Experiment-6

Date: 12/3/25

Aim: Deploy the artifact on the Tcat Server

Procedure:

- Setup Jenkins on AWS.
- Build WAR file.
- Connect to sit machine with git bash.
- Update the apt repository.

> sudo apt-get update

- Install tomcat 9: sudo apt-get install -y tomcat9
- Now install tomcat9 admin as well.
  sudo apt-get install tomcat9-admin.
- To access the tomcat, take the public IP of sit server and add: 8080, copy paste this command on browser.
- Now go to the path: cd /etc/tomcat9
- ll
- Now we need to add user in the tomcat-users.xml

  sudo vim tomcat-users.xml

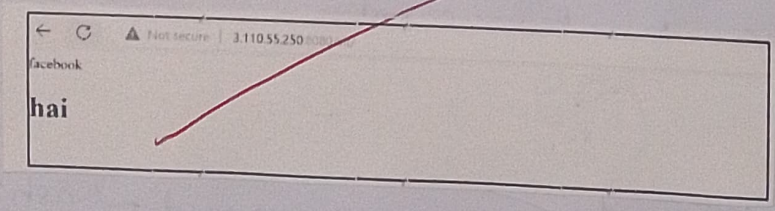
  Go to insert mode (I)

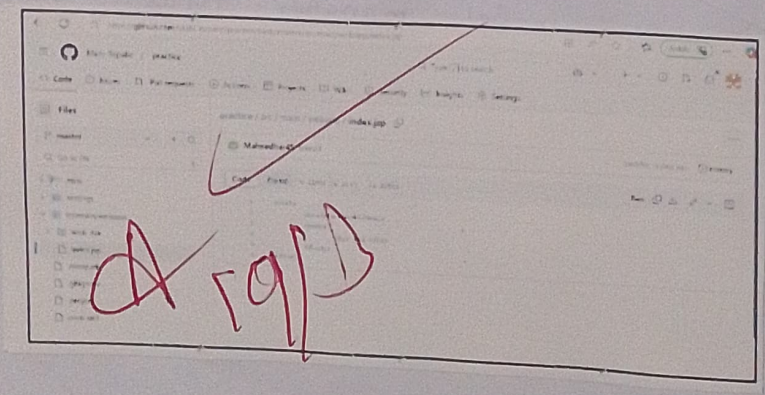
  Add the following lines

  < user username = "ashvith" password = "ash12345"
  roles = "manager-script, manager-status, manager-gui" />

  esc :wq
- Now restart the tomcat service
  sudo service tomcat9 restart
- Now you can add one plugin in the jenkins (deploy to container)
- Manage jenkins > plugins > available plugins > select deploy to container and install.

- Go to jenkins dashboard, select the war job > select configure.
- Select the post build action and search deploy war/ear to a container.
- Add container tomcat 9.
- Add the credentials.
- select the credentials and give the tomcat-url (sit server path along with 8080 port number)
- Click on Apply and save.
- Run the build now
- How can we check the artifact is deployed or not?
- It can be done as follows:
  Take the sit public IP address and port-no. give the context path name.
  ex: http://3.110.55.250:8080/sit
- The output will be displayed.





VIVA QUESTIONS

1. What is an artifact in DevOps?

Ans. An artifact is a compiled or built file, such as a JAR, WAR, binary, or Docker image, created during the build stage of the software development lifecycle and deployed to environments like testing, staging or production.

2. How do you deploy an artifact to a test server?

Ans. Deployment can be done via CI/CD tools (like Jenkins, GitLab CI, GitHub Actions) using SSH, SCP, FTP, or Docker/Kubernetes.
Example using scp:
scp myapp.war user@test-server:/opt/tomcat/webapps/

3. Which tools are commonly used to deploy artifacts?

Ans.
- Jenkins
- Ansible
- GitHub Actions
- GitLab CI/CD
- Docker + Kubernetes
- Nexus/Artifactory (for storing)

4. What is a test environment in DevOps?

Ans. A test environment mimics production where new builds or features are tested before release. It often includes servers, databases and services similar to production.

5. What is the role of Jenkins in deploying artifacts?

Ans. Jenkins automates deployment by pulling the latest build, copying the artifact to the server and restarting services using a pipeline job.

A 19/3

Experiment-H 7

Perform automation using Jenkins

Date: 19/3/25

PROCEDURE

Step-1: Install Jenkins

Install Java

Access Jenkins

Step-2: Configure Jenkins

Step-3: Create a New Job for Automation

Step-4: Create a Pipeline (Optional)

Step-5: Configure Post-Build Actions

Step-6: Schedule Jobs and Monitor

Step-7: Secure and Maintain Jenkins

SOURCECODE

Whenever we make some changes in the source file, jenkins job will automatically fetch the code from github to dev server and build the war file and deploy into sit.

Git commands involved in the automation process are:

- git clone https://github.com/AshvithKumar2005/DevOps2.git
  this copies the remote repository to your local machine.
- ll
- cd practice
- ll
- cd src
- ll
- cd main
- ll
- cd webapp
- ll
- vim index.jsp
- git status
  shows modified/untracked files in working directory
- git add
  stages all changed files for commit
- git status
- git commit -m "updated"

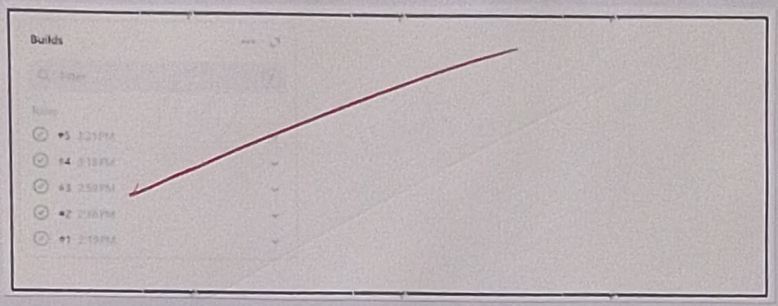
creates a snapshot of changes with message "updated"

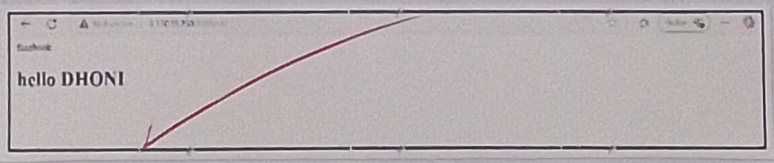
- git push origin master
  uploads local commits to the remote repository's main branch.
- Go to Jenkins > right click on project > configure > triggers > give "* * * * *" in schedule > Apply > Save.
- Now we don't have to build again.

- Jenkins will automatically trigga the job.

#26/2

OUTPUT





9/2612

VIVA QUESTIONS

1. How does Jenkins support Continuous Integration and Continuous Deployment (CI/CD?

Ans. By automating code integration, testing, deployment through pipelines triggered by code changes.

2. Which Java version is recommended for Jenkins installation?

Ans. Java 11

3. What are the key plugins required for basic automation in jenkins, and how do you install them?

Ans. Key plugins: Git, Pipeline, Docker and Ansible. Installation: Manage Jenkins > Manage Plugins > Available > Search and install plugin.

4. What are the different build triggers available in jenkins?

Ans. Poll SCM, Build periodically, GitHub Hook trigger, Trigger via URL.

5. How do you configure email notifications for build results in jenkins?

Ans. Install email extension Plugin, go to manage jenkins > Configure system, set SMTP details, then add post build actions to send emails.

Experiment-9 8

Build and Deploy a Grid for Chrome and Firefox based Testing

Date: 26/3/25

PROCEDURE

Steps to Build and Deploy Selenium Grid

Prerequisites

1. Java Development Kit (JDK) installed on the system.
2. Selenium Server JAR file (latest version).
3. Latest ChromeDriver and GeckoDriver (for Firefox) downloaded.
4. Installed web browsers (Chrome and Firefox).

Step-1: Set Up the Hub

i. Open a terminal or command prompt.

ii. Navigate to the directory containing the Selenium Server JAR file!

iii. Start the Selenium Hub using the following command:

1. java -jar selenium-server-standalone-<version>.jar hub

iv. Verify the hub is running by visiting http://localhost:4444/grid/console in your browser.

Step-2: Set Up Node for Chrome

Step-3: Set Up Node for Firefox

Step-4: Configure the Grid

Step-5: Write and Execute Test Scripts

Step-6: Monitoring and Maintenance

SOURCE CODE

Step 1: Launch AWS EC2 Instance:-
Go to AWS EC2 console.
Launch a new instance with the following settings:
- Name: SeleniumGridServer
- Amazon Machine Image: Ubuntu Server 22.04 LTS (Free Tier)
- Instance Type: t2.micro
- Key Pair: Create New Key Pair or Select one
- Add Security Group Rule:
    1) SSH (default)
    2) Custom TCP (port: 4444, Sourcetype: Anywhere)

Step 2: Connect to EC2 Instance via Mobaxtream
Step 3: Install Docker and Docker Compose:-
1. sudo apt update
To Install Docker
2. sudo apt install -y docker.io
To start and enable Docker
3. sudo systemctl start docker
4. sudo systemctl enable docker
To Install Docker Compose
5. sudo curl -L "https://github.com/docker/compose/releases/download/v2.17.3/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
6. sudo chmod +x /usr/local/bin/docker-compose
To check versions
7. docker --version
8. docker-compose --version
Step 4: Create Selenium Grid with Docker Compose:-
1. mkdir selenium-grid && cd selenium-grid
2. nano docker-compose.yml
Paste this code:

```
version: "3"
services:
  selenium-hub:
```

```
image: selenium/hub:4.0.0-rc-2-20210930
container_name: seleniumHub
ports:
  - "4444:4444"
chrome:
  image: selenium/node-chrome:4.0.0-rc-2-20210930
  container_name: chromeNode
  depends_on:
    - selenium-hub
  environment:
    - SE_EVENT_BUS_HOST = selenium-hub
    - SE_EVENT_BUS_PUBLISH_PORT = 4442
    - SE_EVENT_BUS_SUBSCRIBE_PORT = 4443
  shm_size: 2g
firefox:
  image: selenium/node-firefox:4.0.0-rc-2-20210930
  container_name: firefoxNode
  depends_on:
    - selenium-hub
  environment:
  - SE_EVENT_BUS_HOST = selenium-hub
  - SE_EVENT_BUS_PUBLISH_PORT = 4442
  - SE_EVENT_BUS_SUBSCRIBE_PORT = 4443
  shm_size: 2g
```

Save (Ctrl+O, Enter), then exit (Ctrl+X)

Step 5: Start the Selenium Grid :-

To Run the grid :

```
sudo docker-compose up -d
```

To check containers :

```
sudo docker ps
```

Step 6: Access Selenium Grid UI :-

open in your browser :

```
http://<Your-EC2-Public-IP>:4444/ui
```

OUTPUT

Step 7: Run a Sample Python Test:-

Create a Virtual Environment:

Install Python and Selenium:

```
sudo apt install python3-venv python3-full -y
python3 -m venv venv
source venv/bin/activate
pip install selenium
nano test_grid.py
```

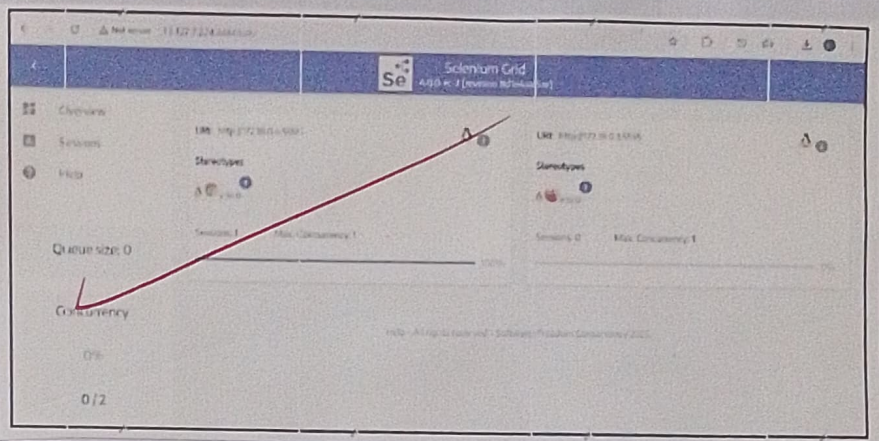
Paste below Code:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
browser = "chrome"
GRID_URL = "http://localhost:4444/wd/hub"
options = None
if browser == "chrome":
    options = webdriver.ChromeOptions()
elif browser == "firefox":
    options = webdriver.FirefoxOptions()
else:
    raise Exception("Unsupported browser!")
driver = webdriver.Remote(
command_executor = GRID_URL,
    options = options
)
driver.get("https://www.google.com")
print("Title:", driver.title)
driver.quit()
```

Save (Ctrl+O, Enter), then exit (Ctrl+X)

```
python3 test_grid.py
```

```
ubuntu@ip-172-31-2-67:~/selenium-grid$ sudo docker-compose up -d
[+] Running 3/3
 ✔ firefox 17 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                    29.2s
 ✔ selenium-hub 12 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                    11.9s
 ✔ chrome 15 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                    33.0s
[+] Running 4/4
 ✔ Network selenium-grid_default  Created                                             0.1s
 ✔ Container seleniumHub          Started                                             1.2s
 ✔ Container chromeNode           Started                                             1.8s
 ✔ Container firefoxNode          Started                                             1.8s
ubuntu@ip-172-31-2-67:~/selenium-grid$
```



VIVA QUESTIONS

1. What are the differences between the Selenium Hub and Nodes?

Ans. The hub is the control server that controls and distributes test to different machines, while Nodes are the machines that execute the test based on Hub's instructions.

2. What is the role of the chromedriver and geckodriver in Selenium Grid?

Ans. Chromedriver and geckodriver act as a bridge between selenium Commands and the respective browsers (chrome and firefox), enabling browser automation on the Nodes.

3. How would you configure a JSON file for a node? Can you give an example of its structure?

Ans. A JSON file for a Node defines it capabilities and the Hub connection detail
E.g :- {"capabilities":[{"browserName":"chrome"}],"configuration":
{"hub":"http://localhost:4444/grid/register"}}

4. How do you configure your test script to use the grid for Chrome testing?

Ans. By setting the RemoteWebDriver to point the Hub URL and specifying chrome capabilities.
Eg: WebDriver driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"), DesiredCapabilities.chrome());

5. What are the best practices for optimizing the performance of Selenium Grid?

Ans. Use parallel execution, limit browser instance per node, distribute load across multiple nodes, monitor node health, and use Docker or kubernetes for scalable and efficient resource management.

Experiment - 9

Date: 9/4/25

Aim: create deployment resource using kubernetes

Procedure:

- Create the google cloud console free account
- It is a two step process
- NOTE: Don't activate the full account
- Once the account is created, you can login to google cloud console.
- Now create the kubernetes cluster
- Open the cloud shell and run the following commands.
- To see the cluster list

  > gcloud container clusters list

- Create a cluster

  > gcloud container clusters create my-cluster --zone us-central1-a

- cluster creation takes 5-10 minutes
- Once the cluster is created you can see the below message automatically.

  STATUS : RUNNING

- Now go and check kubernetes engine --> cluster, you can see my-cluster is running.
- Run the below command:

  > gcloud container clusters get-credentials my-cluster --zone us-central1-a

- To see the list of nodes

  > kubectl get nodes

- Create the pods

  > kubectl run --image tomcat webserver

- To see pods list

  > kubectl get pods

- To get the list of pods along with ip address and which node the pod is running

  > kubectl get pods -o wide

- Actually you can create the pod using definition file.

  > create pd-df1.yaml

  > vim pd-df1.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: jenkins-pod
spec:
  containers:
  - name: myjenkins
    image: jenkins/jenkins
    ports:
    - containerPort: 8080
      hostPort: 8080
```

- for accessing the application, you need to open the port

  > gcloud compute firewall-rules create rule2 --allow tcp:8080

  > kubectl create -f pd-df1.yaml

> kubectl get pods -o wide

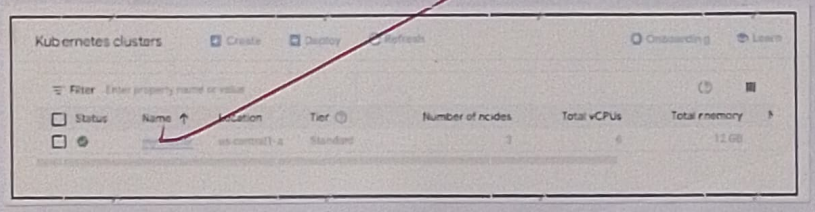
> kubectl get nodes -o wide

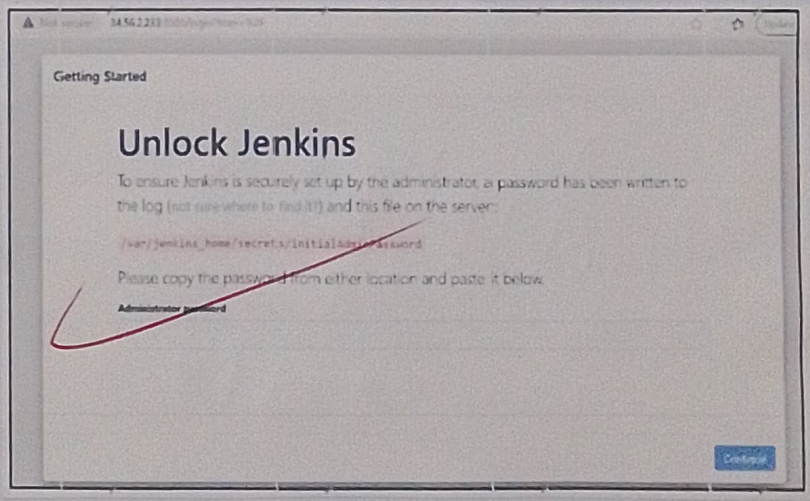
. To access the pod, take the external ip, and add the port no 8080.

. open browser and parse Ipaddress:8080

. Now we can see jenkins.

Output:





16/M

VIVA QUESTIONS

1. What is a Deployment in Kubernetes?

Ans:- A Deployment in Kubernetes is a controller that provides declarat updates for Pods and Replicasets. It allows you to define the desire state for your application and the Deployment Controller will manag the state for you.

2. What command is used to apply a Deployment YAML file?

Ans:- kubectl apply -f deployment.yaml

3. What is the purpose of the replicas field in a Deployment?

Ans:- The replicas field specifies the number of pod replicas that should be running at any given time. Kubernetes ensures that this number is maintained by starting new pods or terminating existing ones.

4. How can you update an application using Deployment?

Ans:- You can update the container image or other configuratim in the Deployment YAML and reapply it using kubectl apply - Kubernetes will perform a rolling update without downtime.

5. How do you roll back a Deployment?

Ans:- kubectl rollout undo deployment/my-app

16/4

Experiment-6 10

Create a Docker Image for any Application using Docker File and Push it to Docker Hub

Date: 16/4/25

PROCEDURE

Step-by-Step Procedure: Build Docker Image and Push to Docker Hub Using Jenkins

Step-1: Install Required Plugins

Step-2: Configure the SSH Plugin

Step-3: Prepare the Environment

1. Install Docker on the Target Node
2. Prepare Dockerfile
3. Commit Dockerfile to Git Repository

Step-4: Jenkins Job for Building and Pushing Docker Image

1. Create New Jenkins Job
2. Link Artifacts from Previous Job
3. Transfer Artifacts to Target Node
4. Build Docker Image
5. Tag and Push Docker Image

Step-5: Configure Docker Credentials in Jenkins

Step-6: Test the Job

Step-7: Troubleshooting Common Issues

1. Permission Denied for Docker Commands
2. Authentication Error

Step-8: Future Enhancements

SOURCE CODE

A) CREATE AN INSTANCE

Launch an EC2 Instance with the name 'nodejs' in AWS.

B) INSTALL nginx, Nodejs, Npm, pm2

. Login connect to the dev terminal through gitBash by pasting the SSH Key of the 'nodejs' instance.

. switch to root

```
> sudo su -
```

. Update the packages

```
> apt update -y
```

. Install Nginx webserver

```
> apt install nginx -y
```

Check the status of nginx webserver

```
> systemctl status nginx
```

. Install Nodejs

```
> apt install nodejs -y
```

Check whether it is installed through version

```
> nodejs --version
```

. Install NPM (Node Package Manager) and check its version

```
> apt install npm -y
> npm --version
```

- Install pm2 (Procen manager)

```
> npm install -g pm2
```

C) CREATING A NODEJS APPLICATION

- First, using nano or any other text editor, create a sample application called hello.js inside the home directory.
- Go to the home directory

```
>cd /home
```

- Open a hello.js file and paste the code

```
>nano hello.js
const http = require ('http');
const hostname = '0.0.0.0';
const port = 3000;
const server = http.createServer ((req, res) => {
        res.statusCode = 200;
        res.setHeader ('Content-Type', 'text/plain');
        res.end('Hello World!\n');
});
server.listen(port, hostname, () => {
  console.log('Server running at http://${hostname}:
                                        ${port}/');
});
```

- Save the file and exit from the editor

```
Save : Ctrl + O
Exit : Ctrl + X
```

- This Node.js application listens on the specified address (0.0.0.0) and port (3000), and returns "Hello World!" with a 200 HTTP success code.

To test your application, type:

> node hello.js

start node application in pm2

> pm2 start hello.js --name app

- set-up node application in nginx as a reverse proxy

The application is running and listening on localhost. To allow the users to access it, set up the Nginx web server as a reverse proxy.

Set up the Nginx configuration in the /etc/nginx/sites-available/example.com file. Open this file for editing:

```
server {
    listen 80;
    server_name 13.208.246.123 (replace with your server ip)
    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Create a symink for this example.com

> ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/

- Restart your nginx webserver

> systemctl restart nginx

- Check your application is running in browser using IP

2) SETTING UP IN DOCKER

- Install docker

> apt install -y docker.io

D.

- Install Docker compose and check its version
  > apt install -y docker-compose
  > docker-compose --version
- Navigate to the application directory
  > cd /home/node
- Create a Dockerfile in the /home/node directory and type this code.

```
#Use Node.js base image
FROM node:12
#Set the working directory inside the container
WORKDIR /app
#Install dependencies
RUN npm install
#Copy the application code
COPY..
#Expose port 3000
EXPOSE 3000
#Run the application
CMD ["node", "hello.js"]
```

- Save it and exit from editor
- Create a .dockerignore file to exclude unnecessary files from the image.

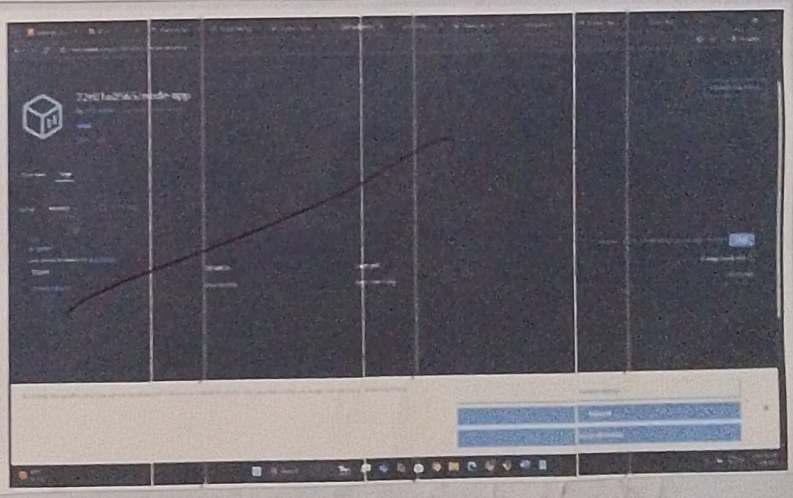
```
node_modules
npm-debug.log
```

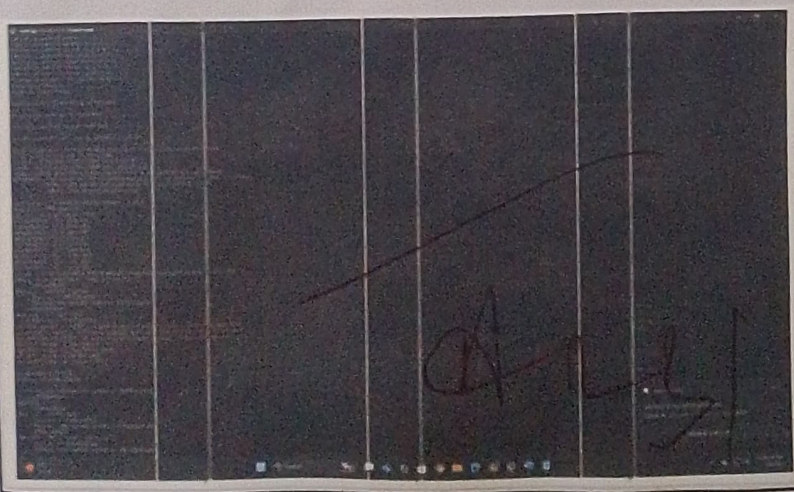
- Save it and exit from editor
- Build the docker image
  > docker build -t arunhub01/node-app:latest . (replace arunhub01 with docker username)
- verify if the image is built or not
  > docker images

OUTPUT

E) PUSH THE IMAGE TO DOCKER HUB

- Login to Docker Hub
  > docker login
  Enter docker hub username and password.
- Push the image to docker hub
  > docker push arunhub01/node-app:latest
- Go to the Docker Hub and check repo is there or not.





4

VIVA QUESTIONS

1. What is Docker, and how does it integrate with Jenkins?

Ans. Docker packages appear in containers, Jenkins uses Docker for consistent builds, tests and deployment across environment

2. What is Docker, and how does it integrate with Jenkins?

Ans. Docker packages app in container, Jenkins uses Docker for consistent builds, lists and deployment across environments

3. What is the purpose of the Dockerfile in this pipeline?

Ans. The Dockerfile defines the container environment, Jenkins uses to build, test and run the application consistently.

4. Why is the source code repository skipped in the Build and Push Image job.

Ans. The source code is already build the job, only needs the compiled output to build and push the image.

5. How is the Docker image pushed to Docker Hub?

Ans. Jenkins uses Docker CLI to tag the image, then authentica and pushes it to Docker Hub repository.

Experiment-11:

Date: 23/4/25

Aim: Setup Grafana for DevOps

Procedure:

- Create a cluster on google cloud
- To get the username of grafana

> kubectl get secret prometheus-grafana -n monitoring -o jsonpath="{.data.admin-user}" | base64 --decode ; echo

- To get the password for grafana

> kubectl get secret prometheus-grafana -n monitoring -o jsonpath="{.data.admin-password}" | base64 --decode ; echo

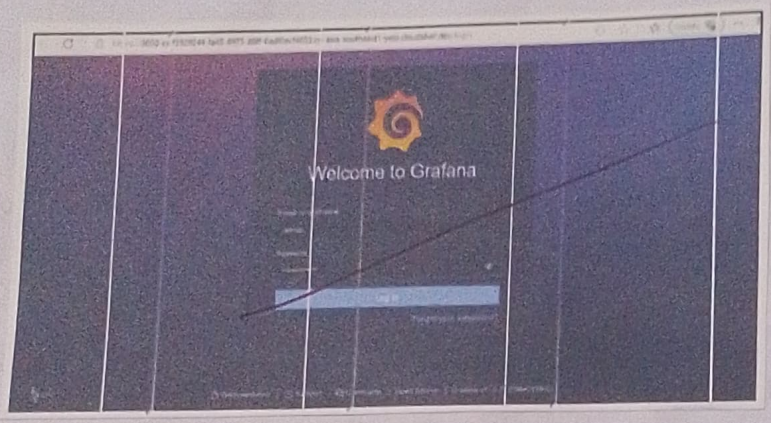
- Port forwarding:

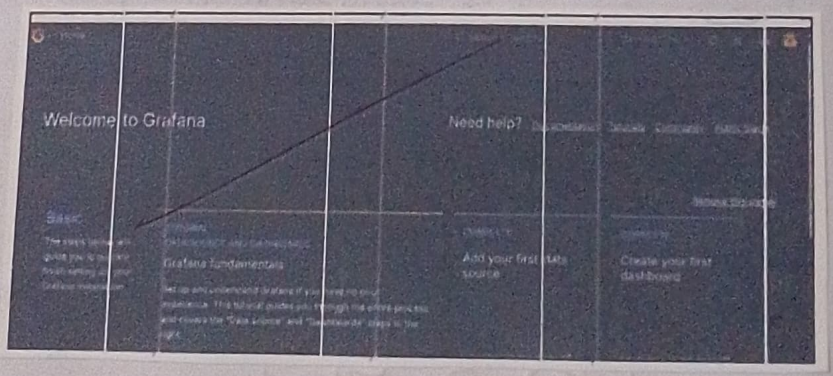
> kubectl port-forward svc/prometheus-grafana 3000:80 -n monitoring

- click on web preview give the post no 3000 and click on change and preview.
- Open in browser.
- Login with "admin" as username and "prom-operator" as password.

23/4

Output:


Welcome to Grafana


Welcome to Grafana
Need help?
Grafana fundamentals
Add your first data source
Create your first dashboard

2/3/4

VIVA QUESTIONS

1. What is Grafana?

Ans. Grafana is an open-source data visualization tool used for monitoring, alerting, and analyzing metrics from various data sources, including Prometheus.

2. What are the main features of Grafana?

Ans. → Interactive and customizable dashboards.
→ Multiple data source support (Prometheus, InfluxDB, Elasticsearch, etc)
→ Alerting and notification system
→ Role-based access control.

3. How do you install Grafana?

Ans. • Using apt:

```
sudo apt-get install -y grafana
```

• Using Docker:

```
docker run -d -p 3000:3000 grafana/grafana
```

• Access it at http://localhost:3000

4. What are Grafana dashboards?

Ans. Dashboards in Grafana are collections of visual panels that display metrics and logs in real-time.

5. How do you connect Grafana to Prometheus?

Ans. • Add Prometheus as a data source.
• Enter Prometheus URL (e.g., http://localhost:9090).
• Save and test the connection.

Experiment-12:

Date: 23/4/25

Aim: Setup Prometheus for DevOps.

Procedure:

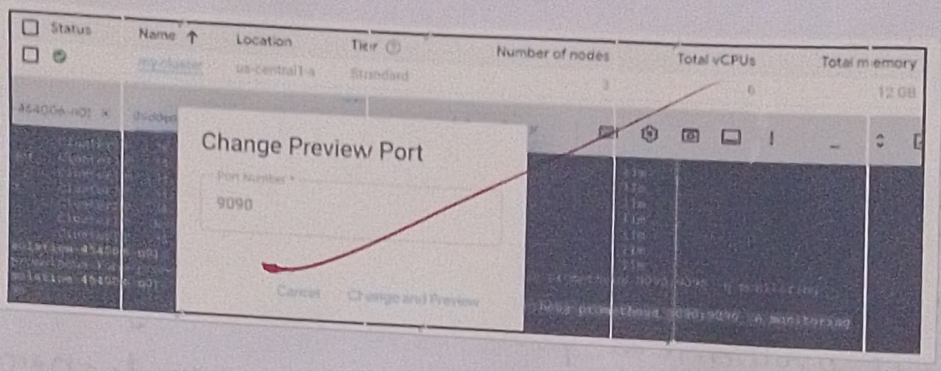
- Create a cluster on google cloud.
- > helm repo add prometheus https://prometheus-community.github.io/helm-charts
- > helm repo update
- > helm install prometheus prometheus-community/kube-prometheus-stack --namespace monitoring --create-namespace
- The above command will install prometheus, alertmanager and grafana.
- Check the prometheus pods and services.

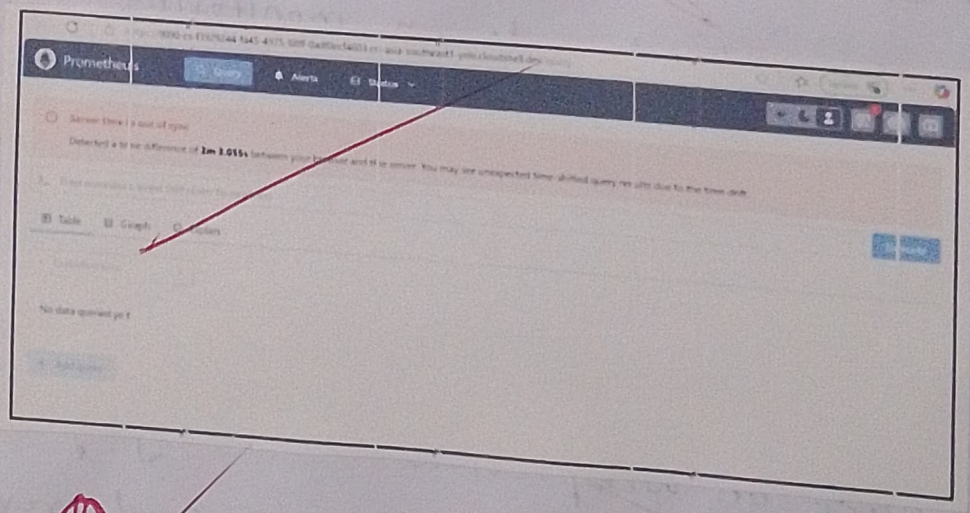
  > kubectl get pods -n monitoring

  > kubectl get svc -n monitoring
- Access prometheus and port forwarding

  > kubectl port-forward svc/prometheus-kube-prometheus-prometheus 9090:9090 -n monitoring
- Click on webpreview and change port number to 9090 and then click on change and preview.
- Now you can see prometheus in the browser.

Output:


Status
Name
Location
Tier
Number of nodes
Total vCPUs
Total memory
us-central1-a
Standard
12 GB
Change Preview Port
Port Number *
9090
Cancel
Change and Preview


Prometheus
Query
Alerts
Status
Table
Graph
No data queried yet

VIVA QUESTIONS

1. What is Prometheus?

Ans. Prometheus is an open-source systems monitoring and alerting toolkit. Particularly well suited for monitoring dynamic, cloud-native environments such as Kubernetes. It uses a pull-based model to scrape metrics from configured endpoints.

2. What are the key concepts in prometheus?

Ans.
- Metrics: Datapoints collected over time, usually in the form of time series.
- PromQL: Prometheus Query Language used to query the collected metrics.
- Exporters: Components that expose metrics in a format that Prometheus can scrape.
- Alertmanager: Manages alerts generated by Prometheus

3. How to install prometheus?

Ans.
```
wget https://github.com/prometheus/prometheus/releases/download/v2.30.0/prometheus-2.30.0.linux-amd64.tar.gz
tar xvfz prometheus-*.tar.gz
cd prometheus-*
./prometheus --config.file=prometheus.yml
```

Docker:

```
docker run -p 9090:9090 prom/prometheus
```

4. How to configure Prometheus?

Ans. Basic prometheus.yml Configuration:

```
global:
  scrape_interval: 15s
```

```
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```

Adding Targets:

```
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
```

5. Basic Queries for PromQL

Ans. Basic:

```
up
rate(http_requests_total[5m])
```

Aggregations:

```
sum(rate(http_requests_total[5m]))
avg_over_time(http_requests_total[5m])
```

Recording Rules:

```
groups:
  - name: example
    rules:
    - record: job:http_inprogress_requests:sum
      expr: sum(http_inprogress_requests) by (job)
```