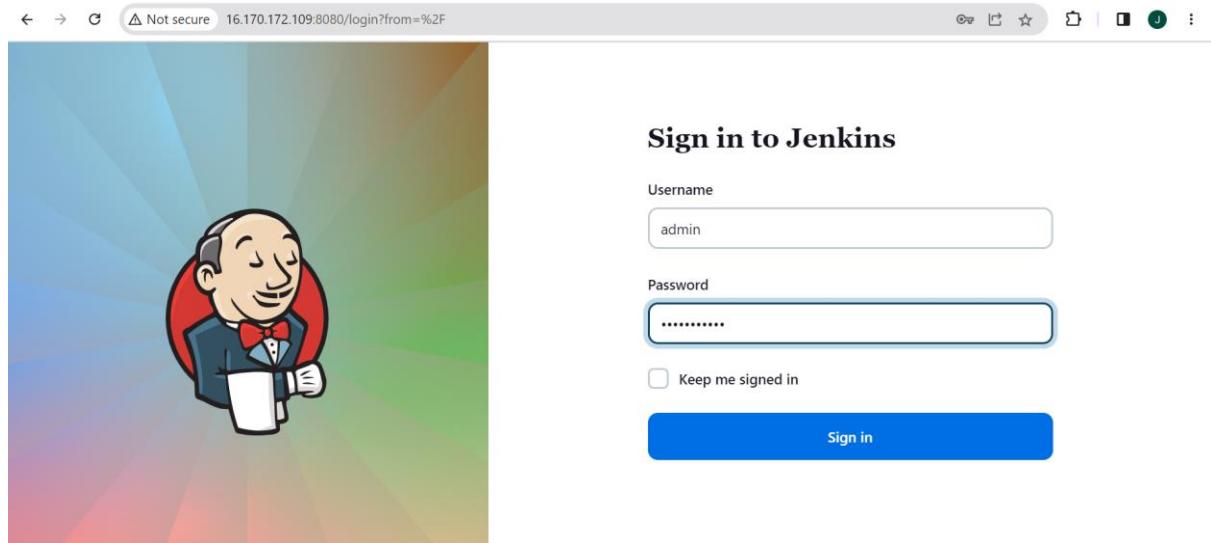
Step 19: Restart Jenkins service to apply the changes.

```
ubuntu@ip-172-31-47-227:~$ sudo systemctl restart jenkins
```

Step 20: Allow Custom TCP port 8080, 8000 in Inbound rules.

Step 21: Now open <Public_IP_Address_of_EC2_Instance>:8080 in Browser to open Jenkins.

Step 22: Login with initialAdminPassword that we retrieved – Install suggested plugins – Setup new username and password.



Step 23: Setup Credentials in Jenkins.

A screenshot of the Jenkins dashboard. At the top, there is a navigation bar with the Jenkins logo and the word "Dashboard". Below the dashboard, there are several menu items: "+ New Item", "People", "Build History", "Project Relationship" (which is highlighted with a yellow box), "Check File Fingerprint", "Manage Jenkins" (which is also highlighted with a yellow box), and "My Views".

<next>

Manage Jenkins

Search settings /

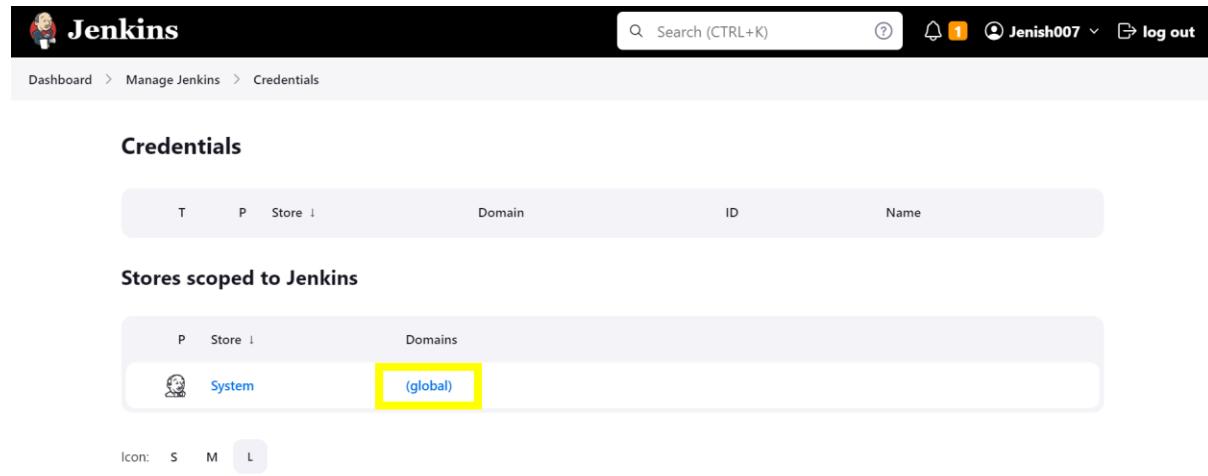
System Configuration

 System Configure global settings and paths.	 Tools Configure tools, their locations and automatic installers.	 Plugins Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
 Nodes Add, remove, control and monitor the various nodes that Jenkins runs jobs on.	 Clouds Add, remove, and configure cloud instances to provision agents on-demand.	

Security

 Security Secure Jenkins; define who is	 Credentials Configure credentials	 Credential Providers Configure the credential providers
--	---	---

<next>



The screenshot shows the Jenkins Manage Jenkins page with the 'Credentials' section selected. A yellow box highlights the 'Credentials' button. Below it, the 'Stores scoped to Jenkins' table has a row where 'System' is selected and '(global)' is highlighted with a yellow box. The 'Icon' column shows icons for System, M, and L.

<next>

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

<next>

Now add your Docker hub User name and password and save it with Id (I have used test1).
Also make necessary changes in the Jenkinsfile like setting **credentials id** and **username of docker**

New credentials

Kind

Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
jenish007

Treat username as secret ?

Password ?
.....

ID ?
test1

Description ?
dockerhub credentials

Create

Step 24: Create Pipeline in Jenkins.



+ New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

<next>

Enter an item name of your wish.

Enter an item name

Required field

 **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a

OK

<next>

Add some description:

General Enabled

Description

Plain text [Preview](#)

<next>

Build Triggers with Poll SCM.

This poll SCM with Schedule ‘* * * * *’ will check the git repository every minute and if a commit is encountered it automatically build up...

Build Triggers

The screenshot shows the 'Build Triggers' section of a Jenkins job configuration. It includes several options: 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. The 'Poll SCM' option is checked and highlighted with a yellow box. Below it is a 'Schedule' field containing the cron expression '* * * * *'. A large rectangular area covers the rest of the page content.

<next>

Display name – The same name given in item name

Advanced Project Options

The screenshot shows the 'Advanced Project Options' section. It features a 'Display Name' field containing the value 'app', which is highlighted with a yellow box. Above the field is an 'Advanced' button with a dropdown arrow and an 'Edited' status indicator.

<next>

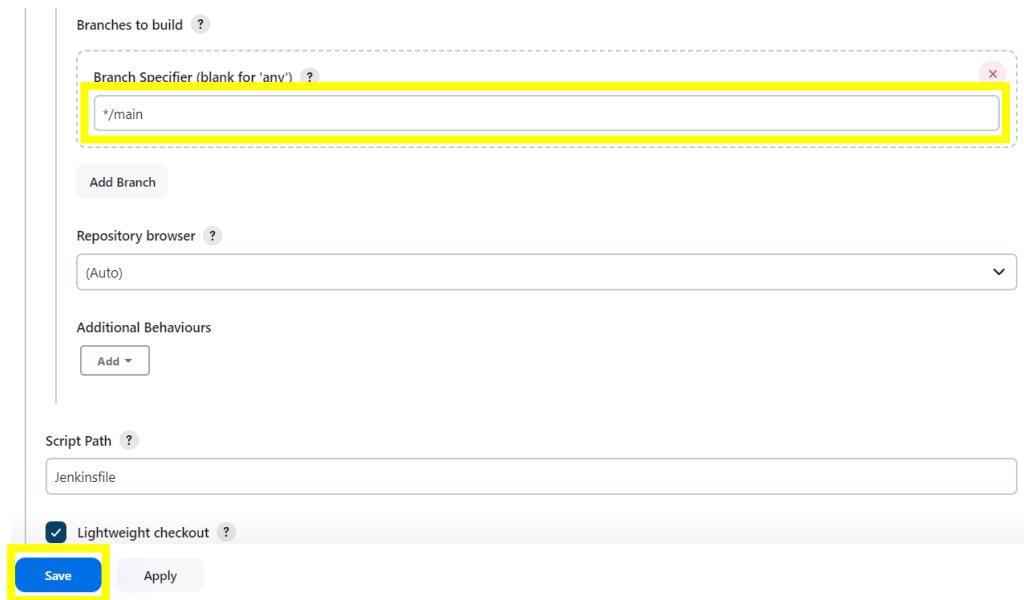
Setting up pipeline definition from my GitHub Repository.

Pipeline

The screenshot shows the 'Pipeline' configuration. Under the 'Definition' section, 'Pipeline script from SCM' is selected, highlighted with a yellow box. In the 'SCM' section, 'Git' is chosen, also highlighted with a yellow box. The 'Repositories' section contains a 'Repository URL' field with the value 'https://github.com/JenishRevaldo/jenkins-demo.git', which is highlighted with a yellow box. The 'Credentials' section shows a dropdown menu with '- none -'. At the bottom, there is an 'Advanced' dropdown menu.

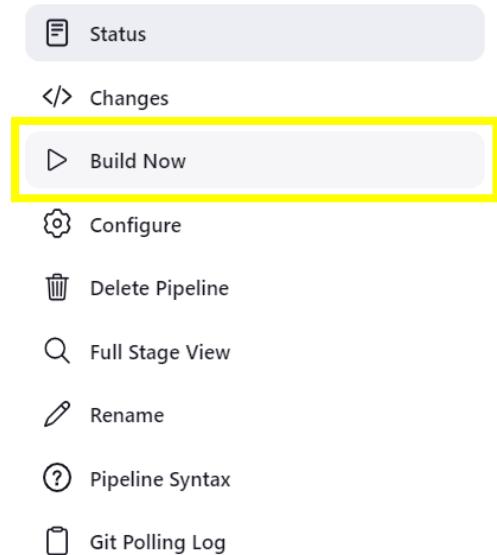
<next>

Change the branch specifier according to your repository and save the pipeline.



<next>

Now build the pipeline.



<next>

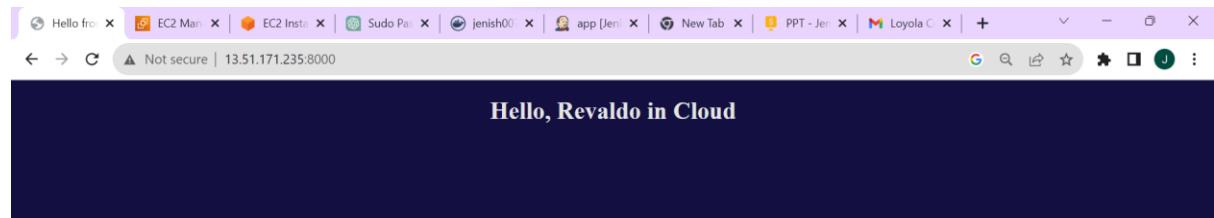
Stages of pipeline will get executed.

Stage View



Step 25: Now open <Public_IP_Address_of_EC2_Instance>:8000 in Browser

Output:



Now you can make changes in the **hello.html** file located in the templates folder in the repository and commit it, within one minute it gets reflect in the end page.

Result:

The local pipeline setup for Git, Jenkins, and Docker has been successfully established on the Ubuntu instance, enabling efficient software development, testing, and deployment processes.

Setting Up Jenkins on AWS EC2 Instance and Creating an Image

Aim:

To install Jenkins on an AWS EC2 instance, create an Amazon Machine Image (AMI) of the instance, launch a new EC2 instance from the AMI, and verify that Jenkins is installed on the new instance.

Procedure:

Step 1: Create an EC2 instance with Ubuntu in Amazon Machine Image.

Step 2: Update package information on the Ubuntu EC2 instance.

```
ubuntu@ip-172-31-30-107:~$ sudo apt-get update
```

```
ubuntu@ip-172-31-30-107:~$ sudo apt update
```

Step 3: Install OpenJDK 17 (Java Runtime Environment).

```
ubuntu@ip-172-31-30-107:~$ sudo apt install openjdk-17-jre
```

Step 4: Check the Java version to ensure it is installed correctly.

```
ubuntu@ip-172-31-30-107:~$ java -version  
openjdk version "17.0.7" 2023-04-18
```

Step 5: Import Jenkins Repository Key.

```
ubuntu@ip-172-31-30-107:~$ curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key |  
sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

Step 6: Add Jenkins Repository to Sources List.

```
ubuntu@ip-172-31-30-107:~$echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]  
https://pkg.jenkins.io/debian binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
```

Step 7: Update the package lists from all repositories, including the newly added Jenkins repository.

```
ubuntu@ip-172-31-30-107:~$ sudo apt-get update
```

Step 8: Install Jenkins.

```
ubuntu@ip-172-31-30-107:~$ sudo apt-get install jenkins
```

Step 9: Start Jenkins service and check its status.

```
ubuntu@ip-172-31-30-107:~$ sudo systemctl start jenkins.service
```

```
ubuntu@ip-172-31-30-107:~$ sudo systemctl status jenkins
```

• *jenkins.service - Jenkins Continuous Integration Server*

Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)

Active: active (running) since Sun 2023-09-24 12:50:18 UTC; 1min 57s ago

Main PID: 6680 (java)

Tasks: 38 (limit: 1111)

Memory: 292.6M

CPU: 1min 17.623s

CGroup: /system.slice/jenkins.service

*└─6680 /usr/bin/java -Djava.awt.headless=true -jar
/usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080*

Step 10: Once Jenkins is installed and configured, navigate to the EC2 Management Console and stop the EC2 instance.

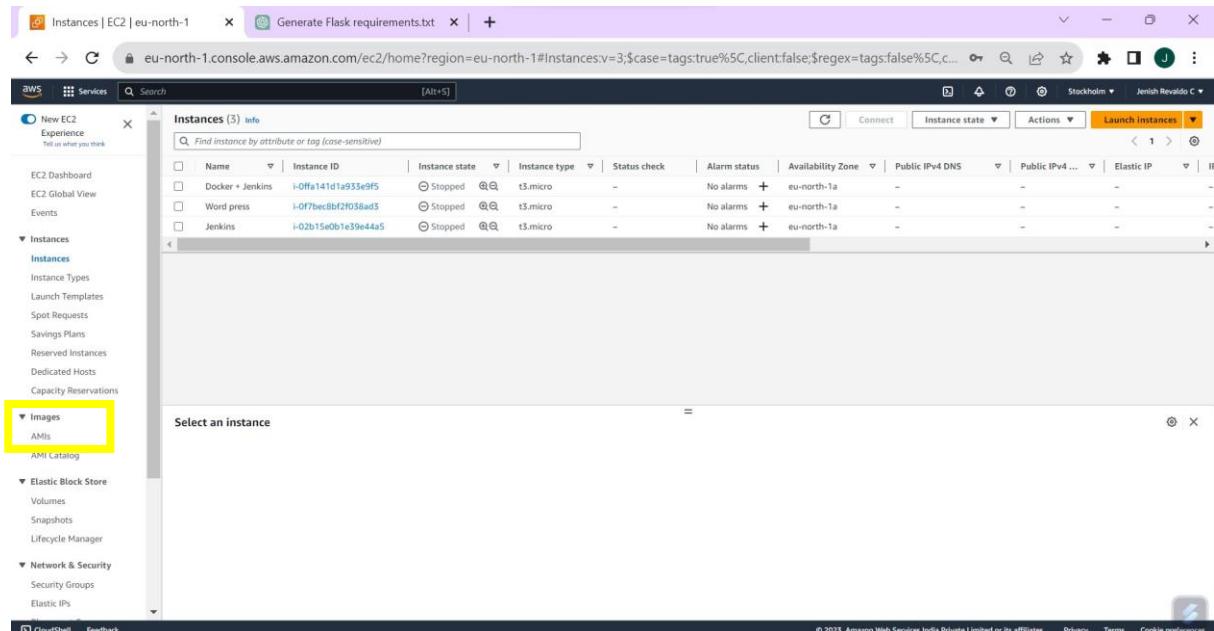
Step 11: Select your instance, right-click, and choose "Image" > "Create Image."

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Images, and Elastic Block Store. The main area displays three instances: Docker + Jenkins, Word press, and Jenkins. The Jenkins instance is selected and has a context menu open. The menu items include Launch instances, Launch instance from template, Migrate a server, Connect, Stop instance, Start instance, Reboot instance, Hibernate instance, Terminate instance, Instance settings, Networking, Security, and Create image. The 'Create image' option is highlighted with a yellow box.

Step 12: Provide a name and description for the image and create it.

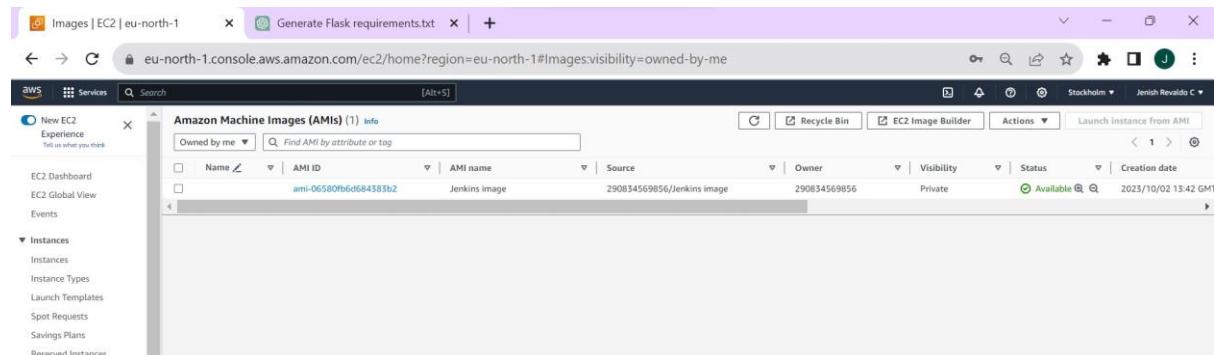
The screenshot shows the 'Create image' wizard. Step 1: Set instance details. It shows the instance ID i-02b15e0b1e39e44a5 (Jenkins). The 'Image name' field is filled with 'Jenkins image' and has a note: 'Maximum 127 characters. Can't be modified after creation.' The 'Image description - optional' field contains 'Image description'. Below these fields are checkboxes for 'No reboot' and 'Enable'. A section for 'Instance volumes' shows one EBS volume attached with a size of 8 GiB, IOPS of 100, throughput of 100 MiB/s, and encrypted status. An 'Add volume' button is available. A note says: 'During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.' A 'Tags - optional' section allows tagging both the image and snapshots together or separately. Finally, a 'Create image' button is highlighted with a yellow box.

Step 13: Navigate to Images > AMIs in the EC2 dashboard to see the images



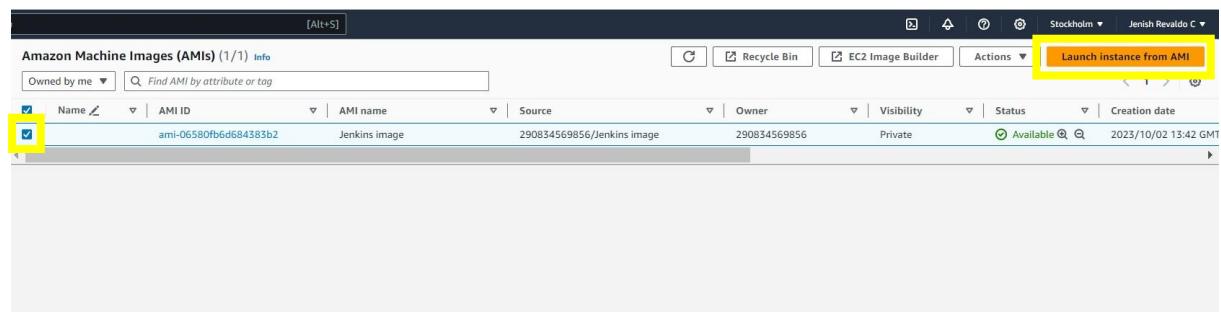
The screenshot shows the AWS EC2 Instances dashboard. On the left, a sidebar menu is open, showing 'Instances' and 'Images' under 'AMIs'. The 'Images' section is highlighted with a yellow box. In the main content area, there is a table titled 'Instances (3) Info' showing three stopped t3.micro instances: 'Docker + Jenkins', 'Word press', and 'Jenkins'. The table includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4, Elastic IP, and IF. At the bottom right of the main area, there is a 'Select an instance' button.

<after navigating>



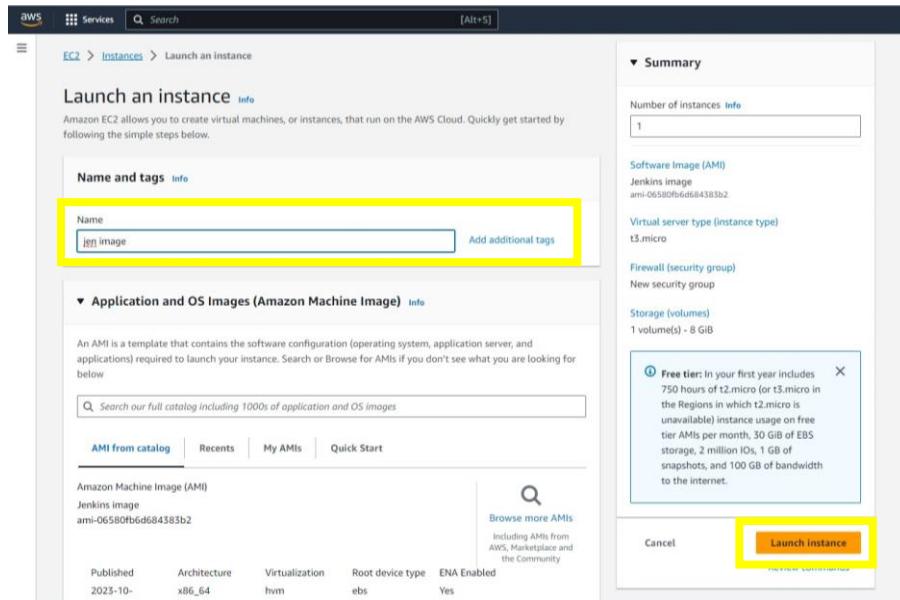
The screenshot shows the AWS EC2 Images dashboard. The left sidebar is collapsed. The main content area is titled 'Amazon Machine Images (AMIs) (1) Info' and shows a single AMI named 'Jenkins image' with the ID 'ami-06580fb6d684383b2'. The table includes columns for Name, AMI ID, AMI name, Source, Owner, Visibility, Status, and Creation date. The 'Status' column shows 'Available' with a green checkmark. At the top right, there are buttons for 'Recycle Bin', 'EC2 Image Builder', 'Actions', and 'Launch Instance from AMI'.

Step 14: To launch a new instance with the created image, select the image and click Launch instance from AMI



The screenshot shows the AWS EC2 Images dashboard, similar to the previous one but with a yellow box highlighting the 'Launch instance from AMI' button at the top right of the table header. The table below shows the same single AMI entry as before.

Step 15: Follow the instance launch wizard, giving name, choosing key pair, configuring security groups and other settings as needed and launch.



Step 16: Now navigate to the EC2 instances dashboard and you can see the newly created instance from image.

Instances (4) Info											
Find instance by attribute or tag (case-sensitive) Actions ▾											
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP	IF	
Docker + Jenkins	i-0ffa141d1a933e9f5	Stopped	t3.micro	-	No alarms	+ eu-north-1a	-	-	-	-	
Word press	i-0f7bec8bf2f058ad3	Stopped	t3.micro	-	No alarms	+ eu-north-1a	-	-	-	-	
Jenkins	i-02b15e0b1e39e44a5	Stopped	t3.micro	-	No alarms	+ eu-north-1a	-	-	-	-	
jen image	i-085f516e570f411b2	Running	t3.micro	Initializing	No alarms	+ eu-north-1a	ec2-51-20-83-139.eu-n...	51.20.83.139	-		

Step 17: Connect to the new EC2 instance using SSH. Check if Jenkins is installed and running.

```
root@ip-172-31-24-134:~# jenkins --version
```

2.424

```
root@ip-172-31-24-134:~# sudo systemctl status jenkins
```

- *jenkins.service - Jenkins Continuous Integration Server*

Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)

Active: active (running) since Mon 2023-10-02 08:22:25 UTC; 2min 50s ago

Main PID: 455 (java)

Tasks: 38 (limit: 1111)

Memory: 308.8M

CPU: 1min 6.098s

CGroup: /system.slice/jenkins.service

```
└─455 /usr/bin/java -Djava.awt.headless=true -jar  
/usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
```

Result:

Thus, set up Jenkins on an EC2 instance, created a reusable image, and launch new instances with Jenkins pre-installed, making it easier to scale and manage Jenkins-based infrastructure in the cloud.

Deploying MySQL on AWS, Creating Snapshots

Aim:

To install and configure MySQL on an AWS EC2 instance, create a snapshot of the database, perform various database operations, and then restore the database from the snapshot.

Procedure:

Step 1: Create an EC2 instance with Ubuntu in Amazon Machine Image

Step 2: Update package information on the Ubuntu EC2 instance.

```
ubuntu@ip-172-31-16-6:~$ sudo apt update
```

Step 3: Install MySQL server on the instance.

```
ubuntu@ip-172-31-16-6:~$ sudo apt install mysql-server
```

Step 4: Check the status of the MySQL service to ensure it's running.

```
ubuntu@ip-172-31-16-6:~$ sudo systemctl status mysql
```

```
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-10-02 09:05:35 UTC; 11s ago
       Process: 2901 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
      Main PID: 2909 (mysqld)
        Status: "Server is operational"
         Tasks: 38 (limit: 1111)
        Memory: 356.9M
           CPU: 1.052s
          CGroup: /system.slice/mysql.service
                  └─2909 /usr/sbin/mysqld
```

Step 5: Navigate to the EC2 Management Console and select your instance volume id.

The screenshot shows the AWS EC2 Instances page. The 'sql' instance is selected. In the Instance Details panel, the Volume ID 'vol-05a29da9b4a11bfb9' is highlighted with a yellow box. The table below shows the volume details:

Volume ID	Device name	Volume size (GiB)	Attachment status	Attachment time	Encrypted	KMS key ID	Delete
vol-05a29da9b4a11bfb9	/dev/sda1	8	Attached	2023/10/02 14:33 GMT+5:30	No	-	Yes

<navigates to volume dashboard>

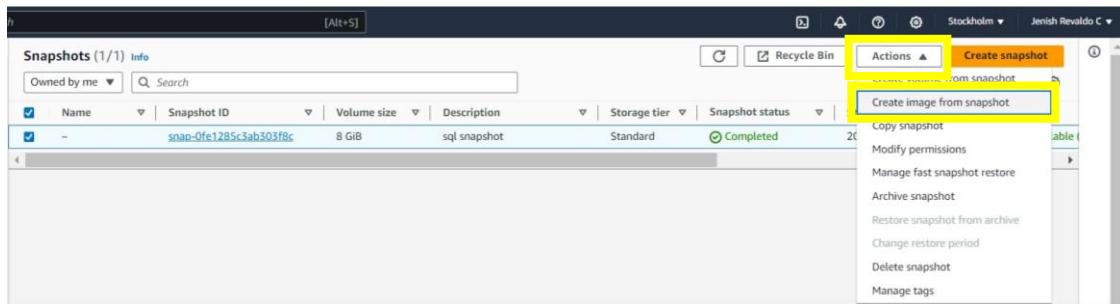
Step 6: Select the volume, Actions > Create snapshot

The screenshot shows the AWS Volumes page. The volume 'vol-05a29da9b4a11bfb9' is selected. The Actions menu is open, and 'Create snapshot' is highlighted with a yellow box. The menu also includes options like 'Create volume', 'Modify volume', 'Delete volume', 'Attach volume', 'Detach volume', 'Force detach volume', 'Manage auto-enabled I/O', 'Manage tags', and 'Fault injection'.

Step 7: Navigate to Elastic Block Store > Snapshots in the EC2 dashboard to see the snapshots.

The screenshot shows the AWS EC2 Dashboard. The 'Solutions' section is expanded, showing 'Images' and 'Elastic Block Store'. Under 'Elastic Block Store', 'Volumes' and 'Solutions' are listed, with 'Solutions' highlighted with a yellow box.

Step 8: Select the snapshot, Actions > Create image from snapshot.



Step 9: Give image name and create image.

EC2 > Snapshots > snap-0fe1285c3ab303f8c > Create image from snapshot

Create image from snapshot Info

Create a new image from a snapshot taken from the root device volume of an instance.

Image settings

Snapshot ID
snap-0fe1285c3ab303f8c

Image name
A descriptive name for the image.
 3 - 128 characters. Valid characters are a-z, A-Z, 0-9, spaces, and - _ . / () [] ' @.

Description
A description for the image.
 255 characters maximum

Block device mappings - optional Info

Provisioned IOPS SSD (io2) volumes with a size greater than 16 TiB, IOPS greater than 64,000, or IOPS:GiB ratio greater than 500:1 are supported only with instance types that support io2 Block Express.

Volume 1

Device type	Device name	Snapshot
Root	/dev/sda1	snap-0fe1285c3ab303f8c

Size (GiB)

Volume type

IOPS

Throughput (MB/s)

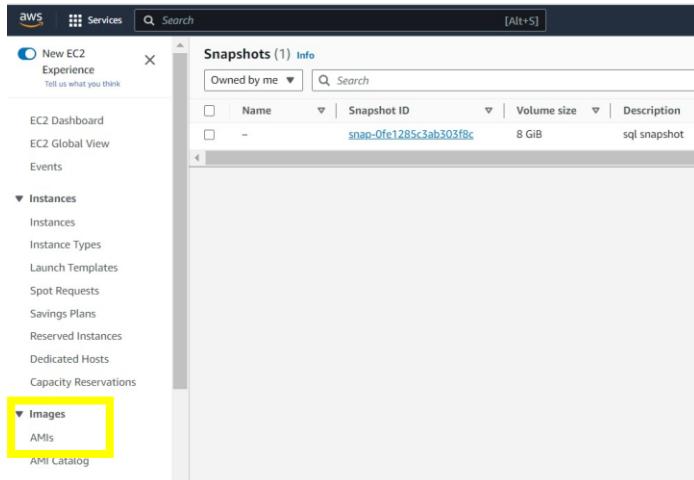
Termination behavior
 Delete on termination

Encryption
 Encrypt volume

Add volume

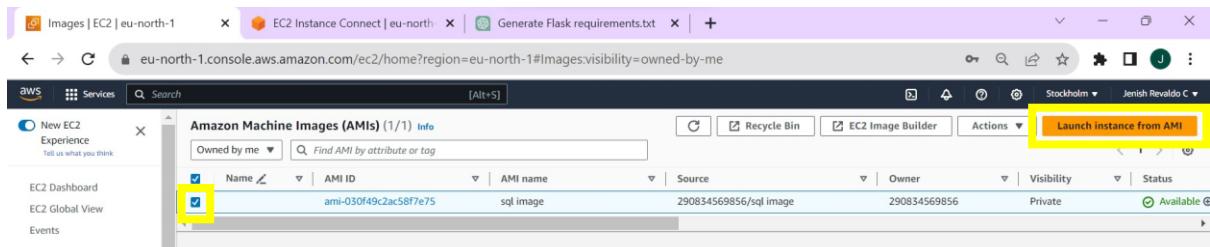
Create image

Step 10: Navigate to Images > AMIs in the EC2 dashboard to see the images

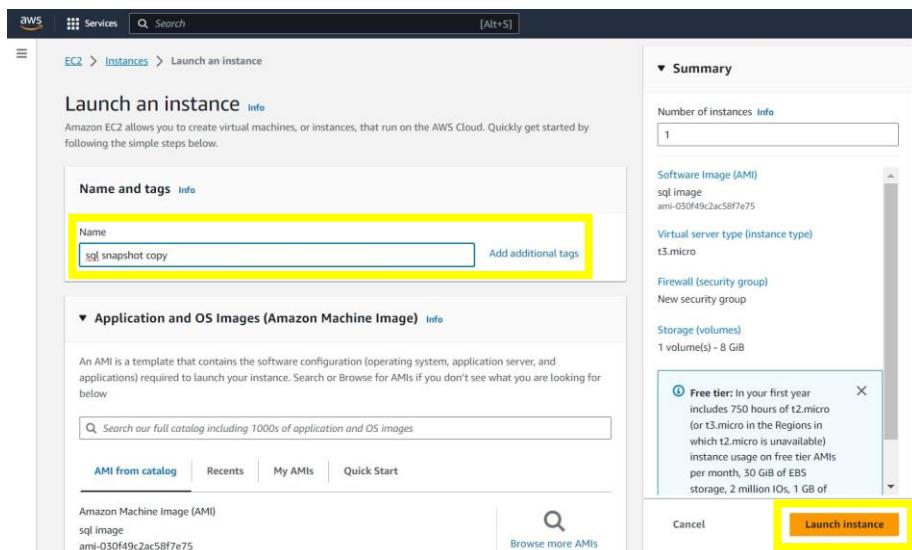


<After Navigating>

Step 11: To launch a new instance with the created image, select the image and click Launch instance from AMI



Step 12: Follow the instance launch wizard, giving name, choosing key pair, configuring security groups and other settings as needed and launch.



Step 13: Now navigate to the EC2 instances dashboard and you can see the newly created instance from image.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
Docker + Jenkins	i-0ffa141d1a933e9f5	Stopped	t3.micro	-	No alarms +	eu-north-1a	-	-
Word press	i-0f7bec8bf2f038ad3	Stopped	t3.micro	-	No alarms +	eu-north-1a	-	-
Jenkins	i-02b15e0b1e39e44a5	Terminated	t3.micro	-	No alarms +	eu-north-1a	-	-
jen image	i-085f516e570f411b2	Terminated	t3.micro	-	No alarms +	eu-north-1a	-	-
sql	i-08a4bd85dcabe54c0	Running	t3.micro	2/2 checks passed	No alarms +	eu-north-1a	ec2-51-20-35-167.eu-n...	51.20.35.16
sql snapshot c...	i-00d1aa2961010df7c	Running	t3.micro	Initializing	No alarms +	eu-north-1a	ec2-16-16-202-238.eu...	16.16.202...

Step 14: Connect to the new EC2 instance using SSH. Check if MySQL is installed and running.

```
ubuntu@ip-172-31-16-6:~$ sudo systemctl status mysql
```

- mysql.service - MySQL Community Server
 Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
 Active: active (running) since Mon 2023-10-02 09:26:21 UTC; 53s ago
 Process: 2901 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
 Main PID: 2909 (mysqld)
 Status: "Server is operational"
 Tasks: 38 (limit: 1111)
 Memory: 356.9M
 CPU: 1.052s
 CGroup: /system.slice/mysql.service
 └─2909 /usr/sbin/mysqld

Result:

Thus, set up MySQL on an EC2 instance, created a snapshot for backup and restored the database to another instance, providing data resilience and disaster recovery capabilities in the cloud.

Deploying WordPress Using Bitnami's Amazon Marketplace AMI and Creating a User Profile

Aim:

To deploy a WordPress instance on Amazon Web Services (AWS) using Bitnami's WordPress Amazon Machine Image (AMI) available on the AWS Marketplace and create a user profile on the WordPress website.

Procedure:

Step 1: In the AWS Management Console, navigate to the EC2 service, Click on "Launch Instance."

- In the "AWS Marketplace" section, search for "Bitnami WordPress."
- Select the Bitnami WordPress AMI from the list.

The screenshot shows the 'Launch an instance' wizard in the AWS Management Console. The 'Name and tags' section has a yellow box around the 'Name' field containing 'wordpress'. The 'Add additional tags' link is visible. The 'Application and OS Images (Amazon Machine Image)' section has a yellow box around the 'Search our full catalog including 1000s of application and OS images' input field. Below it, there are tabs for 'Recents' and 'Quick Start', and a grid of OS/AMI icons including Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. A yellow box highlights the 'Browse more AMIs' button and its description: 'Including AMIs from AWS, Marketplace and the Community'.

Choose an Amazon Machine Image (AMI)

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

The screenshot shows the AWS Lambda console interface. At the top, there is a search bar with the placeholder "word press". Below the search bar, there are three tabs: "Quickstart AMIs (0)", "My AMIs (0)", and "AWS Marketplace AMIs (62)". The "AWS Marketplace AMIs" tab is highlighted with a yellow box. On the left, there is a sidebar with "Refine results" sections for "Categories" (Business Applications, DevOps, Infrastructure Software, Industries) and "Publisher" (Plesk, cloudling, Supported Images). The main content area shows search results for "word press" with 62 results, including a preview of the "WordPress Certified by Bitnami and Automattic" AMI, which is also highlighted with a yellow box. A "Select" button is located at the bottom right of this preview card.

<now launch the instance>

The screenshot shows the "Launch an instance" page in the AWS EC2 Instances section. It includes fields for "Name and tags" (Name: "wordpress"), "Application and OS Images (Amazon Machine Image)" (Search bar: "word press", AMI from catalog: "bitnami-wordpress-6.3.1-13-r15-linux-debian-11-x86_64-hvm-ebs-nami-7d426cb7-9522-", Verified provider), and a summary section on the right. The summary section details the instance configuration: 1 instance, Software Image (AMI) "WordPress Certified by Bitnami...", Virtual server type (instance type) "t3.micro", Firewall (security group) "New security group", Storage (volumes) "1 volume(s) - 10 GiB", and a note about the Free tier. At the bottom right, there is a prominent "Launch instance" button highlighted with a yellow box.

Step 2: Open the public IP address of the created instance

The screenshot shows the "Instances (1) Info" page in the AWS EC2 Instances section. It displays a table with one row for the instance "Word press" (Instance ID: i-0f7bec8bf2f038ad3, Status: Running, Instance type: t3.micro, Public IPv4: ec2-13-53-170-213.eu...). The "Instance ID" column is highlighted with a yellow box. At the top right, there is a "Launch instances" button.

<next>

EC2 > Instances > i-0f7bec8bf2f038ad3

Instance summary for i-0f7bec8bf2f038ad3 (Word press) Info

Updated less than a minute ago

Instance ID i-0f7bec8bf2f038ad3 (Word press)	Public IPv4 address 13.53.170.213 [open address]	Private IPv4 addresses 172.31.23.85
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-13-53-170-213.eu-north-1.compute.amazonaws.com [open address]

Actions

<after opening the public Ip address>

The screenshot shows a web browser window with two tabs: "Instance details | EC2 | eu-north-1" and "User's blog". The "User's blog" tab is active, displaying the WordPress homepage. The URL in the address bar is "https://13.53.170.213". The page content includes the title "Mindblown: a blog about philosophy.", a featured image, and a post titled "Hello world!". The post content reads: "Welcome to WordPress. This is your first post. Edit or delete it, then start writing!".

Step 3: Move to the admin page by adding /wp-admin after the Ip address.

The screenshot shows a web browser window with two tabs: "Get system log | EC2 | eu-north-1" and "User's blog". The "User's blog" tab is active, showing the WordPress login page. The URL in the address bar is "https://13.53.128.10/wp-admin". The page content includes the WordPress logo and a login form with fields for "Username or Email Address" and "Password", and checkboxes for "Remember Me" and "Log In". Below the form are links for "Lost your password?" and "Go to User's blog".

<after adding, the page would be>

The screenshot shows the same WordPress login page as above, but with the URL "https://13.53.128.10/wp-admin" highlighted in yellow in the address bar. The rest of the page content is identical to the previous screenshot.

Step 4: Get username and password from System log (Actions > Monitor and troubleshoot > Get system log)

The screenshot shows the AWS EC2 Instances page. In the top right corner, the 'Actions' button is highlighted with a yellow box. Below it, the 'Get system log' option is also highlighted with a yellow box. The main table lists several EC2 instances, including 'Word press' which is currently running.

<you can see username and password here in system log>

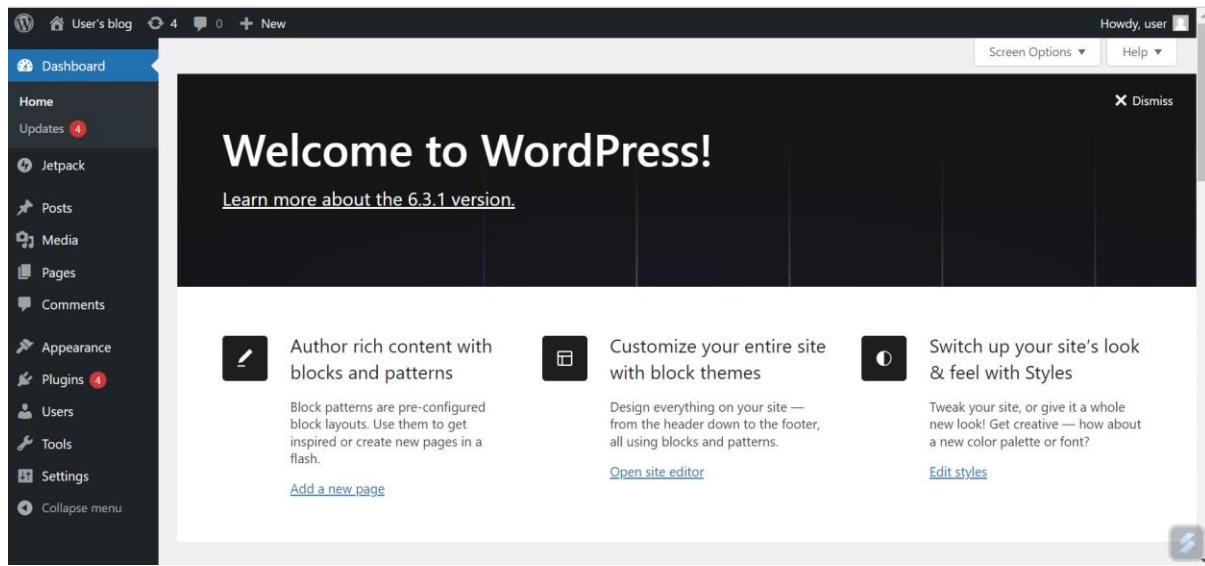
The screenshot shows the 'Get system log' details page for instance i-023bb22106d0b3eb7. The log output is displayed in a large text area. A yellow box highlights a specific section of the log where the Bitnami application password 'pss:7V=SGy/U' is printed. The log also contains other startup messages and configuration details.

```
[ 60.917661] bitnami[618]: 650000+0 records in
[ 60.933642] bitnami[618]: 650000+0 records out
[ 60.934780] bitnami[618]: 665600000 bytes (666 MB, 635 MiB) copied, 3.37504 s, 197 MB/s
[ 62.041607] bitnami[618]: Setting up swapspace version 1, size = 634.8 MiB (65595904 bytes)
[ 62.043817] bitnami[618]: no label, UUID=b2269c54-c2c2-49f8-9a24-0f68db4a64a4
[ 62.554841] Adding 64996k swap on /mnt/.bitnami.swap. Priority:-2 extents:26 across:305444k SSFS
[ 62.063205] bitnami[618]: ## 2023-10-02 10:27:45+00:00 ## INFO ## Running /opt/bitnami/var/init/pre-start/030_get_default_passwords...
[ 62.326108] bitnami[618]: #####
[ 62.328014] bitnami[618]: # #####
[ 62.329744] bitnami[618]: #      Setting Bitnami application password to 'pss:7V=SGy/U' #
[ 62.331495] bitnami[618]: #      (the default application username is 'user') #
[ 62.333050] bitnami[618]: # #####
[ 62.334527] bitnami[618]: #####
```

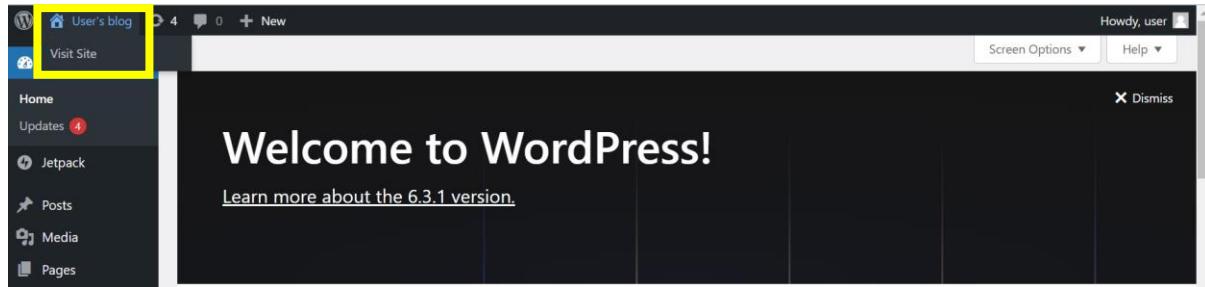
Step 5: Login using the credentials and it will navigate to the dashboard page.



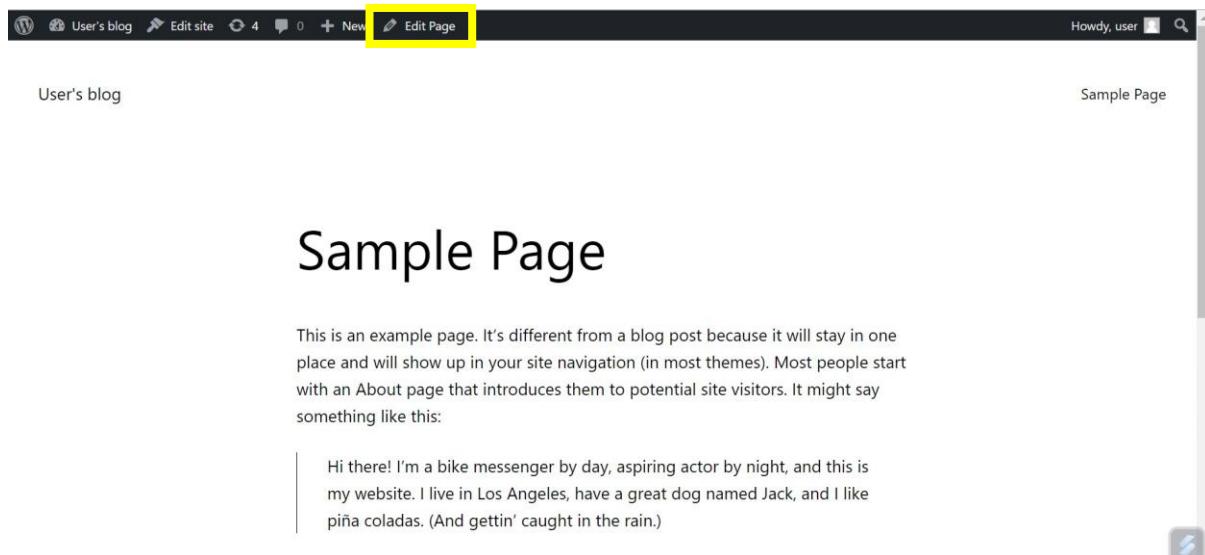
<Dashboard page>



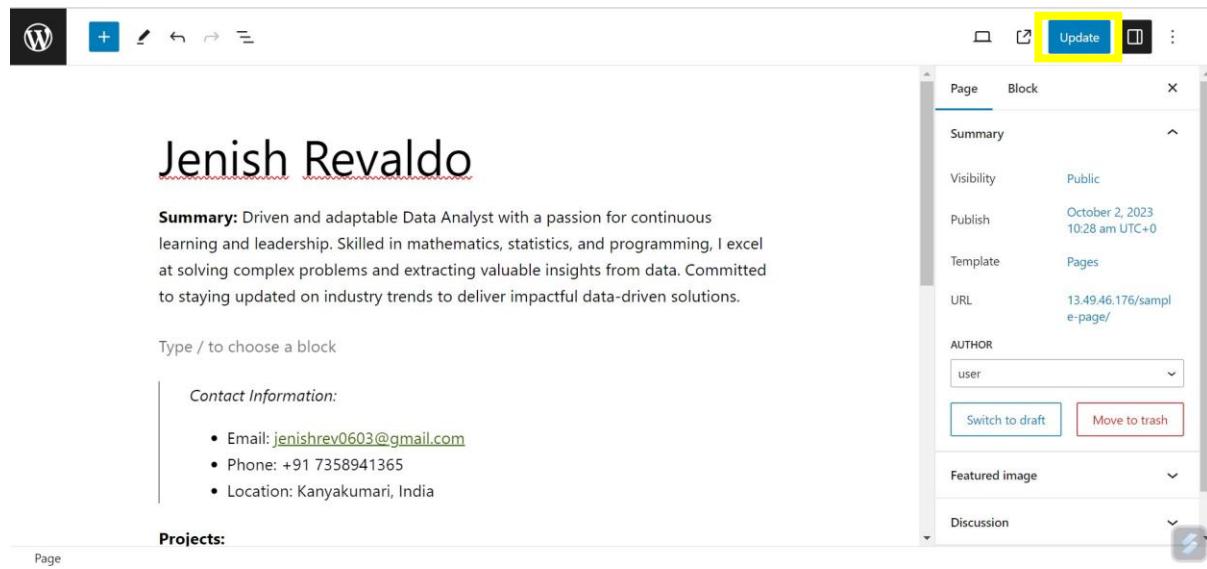
Step 6: Now you can edit the page by navigating to User's blob > Visit Site > Edit Page



<next>

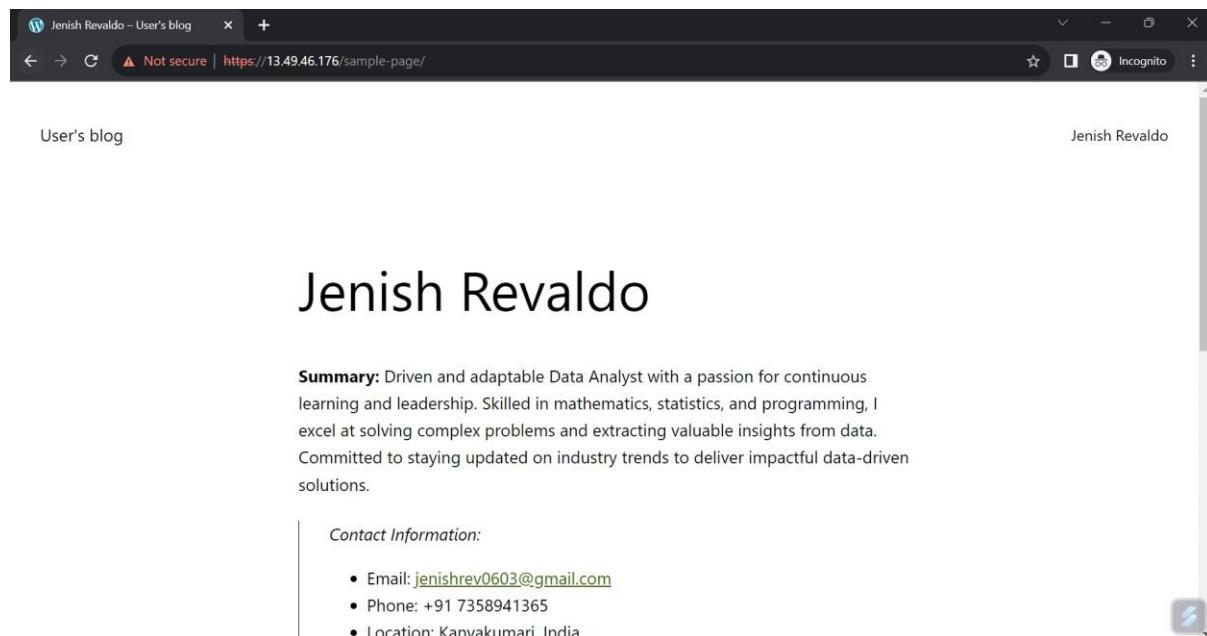


Step 7: Now you can start building your profile, after editing click on ‘Update’.



The screenshot shows the WordPress editor interface. A user profile page titled "Jenish Revaldo" is displayed. The editor's right sidebar is open, showing the "Summary" tab with details like Visibility (Public), Publish date (October 2, 2023, 10:28 am UTC+0), and URL (13.49.46.176/sample-page/). The "AUTHOR" section shows "user". There are buttons for "Switch to draft" and "Move to trash". Below the summary, there are sections for "Contact Information" (listing Email: jenishrev0603@gmail.com, Phone: +91 7358941365, Location: Kanyakumari, India) and "Projects". The top right of the editor has a toolbar with icons for document, list, and update, with the "Update" button highlighted by a yellow box.

Step 8: Now navigate to the public Ip address, your contents will be updated.



The screenshot shows a web browser window with the address bar displaying "Jenish Revaldo – User's blog" and the URL "https://13.49.46.176/sample-page/". The page content is identical to the one in the editor, featuring the title "Jenish Revaldo" and a "Summary" section with the same text as the editor. The browser's status bar shows "User's blog" on the left and "Jenish Revaldo" on the right. The top right of the browser window has standard control buttons for minimize, maximize, and close.

Result:

Thus, deployed a WordPress website using Bitnami's pre-configured AMI from the AWS Marketplace, created and updated a user profile on the WordPress platform.

Deploying a Website in Azure Web App

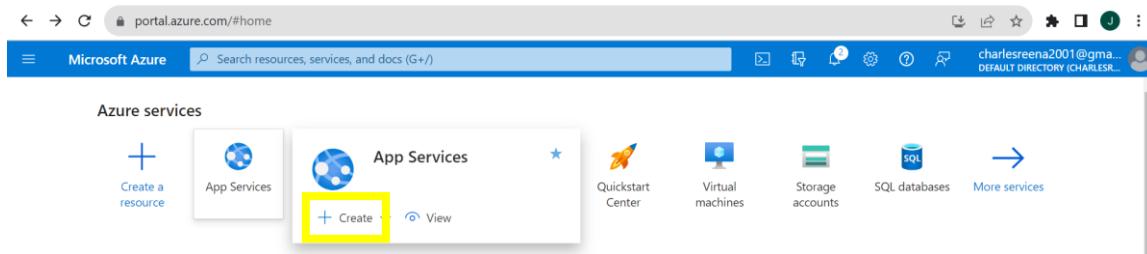
Aim:

To deploy a website to Azure Web App using the Azure CLI, ensuring that the website is accessible online.

Procedure:

Step 1: Create a Web App in Azure App Service

1. In the Azure Portal, navigate to the Azure App Service section.
2. Click on "Create" to create a new web app.



<next>

A screenshot of the 'App Services' blade in the Azure portal. The left sidebar has a tree view with 'Web App' selected and highlighted with a yellow box. Other items in the tree include 'Static Web App', 'Web App + Database', and 'WordPress on App Service'. The main content area shows various filtering and sorting options for the list of web apps. A yellow box highlights the 'Web App' item in the sidebar.

3. Provide a unique name for your web app. Choose your subscription, resource group, and App Service plan (or create a new one).
4. Select your runtime stack (e.g., .NET, Node.js, Python). Configure other settings like the operating system, region, and whether you want to enable or disable application insights.
5. Review your settings and click "Create" to provision the web app.

Microsoft Azure Search resources, services, and docs (G+/-)

Home > App Services >

Create Web App

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *
 Resource Group *

Instance Details

Need a database? [Try the new Web + Database experience.](#)

Name * .azurewebsites.net

Publish * Code Docker Container Static Web App

<next>

Runtime stack *

Operating System * Linux Windows

Region *
 Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Windows Plan (East US) *

Pricing plan **Free F1** (Shared infrastructure)

Zone redundancy

[Review + create](#) [< Previous](#) [Next : Deployment >](#)

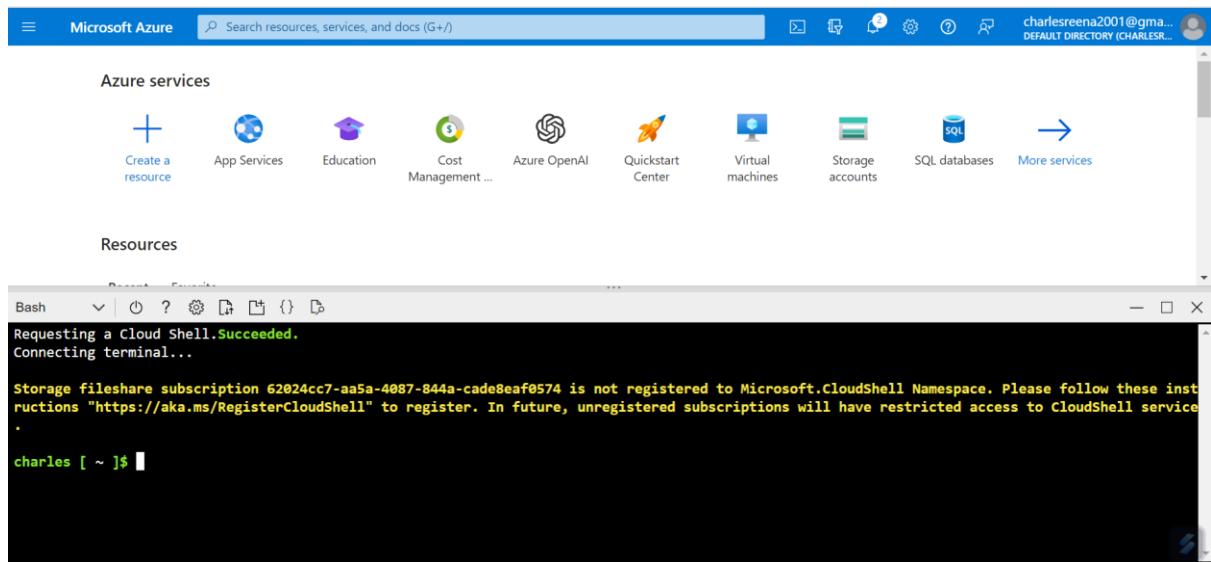
Step 2: Connect to the cloud shell using Bash

portal.azure.com/#home

Microsoft Azure Search resources, services, and docs (G+/-)

Azure services

<then it shows the cloud shell>



Step 3: Use the ‘**wget**’ command to download the website files from a source URL. In this example, we downloaded a zip file named "antique-cafe.zip."

```
charles [ ~ ]$ wget https://www.free-css.com/assets/files/free-css-  
templates/download/page295/antique-cafe.zip
```

Step 4: Use the ‘**unzip**’ command to extract the contents of the downloaded zip file. This will create a directory containing the website files.

```
charles [ ~ ]$ ls  
  
antique-cafe.zip azure-mol-samples-2nd-ed clouddrive
```

```
charles [ ~ ]$ unzip antique-cafe.zip
```

Step 5: Set your Git user email and name using the following commands

```
charles [ ~ ]$ git config --global user.email "your-email@example.com"  
  
charles [ ~ ]$ git config --global user.name "your-username"
```

Step 6: Change your working directory to the directory containing your website files

```
charles [ ~ ]$ ls
```

```
2126_antique_cafe antique-cafe.zip azure-mol-samples-2nd-ed clouddrive
```

```
charles [ ~ ]$ cd 2126_antique_cafe
```

```
charles [ ~/2126_antique_cafe ]$ ls -ltr
```

```
total 36
```

```
-rw-r--r-- 1 charles charles 510 Jul 30 2019 'ABOUT THIS TEMPLATE.txt'
```

```
drwxr-xr-x 2 charles charles 4096 Sep 29 2021 webfonts
```

```
drwxr-xr-x 2 charles charles 4096 Sep 29 2021 js
```

```
drwxr-xr-x 2 charles charles 4096 Sep 29 2021 img
```

```
-rw-r--r-- 1 charles charles 15522 Sep 30 2021 index.html
```

```
drwxr-xr-x 2 charles charles 4096 Sep 30 2021 css
```

Step 7: Initialize a Git Repository

```
charles [ ~/2126_antique_cafe ]$ git init
```

Step 8: Add and Commit Website Files

```
charles [ ~/2126_antique_cafe ]$ git add .
```

```
charles [ ~/2126_antique_cafe ]$ git commit -m "css!"
```

Step 9: Use the Azure CLI to configure deployment credentials for your Azure Web App.

```
charles [ ~/2126_antique_cafe ]$ az webapp deployment source config-local-git --name  
pizza-launchess --resource-group MOL-AppService
```

Step 10: Retrieve the publishing credentials for your Azure Web App using the Azure CLI.

```
charles [ ~/2126_antique_cafe ]$ az webapp deployment list-publishing-credentials --name pizza-launchess --resource-group MOL-AppService
```

```
{  
  "id": "/subscriptions/62024cc7-aa5a-4087-844a-  
cade8eaf0574/resourceGroups/MOL-AppService/providers/Microsoft.Web/sites/pizza-  
launchess/publishingcredentials/$pizza-launchess",  
  "kind": null,  
  "location": "East US",  
  "name": "pizza-launchess",  
  "publishingPassword":  
"6bBoiQrt2WlNrzWlaXG3bKjDSsqmDlc6QetjJPdoynCckh1W6nKhkCljjcZ3",  
  "publishingPasswordHash": null,  
  "publishingPasswordHashSalt": null,  
  "publishingUserName": "$pizza-launchess",  
  "resourceGroup": "MOL-AppService",  
  "scmUri": "https://$pizza-  
launchess:6bBoiQrt2WlNrzWlaXG3bKjDSsqmDlc6QetjJPdoynCckh1W6nKhkCljjcZ3  
@pizza-launchess.scm.azurewebsites.net",  
  "type": "Microsoft.Web/sites/publishingcredentials"  
}
```

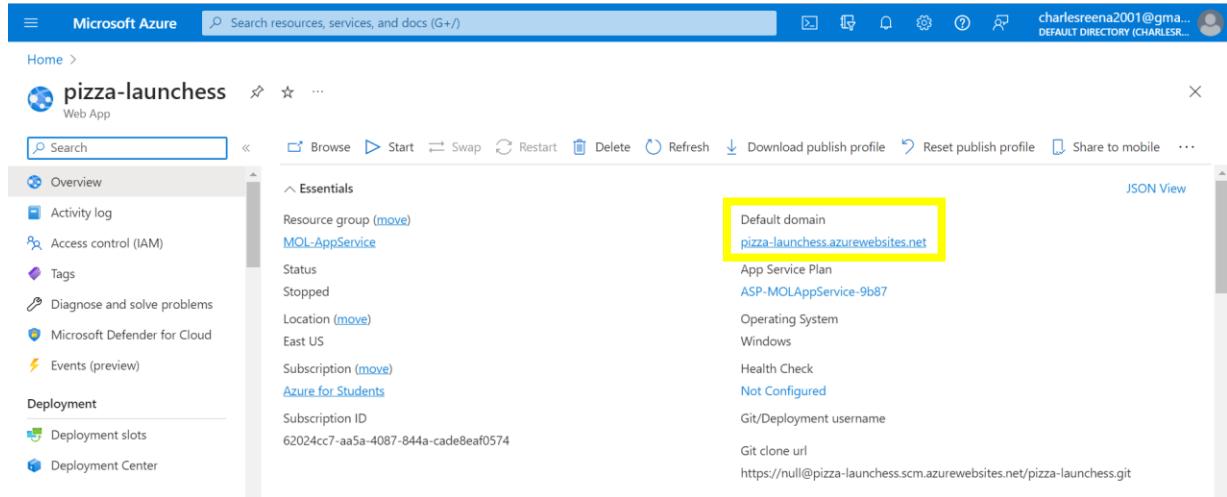
Step 11: Add the Azure Git repository as a remote to your local Git repository. Use the actual credentials you obtained in previous step.

```
charles [ ~/2126_antique_cafe ]$ git remote add azure 'https://$pizza-  
launchess:6bBoiQrt2WlNrzWlaXG3bKjDSsqmDlc6QetjJPdoynCckh1W6nKhkCljjcZ3@pizza-  
launchess.scm.azurewebsites.net'
```

Step 12: Push Website to Azure

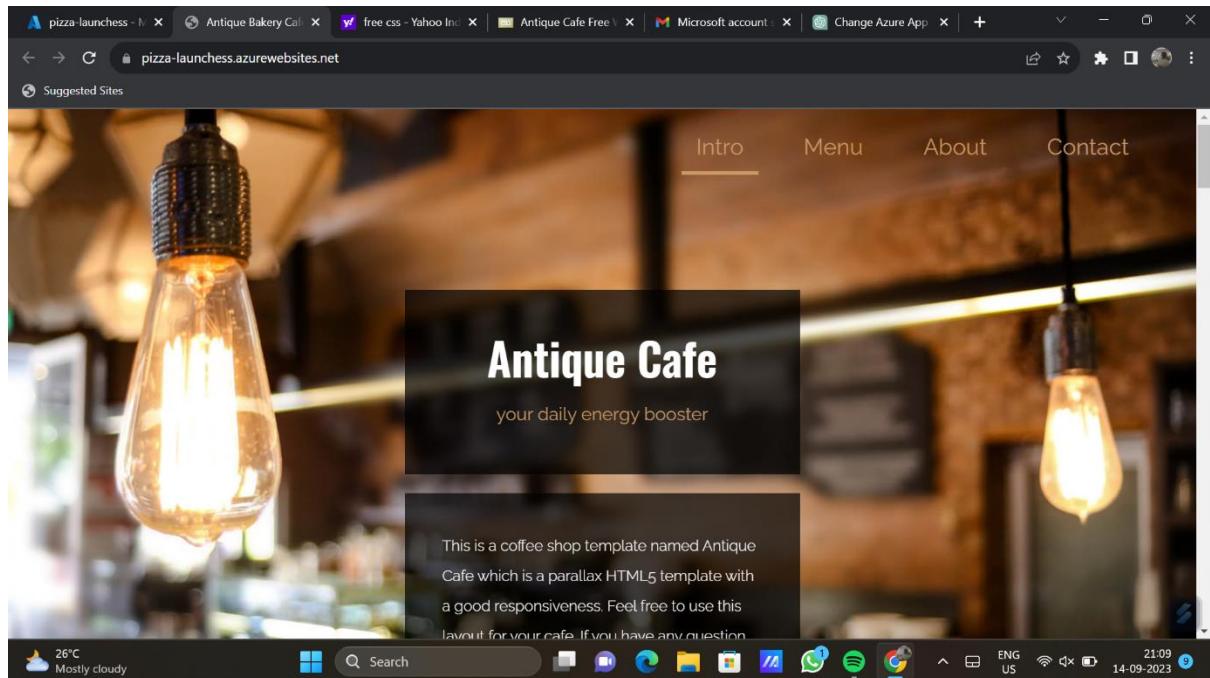
```
charles [ ~/2126_antique_cafe ]$ git push -f azure master
```

Step 13: Navigate to the default domain you obtained from Web App Overview



The screenshot shows the Microsoft Azure portal interface. In the top navigation bar, there's a search bar and a user profile icon. Below it, the URL 'pizza-launchess' is shown under 'Web App'. On the left, there's a sidebar with 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Microsoft Defender for Cloud', 'Events (preview)', 'Deployment slots', and 'Deployment Center'. The main content area is titled 'Essentials' and includes fields for 'Resource group (move)' (MOL-AppService), 'Status' (Stopped), 'Location (move)' (East US), 'Subscription (move)' (Azure for Students), 'Subscription ID' (62024cc7-aa5a-4087-844a-cade8eaf0574), 'App Service Plan' (ASP-MOLAppService-9b87), 'Operating System' (Windows), 'Health Check' (Not Configured), and 'Git/Deployment username' (https://null@pizza-launchess.scm.azurewebsites.net/pizza-launchess.git). A yellow box highlights the 'Default domain' field, which contains 'pizza-launchess.azurewebsites.net'.

Output:



Result:

Thus, our website is successfully deployed to Azure Web App, and it is accessible online via the provided URL.

Azure Kubernetes Service (AKS)

Aim:

To Create an Azure Kubernetes Service (AKS) cluster, connect to it, and deploy a sample application to the cluster.

Procedure:

Step 1: Creating a Kubernetes cluster

Create an AKS cluster using Azure portal.

Create resource → Containers → Azure Kubernetes Services → Create.

The screenshot shows the Microsoft Azure portal homepage. At the top, there's a navigation bar with the Microsoft Azure logo, a search bar, and various icons. Below the navigation bar, there's a section titled "Azure services" with a "Create a resource" button highlighted by a yellow box. Other service icons include Azure DevOps organizations, App Services, Education, Cost Management ..., Azure OpenAI, Quickstart Center, Virtual machines, Storage accounts, and a "More services" button. Below this, there's a "Resources" section with "Recent" and "Favorite" links.

<select containers>

[Home](#) >

Create a resource

[Get Started](#)

[Recently created](#)

Categories

[AI + Machine Learning](#)

[Analytics](#)

[Blockchain](#)

[Compute](#)

[Containers](#)

[Databases](#)

<In Azure Kubernetes Services, Select Create>

Create a resource ...

Get Started
Recently created

Popular Azure services See more in All services

Popular Marketplace products See more in Marketplace

Azure Kubernetes Service (AKS)
Create | Docs | MS Learn

Web App for Containers
Create | Docs | MS Learn

Batch Service
Create | Docs | MS Learn

NVIDIA GPU-Optimized VM
Create | Learn more

Windows Server 2022 Core Datacenter Minimal OS
Create | Learn more

Basic
Create | Learn more

Step 2: Fill out the settings and configuration

Basics tab:

1. Give a resource group and a cluster name
2. Select region as US East
3. Cluster preset configuration as Dev/Test Nodes tab:

Create Kubernetes cluster ...

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline.

[Learn more ↗](#)

Project details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Azure for Students

Resource group * ⓘ

MOL-AppService

[Create new](#)

Cluster details

Cluster preset configuration

Dev/Test

To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time.

[Learn more and compare presets](#)

Create Kubernetes cluster ...

Cluster preset configuration

Dev/Test

To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time.
[Learn more and compare presets](#)

Kubernetes cluster name * ⓘ

MyAKS

Region * ⓘ

(US) East US

Availability zones ⓘ

Zones 1,2,3

AKS pricing tier ⓘ

Free

Kubernetes version * ⓘ

1.26.6 (default)

Automatic upgrade ⓘ

Enabled with patch (recommended)

Choose between local accounts or Azure AD for authentication and Azure RBAC or Kubernetes RBAC for your authorization needs.

< Previous

Next : Node pools >

Review + create

<Networking tab>

1. Select Network configuration as kubenet

Create Kubernetes cluster ...

Network configuration ⓘ

kubenet

Best for smaller node pools. Each pod is assigned a logically different IP address from the subnet for simpler setup

Azure CNI

Best for larger node pools. Each node and pod is assigned a unique IP for advanced configurations

Bring your own virtual network ⓘ



DNS name prefix * ⓘ

MyAKS-dns

Network policy ⓘ

None

Allow all ingress and egress traffic to the pods

Calico

Open-source networking solution. Best for large-scale deployments with strict security requirements

Azure

Native networking solution. Best for simpler deployments with basic security and networking requirements

< Previous

Next : Integrations >

Review + create

<Integrations tab>

Changes are not necessary. Proceed as it is.

Create Kubernetes cluster

Basics Node pools Networking Integrations Advanced Tags Review + create

Connect your AKS cluster with additional services.

Microsoft Defender for Cloud
Microsoft Defender for Cloud provides unified security management and advanced threat protection across hybrid cloud workloads. [Learn more](#)

Enable basic plan for free

Azure Container Registry
Connect your cluster to an Azure Container Registry to enable seamless deployments from a private image registry.
[Learn more about Azure Container Registry](#)

Container registry

Azure Monitor
In addition to the CPU and memory metrics included in AKS by default, you can enable Container Insights for more comprehensive data on the overall performance and health of your cluster. Billing is based on data ingestion and retention

< Previous Next : Advanced > Review + create

<Advanced tab>

Changes are not necessary. Proceed as it is.

Create Kubernetes cluster

Basics Node pools Networking Integrations Advanced Tags Review + create

Enable secret store CSI driver

Infrastructure resource group [Edit](#)

< Previous Next : Tags > Review + create

<Tags tab>

Changes are not necessary. Proceed as it is.

Create Kubernetes cluster ...

Basics Node pools Networking Integrations Advanced **Tags** Review + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more about tags](#)

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

Name ⓘ	Value ⓘ	Resource
<input type="text"/>	:	<input type="text"/> 2 selected ▾

< Previous Next : Review + create > **Review + create**

<Review and create tab>

Click create after validation is passed

Microsoft Azure Search resources, services, and docs (G+/-)

Home > Create a resource >

Create Kubernetes cluster ...

Validation passed

Basics Node pools Networking Integrations Advanced Tags **Review + create**

Basics

Subscription	Azure for Students
Resource group	MOL-AppService
Region	East US
Kubernetes cluster name	MyAKS
Kubernetes version	1.26.6
Automatic upgrade	Patch

Node pools

< Previous Next > **Create** Download a template for automation

Step 3: Wait for the deployment to complete.

The screenshot shows the Azure portal interface for a deployment named "microsoft.aks-20231004223512". The main heading says "Deployment is in progress". Below it, under "Deployment details", there is a table with one row:

Resource	Type	Status	Operation details
InsightsActionGroupDep	Microsoft.Resources/d...	Created	Operation details

At the bottom, there are two buttons: "Give feedback" and "Tell us about your experience with deployment".

<after deployment, click on go to resources>

The screenshot shows the Azure portal interface for the same deployment, now marked as "Your deployment is complete". The deployment details show the name, start time, subscription, resource group, and correlation ID. Below the details, there are sections for "Deployment details" and "Next steps". Under "Next steps", there are four recommended actions: "Create a quick start application", "Create a Kubernetes deployment", "Integrate automatic deployments within your cluster", and "Connect to cluster". The "Go to resource" button is highlighted with a yellow box.

Step 4: Connect to Cloud shell using PowerShell

The screenshot shows the Azure portal interface for a Kubernetes service named "MyAKS". The "Overview" tab is selected. In the "Essentials" section, the following information is displayed:

Resource group	Kubernetes version
MOL-AppService	1.26.6
Status	API server address
Succeeded (Running)	myaks-dns-gd1w40js.hcp.eastus.azmk8s.io
Location	Network type (plugin)
East US	Kubenet
Subscription	Node pools
Azure for Students	1 node pool

Step 5: Connect to the AKS Cluster

1. Run the following command to connect to your AKS cluster

```
PS /home/charles> az aks get-credentials --resource-group MOL-AppService --name MyAKS
```

Merged "MyAKS" as current context in /home/charles/.kube/config

This command configures your local **kubectl** to use the AKS cluster.

2. Verify that you are connected to the AKS cluster (display information about the nodes in your AKS cluster).

```
PS /home/charles> kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
aks-agentpool-11990967-vmss000000	Ready	agent	7m22s	v1.26.6
aks-agentpool-11990967-vmss000001	Ready	agent	7m28s	v1.26.6

Step 6: Check for Existing Pods and Resources

1. Check for existing pods in the default namespace:

```
PS /home/charles> kubectl get pods
```

No resources found in default namespace.

2. Check for all resources (including services, deployments, and pods):

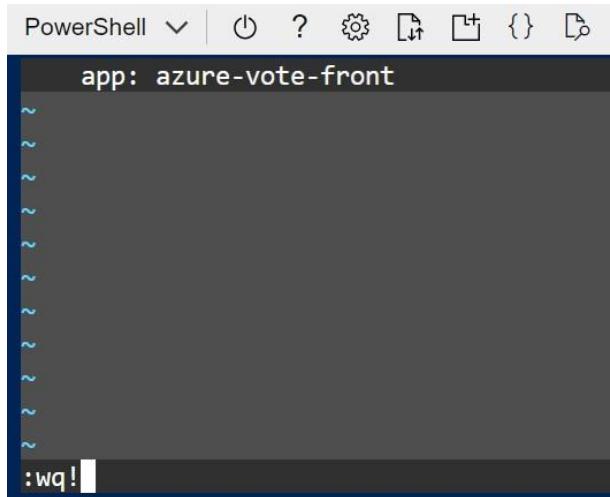
```
PS /home/charles> kubectl get all
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP	8m47s

Step 7: Create and Apply a Kubernetes Manifest

1. Create or edit a Kubernetes manifest file (in this case, **askapp.yml**):

```
PS /home/charles> vi askapp.yml
```



A screenshot of a terminal window titled "PowerShell". The status bar at the top shows "app: azure-vote-front". The main terminal area is dark gray and appears mostly empty, with a few small white marks. In the bottom right corner of the terminal window, there is a small white box containing the text ":wq!".

Paste the yml contents and Finally add :wq! and click enter.

Contents are obtained from <https://github.com/Azure-Samples/azure-voting-app-redis/blob/master/azure-vote-all-in-one-redis.yaml>

2. List the files in your current directory to ensure **askapp.yml** is present:

```
PS /home/charles> ls
```

*2126_antique_cafe antique-cafe.zip askapp.yml azure-mol-samples-2nd-ed
clouddrive Microsoft*

3. Apply the Kubernetes manifest to create the resources:

```
PS /home/charles> kubectl apply -f askapp.yml
```

deployment.apps/azure-vote-back created

service/azure-vote-back created

deployment.apps/azure-vote-front created

service/azure-vote-front created

Step 8: Check Deployments, Services, and Pods

1. Check the status of deployments:

```
PS /home/charles> kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
azure-vote-back	1/1	1	1	30s
azure-vote-front	1/1	1	1	30s

2. Check the status of services (This should display the services and their external IP addresses).

```
PS /home/charles> kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
azure-vote-back	ClusterIP	10.0.80.231	<none>	6379/TCP	75s
azure-vote-front	LoadBalancer	10.0.233.213	52.149.236.217	80:32717/TCP	74s
kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP	16m

3. Retrieve information about ReplicaSets

```
PS /home/charles> kubectl get rs
```

NAME	DESIRED	CURRENT	READY	AGE
azure-vote-back-66c88ccc8	1	1	1	2m18s
azure-vote-front-85dc674b97	1	1	1	2m18s

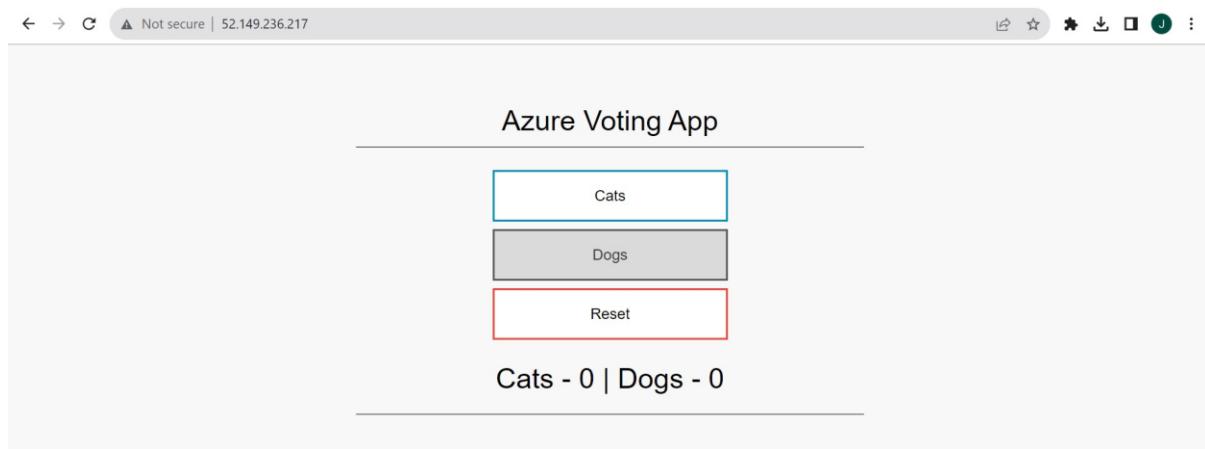
4. Verify that pods are running:

```
PS /home/charles> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
azure-vote-back-66c88ccc8-h572x	1/1	Running	0	2m33s
azure-vote-front-85dc674b97-c8zhh	1/1	Running	0	2m33s

Step 9: Paste the external IP address got in step 8.2

Output:



Result:

Thus, Created an Azure Kubernetes Service (AKS) cluster and deployed a sample application to the cluster.