

PROBLEM STATEMENT

OBJECTIVE:

Dockerize a simple Node.js app and deploy it on an AWS EC2 instance, this application will interact with a MySQL database hosted on AWS RDS, and set up a CI/CD pipeline with either AWS CodePipeline or Jenkins. Provision the infrastructure on AWS using terraform.

REQUIREMENTS:

- ➔ Dockerize a Simple Node.js App
- ➔ Set Up AWS Resources
- ➔ Deploy Node.js App on EC2
- ➔ CI/CD Pipeline
 - ❖ Option 1: AWS CodePipeline:
 - ❖ Option 2: Jenkins
- ➔ Monitoring
- ➔ Set Up above AWS Resources with Terraform

SUBMISSIONS DETAILS:

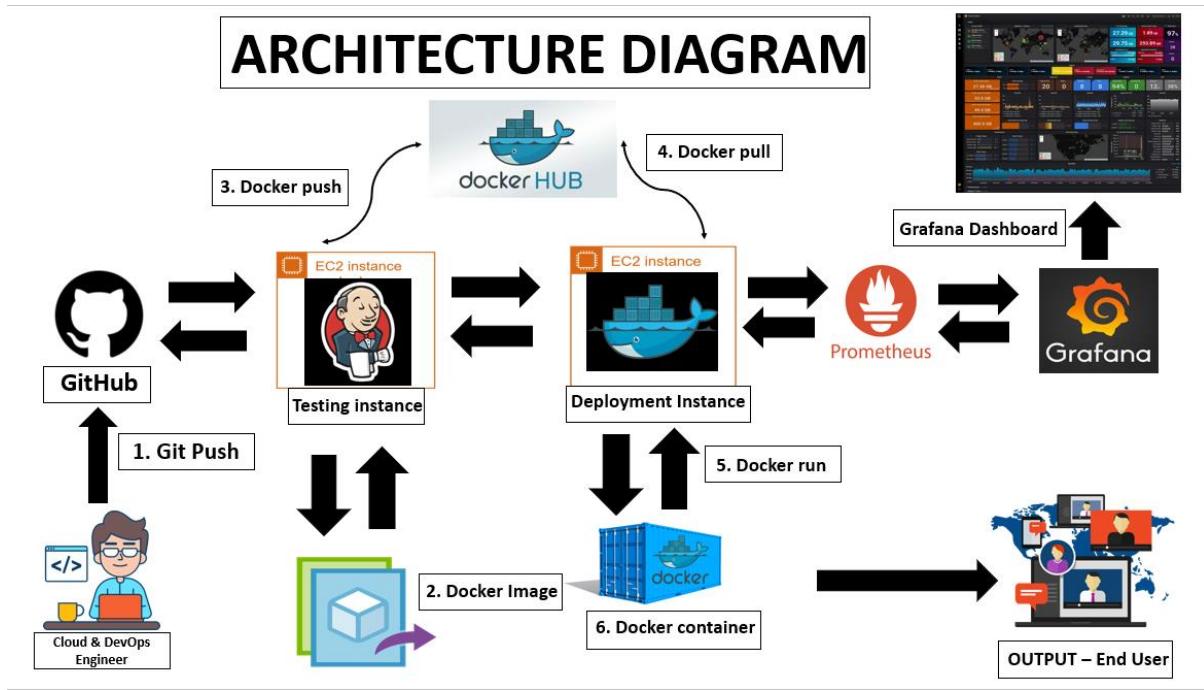
- ➔ Submit task in github repository link
- ➔ GitHub repo containing:
 - ❖ Application code.
 - ❖ Dockerfile for containerization.
 - ❖ Jenkinsfile for CI/CD pipeline.
 - ❖ Terraform template for infrastructure provisioning.
 - ❖ README file with setup and run instructions.
 - ❖ Brief overview of design decisions with workflow diagram.
 - ❖ Description of any challenges faced during implementation.

SOLUTION:

PRE-REQUIREMENTS:

Cloud	AWS
IAC	Terraform
Environment	Nodejs
Containerization	Docker
CI/CD	Jenkins
Monitoring	CloudWatch, Prometheus & Grafana
Version control	Git & GitHub

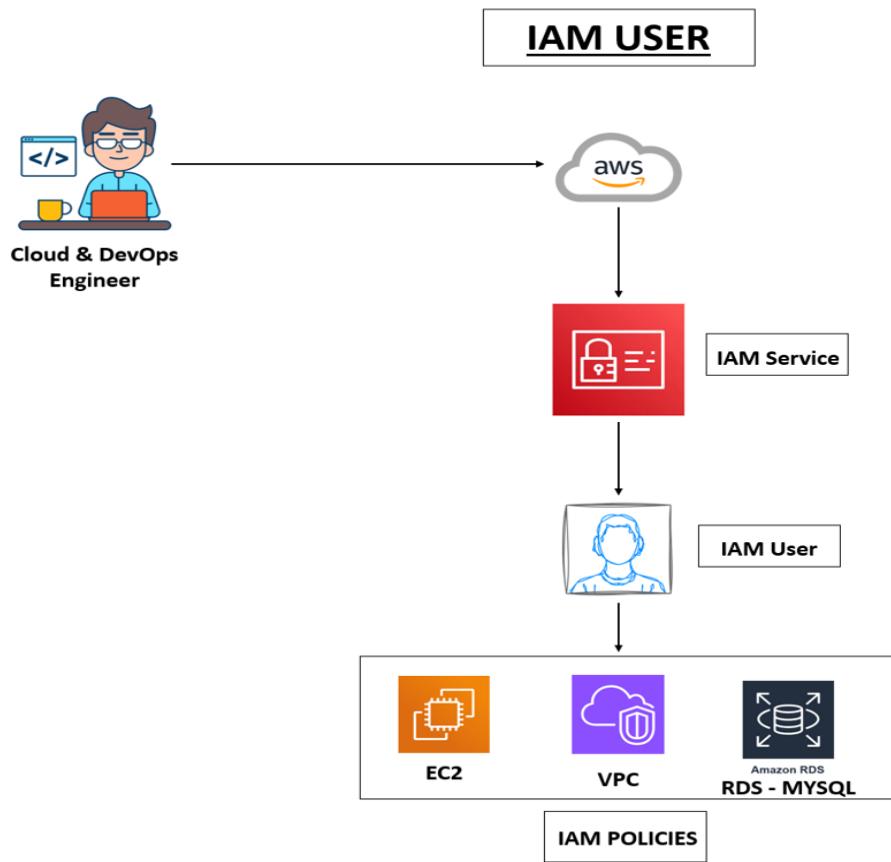
ARCHITECTURE DIAGRAM:



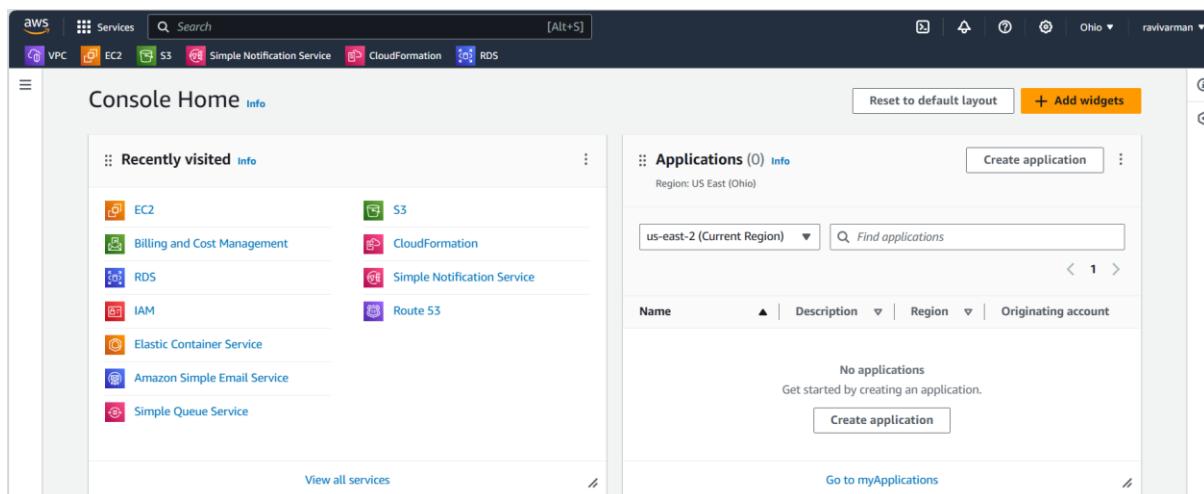
GitHub Repository: https://github.com/Ravivarman16/mysql-nodejs-application_deployment.git

STEP:1 – CREATING AN IAM USER FOR TERRAFORM:

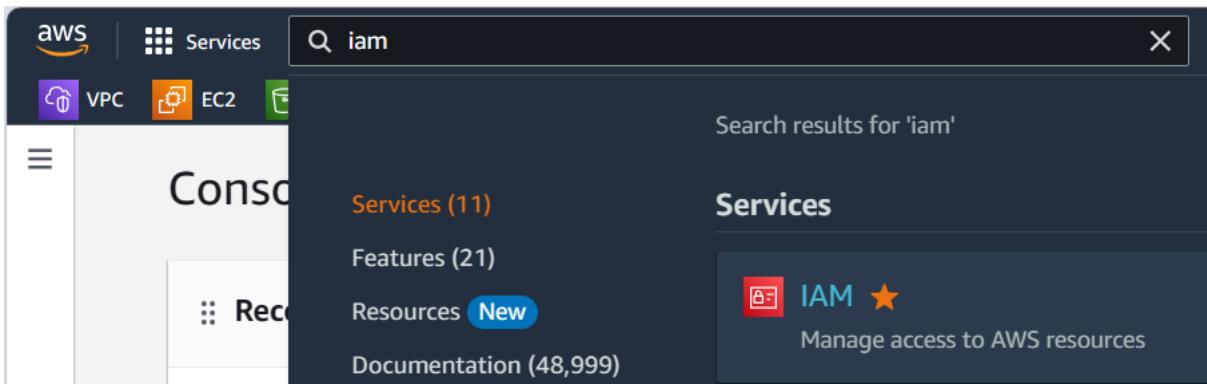
OVERVIEW DIAGRAM



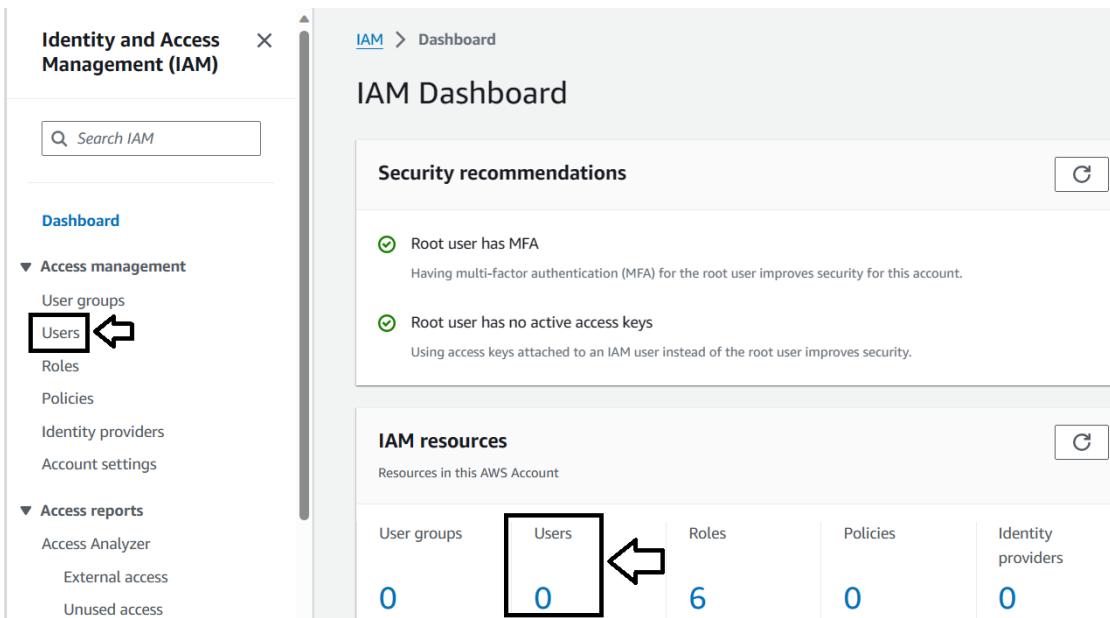
→ First, we need to login into our **AWS Management Console**:



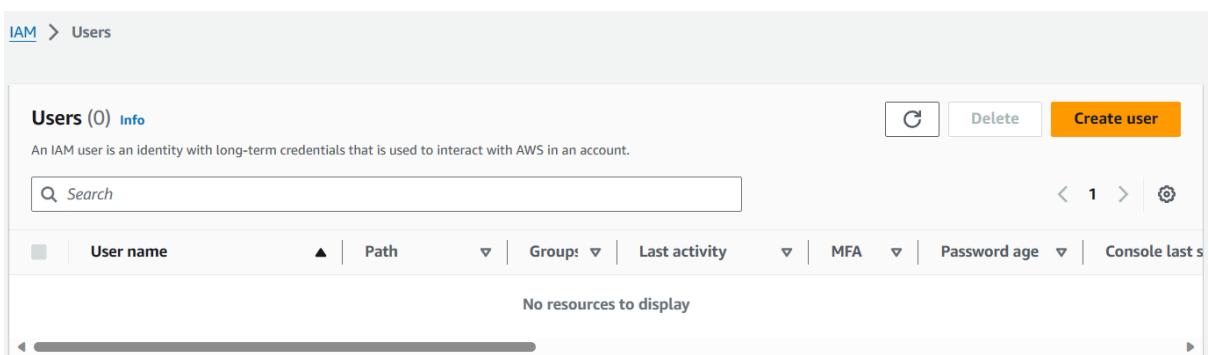
→ Then search IAM on service panel, click that one:



→ Then on IAM Management console: we need to create **IAM user** for that, **click users on left side of the console or click users at centre of the screen according to your wish:**



→ Then click **create user** option:



→ Then name IAM User as Terraform and click next:

Specify user details

User details

User name

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

[Learn more](#)

Cancel **Next**

→ Then select **attach policies directly** option to attach policies we required for this task:

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

→ Then we need to attach policies for this task, the required policies are

- ❖ **AmazonEC2FullAccess**
- ❖ **AmazonRDSFullAccess**
- ❖ **AmazonVPCFullAccess**

Attach these policies, click next:

Permissions policies (1168)

Choose one or more policies to attach to your new user

Policy name [Edit](#)

[AmazonEC2FullAccess](#)

Permissions policies (1/1168)

Choose one or more policies to attach to your new user

Policy name [Edit](#)

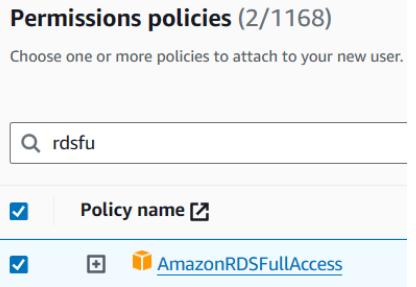
[AmazonVPCFullAccess](#)

Permissions policies (2/1168)

Choose one or more policies to attach to your new user.

Policy name

 [AmazonRDSFullAccess](#)

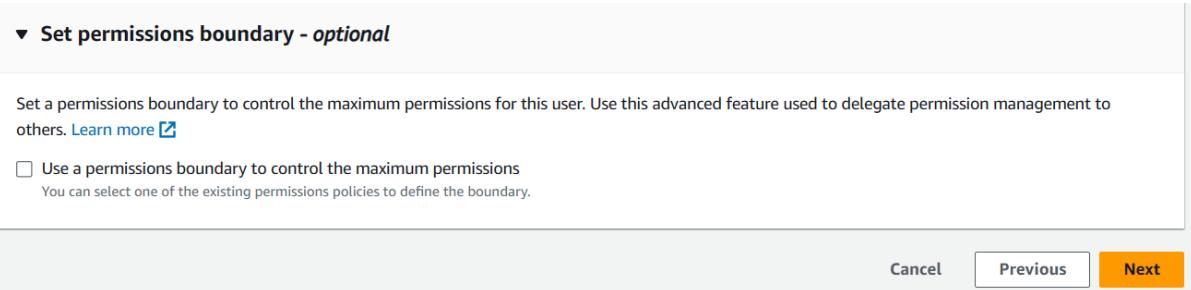


▼ Set permissions boundary - optional

Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. [Learn more](#)

Use a permissions boundary to control the maximum permissions
You can select one of the existing permissions policies to define the boundary.

Cancel Previous Next



→ After that just **review and create** page will appear, just review the configurations, click **create user**:

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name Terraform	Console password type None	Require password reset No
------------------------	-------------------------------	------------------------------

Permissions summary

Name	Type	Used as
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonRDSFullAccess	AWS managed	Permissions policy
AmazonVPCFullAccess	AWS managed	Permissions policy

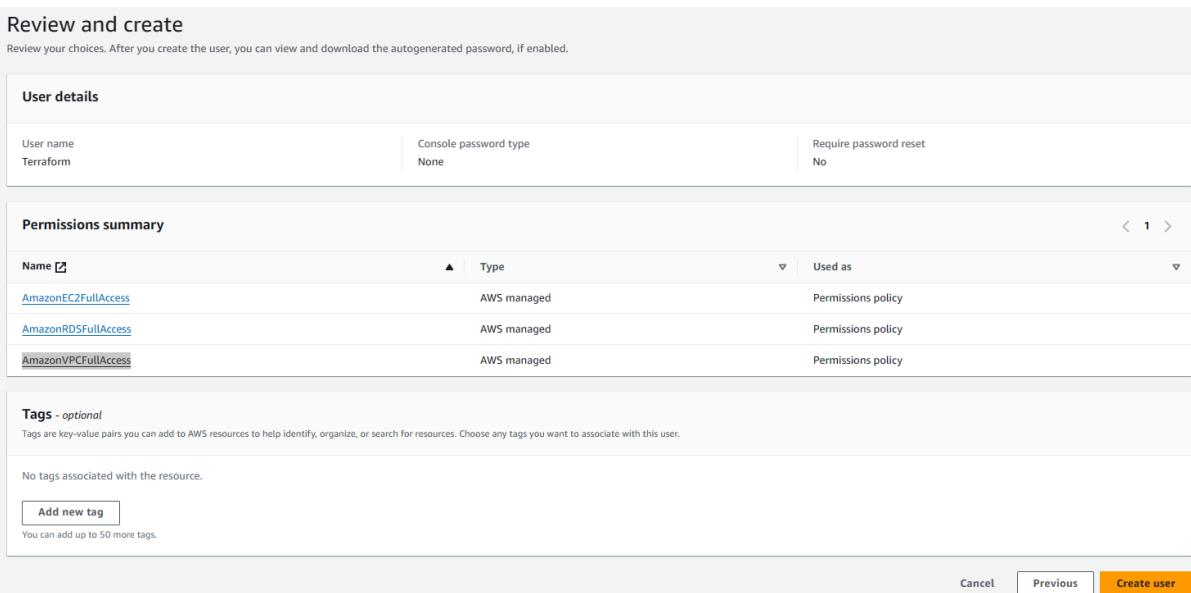
Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

You can add up to 50 more tags.

Cancel Previous Create user



→ The user has been created successfully:

⌚ User created successfully

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

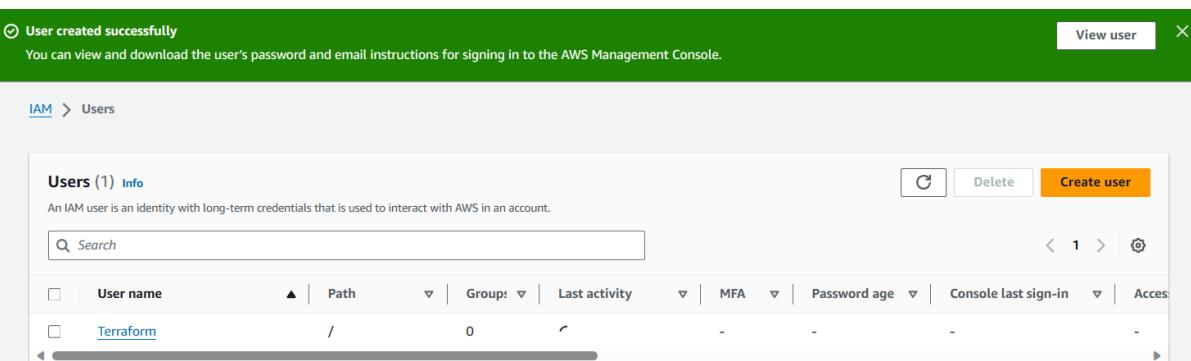
[View user](#) ×

[IAM](#) > [Users](#)

Users (1) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access keys
Terraform	/	0	-	-	-	-	-



→ Then click the **created IAM user**, click **Create access key** option to **create access and secret key** for above user:

The screenshot shows the AWS IAM Users page. A user named "Terraform" has been created. The "Summary" section displays the ARN (arn:aws:iام:619355063360:user/Terraform), which is disabled for console access. There is one access key listed, with a "Create access key" button available.

ARN	Console access	Access key 1
arn:aws:iام:619355063360:user/Terraform	Disabled	Create access key

Created: December 23, 2023, 13:23 (UTC+05:30)

Last console sign-in: -

→ Then click the use case as **Command line interface (CLI)**: Just **acknowledge the confirmation, click next**:

This step in the wizard asks for the use case for the access key. The "Command Line Interface (CLI)" option is selected, with a note explaining it enables the AWS CLI to access the account.

Use case

Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

Alternatives recommended

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation

I understand the above recommendation and want to proceed to create an access key.

[Cancel](#) [Next](#)

→ Then just click **create access key**:

This step allows setting a description tag for the access key. A text input field is provided for this purpose.

Description tag value
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

[Cancel](#) [Previous](#) [Create access key](#)

→ The access and secret key have been created successfully:

Retrieve access keys Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
------------	-------------------

AKIAZANDQIRAPG7GCZ7D	***** Show
----------------------	----------------------------

Access key best practices

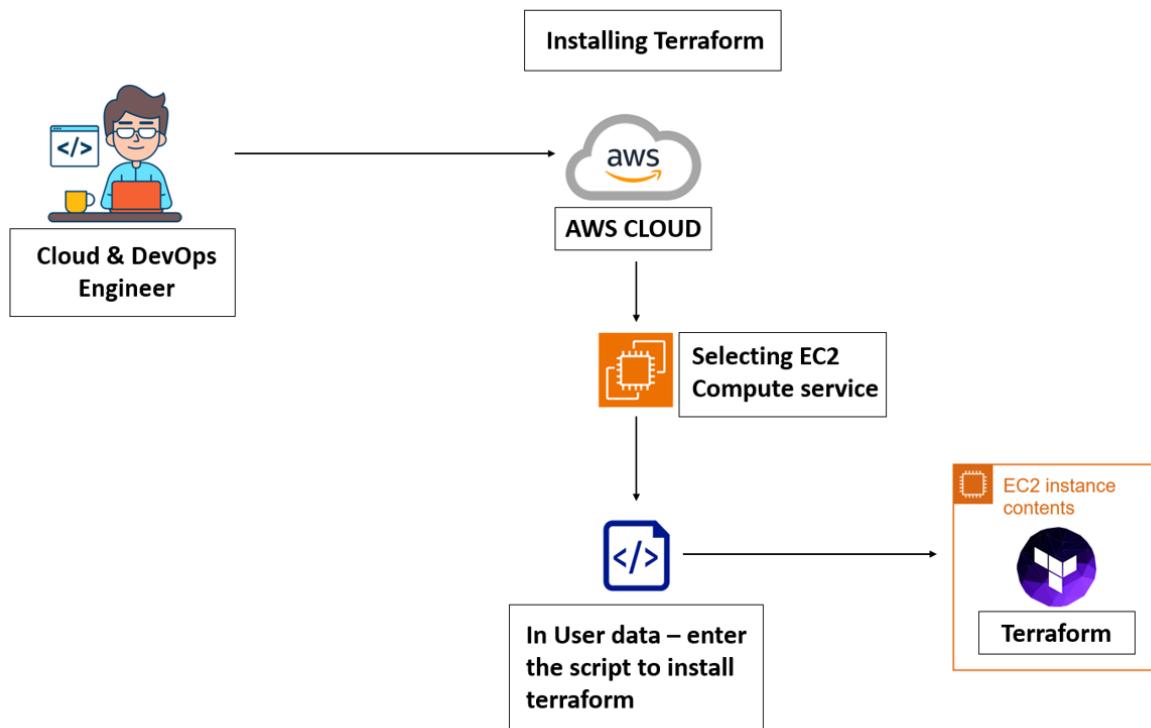
- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

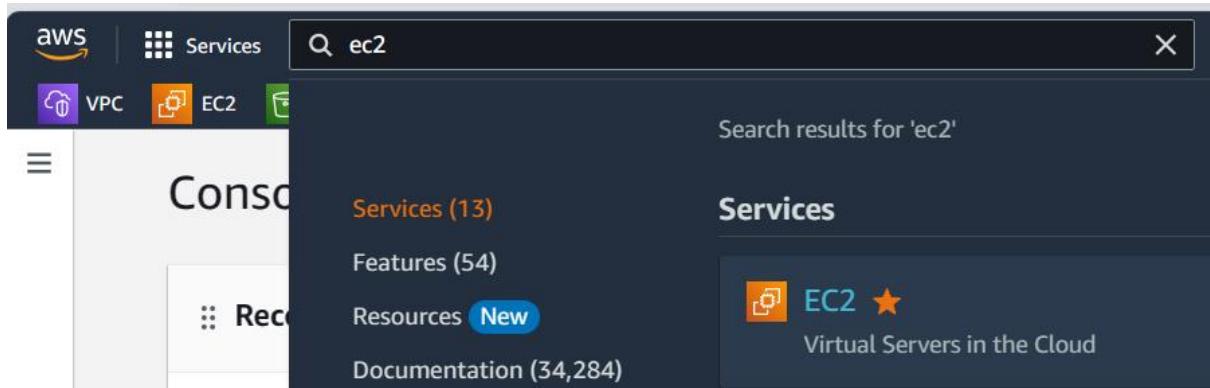
[Download .csv file](#)

[Done](#)

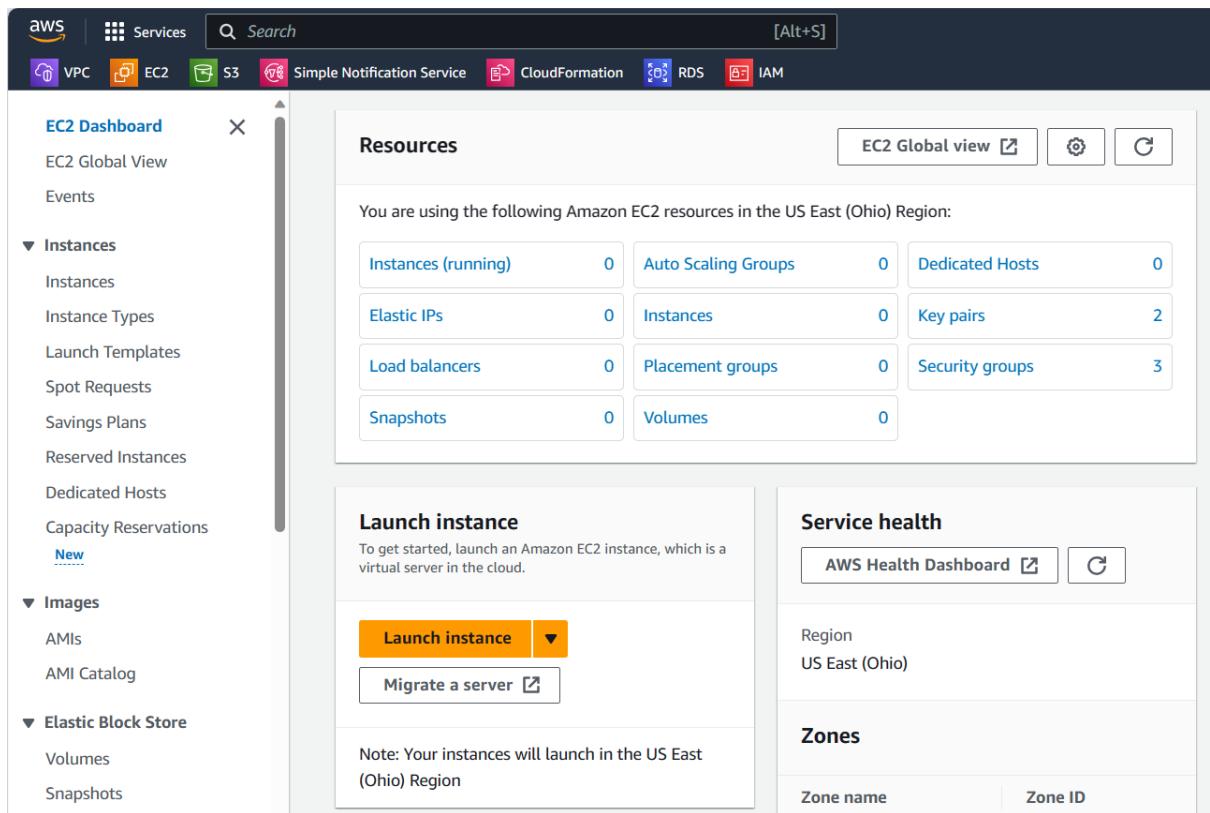
STEP:2 – CREATING AN INSTANCE FOR TERRAFORM: OVERVIEW – DIAGRAM



→ Search EC2 on service panel, click that one:



→ Then on EC2 Management Console, just click **launch instance** option:



→ Then name the instance according to your wish:

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines following the simple steps below.

Name and tags [Info](#)

Name

Terraform

→ Then select the **operating system** according to your wish, here I am selecting **ubuntu** as operating system:

Quick Start



Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type ami-05fb0b8c1424f266b (64-bit (x86)) / ami-0748d13ffbc370c2b (64-bit (Arm)) Virtualization: hvm ENA enabled: true Root device type: ebs	Free tier eligible
--	--------------------

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-12-07

Architecture

AMI ID

ami-05fb0b8c1424f266b

Verified provider

→ Then select the **instance type** as **t2.micro**

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0116 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand RHEL base pricing: 0.0716 USD per Hour

All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

➔ Then select the **keypair**:

▼ 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*

ravi2

 [Create new key pair](#)

➔ Then under the **network settings**, select the **subnet** according to your wish:

VPC - *required* [Info](#)

vpc-0a095731c8715a9df
172.31.0.0/16

(default)

Subnet [Info](#)

subnet-0eb1dcebaaca4e0c7
VPC: vpc-0a095731c8715a9df Owner: 619355063360 Availability Zone: us-east-2a
IP addresses available: 4091 CIDR: 172.31.0.0/20

Auto-assign public IP [Info](#)

Enable

➔ Then name the security group as **terraform-sc**:

Security group name - *required*

Terraform-sc

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-:/@#=;&[]!\$*

Description - *required* [Info](#)

Terraform-sc

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

[Remove](#)

Type [Info](#)

ssh

Protocol [Info](#)

TCP

Port range [Info](#)

22

Source type [Info](#)

Anywhere

Source [Info](#)

Add CIDR, prefix list or security

Description - *optional* [Info](#)

e.g. SSH for admin desktop

0.0.0.0/0 

➔ Then keep the default storage as it is:

▼ Configure storage [Info](#) [Advanced](#)

1x GiB [▼](#) Root volume (Not encrypted)

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

[Add new volume](#)

→ Then under advanced settings, user data section, enter the terraform script to install **terraform & awscli**. Click launch instance:

Script contains:

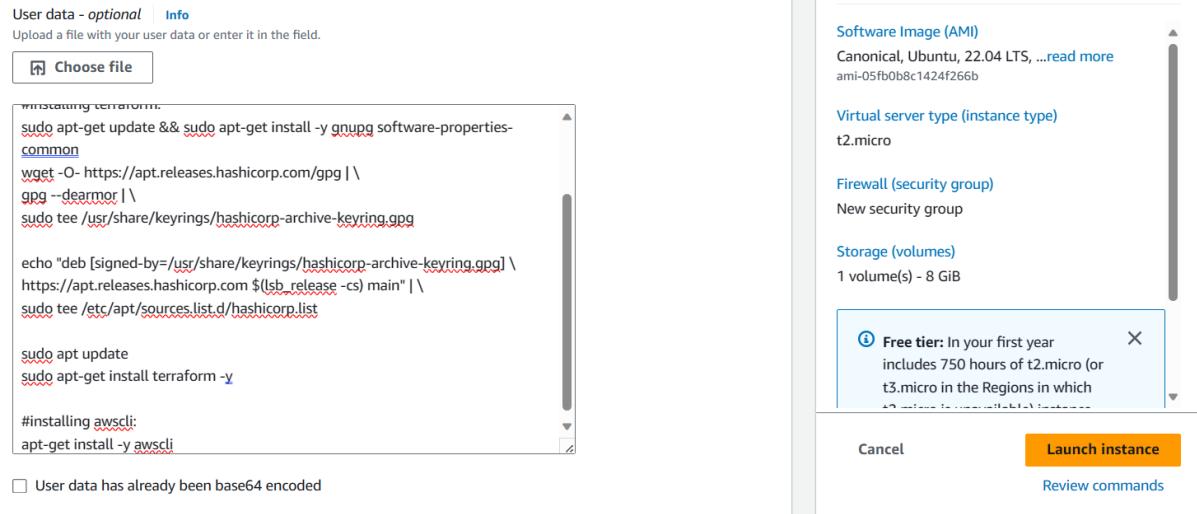
```
#!/bin/bash
#updating the os:
apt-get update

#installing terraform:
sudo apt-get update && sudo apt-get install -y gnupg
software-properties-common
wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-
keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main"
| \
sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update
sudo apt-get install terraform -y

#installing awscli:
apt-get install -y awscli
```



➔ The instance has launched successfully:

The screenshot shows the AWS EC2 Instances page. A green success message box displays: "Success" and "Successfully initiated launch of instance (i-0f44c8bc5fbcbc5cf)". Below it, a table lists one instance:

Instances (1)		Info	<input type="button" value="C"/>	<input type="button" value="Connect"/>	Instance state
	Name	Instance ID	Instance state	Instance type	Status check
<input type="checkbox"/>	Terraform	i-0f44c8bc5fbcbc5cf	Running	t2.micro	Initializing

➔ Then connect the instance and verify whether **terraform & awscli** is installed or not by using command:

```
terraform --version
aws --version
```

The screenshot shows a terminal window with the following output:

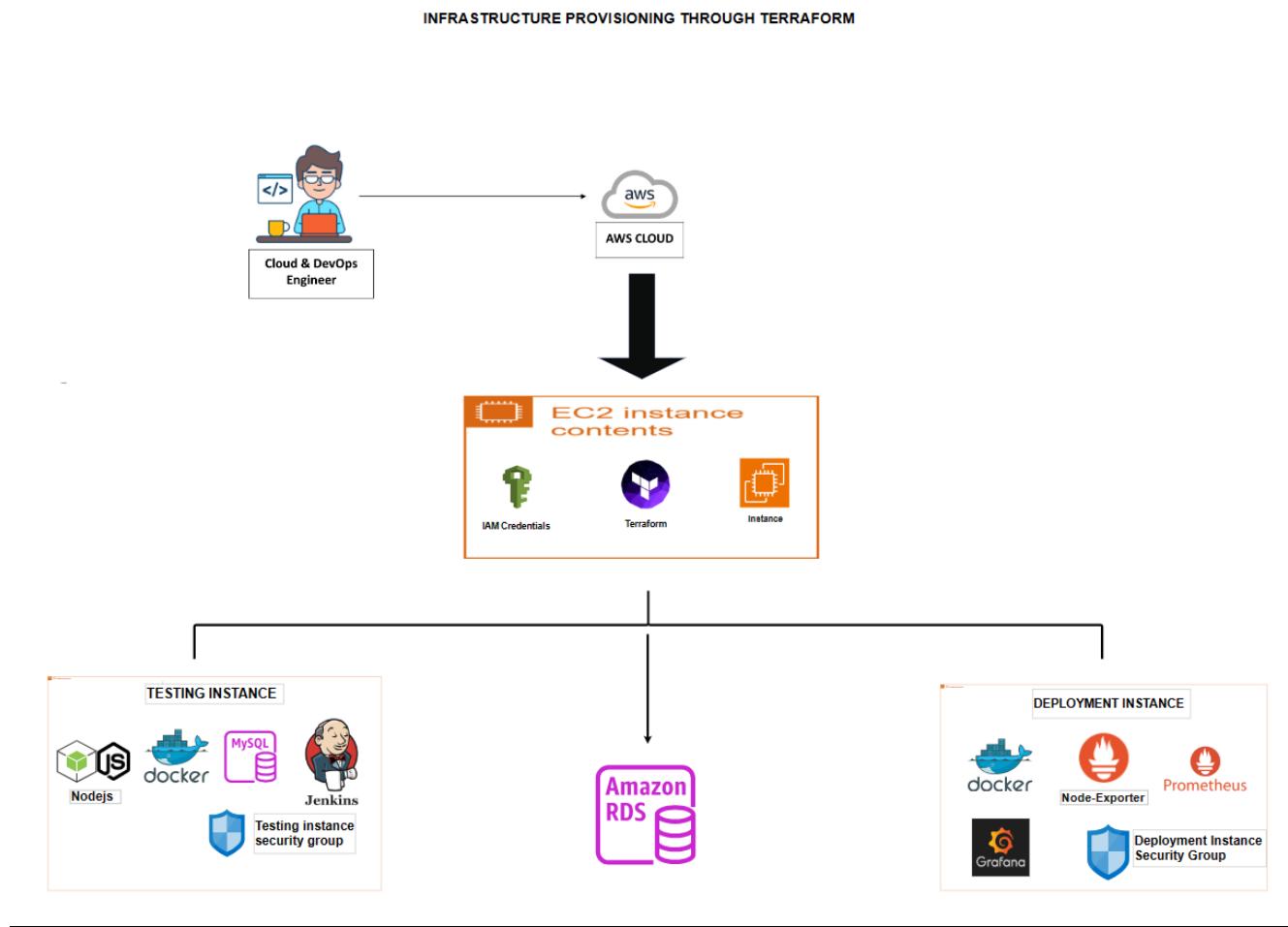
```

root@ip-172-31-10-219:/home/ubuntu# terraform --version
Terraform v1.6.6
on linux_amd64
root@ip-172-31-10-219:/home/ubuntu# aws --version
aws-cli/1.22.34 Python/3.10.12 Linux/6.2.0-1017-aws botocore/1.23.34
root@ip-172-31-10-219:/home/ubuntu#

```

STEP:3 – PROVISIONING THE INFRASTRUCTURE THROUGH TERRAFORM

OVERVIEW – DIAGRAM



- First, we need to **configure the aws credentials of IAM User** for terraform to use it for creating the required resources for this task:

```
root@ip-172-31-10-219:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIAZANDQIRAPG7GCZ7D
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-2
Default output format [None]: json
root@ip-172-31-10-219:/home/ubuntu#
```

- Creating a terraform script for provisioning the infrastructure required for this task:

Terraform script contains:

```

# 1.Providing aws details:
provider "aws" {
  profile = "default"
  region  = "us-east-2"
}

# 2.Creating security group for testing instance:
resource "aws_security_group" "testing-sc" {
  name        = "testing-instance"
  description = "security group for AWS EC2 instances"

  # Ingress rules (inbound traffic)
  # Allow SSH (port 22) from anywhere
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow application & jenkins (port 8080) from anywhere
  ingress {
    from_port   = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow mysql database (port 3306) from anywhere
  ingress {
    from_port   = 3306
    to_port     = 3306
    protocol    = "tcp"
    cidr_blocks = ["172.31.0.0/16"]
  }

  # Allow HTTP (port 80) from anywhere
  ingress {

```

```

        from_port    = 80
        to_port     = 80
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

# Egress rules (outbound traffic)
egress {
    from_port    = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
}

tags = {
    Name = "testing-instance"
}
}

# 3.Creating security group for deployment instance:
resource "aws_security_group" "deployment-sc" {
    name          = "deployment-instance"
    description   = "security group for AWS EC2 instances"

# Ingress rules (inbound traffic)
# Allow SSH (port 22) from anywhere
ingress {
    from_port    = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
}

# Allow HTTP (port 80) from anywhere
ingress {
    from_port    = 80
    to_port     = 80
    protocol    = "tcp"
}
}

```

```
    cidr_blocks = ["0.0.0.0/0"]
}

# Allow node-exporter (port 9100) from anywhere
ingress {
    from_port    = 9100
    to_port      = 9100
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
}

# Allow prometheus (port 9090) from anywhere
ingress {
    from_port    = 9090
    to_port      = 9090
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
}

# Allow grafana (port 3000) from anywhere
ingress {
    from_port    = 3000
    to_port      = 3000
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
}

# Allow docker-daemon (port 9323) from anywhere
ingress {
    from_port    = 9323
    to_port      = 9323
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
}

# Egress rules (outbound traffic)
egress {
    from_port    = 0
```

```

        to_port      = 0
        protocol     = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }

tags = {
    Name = "deployment-instance"
}
}

#4.Creating testing instance:
resource "aws_instance" "jenkins-instance" {
    ami           = "ami-07b36ea9852e986ad"
    instance_type = "t2.small"
    availability_zone = "us-east-2a"
    key_name       = "ravi2"
    vpc_security_group_ids = ["${aws_security_group.testing-sc.id}"]
    user_data       = "${file("testing.sh")}"

tags = {
    Name = "testing-instance"
}
}

#5.Creating deployment instance:
resource "aws_instance" "deployment-instance" {
    ami           = "ami-07b36ea9852e986ad"
    instance_type = "t2.micro"
    availability_zone = "us-east-2b"
    key_name       = "ravi2"
    vpc_security_group_ids =
["${aws_security_group.deployment-sc.id}"]
    user_data       = "${file("deployment.sh")}"

tags = {
    Name = "deployment-instance"
}
}
```

```

}

#6.Creating mysql database:
resource "aws_db_instance" "mydb" {
  identifier          = "nodejs-db"
  allocated_storage   = 20
  storage_type        = "gp2"
  db_name             = "nodejs"
  engine              = "mysql"
  engine_version      = "5.7"
  instance_class      = "db.t2.micro"
  username            = "admin"
  password            = "nodejs123"
  parameter_group_name = "default.mysql5.7"
  availability_zone   = "us-east-2c"
  skip_final_snapshot = true
  vpc_security_group_ids = ["${aws_security_group.testing-sc.id}"]
  tags = {
    Name = "nodejs-db"
  }
}

#7. Getting the output of mysql database:
output "rds_endpoint" {
  value = aws_db_instance.mydb.endpoint
}

```

testing.sh script contains:

```

#!/bin/bash
set -e # Exit script if any command returns a non-zero
status

#1. Updating the operating system:
apt-get update

#2. Installing java:

```

```
apt-get install -y openjdk-17-jre

#3. Installing jenkins:
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
apt-get update
apt-get install jenkins -y

#4. Installing docker:
apt-get install -y docker.io
chmod 777 /var/run/docker.sock

#5. Installing nodejs:
cd ~
curl -sL https://deb.nodesource.com/setup_16.x -o
/tmp/nodesource_setup.sh
sudo bash /tmp/nodesource_setup.sh
sudo apt-get install -y nodejs

#6. installing mysql:
apt install mysql-client-core-8.0 -y
```

deployment.sh script contains:

```
#!/bin/bash

#1. Updating the OS:
apt-get update

#2. Installing docker:
apt-get install -y docker.io
chmod 777 /var/run/docker.sock
```

```
#3. Installing node-exporter:  
#!/bin/bash  
NODE_EXPORTER_VERSION="1.7.0"  
wget  
https://github.com/prometheus/node_exporter/releases/download/v${NODE_EXPORTER_VERSION}/node_exporter-  
${NODE_EXPORTER_VERSION}.linux-amd64.tar.gz  
tar -xvf node_exporter-${NODE_EXPORTER_VERSION}.linux-  
amd64.tar.gz  
cd node_exporter-${NODE_EXPORTER_VERSION}.linux-amd64  
cp node_exporter /usr/local/bin  
  
# create user  
useradd --no-create-home --shell /bin/false node_exporter  
  
chown node_exporter:node_exporter  
/usr/local/bin/node_exporter  
  
echo '[Unit]  
Description=Node Exporter  
Wants=network-online.target  
After=network-online.target  
  
[Service]  
User=node_exporter  
Group=node_exporter  
Type=simple  
ExecStart=/usr/local/bin/node_exporter  
  
[Install]  
WantedBy=multi-user.target' >  
/etc/systemd/system/node_exporter.service  
  
# enable node_exporter in systemctl  
systemctl daemon-reload  
systemctl start node_exporter  
systemctl enable node_exporter
```

```
#4. Installing prometheus:  
#!/bin/bash  
PROMETHEUS_VERSION="2.48.0"  
wget  
https://github.com/prometheus/prometheus/releases/download/v${PROMETHEUS_VERSION}/prometheus-  
${PROMETHEUS_VERSION}.linux-amd64.tar.gz  
tar -xvf prometheus-${PROMETHEUS_VERSION}.linux-  
amd64.tar.gz  
cd prometheus-${PROMETHEUS_VERSION}.linux-amd64/  
# if you just want to start prometheus as root  
./prometheus --config.file=prometheus.yml  
  
# create user  
useradd --no-create-home --shell /bin/false prometheus  
  
# create directories  
mkdir -p /etc/prometheus  
mkdir -p /var/lib/prometheus  
  
# set ownership  
chown prometheus:prometheus /etc/prometheus  
chown prometheus:prometheus /var/lib/prometheus  
  
# copy binaries  
cp prometheus /usr/local/bin/  
cp promtool /usr/local/bin/  
  
chown prometheus:prometheus /usr/local/bin/prometheus  
chown prometheus:prometheus /usr/local/bin/promtool  
  
# copy config  
cp -r consoles /etc/prometheus  
cp -r console_libraries /etc/prometheus  
cp prometheus.yml /etc/prometheus/prometheus.yml  
  
chown -R prometheus:prometheus /etc/prometheus/consoles
```

```
chown -R prometheus:prometheus
/etc/prometheus/console_libraries

# setup systemd
echo '[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--
web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target' >
/etc/systemd/system/prometheus.service

systemctl daemon-reload
systemctl enable prometheus
systemctl start prometheus

#5. Installing grafana:
#!/bin/bash
echo 'deb https://packages.grafana.com/oss/deb stable main'
>> /etc/apt/sources.list
curl https://packages.grafana.com/gpg.key | sudo apt-key add -
sudo apt-get update
sudo apt-get -y install grafana

systemctl daemon-reload
```

```
systemctl start grafana-server  
systemctl enable grafana-server.service
```

→ The above resources have been created:

```
root@ip-172-31-10-219:/home/ubuntu# vi instance.tf  
root@ip-172-31-10-219:/home/ubuntu# vi testing.sh  
root@ip-172-31-10-219:/home/ubuntu# vi deployment.sh
```

→ First, we need to perform **terraform init** command: to initialize the plugins for terraform to control our aws cloud:

```
root@ip-172-31-10-219:/home/ubuntu# terraform init  
  
Initializing the backend...  
  
Initializing provider plugins...  
- Finding latest version of hashicorp/aws...  
- Installing hashicorp/aws v5.31.0...  
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)  
  
Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.  
  
Terraform has been successfully initialized!  
  
You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.  
  
If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.  
root@ip-172-31-10-219:/home/ubuntu#
```

→ Then **terraform fmt** to format the syntax of terraform script:

```
root@ip-172-31-10-219:/home/ubuntu# terraform fmt  
instance.tf  
root@ip-172-31-10-219:/home/ubuntu#
```

→ Then **terraform validate** to validate the keywords of terraform script:

```
root@ip-172-31-10-219:/home/ubuntu# terraform validate  
Success! The configuration is valid.  
root@ip-172-31-10-219:/home/ubuntu#
```

→ Then **terraform plan**, for the blueprint of terraform execution:

```
root@ip-172-31-10-219:/home/ubuntu# terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_db_instance.mydb will be created
+ resource "aws_db_instance" "mydb" {
    + address = (known after apply)
    + allocated_storage = 20
    + apply_immediately = false
    + arn = (known after apply)
    + auto_minor_version_upgrade = true}
```

→ Then **terraform apply -auto-approve** command, to initiate terraform to create above resources:

```
root@ip-172-31-10-219:/home/ubuntu# terraform apply -auto-approve
```

```
Plan: 5 to add, 0 to change, 0 to destroy.
```

```
Changes to Outputs:
+ rds_endpoint = (known after apply)
aws_security_group.testing-sc: Creating...
aws_security_group.deployment-sc: Creating...
aws_security_group.testing-sc: Creation complete after 2s [id=sg-069c69df61373e3c4]
aws_security_group.deployment-sc: Creation complete after 2s [id=sg-030220b5aa4ffba5f]
aws_db_instance.mydb: Creating...
aws_instance.jenkins-instance: Creating...
aws_instance.deployment-instance: Creating...
aws_db_instance.mydb: Still creating... [10s elapsed]
aws_instance.jenkins-instance: Still creating... [10s elapsed]
aws_instance.deployment-instance: Still creating... [10s elapsed]
aws_db_instance.mydb: Still creating... [20s elapsed]
aws_instance.jenkins-instance: Still creating... [20s elapsed]
aws_instance.deployment-instance: Still creating... [20s elapsed]
```

→ Terraform has done its job, now we need to check the console:

```
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
```

```
Outputs:
```

```
rds_endpoint = "nodejs-db.chttjdyzo3c7.us-east-2.rds.amazonaws.com:3306"
root@ip-172-31-10-219:/home/ubuntu#
```

→ **EC2 instance page:**

Instances (3) Info		Connect	Instance state	Actions	Launch instances	
		Find Instance by attribute or tag (case-sensitive)				
		Instance state = running X	Clear filters			
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
						Availability Zone
	Terraform	i-0f44c8bc5fbcb5cf	Running Q Q	t2.micro	2/2 checks passed	No alarms +
	testing-instance	i-0f1ea59e14ee0184	Running Q Q	t2.small	2/2 checks passed	No alarms +
	deployment-instance	i-06680a8416bf02a1b	Running Q Q	t2.micro	2/2 checks passed	No alarms +

➔ Testing instance details:

Instance: i-0f1eae59e14ee0184 (testing-instance)

Security groups

sg-069c69df61373e3c4 (testing-instance)

➔ Deployment instance details:

Instance: i-06680a8416bf02a1b (deployment-instance)

Security groups

sg-030220b5aa4ffba5f (deployment-instance)

➔ Testing security group:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
	Type	Protocol	Port range	Source	Description - optional
sgr-0188a65a152cba47d	HTTP	TCP	80	Custom	Q 0.0.0.0/0 X
sgr-02012600e682ffd89	SSH	TCP	22	Custom	Q 0.0.0.0/0 X
sgr-02b61a3b71b64c892	Custom TCP	TCP	8080	Custom	Q 0.0.0.0/0 X
sgr-04b68d257eebfc847	MySQL/Aurora	TCP	3306	Custom	Q 172.31.0.0/16 X

Add rule

➔ Deployment security group:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
	Type	Protocol	Port range	Source	Description - optional
sgr-09ee674d739e76db	Custom TCP	TCP	9525	Custom	Q 0.0.0.0/0 X
sgr-0f35a9dcfc870e5f	Custom TCP	TCP	9090	Custom	Q 0.0.0.0/0 X
sgr-018aa93d80e23a027	Custom TCP	TCP	9100	Custom	Q 0.0.0.0/0 X
sgr-068d86d029efe7cd2	SSH	TCP	22	Custom	Q 0.0.0.0/0 X
sgr-028c1c0c71b61da4f	HTTP	TCP	80	Custom	Q 0.0.0.0/0 X
sgr-024a189925a2b4804	Custom TCP	TCP	3000	Custom	Q 0.0.0.0/0 X

Add rule

→ MySQL RDS database:

The screenshot shows the AWS RDS 'Databases' page. A single database named 'nodejs-db' is listed. It is in the 'Available' status, running MySQL Community Engine, and is located in the us-east-2c region. The instance class is db.t2.micro, with 4.24% CPU usage and 0 connections. The database is currently inactive.

Instance			
Configuration	Instance class	Storage	Performance Insights
DB instance ID nodejs-db	Instance class db.t2.micro	Encryption Not enabled	Performance Insights enabled Turned off
Engine version 5.7.44	vCPU 1	Storage type General Purpose SSD (gp2)	
DB name nodejs	RAM 1 GB	Storage 20 GiB	
License model General Public License	Availability	Provisioned IOPS -	

→ By using the above **terraform code**, the resources created are:

Security groups:

TESTING INSTANCE SC	
Rules	Port number
HTTP	80
SSH	22
Jenkins & Application	8080
Database	3306
DEPLOYMENT INSTANCE SC	
Rules	Port number
HTTP	80
SSH	22
Grafana	3000
Prometheus	9090
Docker – daemon	9323
Node-exporter	9100

Instances:

- ❖ Testing instance:
 - Instance type: t2.small
 - Os: Ubuntu

- **Security group:** testing instance
- **Services and packages installed:**
 - Nodejs
 - Docker
 - Jenkins
 - MySQL

❖ **Deployment instance:**

- **Instance type:** t2.micro
- **Os:** Ubuntu
- **Security group:** deployment instance
- **Services and packages installed:**
 - Docker
 - Node-exporter
 - Grafana
 - Prometheus

❖ **Database: MySQL RDS Database**

- **Database name:** nodejs
- **Database endpoint:** nodejs-db.chttjdyzo3c7.us-east-2.rds.amazonaws.com
- **User:** admin

STEP:4 – TESTING THE APPLICATION ON LOCAL ENVIRONMENT

- ➔ First connecting the testing instance, and checking the required resources has installed or not:

```
root@ip-172-31-3-204:/home/ubuntu# node -v
v16.20.2
root@ip-172-31-3-204:/home/ubuntu# npm -v
8.19.4
root@ip-172-31-3-204:/home/ubuntu# docker -v
Docker version 24.0.5, build 24.0.5-0ubuntu1~20.04.1
root@ip-172-31-3-204:/home/ubuntu# jenkins --version
2.426.2
root@ip-172-31-3-204:/home/ubuntu# java --version
openjdk 17.0.9 2023-10-17
OpenJDK Runtime Environment (build 17.0.9+9-Ubuntu-120.04)
OpenJDK 64-Bit Server VM (build 17.0.9+9-Ubuntu-120.04, mixed mode, sharing)
root@ip-172-31-3-204:/home/ubuntu# mysql
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' (2)
root@ip-172-31-3-204:/home/ubuntu#
```

→ Then we need to test the application on local environment:

```
root@ip-172-31-3-204:/home/ubuntu# git clone https://github.com/Ravivarman16/mysql-nodejs-application_deployment.git
Cloning into 'mysql-nodejs-application_deployment'...
Username for 'https://github.com': Ravivarman16
Password for 'https://Ravivarman16@github.com':
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 24 (delta 1), reused 21 (delta 1), pack-reused 0
Unpacking objects: 100% (24/24), 23.45 KiB | 2.93 MiB/s, done.
root@ip-172-31-3-204:/home/ubuntu# cd mysql-nodejs-application_deployment/
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# ls
README.md app.js package-lock.json package.json public_routes t_user.sql views
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment#
```

→ First, we need to connect the database:

```
root@ip-172-31-3-204:/home/ubuntu# mysql -u admin -p nodejs -h nodejs-db.chttjdyzo3c7.us-east-2.rds.amazonaws.com
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 5.7.44 Please upgrade to 8.0 or opt-in to the paid RDS Extended Support service before 5.7 reaches end of standard support on 29 February, 2024: https://a.co/hQqIn0

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.
mysql> ■
```

→ Then we need to create table by using above database:

```
mysql>
mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| innodb        |
| mysql          |
| nodejs         |
| performance_schema |
| sys            |
+-----+
6 rows in set (0.00 sec)

mysql> use nodejs
Display all 759 possibilities? (y or n)
mysql> use nodejs;
Database changed
mysql> CREATE TABLE IF NOT EXISTS `t_user` (
    ->   `user_id` int(11) NOT NULL AUTO_INCREMENT,
    ->   `name` varchar(100) NOT NULL,
    ->   `email` varchar(100) NOT NULL,
    ->   `password` varchar(100) NOT NULL,
    ->   `created` DATE,
    ->   PRIMARY KEY (`user_id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=6 ;
Query OK, 0 rows affected (0.02 sec)

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

→ Then on application code we need to make few changes:

app.js contains:

```
app.use(connection(mysql, {
  host: "nodejs-db.chttjdyzo3c7.us-east-2.rds.amazonaws.com",
  user: "admin",
  password: "nodejs123",
  database: "nodejs",
  debug: false
}), "request");
```

routes/index.js contains:

```
connection = mysql.createConnection({
  host: "nodejs-db.chttjdyzo3c7.us-east-2.rds.amazonaws.com",
  user: "admin",
  password: "nodejs123",
  database: "nodejs",
  debug: false
});
```

→ Then we need to install an application dependency by using **npm install** command:

```
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# npm install
added 98 packages, and audited 99 packages in 3s
11 packages are looking for funding
  run `npm fund` for details
3 vulnerabilities (2 moderate, 1 critical)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 8.19.4 -> 10.2.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.5
npm notice Run npm install -g npm@10.2.5 to update!
npm notice
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment#
```

→ Then we need to start this application by using **node app.js** as command:

```
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# node app.js
Server is listening on port 8080
Database is connected
|
```

Application output:

The screenshot shows a web browser window with the address bar displaying "Not secure | 3.15.0.163:8080". The main content area features a large "Welcome!" heading. Below it, a descriptive text states: "This application allows users to sign-up and view other users within the database once logged in." At the bottom, there are two buttons: "Login" and "Sign Up".

Login Sign Up

Email

Password

➔ Checking the application by entering the signup details:

The screenshot shows the "Sign Up" form from the previous screenshot. It includes fields for Name, Email, and Password, each containing the text "master". A green "Sign Up!" button is visible at the bottom.

Sign Up

Name

Email

Password

➔ The details have been added: then click the logout option:

All Users

Below is a list of all user information.

[Logout](#)

ID	Name	Email	Password	Date Added
1	master	master@gmail.com	master1234	Sat Dec 23 2023 00:00:00 GMT+0000 (Coordinated Universal Time)

→ Now try to login with those credentials:

Welcome!

This application allows users to sign-up and view other users within the database once logged in

[Login](#) [Sign Up](#)

Email

master@gmail.com

Password

[Login!](#)

→ We can able to login successfully with those credentials. The application is working fine:

All Users

Below is a list of all user information.

[Logout](#)

ID	Name	Email	Password	Date Added
1	master	master@gmail.com	master1234	Sat Dec 23 2023 00:00:00 GMT+0000 (Coordinated Universal Time)

→ The command line output of the application:

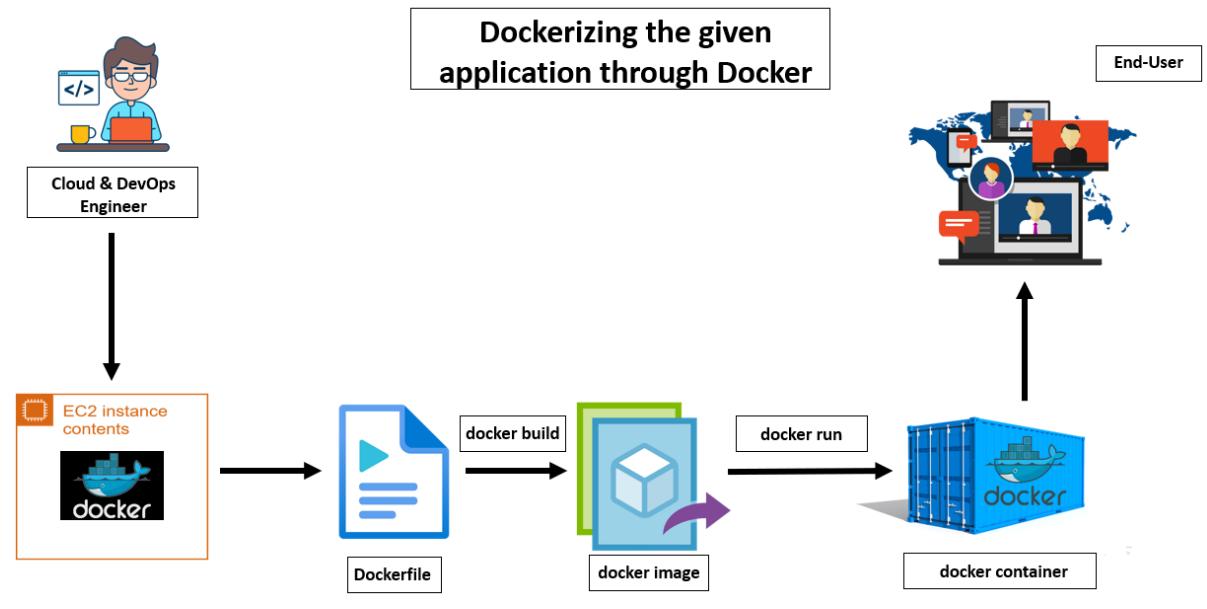
```

Database is connected
[
  RowDataPacket {
    user_id: 6,
    name: 'master',
    email: 'master@gmail.com',
    password: 'master1234',
    created: 2023-12-23T00:00:00.000Z
  }
]
master@gmail.com

```

STEP:5 – DOCKERIZING THE APPLICATION THROUGH DOCKER

OVERVIEW – DIAGRAM



→ Now we need to Dockerize this application, for that we need to create a dockerfile:

Dockerfile contains:

```
#choosing the base image:
FROM node:16-alpine
```

```

#choosing working directory for the application:
WORKDIR /app

#copying the application code:
COPY . .

#installing the dependencies:
RUN npm install

#exposing the application:
EXPOSE 8080

#starting the application:
CMD [ "node", "app.js" ]

```

- Then building a docker image by using above dockerfile with the command:

docker build -t <image name> .

```

root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# vi dockerfile
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# docker build -t ravivarman46/mysql-nodejs:app1 .
DEPRECATION: The legacy builder is deprecated and will be removed in a future release.
              Install the buildx component to build images with BuildKit:
              https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 11.31MB
Step 1/6 : FROM node:16-alpine
16-alpine: Pulling from library/node
7264a8db6415: Pull complete
eee371b9ce3f: Pull complete
93b3025fe103: Pull complete
d9059661ce70: Pull complete
Digest: sha256:a1f9d027912b58a7c75be7716c97cfbc6d3099f3a97ed84aa490be9dee20e787
Status: Downloaded newer image for node:16-alpine
--> 2573171e0124
Step 2/6 : WORKDIR /app
--> Running in 9c87d8939a98
Removing intermediate container 9c87d8939a98
--> ea94cbfb545
Step 3/6 : COPY . .
--> 3f34204bc5db
Step 4/6 : RUN npm install
--> Running in 171566b75bc0
up to date, audited 99 packages in 1s

11 packages are looking for funding
  run `npm fund` for details

```

- Checking the image is created or not by using the command:

docker images

```

Step 6/6 : CMD [ "node", "app.js" ]
--> Using cache
--> 029ef293c265
Successfully built 029ef293c265
Successfully tagged ravivarman46/mysql-nodejs:app1
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
ravivarman46/mysql-nodejs   app1      029ef293c265  3 minutes ago  130MB
node                16-alpine  2573171e0124  4 months ago  118MB
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment#

```

→ Then a running a container by using above image:

```
docker run -d -it --name nodejs -p 80:8080 <image name>
```

```

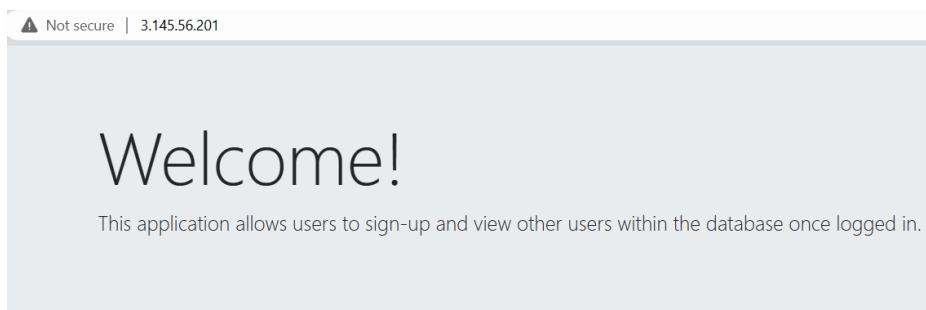
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# docker run -d -it --name nodejs -p 80:8080 ravivarman46/mysql-nodejs:app1
c87dc1b6fd3db2bd15e62b75ea03f0a63c5c610bf397da886b8d02a936c00018
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# docker ps
CONTAINER ID   IMAGE           COMMAND            CREATED          STATUS          PORTS
MES
c87dc1b6fd3d   ravivarman46/mysql-nodejs:app1   "docker-entrypoint.s..."   8 seconds ago   Up 7 seconds   0.0.0.0:80->8080/tcp, :::80->8080/tcp   no
dejs

```

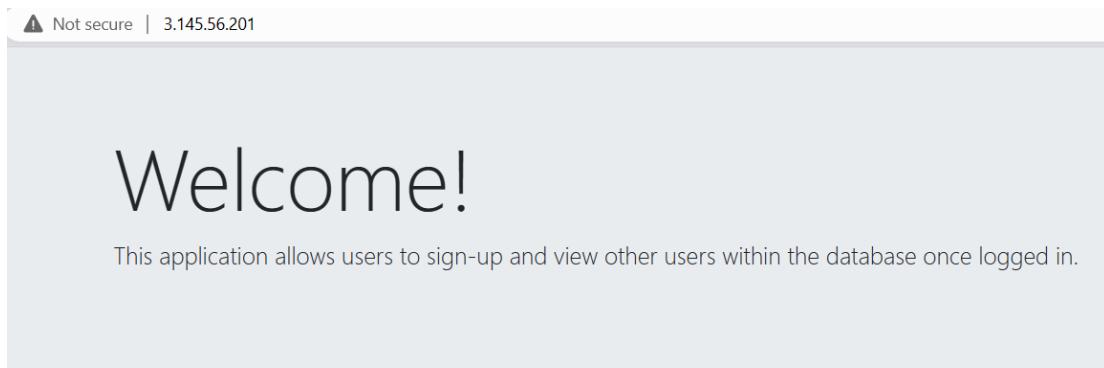
→ Container output checking, by using previous credentials created while testing in local environment:

Email id: master@gmail.com

Password: master1234



→ Entering the above details in login session and click login:



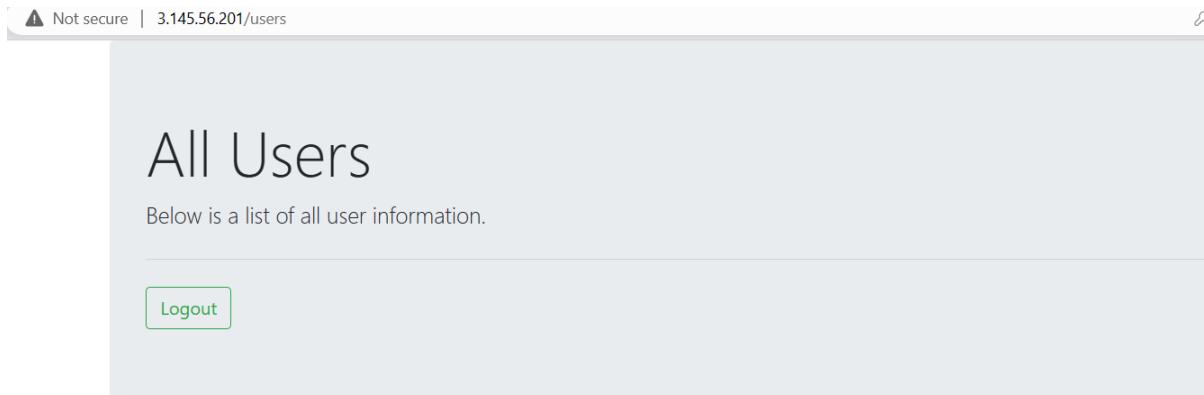
Login [Sign Up](#)

Email

Password

[Login!](#)

➔ We can able to login into the application with the previous credentials:



ID	Name	Email	Password	Date Added
1	master	master@gmail.com	master1234	Sat Dec 23 2023 00:00:00 GMT+0000 (Coordinated Universal Time)

Dockerization had been successfully

STEP:6 – SETTING UP THE JENKINS CONSOLE

→ Copy the public ip address of the testing instance, paste it on browser:

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

/var/lib/jenkins/secrets/initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password

→ Then copy the path and paste it on command line along with **cat** command to get the password:

```
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# cat /var/lib/jenkins/secrets/initialAdminPassword  
6c9209b628414872b23b7cbe91580ad0  
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment#
```

→ Then click continue after the entering the password:

/var/lib/jenkins/secrets/initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password

.....

Continue

→ Then select **install suggested plugins** option:

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different need:

Install suggested plugins

Select plugins to install

Install plugins the Jenkins community finds most useful.

Select and install plugins most suitable for your needs.

➔ The plugins will start to download:

Getting Started

The screenshot shows the Jenkins 'Getting Started' page. At the top, there's a large 'Getting Started' title with a progress bar below it. Below the title is a grid of plugin cards. Some cards have a green checkmark icon, while others have a question mark icon. The cards are arranged in rows:

✓ Folders	✓ OWASP Markup Formatter	⌚ Build Timeout	⌚ Credentials Binding	** Ionicons API Folders OWASP Markup Formatter ** JSON Path API ** Structs ** bouncycastle API
⌚ Timestamper	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle	
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline: Stage View	
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication	
⌚ LDAP	⌚ Email Extension	⌚ Mailer		

➔ Then setup the **login credentials for Jenkins**, after that click save and continue:

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

Jenkins 2.426.2

Skip and continue as admin

Save and Continue

➔ Then click save and finish:

Getting Started

Instance Configuration

Jenkins URL:

http://3.145.56.201:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

- Then the Jenkins console is ready for using, click **start using Jenkins option:**

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

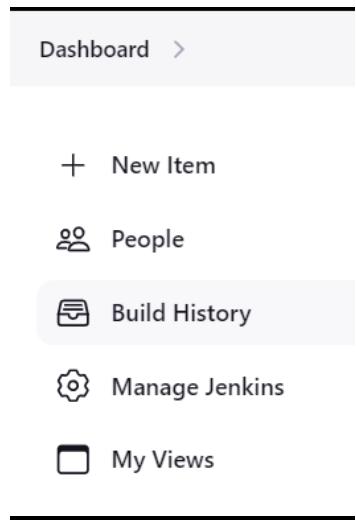
- Jenkins console is available for using:

The screenshot shows the Jenkins dashboard with the following elements:

- Header:** Jenkins logo, Dashboard >, Search (CTRL+K), Help icon, Jenkins user icon, and Log Out link.
- Sidebar:** New Item, People, Build History, Manage Jenkins, My Views, Build Queue (No builds in the queue), and Build Executor Status (1 Idle, 2 Idle).
- Main Content:**
 - Welcome to Jenkins!**: A message stating "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project".
 - Start building your software project**: A button to "Create a job" and a "+ Add description" link.
 - Set up a distributed build**: Links to "Set up an agent", "Configure a cloud", and "Learn more about distributed builds".

STEP:7 – SETTING UP THE DOCKERHUB, GITHUB & SSH CREDENTIALS:

→ On Jenkins dashboard, we can able to see **manage Jenkins**, click that one:



→ Then click **system** on manage Jenkins page:

System Configuration

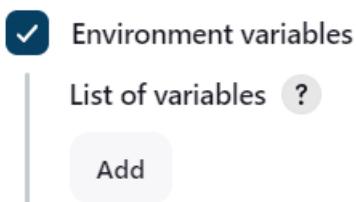


→ Then click **environment variables** under **global properties**:

Global properties

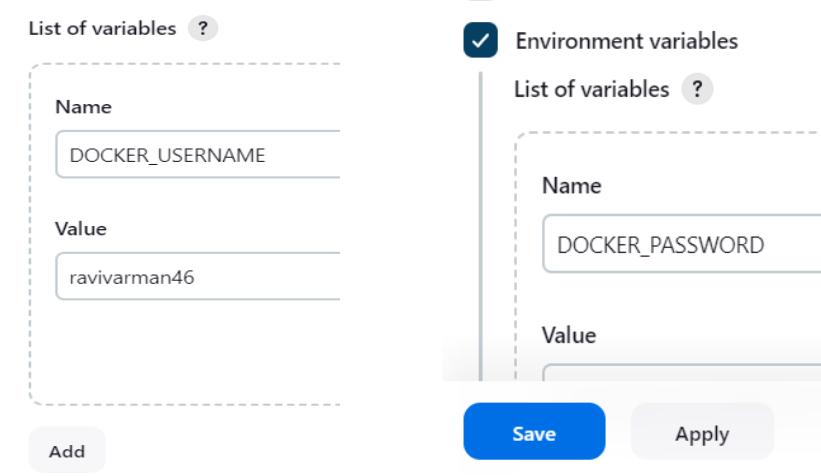
- Disable deferred wipeout on this node ?
- Environment variables
- Tool Locations

→ Then click **add** option:



- Then enter the environmental variables for **DockerHub credentials**:
- ❖ **DOCKER_USERNAME**
 - ❖ **DOCKER_PASSWORD**

After adding the environment variables, **click apply & save**:

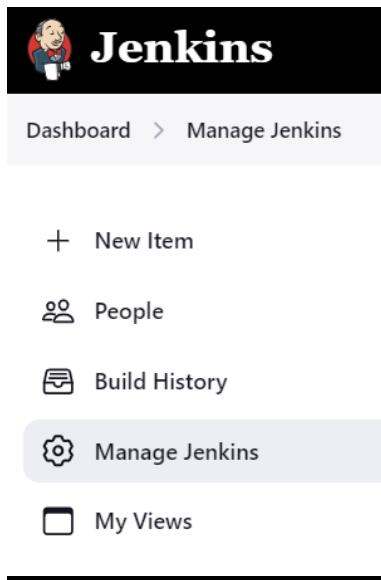


The screenshot shows the Jenkins 'Environment variables' configuration page. It displays two environment variables:

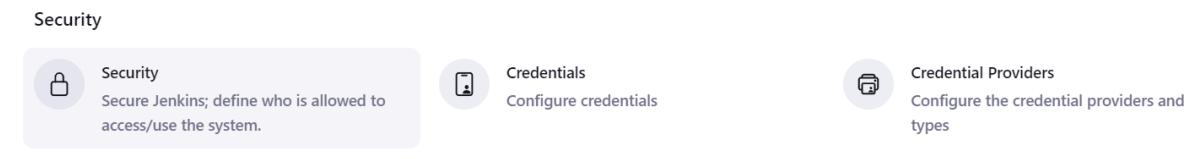
- DOCKER_USERNAME**: Value - ravivarman46
- DOCKER_PASSWORD**: Value - (redacted)

At the bottom, there are 'Save' and 'Apply' buttons.

- For adding GitHub credentials, click **manage Jenkins**:



- Then under manage Jenkins, click **credentials**:



→ Then click **global**:

The screenshot shows the Jenkins 'Credentials' page under 'Manage Jenkins'. The top navigation bar includes 'Dashboard', 'Manage Jenkins', and 'Credentials'. The main title is 'Credentials'. Below it, a sub-section titled 'Stores scoped to Jenkins' is shown. This section has a header with filters 'P', 'Store ↓', 'Domains', 'ID', and 'Name'. A single entry is listed: 'System' (Kind) with '(global)' (Domain). Below this, there are icons for sorting by 'Icon: S M L'.

→ Then click **add credentials**:

The screenshot shows the 'Global credentials (unrestricted)' page under 'Manage Jenkins'. The top navigation bar includes 'Dashboard', 'Manage Jenkins', 'Credentials', 'System', and 'Global credentials (unrestricted)'. The main title is 'Global credentials (unrestricted)'. A blue button on the right says '+ Add Credentials'. Below the title, a message states 'Credentials that should be available irrespective of domain specification to requirements matching.' A table header with columns 'ID', 'Name', 'Kind', and 'Description' is shown, followed by a message: 'This credential domain is empty. How about [adding some credentials?](#)'

→ Then add **GitHub Credentials**, after adding the credentials click **create** option:

The screenshot shows the 'New credentials' page under 'Manage Jenkins'. The top navigation bar includes 'Dashboard', 'Manage Jenkins', 'Credentials', 'System', and 'Global credentials (unrestricted)'. The main title is 'New credentials'. A form for 'Kind' is filled with 'Username with password'. Under 'Scope', 'Global (Jenkins, nodes, items, all child items, etc)' is selected. The 'Username' field contains 'Ravivarman16'. The 'Treat username as secret' checkbox is unchecked. The 'Password' field is empty. At the bottom, there is an 'ID' field and a 'Create' button.

→ The GitHub credentials has been added successfully:

The screenshot shows the Jenkins 'Global credentials (unrestricted)' configuration page. At the top, there's a breadcrumb navigation: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted). A blue button labeled '+ Add Credentials' is visible in the top right. The main table lists one credential entry:

ID	Name	Kind	Description
	git	Ravivarman16/******** (git)	Username with password

Below the table, there are filter buttons for 'Icon', 'S', 'M', and 'L'.

→ Like this we need to add SSH credentials: under **kind select SSH Username with private key**:

New credentials

Kind

SSH Username with private key

Scope ?

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

→ Under ID, I am mentioning it as ssh:

ID ?

ssh

Description ?

ssh

Username

ubuntu

→ Then under private key, I am adding the **keypair of the deployment instance**: after that click create:

Private Key

Enter directly

Key

```
bAp95AECgYB1zWSH7jIiiDqp5atCN6hunDgg3XFWUBLnaNM5PGXUXrJfPMp+K2WG  
26JA7u4bqrHVttWlhZE5rIq55C1EA9+byn8DSjkFRQahC3vRio3Azl5XIGTk3yF  
nnPyE/Hdkkt+QGpxCy8ps6vf2xrjyc+N5UxmdZ36n8AitmP06JdRRw==  
-----END RSA PRIVATE KEY-----
```

Passphrase

Create

→ The ssh credentials has been added successfully:

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

+ Add Credentials

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

ID	Name	Kind	Description	Actions
git	Ravivarman16/******** (git)	Username with password	git	
ssh	ubuntu (ssh)	SSH Username with private key	ssh	

Icon: S M L

STEP:8 - SETTING UP SSH CONNECTION

→ For setting up **SSH connection**, we need to download **two plugins**, for that click **manage Jenkins**:

Dashboard >

+ New Item

People

Build History

Manage Jenkins

My Views

➔ Under manage Jenkins, click **plugins**:

The screenshot shows the 'Manage Jenkins' page with a search bar at the top. Below it is the 'System Configuration' section. It contains several items with icons: 'System' (gear icon), 'Tools' (wrench icon), 'Nodes' (monitor icon), and 'Clouds' (cloud icon). The 'Plugins' item (key icon) is highlighted with a red box. A tooltip for 'Plugins' says: 'Add, remove, disable or enable plugins that can extend the functionality of Jenkins.'

➔ Then under plugins, click **available plugins**:

The screenshot shows the 'Plugins' page with a navigation bar at the top: 'Dashboard > Manage Jenkins > Plugins'. Below it is a list of tabs: 'Updates' (down arrow icon), 'Available plugins' (selected, folder icon), 'Installed plugins' (key icon), 'Advanced settings' (gear icon), and 'Download progress' (list icon). A red box highlights the 'Available plugins' tab.

➔ We need to download these **two plugins** for SSH connection to an **EC2 instance**:

- ❖ **SSH**
- ❖ **SSH Agent**

After selecting these two plugins, click **install** option:

A screenshot of the Jenkins plugin search interface. The search bar at the top contains the text 'ssh'. Below the search bar, there are two tabs: 'Install' and 'Name ↓'. Under the 'Install' tab, a plugin named 'SSH 2.6.1' is listed with a checked checkbox. A blue button labeled 'Build Wrappers' is shown below the plugin name. A brief description states: 'This plugin executes shell commands remotely using SSH protocol.'

A screenshot of the Jenkins plugin search interface, showing the results for 'ssh'. The search bar at the top contains the text 'ssh'. Below the search bar, there are two tabs: 'Install' and 'Name ↓'. Under the 'Install' tab, a plugin named 'SSH Agent 346.vda.a.c4f2c8e50' is listed with a checked checkbox. A blue button labeled 'Install' is shown to the right. A brief description states: 'This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins.' To the right of the description, it says '1 mo 21 days ago'.

→ The plugins have been installed successfully:

A screenshot of the Jenkins 'Plugins' page under 'Manage Jenkins'. The navigation bar shows 'Dashboard > Manage Jenkins > Plugins'. On the left, there's a sidebar with options: 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress' (which is highlighted). On the right, there's a 'Download progress' section with a table. The table has two columns: 'Preparation' and a list of Jenkins core API components. Each component has a green checkmark icon next to it, indicating success. The components listed are: Ionicons API, Folders, OWASP Markup Formatter, JSON Path API, and Structs.

Preparation	
Ionicons API	Success
Folders	Success
OWASP Markup Formatter	Success
JSON Path API	Success
Structs	Success

→ Then under **manage Jenkins** click system:

System Configuration

A screenshot of the Jenkins 'System Configuration' page under 'Manage Jenkins'. It features a large gear icon and the text 'System'. Below that, it says 'Configure global settings and paths.'.

→ Then under **SSH remote hosts**, click that add option:

SSH remote hosts

SSH sites

SSH sites that projects will want to connect

Add

→ Then under **hostname**, enter the **public Ip of deployment instance**:

SSH remote hosts

SSH sites

SSH sites that projects will want to connect

Hostname ?

3.20.227.222

→ Then under **port** enter **SSH port number 22**, under credentials select the **SSH Credentials we added**.

Port ?

22

Credentials

ubuntu (ssh)

+ Add ▾

Pty ?

→ Then on left side we can able to **check connection**, click that one to check whether the **testing instance can reach to deployment instance**: we should able to see **successful connection**



STEP:9 - CREATING A JENKINSFILE FOR CI/CD PIPELINE

- We need to create Jenkinsfile for CI/CD Pipeline by using above application:

Jenkinsfile contains:

```
pipeline {
    agent any

    stages {

        stage ('Docker login & Building image') {
            steps {
                sh 'docker login -u $DOCKER_USERNAME -p
$DOCKER_PASSWORD'
                sh 'docker build -t $DOCKER_USERNAME/mysql-
nodejs:app1 .'
            }
        }

        stage ('Pushing the Docker image to DockerHub') {
            steps {
                sh 'docker push $DOCKER_USERNAME/mysql-
nodejs:app1'
            }
        }
    }
}
```

```

stage('Deploy to Deployment server') {
    steps {
        script {
            sshagent(['ssh']) {
                // Execute the command within the
                sshagent block using sh step
                sh 'ssh -o StrictHostKeyChecking=no
ubuntu@3.20.227.222 "docker run -d -it --name nodejs -p
80:8080 ravivarman46/mysql-nodejs:app1"'
            }
        }
    }
}

```

→ Then pushing the **contains** and **source** files to GitHub repository:

```

root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# vi Jenkinsfile
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# ls
Jenkinsfile README.md app.js dockerfile node_modules package-lock.json package.json public routes t_user.sql views
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment#

```

```

root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   app.js
    modified:   routes/index.js

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Jenkinsfile
    dockerfile
    node_modules/

no changes added to commit (use "git add" and/or "git commit -a")
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# git add .
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment#

```

```

root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# git commit -m "cicd pipeline"
[main 87e7e53] cicd pipeline
Committer: root <root@ip-172-31-3-204.us-east-2.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.

```

```
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment# git push origin main
Username for 'https://github.com': Ravivarman16
Password for 'https://Ravivarman16@github.com':
Enumerating objects: 2116, done.
Counting objects: 100% (2116/2116), done.
Compressing objects: 100% (1998/1998), done.
Writing objects: 100% (2112/2112), 1.96 MiB | 4.68 MiB/s, done.
Total 2112 (delta 506), reused 0 (delta 0)
remote: Resolving deltas: 100% (506/506), completed with 2 local objects.
To https://github.com/Ravivarman16/mysql-nodejs-application_deployment.git
  38b9dfe..87e7e53  main -> main
root@ip-172-31-3-204:/home/ubuntu/mysql-nodejs-application_deployment#
```

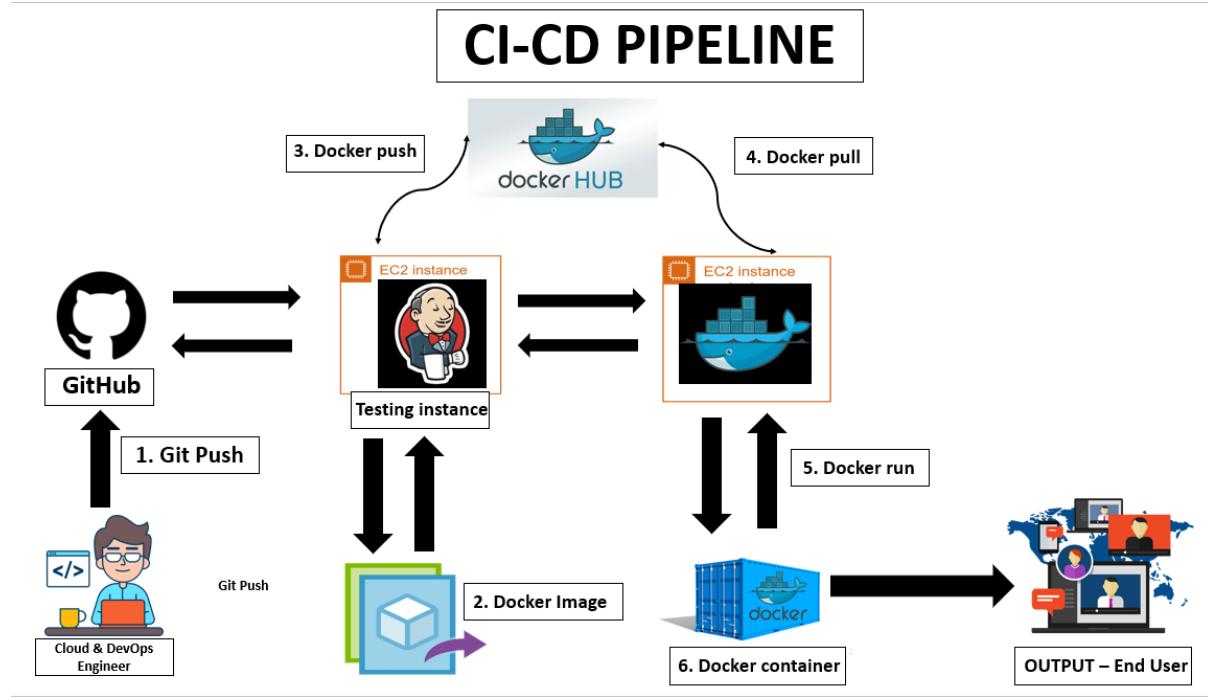
→ source code has been pushed to GitHub repository from local repository successfully. Confirm it by checking in GitHub

The screenshot shows a GitHub repository page for 'mysql-nodejs-application_deployment'. The repository is private. The main branch has 1 branch and 0 tags. The commit history shows the following commits:

Commit	Author	Message	Time	Commits
87e7e53	root	cicd pipeline	1 minute ago	3 Commits
node_modules		cicd pipeline	1 minute ago	
public		application files	1 hour ago	
routes		cicd pipeline	1 minute ago	
views		application files	1 hour ago	
Jenkinsfile		cicd pipeline	1 minute ago	
README.md		Initial commit	1 hour ago	

STEP:10 - CREATING A CI/CD PIPELINE IN JENKINS

OVERVIEW – DIAGRAM



→ On GitHub repository, click settings:

Screenshot of the GitHub repository settings page for 'mysql-nodejs-application_deployment'. The top navigation bar shows 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'. The repository name 'mysql-nodejs-application_deployment' is visible in the title bar. The left sidebar menu includes 'Code and automation' with options: 'Branches', 'Tags', 'Rules', 'Actions', 'Webhooks', 'Codespaces', and 'Pages'.

→ Then click **webhooks** on left side of screen:

- Code and automation
- Branches
- Tags
- Rules
- Actions
- Webhooks
- Codespaces
- Pages

→ Then click **add webhook**:

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

→ Then under **payload URL**, enter Jenkins URL along with **github-webhook**:

Webhooks / Add webhook

We'll send a POST request to the URL below with format you'd like to receive (JSON, x-www-form-urlencoded documentation).

Payload URL *

`http://3.145.56.201:8080/github-webhook/`

→ Then select the **event** according to your preferences:

Which events would you like to trigger this webhook?

- Just the push event.
- Send me everything.
- Let me select individual events.

Active

We will deliver event details when this hook is triggered.

Add webhook

→ We can able to see webhooks has been successfully for **automatic trigger**:

Webhooks

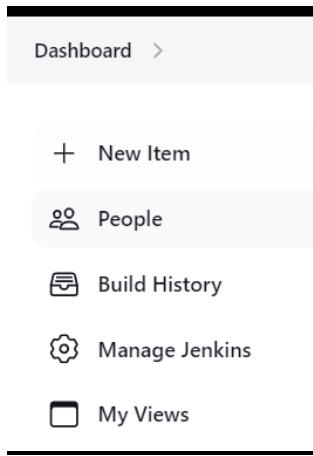
Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

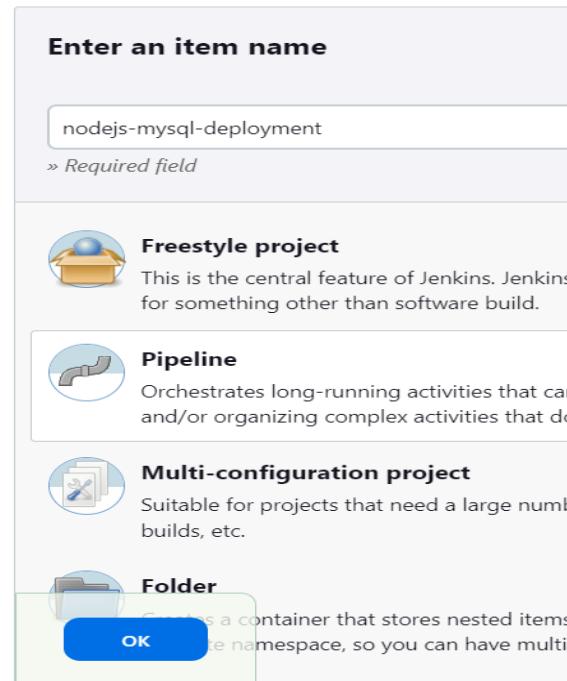
✓ `http://3.145.56.201:8080/github-we... (all events)`

Edit Delete

→ Then on **Jenkins dashboard**, click **new item** to create a new job:



→ Then **enter name for the job**, select **pipeline** and click **okay**:



→ Then under **General**, give **description** according to the job:

General

Description	NodeJS-MySQL-deployment pipeline
-------------	----------------------------------

→ Then under **build triggers**, select **GitHub hook trigger for GITScm polling option:**

Build Triggers

- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?
- Quiet period ?
- Trigger builds remotely (e.g., from scripts) ?

→ Then under **Pipeline** select **Pipeline script from SCM, select Git:**

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

→ Then enter the **GitHub repository link** under **Repository URL:**

Repositories ?

Repository URL ?

https://github.com/Ravivarman16/mysql-nodejs-application_deployment.git

Credentials ?

Ravivarman16/******** (git)

+ Add ▾

→ then enter the **branch** according to your GitHub repository:

The screenshot shows the Jenkins Pipeline configuration page for a job named "nodejs-mysql-deployment". The left sidebar has tabs for "General", "Advanced Project Options", and "Pipeline", with "Pipeline" currently selected. The main configuration area includes sections for "Branches to build" (specifying "*/main") and "Repository browser" (set to "(Auto)"). Under "Additional Behaviours", there is an "Add" dropdown. The "Script Path" is set to "Jenkinsfile", and the "Lightweight checkout" option is checked. At the bottom are "Save" and "Apply" buttons.

→ Pipeline job has been created successfully, now click **build now** option to start pipeline job:

The screenshot shows the Jenkins Pipeline job status page for "nodejs-mysql-deployment". The left sidebar lists options: "Status", "Changes", "Build Now", "Configure", "Delete Pipeline", "Full Stage View", "Rename", and "Pipeline Syntax". The "Status" tab is active. The main area is titled "Stage View" and displays a message: "No data available. This Pipeline has not yet run." Below this is a "Permalinks" section. At the bottom, there is a "Build History" tab and a "trend" dropdown.

→ The job has started successfully:

The screenshot shows the Jenkins Build History page. At the top, there's a header with a sun icon and the text "Build History". Below it is a search bar with the placeholder "Filter builds..." and a dropdown menu set to "trend". A single build entry is listed: "#1 Dec 23, 2023, 10:48 AM". To the right of the build entry are three small icons: a red square with a white "X", an upward arrow, and a downward arrow. At the bottom of the list are two links: "Atom feed for all" and "Atom feed for failures".

→ The job has executed successfully: click the build number 1:

The screenshot shows the Jenkins Pipeline stage view for build #1 of the "nodejs-mysql-deployment" pipeline. The pipeline name is displayed at the top. On the left, there's a sidebar with various pipeline management options like Status, Changes, Build Now, Configure, Delete Pipeline, and Full Stage View. The main area is titled "Stage View" and shows four stages: "Declarative: Checkout SCM", "Docker login & Building image", "Pushing the Docker image to DockerHub", and "Deploy to Deployment server". Each stage has a progress bar indicating its duration: 1s, 6s, 4s, and 6s respectively. Below the stages, a summary states "Average stage times: (Average full run time: ~25s)". A detailed view of build #1 is shown, with a timestamp of "Dec 23 16:18" and a note "No Changes". At the bottom, there's a "Permalinks" section with up and down arrows.

→ Then click **console output**, to know how Jenkins executed this job:

The screenshot shows the Jenkins Build #1 console output page. The top navigation bar includes "Dashboard", "nodejs-mysql-deployment", and "#1". The main content area starts with a "Status" button (which is green for success) and the text "Build #1 (Dec 23, 2023, 10:48:54 AM)". Below this are several pipeline management options: "Changes", "Console Output" (which is currently selected), "Edit Build Information", "Delete build '#1'", and "Git Build Data". To the right of these options, there's information about the build: "Started by user jenkins", a "git" icon, "Revision: 87e7e53c450c40b8b244fd3318ab99ba4bf59a05", "Repository: https://github.com/Ravivarman16/mysql-nodejs-application_deployment.git", and a "refs/remotes/origin/main" link. The entire page is framed by a thick black border.

Dashboard > nodejs-mysql-deployment > #1

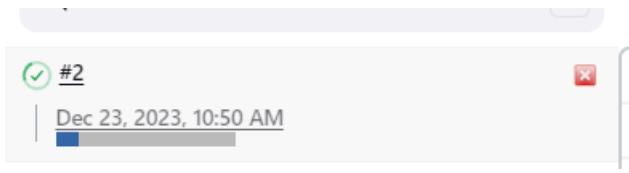
Status **Console Output** Changes View as plain text Edit Build Information

Started by user Jenkins
Obtained Jenkinsfile from git https://github.com/Ravivarman16/mysql-nodejs-application_deployment.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/nodejs-mysql-deployment
[Pipeline] {

Dashboard > nodejs-mysql-deployment > #1

```
828f86ba7f3d: Download complete
eee371b9ce3f: Pull complete
93b3025fe103: Pull complete
d9059661ce70: Pull complete
1d7e7c1a3e55: Pull complete
b2e8e0c026fb: Pull complete
828f86ba7f3d: Pull complete
Digest: sha256:36e73c84796b1b03f4ccf06c4b8dc0f133cc5298d87da84ffb0b5d998c7c41bf
Status: Downloaded newer image for ravivarman46/mysql-nodejs:app1
389286a3020e7a48489ccf6ed5be550b8694238ebbb2935a8ada594240491f93
[Pipeline] }
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 2340 killed;
[ssh-agent] Stopped.
[Pipeline] // sshagent
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

➔ Then build the job again:



➔ The job has failed, to know about the failed status, click the **console output**:

Status  **nodejs-mysql-deployment**

- </> Changes NodeJS-MySQL-deployment pipeline
- ▷ Build Now
- ⚙ Configure
- trash Delete Pipeline
- 🔍 Full Stage View
- ✍ Rename
- ⓘ Pipeline Syntax
- 📋 GitHub Hook Log

Stage View

Declarative: Checkout SCM	Docker login & Building image	Pushing the Docker image to DockerHub	Deploy to Deployment server
920ms	3s	2s	4s
Average stage times: (Average full run time: ~25s)			
#2 Dec 23 16:20 No Changes	585ms	1s	1s 1s failed

Build History **trend** ▾

➔ The job is failed because already a container is running on port number 80. For this we need to alter our Jenkinsfile.

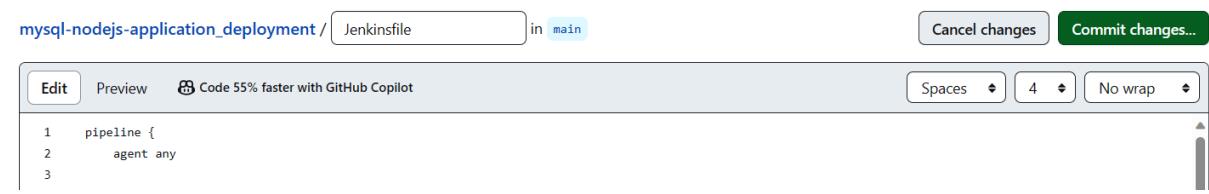
```
+ ssh -o StrictHostKeyChecking=no ubuntu@3.20.227.222 docker run -d -it --name nodejs -p 80:8080 ravivarman46/mysql-nodejs:app1
docker: Error response from daemon: Conflict. The container name "/nodejs" is already in use by container
"389286a3020e7a48489ccf6ed5be550b8694238ebbb2935a8ada594240491f93". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
[Pipeline]
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 2508 killed;
[ssh-agent] Stopped.
[Pipeline] // sshagent
[Pipeline]
[Pipeline] // script
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 125
Finished: FAILURE
```

➔ Jenkinsfile altering portion: update this portion on Jenkinsfile:

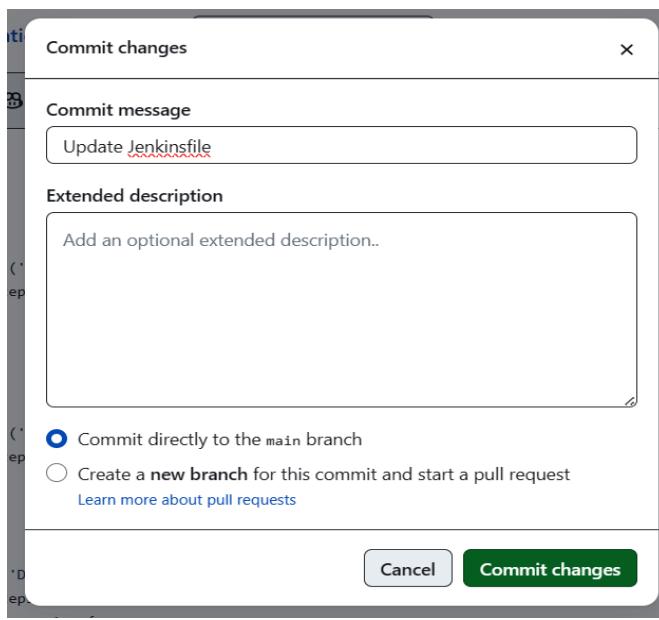
```
script {
    sshagent(['ssh']) {
        // Execute the command within the
        sshagent block using sh step
```

```
sh 'ssh -o StrictHostKeyChecking=no
ubuntu@3.20.227.222 "docker stop nodejs; docker rm nodejs;
docker run -d -it --name nodejs -p 80:8080
ravivarman46/mysql-nodejs:app1"'
}'
```

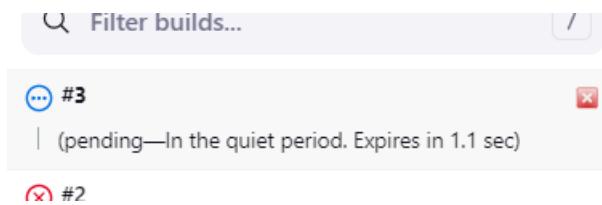
→ Click **commit changes**:



→ Again, click commit changes:



→ The job got triggered automatically when we made a change in GitHub:



→ The job has executed successfully:

Dashboard > nodejs-mysql-deployment >

Status ✖ nodejs-mysql-deployment

- </> Changes NodeJS-MySQL-deployment pipeline
- ▷ Build Now
- ⚙ Configure
- trash Delete Pipeline
- 🔍 Full Stage View
- edit Rename
- info Pipeline Syntax
- file GitHub Hook Log

Stage View

Declarative: Checkout SCM	Docker login & Building image	Pushing the Docker image to DockerHub	Deploy to Deployment server
Average stage times: (Average full run time: ~25s) 792ms	4s	3s	7s
#3 Dec 23 16:23 1 commit	535ms	4s	3s
	12s		

Build History trend ▾

→ Now build the job again, to know whether pipeline is working fine. As per the result pipeline is working perfectly.

Dashboard > nodejs-mysql-deployment >

Status ✖ nodejs-mysql-deployment

- </> Changes NodeJS-MySQL-deployment pipeline
- ▷ Build Now
- ⚙ Configure
- trash Delete Pipeline
- 🔍 Full Stage View
- edit Rename
- info Pipeline Syntax
- file GitHub Hook Log

Stage View

Declarative: Checkout SCM	Docker login & Building image	Pushing the Docker image to DockerHub	Deploy to Deployment server
Average stage times: (Average full run time: ~22s) 699ms	3s	2s	8s
#4 Dec 23 16:23 No Changes	423ms	1s	1s
#3 Dec 23 16:23 1 commit	535ms	4s	3s
#2 Dec 23 16:20 No Changes	585ms	1s	1s
			1s failed

Build History trend ▾

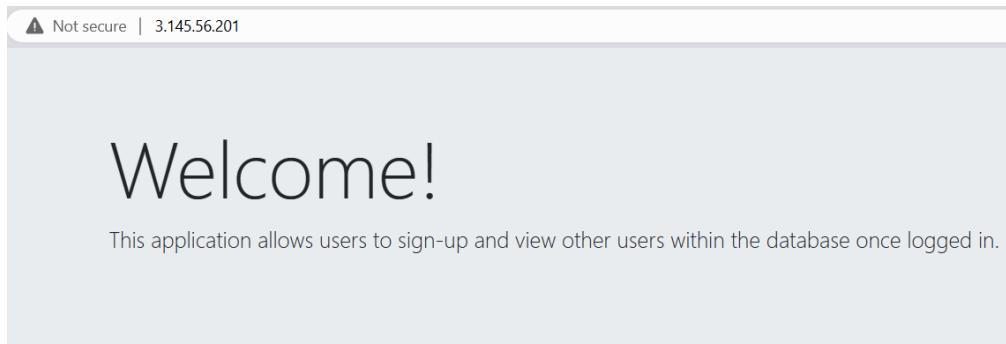
Filter builds... /

#4 | Dec 23, 2023, 10:53 AM

#3 | Dec 23, 2023, 10:53 AM

#2 | Dec 23, 2023, 10:53 AM

→ Checking the container, by adding new credentials from testing instance:



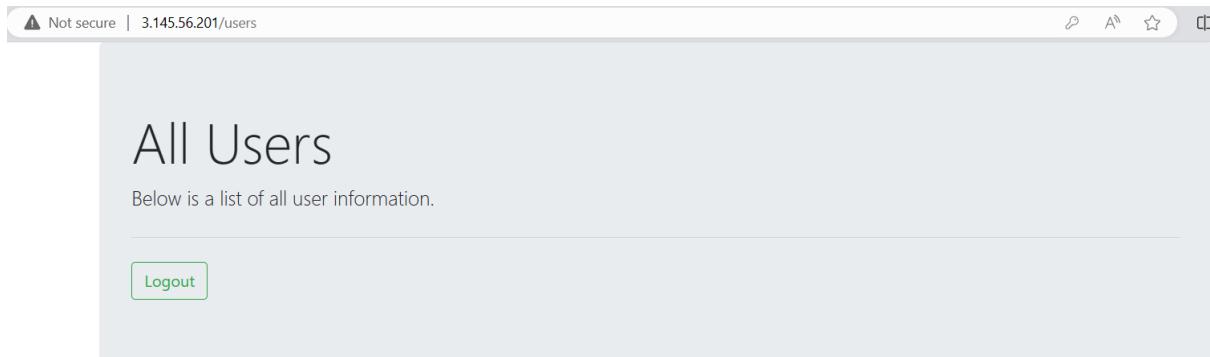
[Login](#) [Sign Up](#)

Name

Email

Password

[Sign Up!](#)



ID	Name	Email	Password	Date Added
1	master	master@gmail.com	master1234	Sat Dec 23 2023 00:00:00 GMT+0000 (Coordinated Universal Time)
2	jenkins	jenkins@gmail.com	jenkins12345	Sat Dec 23 2023 00:00:00 GMT+0000 (Coordinated Universal Time)

→ Trying to login with the new credentials on deployment instance where we deployed the container through Jenkins pipeline.

⚠ Not secure 3.20.227.222/users

All Users

Below is a list of all user information.

[Logout](#)

ID	Name	Email	Password	Date Added
1	master	master@gmail.com	master1234	Sat Dec 23 2023 00:00:00 GMT+0000 (Coordinated Universal Time)
2	jenkins	jenkins@gmail.com	jenkins12345	Sat Dec 23 2023 00:00:00 GMT+0000 (Coordinated Universal Time)

→ Checking the deployment instance:

```
root@ip-172-31-23-65:/home/ubuntu# docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
ravivarman46/mysql-nodejs  app1    67b9173a1b7e  10 minutes ago  132MB
root@ip-172-31-23-65:/home/ubuntu# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED        STATUS        PORTS
 NAMES
2b30f398d7eb   ravivarman46/mysql-nodejs:app1   "docker-entrypoint.s..."  5 minutes ago   Up 5 minutes   0.0.0.0:80->8080/tcp, :::80->8080/tcp
nodejs
root@ip-172-31-23-65:/home/ubuntu#
```

→ Checking the DockerHub:

Images [Give feedback](#)

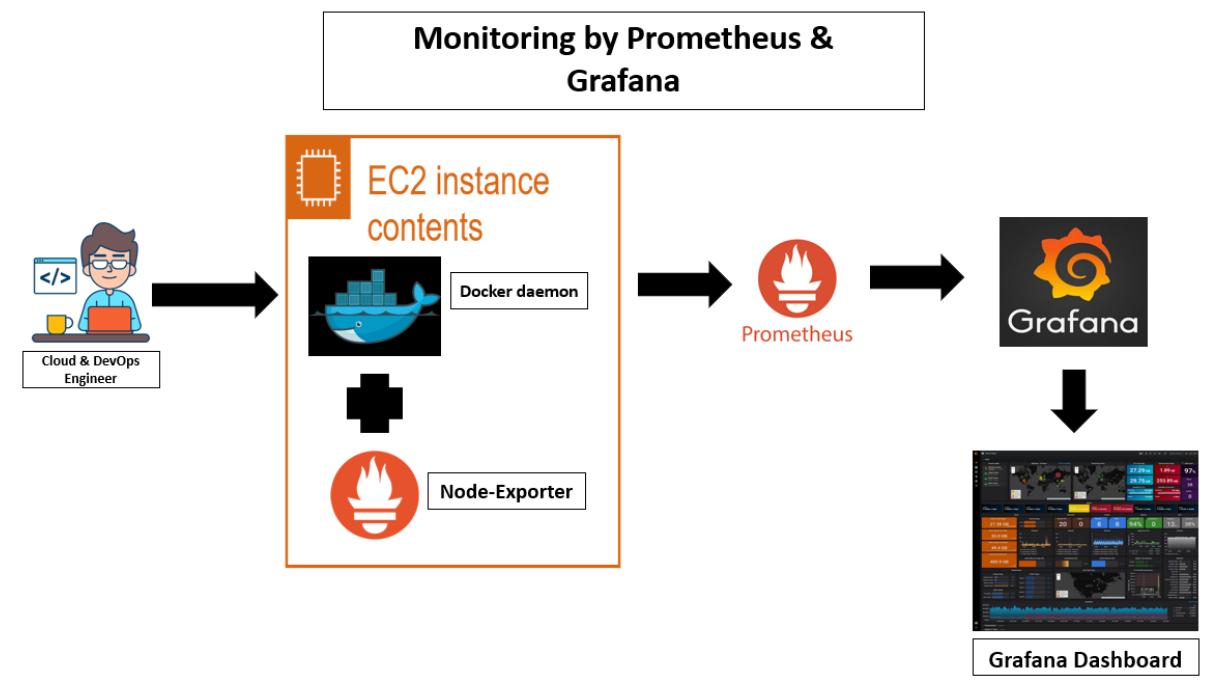
Local **Hub** Artifactory [EARLY ACCESS](#)

ravivarman46 ▾ Search

Tags	OS	Vulnerabilities	Last pushed	Size
 ravivarman46/mysql-nodejs app1		 Inactive	5 minutes ago	47.19 MB

STEP:11 - SETTING UP THE MONITORING THROUGH PROMETHEUS & GRAFANA

OVERVIEW – DIAGRAM



- On the deployment instance, we need to create **daemon.json** under **/etc/docker** for getting metrics from docker daemon.
- After that **restart the docker service**.

```
root@ip-172-31-23-65:/home/ubuntu# cd /etc/docker/
root@ip-172-31-23-65:/etc/docker# ls
root@ip-172-31-23-65:/etc/docker# nano daemon.json
root@ip-172-31-23-65:/etc/docker# cat daemon.json
{
  "metrics-addr" : "0.0.0.0:9323",
  "experimental" : true
}
root@ip-172-31-23-65:/etc/docker# service docker restart
```

- Then we need to add this one to Prometheus service by altering **prometheus.yaml** file which will be under this location **/etc/prometheus/prometheus.yaml**

```

scrape_configs:
  # The job name is added as a label `job`
  - job_name: "prometheus"

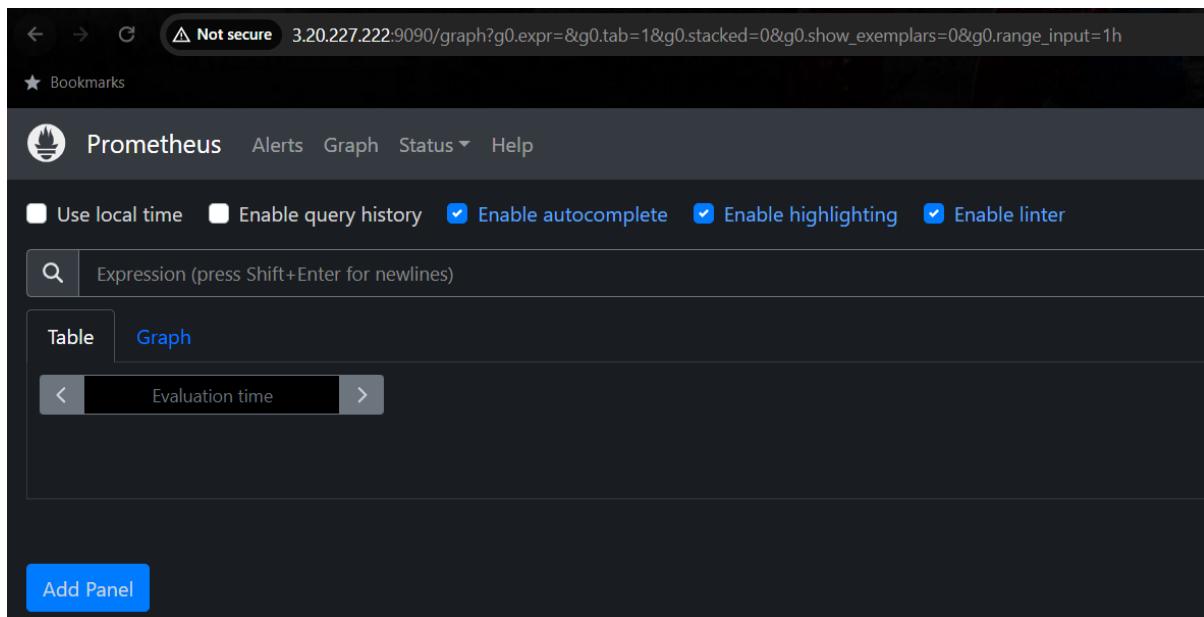
    <# metrics_path defaults to '/metrics'
       # scheme defaults to 'http'.
    > Build Now
    static_configs:
      - targets: ["localhost:9090"]

      - job_name: "node_exporter"
        scrape_interval: 5s
        static_configs:
          - targets: ['localhost:9100']

      - job_name: "Docker-container"
        static_configs:
          - targets: ["localhost:9323"]

```

- ➔ Then restart **Prometheus service** by using the command: **service prometheus restart**
- ➔ Then enter the public Ip address of deployment instance on browser along with the port number **9090**.



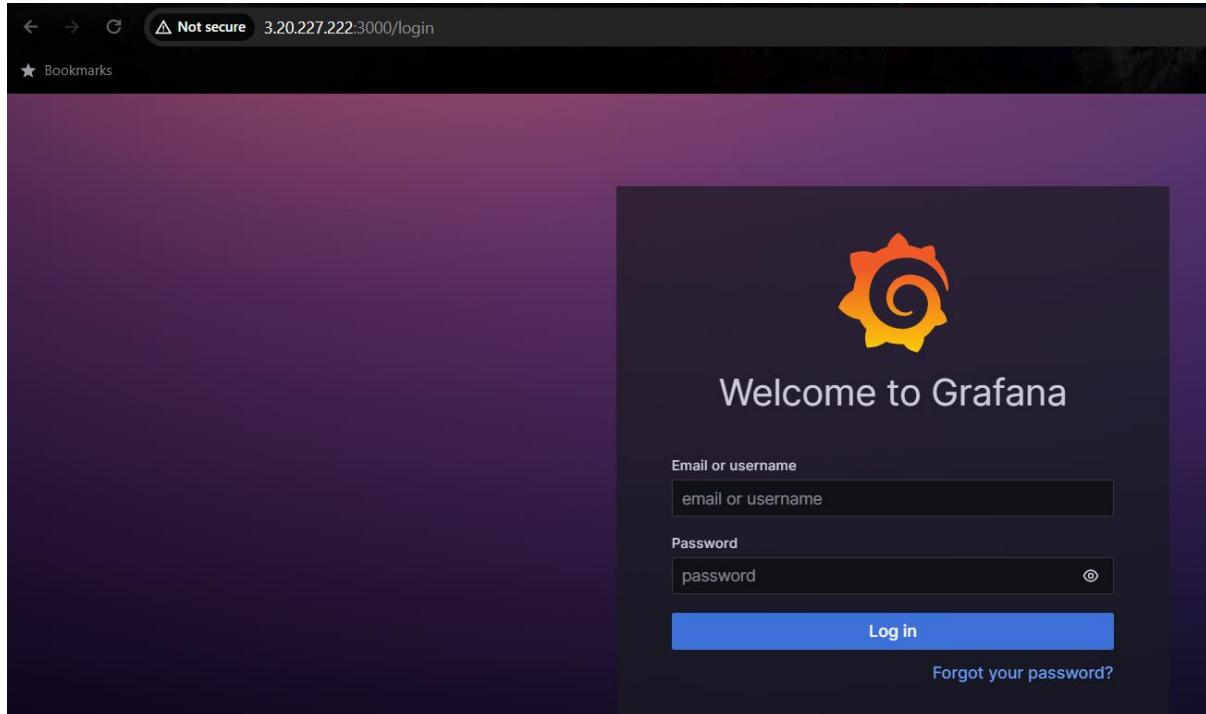
- ➔ Then click the status, under that select targets:

The screenshot shows the Prometheus Targets page. At the top, there are navigation links: Prometheus, Alerts, Graph, Status ▾, and Help. Below the header, there's a search bar with a magnifying glass icon and a placeholder "Filter by endpoint or labels". A dropdown menu "All scrape pools ▾" is open, showing options: All, Unhealthy, and Collapse All. The main content area displays three groups of targets:

- Docker Job (1/1 up)**: Shows one target at <http://localhost:9323/metrics> in the UP state. Labels: instance="localhost:9323", job="Docker Job". Last Scrape: 14.666s ago. Scrape Duration: 6.920ms.
- node_exporter (1/1 up)**: Shows one target at <http://localhost:9100/metrics> in the UP state. Labels: instance="localhost:9100", job="node_exporter". Last Scrape: 1.267s ago. Scrape Duration: 14.957ms.
- prometheus (1/1 up)**: Shows one target at <http://localhost:9090/metrics> in the UP state. Labels: instance="localhost:9090", job="prometheus". Last Scrape: 4.360s ago. Scrape Duration: 7.269ms.

We could able to see all the targets is up and running successfully.

- Then enter the public Ip address of deployment instance along with the port number **3000**, for login into Grafana.



- For login into Grafana, **the username & password is admin**. Click log in

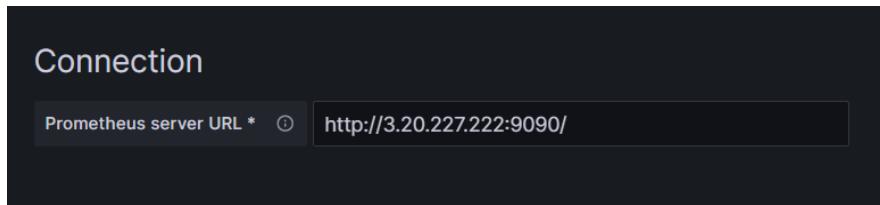
→ Now we need to add **Prometheus** to Grafana server, for that click **data sources**:

The screenshot shows the Grafana home page with a dark theme. At the top right, there's a search bar and a navigation bar with links to Documentation, Tutorials, Community, and Public Slack. Below the header, there are three main sections: 'Basic' (with a 'Tutorial' link), 'DATA SOURCES' (with a 'Add your first data source' link), and 'DASHBOARDS' (with a 'Create your first dashboard' link). At the bottom, there are links for 'Dashboards', 'Starred dashboards', 'Latest from the blog', and a news item about 'Open source at Grafana Labs in 2023: Year in review'.

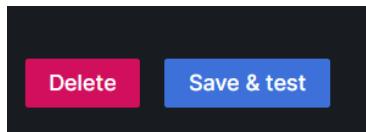
→ Then select **Prometheus**:

The first screenshot shows the 'Add data source' page where 'Prometheus' is selected from a list of time series databases. The second screenshot shows the configuration page for the 'prometheus' data source, which is currently set to 'Type: Prometheus'. It includes tabs for 'Settings' (which is active) and 'Dashboards'. A note at the bottom says 'Configure your Prometheus data source below' and 'Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted da...'. The configuration fields show 'Name' set to 'prometheus' and a 'Default' toggle switch turned on.

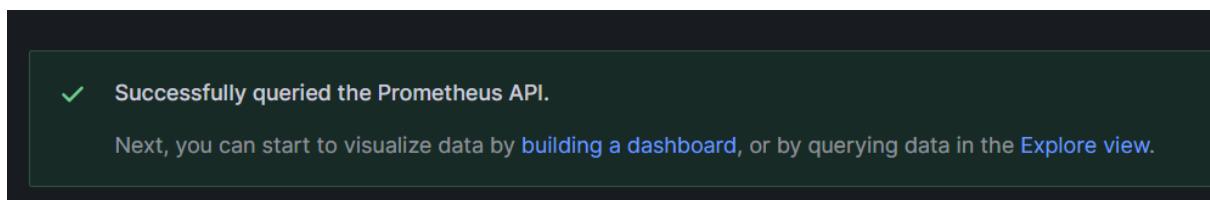
→ Then enter the **Prometheus URL**:



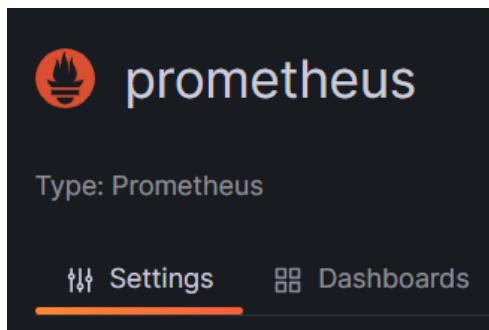
→ After that **click save & test**, to know whether Grafana can able to connect Prometheus and get the logs from it.



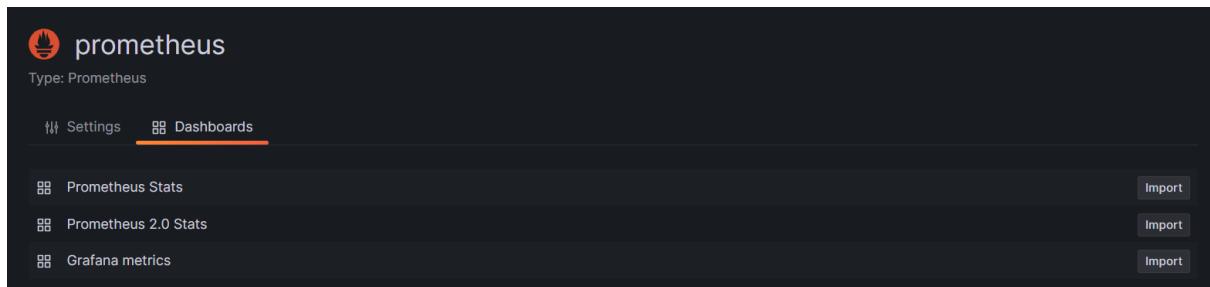
→ We could able to see **Grafana can able to connect Prometheus successfully**.

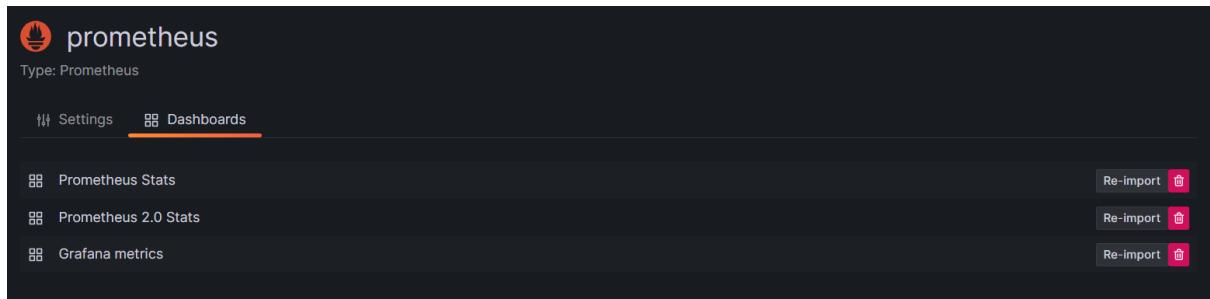


→ Then at the top of screen we could able to see **dashboards**, click that one:

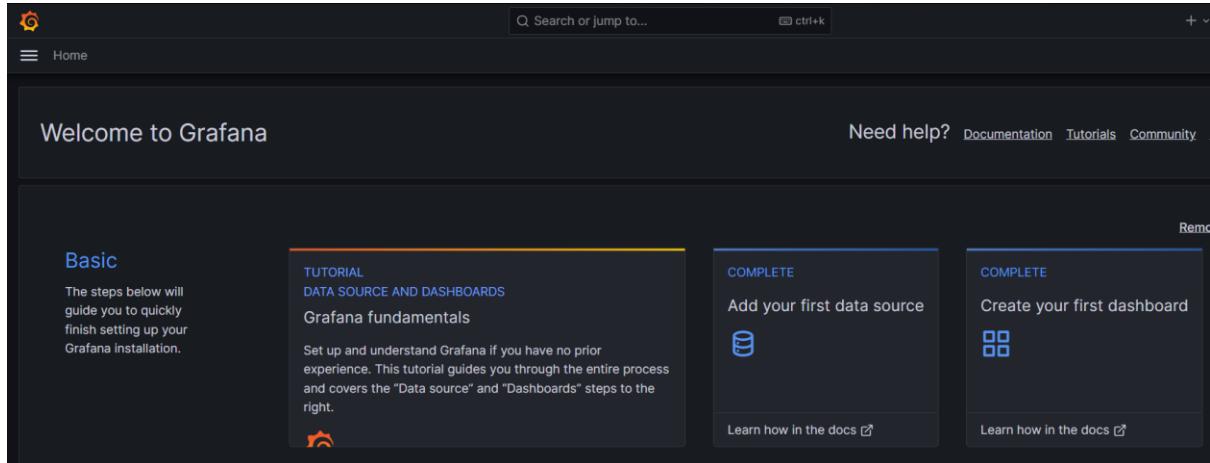


→ Then to get the **logs from Prometheus**, click **import option for all three**:

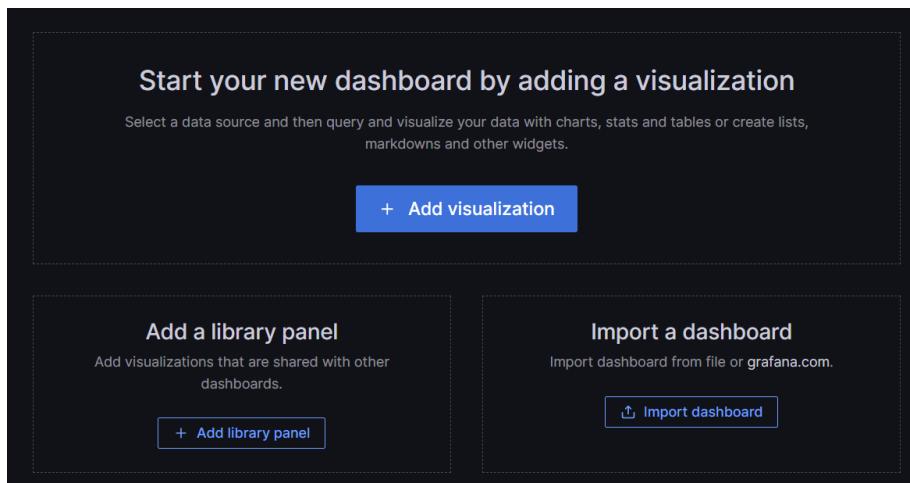




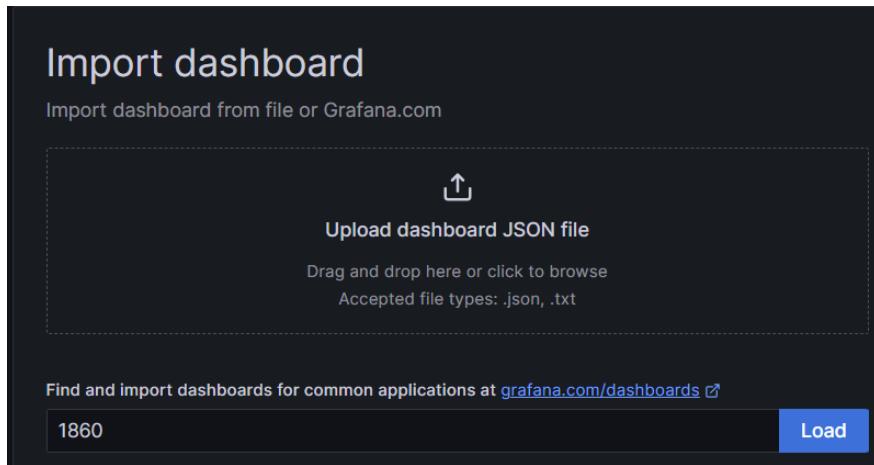
→ Then for creating dashboards, on home screen **click dashboard**:



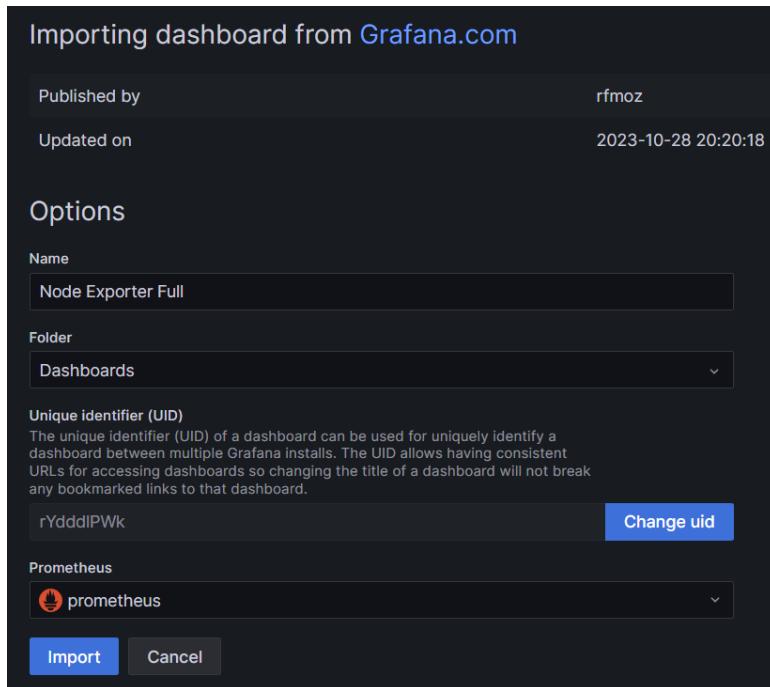
→ Then select **import a dashboard** option:



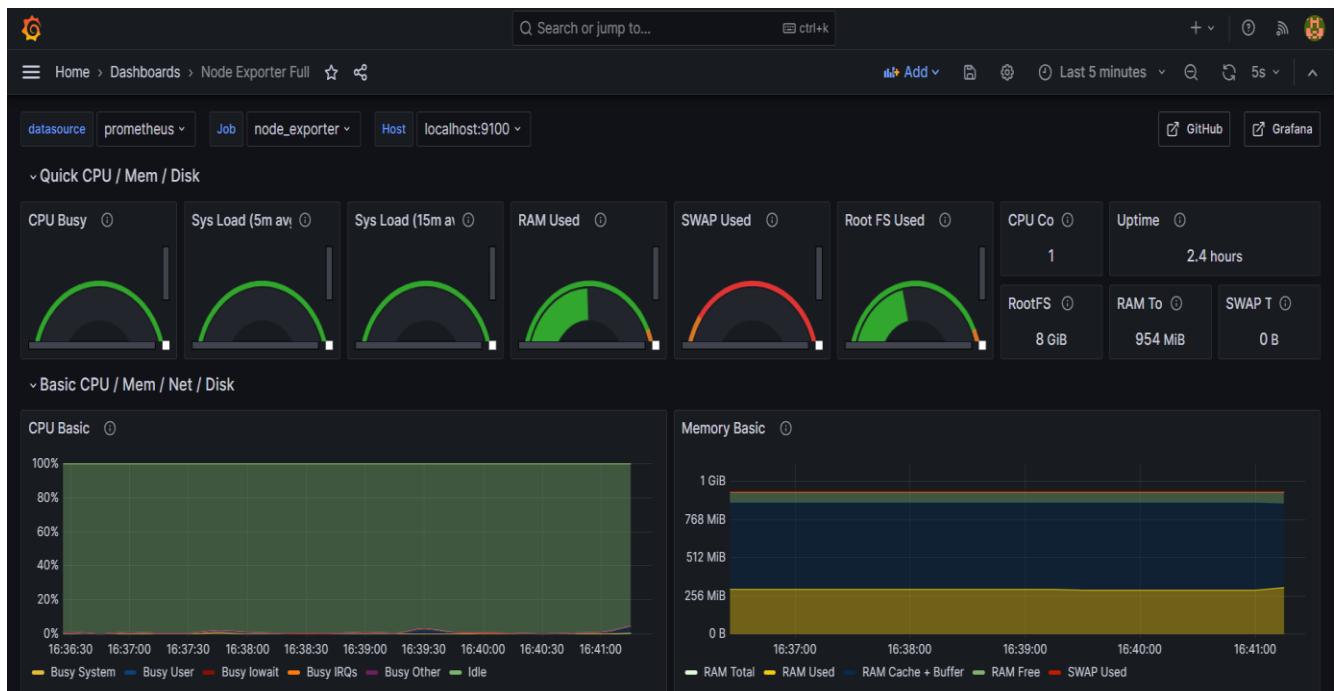
→ Then enter **1860** which is **node exporter dashboard number**, click load option:



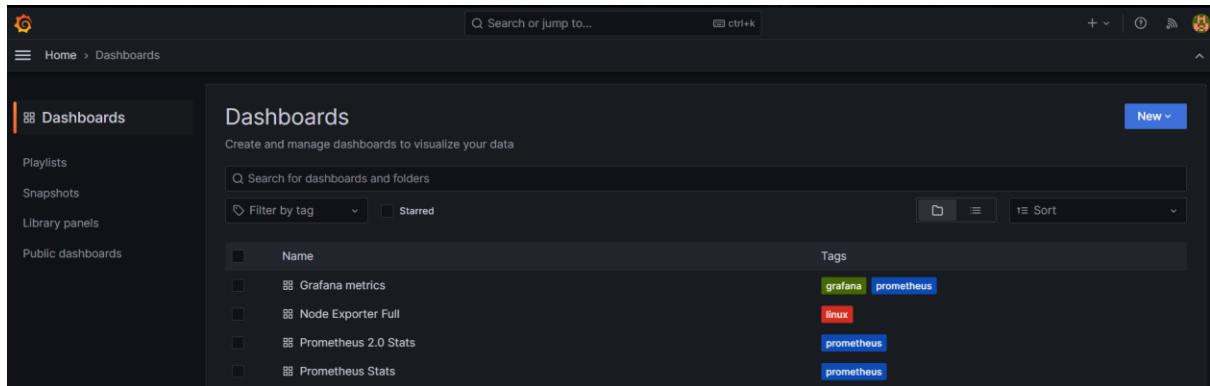
→ Then under data source, select **Prometheus** click the **import** option:



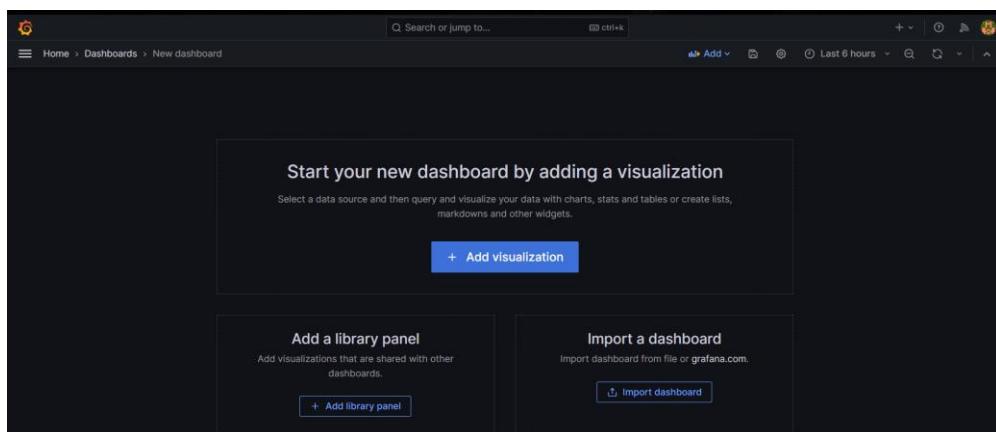
→ The dashboard has been created successfully for **deployment instance**, where the container is running.



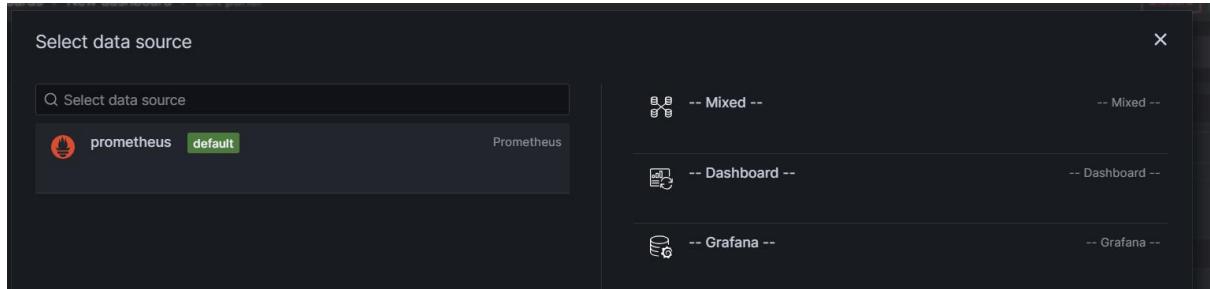
→ To create our own dashboard, click new option on right side of dashboard screen:



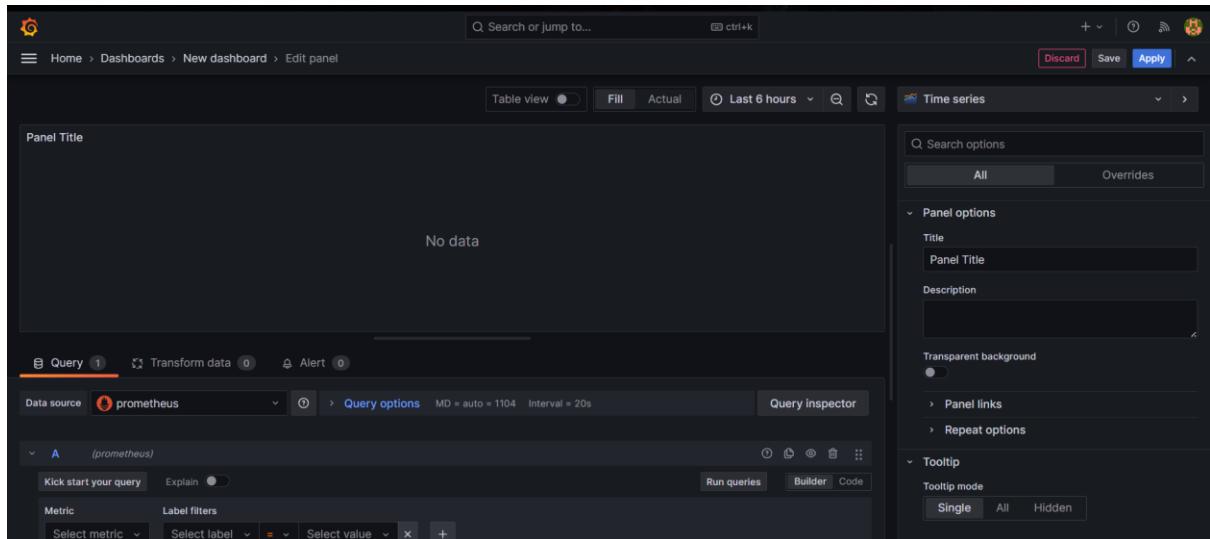
→ Then click add visualization:



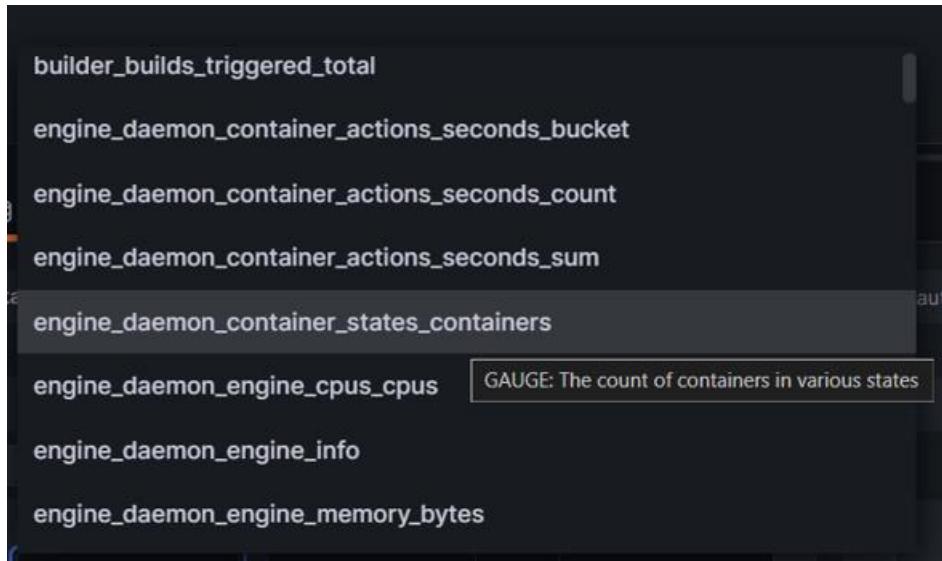
→ Then select Prometheus as data source:



→ Then in the dashboard, we need to make a few changes, for creating our own dashboard, **for that click metrics**:



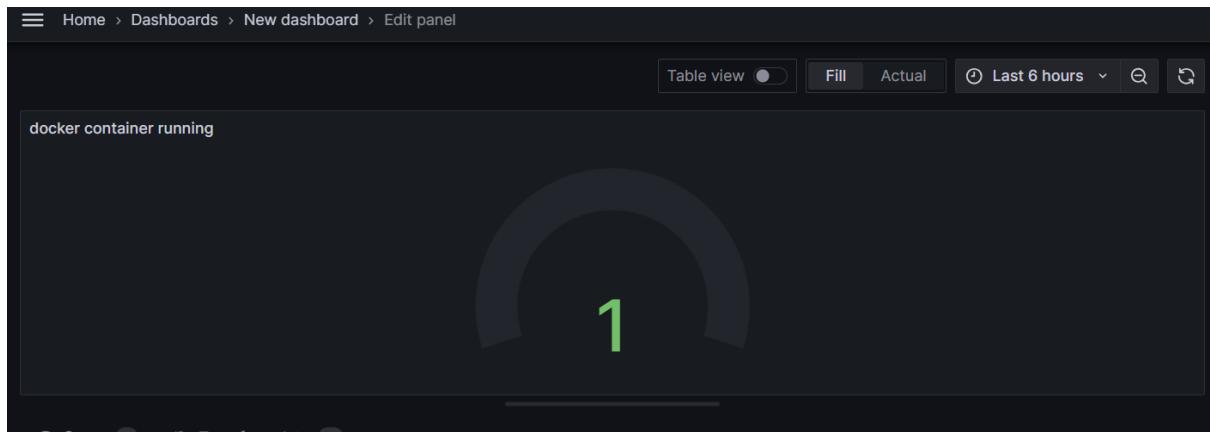
→ Select **engine_daemon_container_states_containers** option:



→ Then select the **state running**, click **run queries**

The screenshot shows the Grafana query editor interface. At the top, there's a search bar with "engine_daemon_container_states_containers" and a label filter section where "state" is set to "running". Below the search bar is a button labeled "+ Operations". The main area contains the query code: `engine_daemon_container_states_containers{state="running"}`. To the right, there are buttons for "Run queries", "Builder", and "Code".

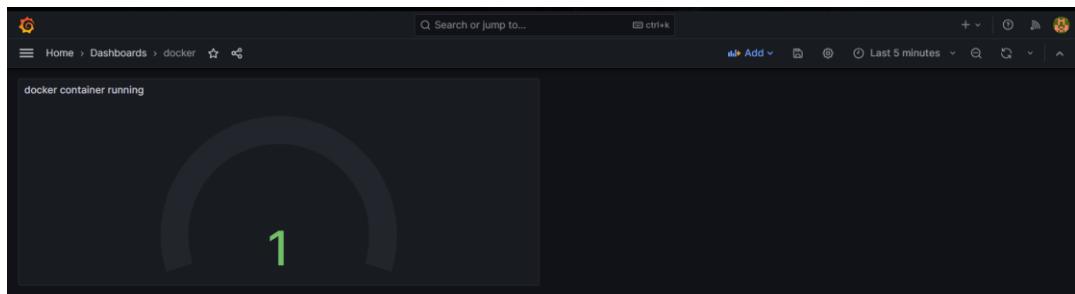
→ We could able to see in the dashboard that **1 container is running**



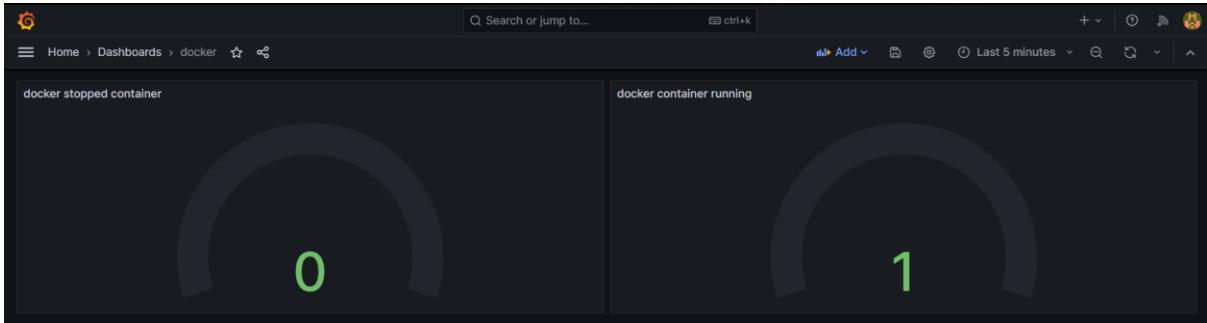
→ Then save this dashboard:

The screenshot shows the "Save dashboard" dialog box. Under the "Details" tab, the "Title" field is filled with "docker". There are also fields for "Description" and "Folder" (set to "Dashboards"). At the bottom, there are "Cancel" and "Save" buttons.

→ Like this we can add more features to this dashboard by **clicking add option**:

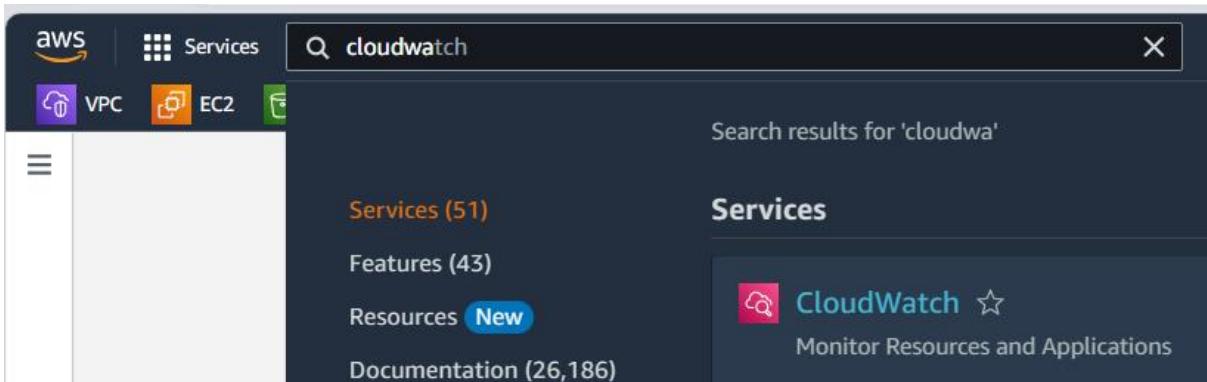


→ Here we had setup two features – **container running & stopped container**:

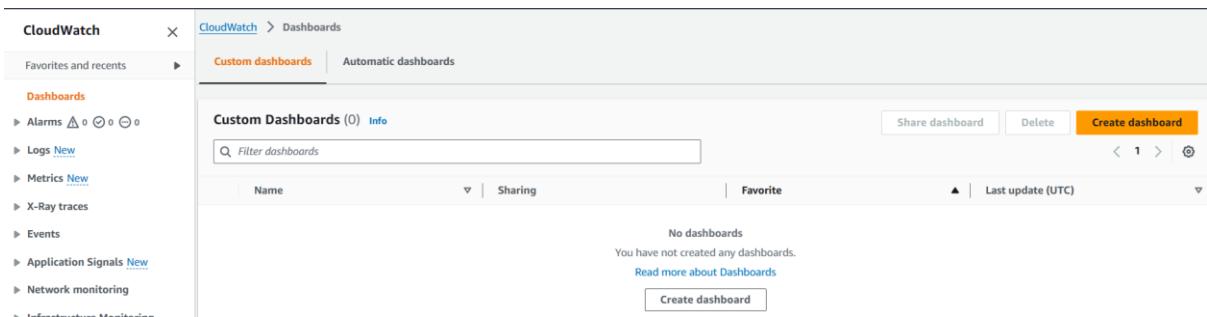


STEP:12 - SETTING UP THE MONITORING & ALTERING THROUGH CLOUDWATCH (OPTIONAL)

→ On service panel search **CloudWatch**, click that one:



→ Then on left side, we can able to see **dashboards**, select that one.
→ Then select **create dashboard**:



→ Then name dashboard, click create dashboard:

Create new dashboard

Dashboard name

nodejs-mysql-deployment

Valid characters in dashboard names include "0-9A-Za-z-_".

Cancel Create dashboard

→ Then select the widget type, according to your preferences:

Add widget

Data sources types - new

Cloudwatch

Other content types

Create data sources

Widget Configuration

Metrics Logs Alarms

Widget type

Line Compare metrics over time

Data table Compare metrics values over time in a table

Number Instantly see the latest value for a metric

Gauge See the latest value of a metric within a range

Stacked area Compare the total over time

Bar Compare categories of data

Pie Show percentage or proportional data

Explorer A single widget with multiple tag-based graphs

Cancel Next

→ Then select EC2:

Add metric graph

Untitled graph

Your CloudWatch graph is empty.
Select some metrics to appear here.

Browse Multi source query - new Graphed metrics Options Source

Metrics (1,762)

Ohio Q Search for any metric, dimension, resource id or account id

ApplicationELB	50	DynamoDB	14	EBS	189	EC2	319
Events	1	Logs	14	RDS	915	S3	2

Add math Alarm recommendations Graph with SQL

→ Then select **per-instance metrics**:

The screenshot shows the CloudWatch Metrics console with the following interface elements:

- Top navigation bar: Browse, Multi source query - new, Graphed metrics, Options, Source.
- Region dropdown: Ohio.
- Filter: All > EC2.
- Search bar: Search for any metric, dimension, resource id or.
- Section title: Metrics (319).
- Table header: Per-Instance Metrics, 319.
- Table body: A single row is visible with the instance ID i-06680a8416bf02a1b, edit, and delete icons.

→ Then copy the instance id of deployment instance, and paste it:

The screenshot shows the CloudWatch Metrics console with the following interface elements:

- Top navigation bar: Browse, Multi source query - new, G.
- Region dropdown: Ohio.
- Filter: All > EC2 > Per-Inst.
- Table body: A single row is selected with the instance ID i-06680a8416bf02a1b, edit, and delete icons.

→ Then select the metrics type according to your preferences, but here I am selecting **CPUUtilization**: click next:

The screenshot shows the CloudWatch Metrics Options screen with the following interface elements:

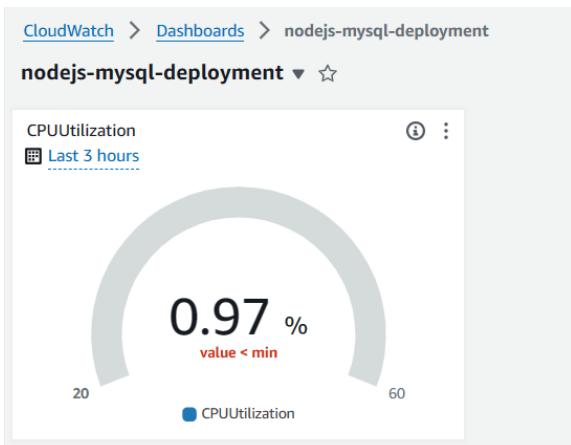
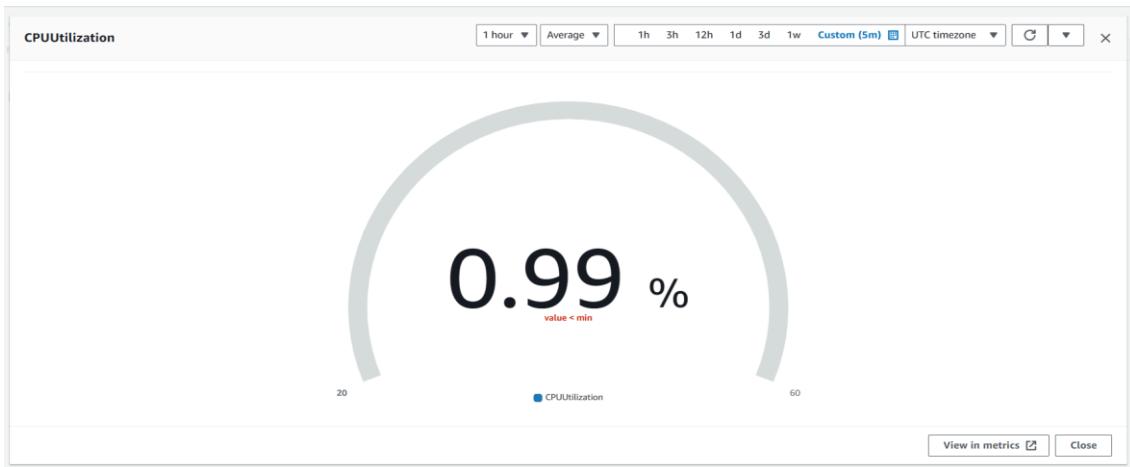
- Selected metric: deployment-instance.
- Selected metric type: CPUUtilization.

→ Then select range according to your preferences, click create dashboard:

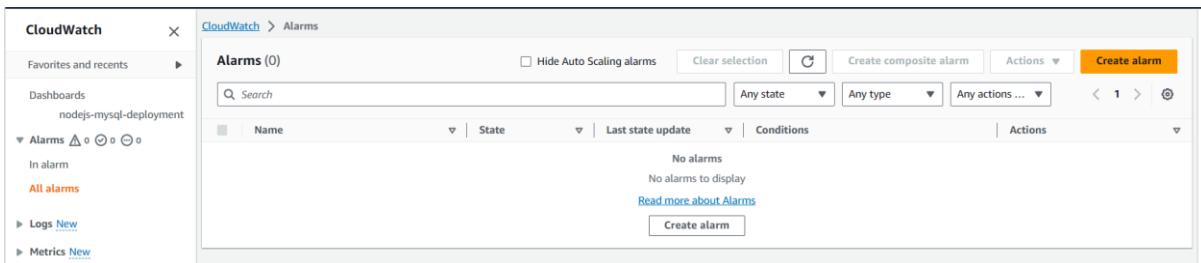
The screenshot shows the CloudWatch Metrics Options screen with the following interface elements:

- Legend position: Bottom (selected).
- Gauge range: Min 20, Max 60.
- Horizontal annotations / thresholds: Add threshold.
- Value: Latest value shows the value from the most recent period of your chosen time range.

→ The dashboard has been setup-ed successfully:



→ Then for alerting, we need to **create CloudWatch alarm**, for that select **In alarm -> create alarm**:



→ Then click **select metrics**:

The "Select metric and conditions" step of the CloudWatch Metrics Selection wizard. The first section is titled "Metric" and contains a "Graph" preview area. The preview area has a "Select metric" button. At the bottom right, there are "Cancel" and "Next" buttons.

→ Then click EC2:

Browse Multi source query - new Graphed metrics Options Source Add math ▾ Add query ▾

Metrics (1,763) Alarm recommendations Graph with SQL Graph search

Ohio ▾ Search for any metric, dimension, resource id or account id

ApplicationELB	50	DynamoDB	14	EBS	189	EC2	319
Events	1	Logs	14	RDS	215	S3	2

Select a single metric to continue

→ Then click per-instance metrics:

Browse | [Multi source query - new](#) | [Graphed metrics](#) | [Options](#) | [Source](#)

Metrics (319)

Ohio ▾ All > EC2 Search for any metric, dimension, resource id

Per-Instance Metrics	319

→ Then select the metrics type for deployment instance: **click select metric:**

Browse	Multi source query - new	Graphed metrics (1)	Options	Source	Add math ▾	Add query ▾
<input type="checkbox"/>	deployment-instance	i-06680a8416bf02a1b	CPUSurplusCreditsCharged ⓘ	No alarms		
<input type="checkbox"/>	deployment-instance	i-06680a8416bf02a1b	CPUSurplusCreditBalance ⓘ	No alarms		
<input type="checkbox"/>	deployment-instance	i-06680a8416bf02a1b	DiskWriteOps ⓘ	No alarms		
<input checked="" type="checkbox"/>	deployment-instance	i-06680a8416bf02a1b	CPUUtilization ⓘ	No alarms		
<input type="checkbox"/>	deployment-instance	i-06680a8416bf02a1b	DiskReadOps ⓘ	No alarms		

→ The metrics has been selected successfully for the alarm:

Specify metric and conditions

Alarm recommendations
[View details](#)

Metric
[Edit](#)

Graph

This alarm will trigger when the blue line goes above the red line for 1 datapoints within 5 minutes.

Percent

74.7

37.6

0.6

08:30 09:30 10:30 11:30

CPUUtilization

X

Namespace

Metric name

InstanceId

Instance name

Statistic

X

Period

▼

→ Then select the **threshold as static**, and enter the threshold value according to your preferences: then click next

Conditions

Threshold type

- Static Use a value as a threshold
- Anomaly detection Use a band as a threshold

Whenever CPUUtilization is...

Define the alarm condition.

- Greater > threshold
- Greater/Equal >= threshold
- Lower/Equal <= threshold
- Lower < threshold

than...

Define the threshold value.

60

Must be a number

▶ Additional configuration

Cancel **Next**

→ Then for the **notification**: select an existing SNS Topic, or else click **create new topic** for sending notification through email.

Configure actions

Notification

Alarm state trigger

Define the alarm state that will trigger this action.

In alarm The metric or expression is outside of the defined threshold.

OK The metric or expression is within the defined threshold.

Insufficient data The alarm has just started or not enough data is available.

Remove

Send a notification to the following SNS topic

Define the SNS (Simple Notification Service) topic that will receive the notification.

Select an existing SNS topic

Create new topic

Use topic ARN to notify other accounts

Send a notification to...

cloudformation_trigger X

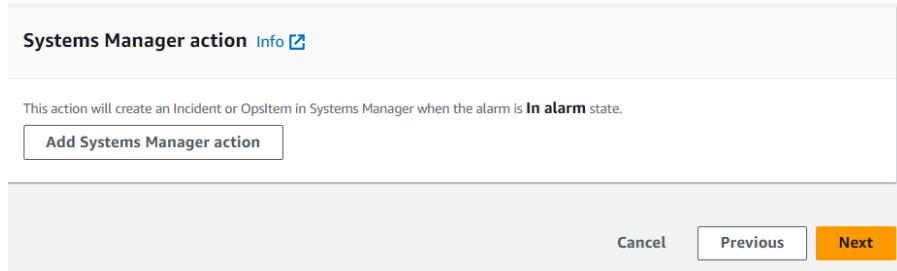
Only email lists for this account are available.

Email (endpoints)

jravi1605@gmail.com - View in SNS Console

Add notification

→ Then click **next**:



→ Then enter **alarm name** according to your preferences, click **next**

This screenshot shows the 'Add name and description' configuration screen. It has a title 'Name and description'. Under 'Alarm name', the value 'nodejs-application' is entered. In the 'Alarm description - optional' section, there is a 'Edit' button and a 'Preview' button. The preview area contains the following Markdown code:
This is an H1
double asterisks will produce strong character
This is [an example](https://example.com/) inline link.
Up to 1024 characters (0/1024).
A note at the bottom states: 'Markdown formatting is only applied when viewing your alarm in the console. The description will remain in plain text in the alarm notifications.' At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons, where 'Next' is highlighted in orange.

→ Then just **review and create** page will appear, just review the configurations, **click create alarm**:

This screenshot shows the 'Preview and create' configuration screen. It starts with 'Step 1: Specify metric and conditions' with an 'Edit' button. Below that, a 'Metric' section is shown. At the bottom, there are 'Cancel', 'Previous', and 'Create alarm' buttons, where 'Create alarm' is highlighted in orange.

→ The alarm has been set successfully:

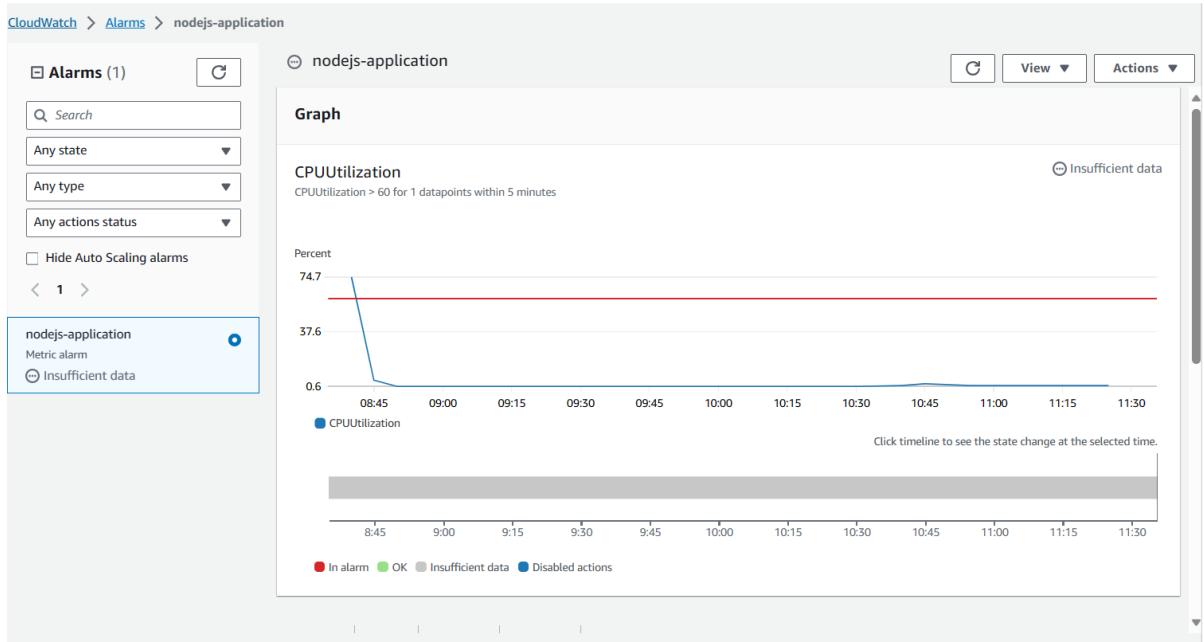
The screenshot shows the CloudWatch Alarms interface. At the top, a green banner displays the message "Successfully created alarm nodejs-application." Below the banner, the title "CloudWatch > Alarms" is visible. The main area is titled "Alarms (1)". There are several buttons at the top right: "Hide Auto Scaling alarms" (unchecked), "Clear selection" (button), "Create composite alarm" (button), "Actions" (dropdown), and "Create alarm" (button). A search bar is present. The table below lists one alarm:

Name	State	Last state update	Conditions	Actions
nodejs-application	Insufficient data	2023-12-23 11:35:12	CPUUtilization > 60 for 1 datapoints within 5 minutes	Actions enabled

In the second screenshot, the same interface is shown, but the alarm is now in an "OK" state. The "nodejs-application" row is highlighted with a blue border.

Name	State	Last state update	Conditions	Actions
nodejs-application	OK	2023-12-23 11:35:55	CPUUtilization > 60 for 1 datapoints within 5 minutes	Actions enabled

→ To know more about alarm, select that alarm:



*****THE END*****