ProgressSoft Assignment Solutions:

Mohamed Mohamed Shehata Badawy

Backend Developer & Cloud DevOps Engineer

Github Repository

https://github.com/shehata18/ProgressSoft-Assignment

ProgressSoft Assignment Solutions:

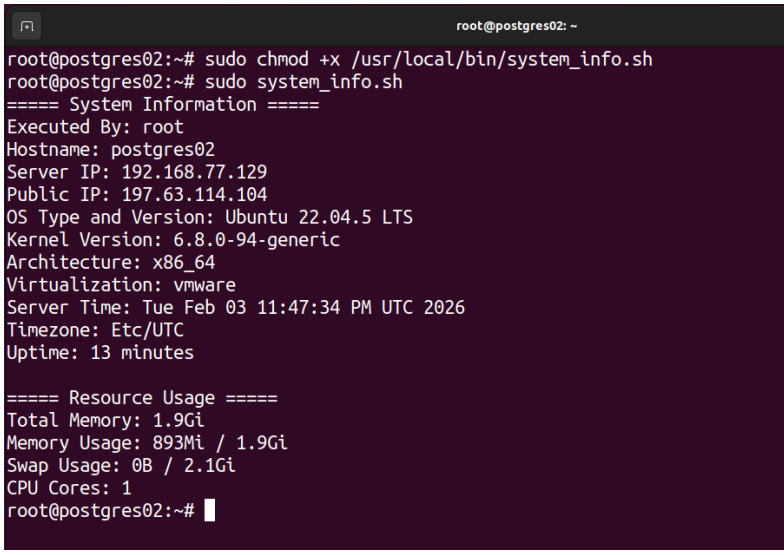
Section 1: Linux

1. OS Information Script

To gather comprehensive system information and resource utilization details, a Bash script named system_info.sh was developed. This script leverages standard Linux commands to extract data such as the executing user, hostname, IP addresses, operating system type and version, kernel version, architecture, virtualization technology, server time, time zone, system uptime, total memory, memory usage, swap usage, and the number of CPU cores. The script is designed to provide a quick overview of the system's configuration and current state.

Script Content (system_info.sh):

```
#!/bin/bash

export LANG=en_US.UTF-8
export LC_ALL=en_US.UTF-8

echo "===== System Information ====="
echo "Executed By: $(whoami)"
echo "Hostname: $(hostname)"
echo "Server IP: $(hostname -I | awk '{print $1}')"
echo "Public IP: $(curl -s ifconfig.me)"
echo "OS Type and Version: $(lsb_release -d | cut -f2)"
echo "Kernel Version: $(uname -r)"
echo "Architecture: $(uname -m)"
echo "Virtualization: $(systemd-detect-virt)"
echo "Server Time: $(LC_TIME=en_US.UTF-8 date '+%a %b %d %I:%M:%S %p %Z %Y')"
echo "Timezone: $(timedatectl show -p Timezone --value)"
echo "Uptime: $(awk '{print int($1/60) " minutes"}' /proc/uptime)"

echo "===== Resource Usage ====="
echo "Total Memory: $(free -h | awk '/Mem:/ {print $2}')"
echo "Memory Usage: $(free -h | awk '/Mem:/ {print $3 " / " $2}')"
echo "Swap Usage: $(free -h | awk '/Swap:/ {print $3 " / " $2}')"
echo "CPU Cores: $(nproc)"
```



2. User and Group Management:

To establish a new user account with specific group memberships, the following commands were utilized. A user named PS was created, assigned to the primary group PSGroup , and also added to the secondary group dba . This setup ensures appropriate permissions and access control for the user within the system.

```
sudo groupadd PSgroup

sudo groupadd dba

sudo useradd -m -g PSgroup -G dba PS

sudo passwd PS

id PS
```

3. Root Password Modification:

```
sudo passwd root
```

```
root@postgres02:~#
root@postgres02:~# sudo groupadd PSgroup
root@postgres02:~# sudo groupadd dba
root@postgres02:~# sudo useradd -m -g PSgroup -G dba PS
root@postgres02:~# sudo passwd PS
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
root@postgres02:~#
root@postgres02:~#
root@postgres02:~# passwd root
New password:
Retype new password:
passwd: password updated successfully
root@postgres02:~#
```

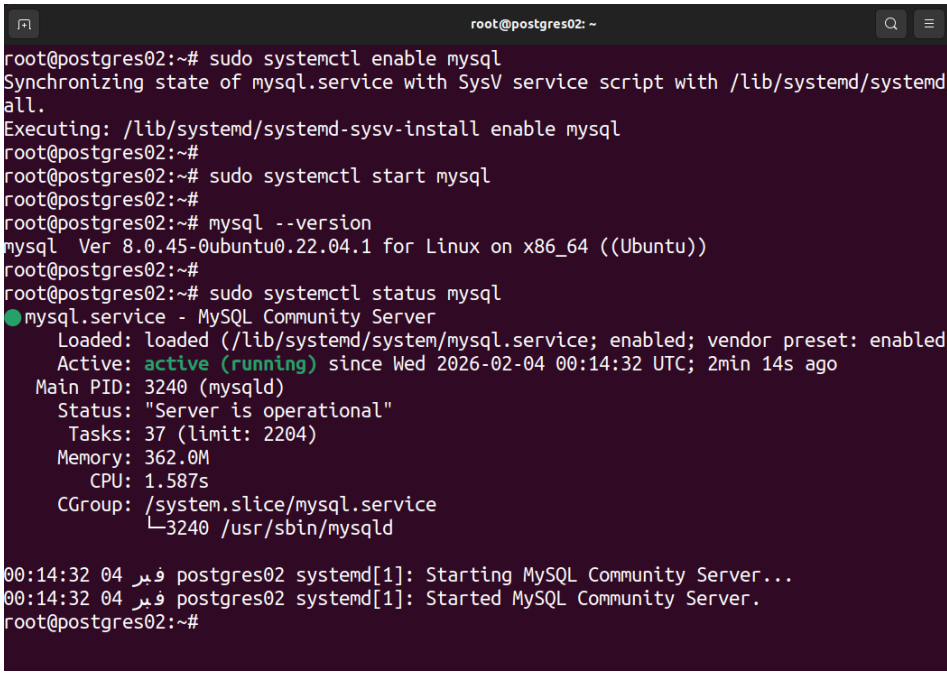
4. Software Installation (MySQL & HAProxy)

MySQL Server and HAProxy were installed on the Ubuntu system using the apt install command. MySQL provides robust relational database capabilities, while HAProxy acts as a high-performance load balancer and reverse proxy. Verifying their installed versions confirms successful deployment.

Install MySQL

```
sudo apt install mysql-server -y

sudo systemctl enable mysql

sudo systemctl start mysql
```

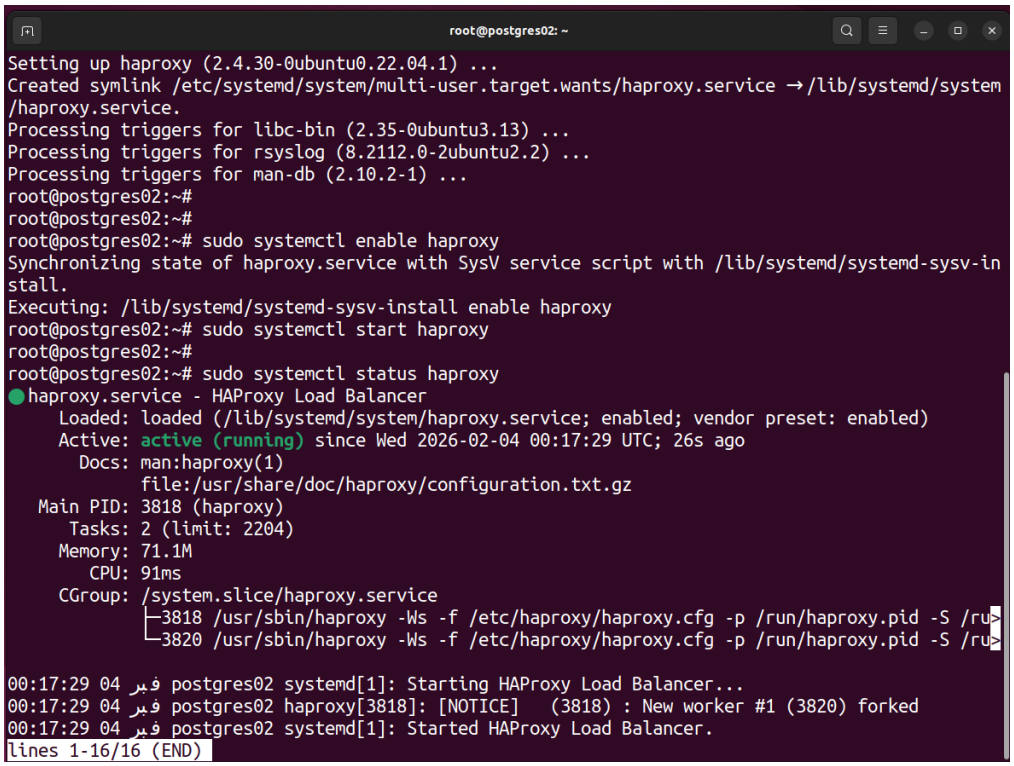
```
mysql --version
```



Install HAProxy

```
sudo apt install haproxy -y

sudo systemctl enable haproxy

sudo systemctl start haproxy
```

```
haproxy -v
```



5. Allow TCP/UDP only on port 3306 (Firewall)

Step 1: Enable UFW

```
sudo ufw enable
```

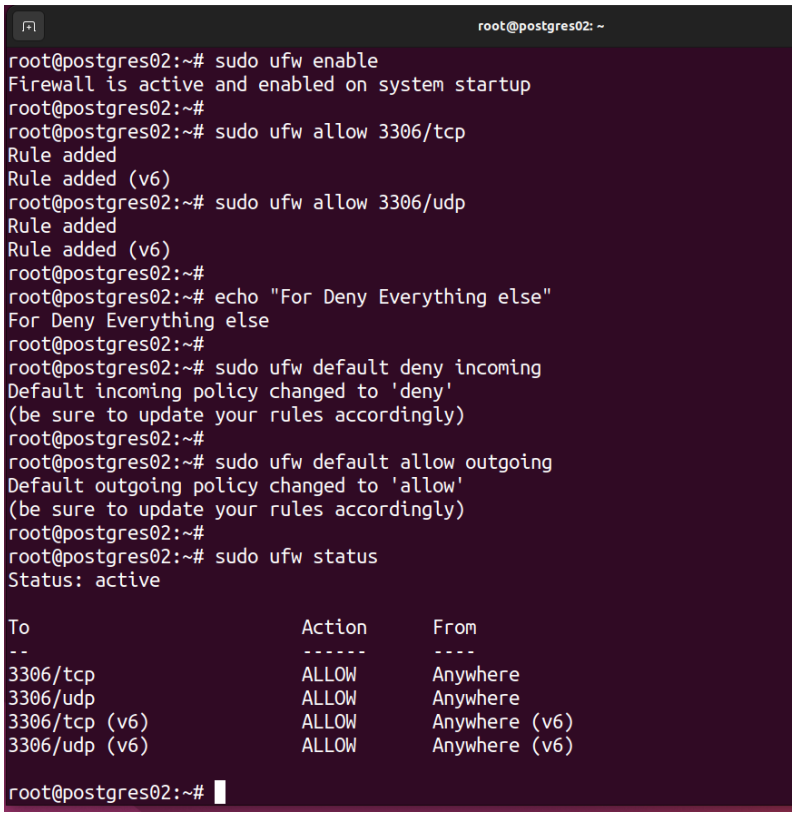
Step 2: Allow MySQL Port (TCP & UDP)

```
sudo ufw allow 3306/tcp
sudo ufw allow 3306/udp
```

Step 3: Deny everything else

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
```

Check status => sudo ufw status

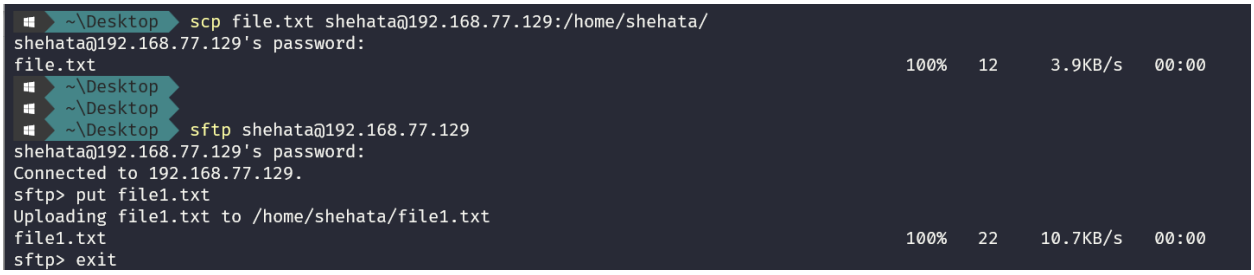


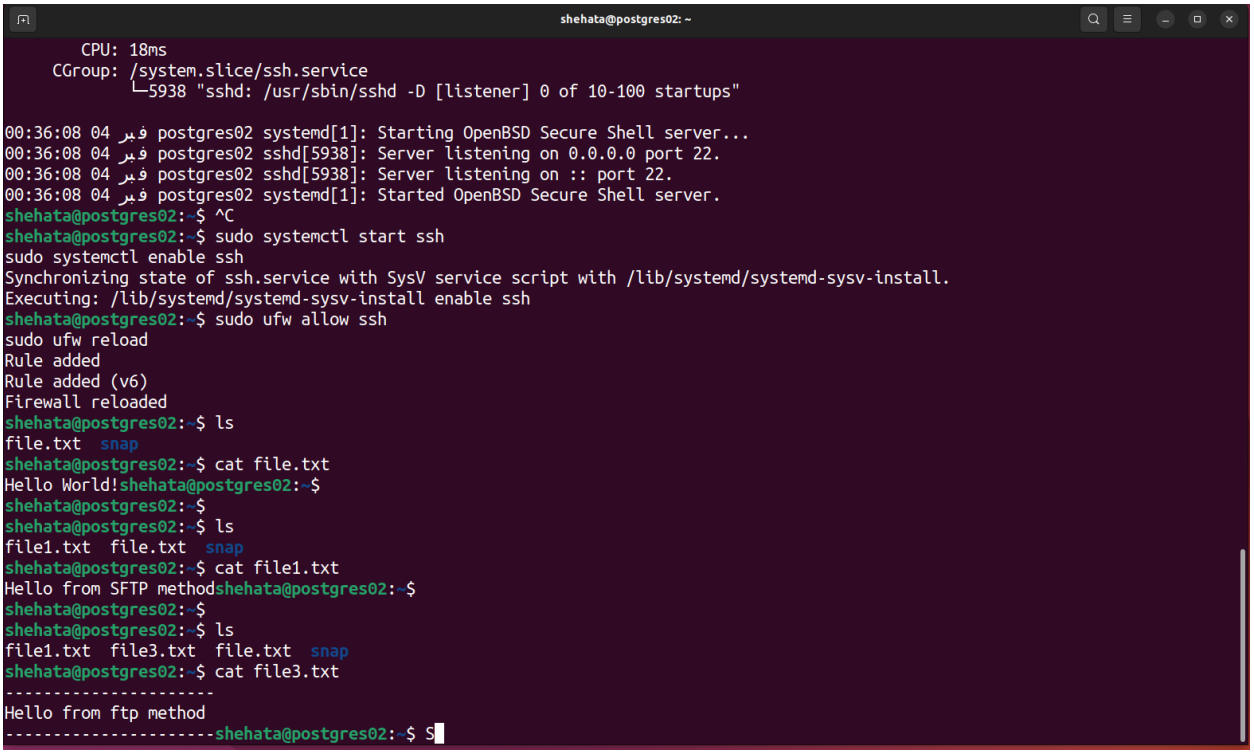
6. File Transfer Simulation

To demonstrate file transfer capabilities, a simulation was performed where a file from the local machine is copied to the virtual machine. While various methods exist, **scp** (Secure Copy Protocol) is a common command-line tool for secure file transfers over SSH.

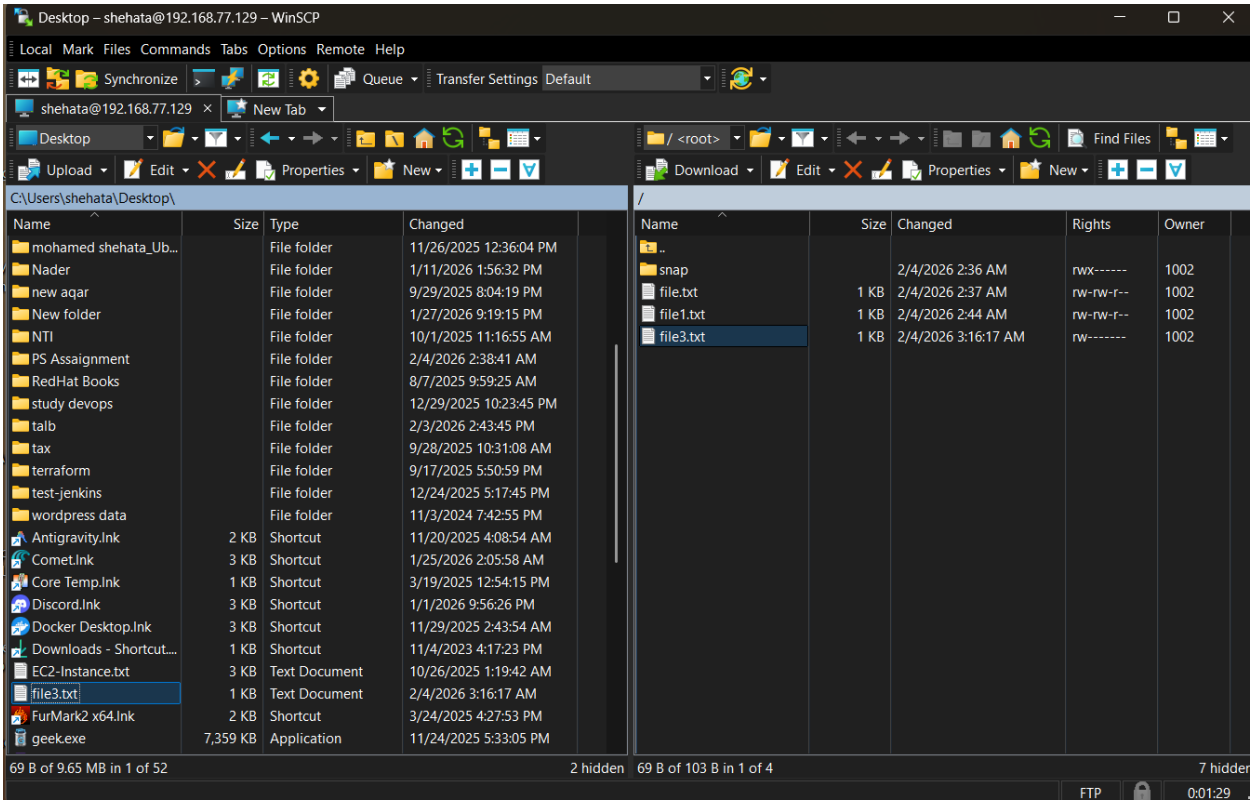
From windows terminal or powershell

```
scp file.txt shehata@192.168.77.129:/home/shehata/
sftp shehata@192.168.77.129
put file.txt
```





We can use WinSCP GUI at windows and use FTP



7. **Read about Linux in general (architecture, directories, distributions, etc.).**
   - **small summary:**

     Linux is a free, open-source operating system based on Unix. It is mainly used in servers, cloud computing, DevOps, and embedded systems.

     Linux has a layered architecture: hardware at the bottom, the kernel in the core (managing processes, memory, files, devices, and networking), and user space on top where applications and users interact with the system.

     Linux uses a single filesystem hierarchy starting from the root directory /. Important directories include /etc for configuration, /home for users, /var for logs and variable data, /bin for commands, and /proc for process and kernel information.

     A Linux distribution (distro) is the Linux kernel combined with system tools and a package manager. Popular distros include **Ubuntu**, **Debian**, **CentOS**, **RHEL**, and **Arch Linux**.

     Linux is multi-user, secure, stable, and highly customizable, which makes it the most widely used operating system for servers and cloud environments.

Section 2: SQL

Q: Read about SQL/PLSQL and DML, DDL, and DCL?

SQL (Structured Query Language):

- SQL is a standard language for managing and manipulating relational databases

- Used to query, insert, update, and delete data

PL/SQL (Procedural Language/SQL):

- Oracle's procedural extension to SQL

- Allows you to write programs with variables, conditions, loops, and exception handling

- Combines SQL's data manipulation with procedural programming features

DML (Data Manipulation Language):

- Commands used to manipulate data in database tables

- Examples: SELECT, INSERT, UPDATE, DELETE

- Used for querying and modifying data

DDL (Data Definition Language):

- Commands used to define database structure/schema

- Examples: CREATE, ALTER, DROP, TRUNCATE

- Used for creating and modifying database objects

DCL (Data Control Language):

- Commands used to control access to data

- Examples: GRANT, REVOKE

- Used for managing permissions and security

Q: What are the different types of JOINs?

1. INNER JOIN:

- Returns only matching rows from both tables
- Most common type of join

2. LEFT JOIN (LEFT OUTER JOIN):

- Returns all rows from left table and matching rows from right table
- NULL values for non-matching rows from right table

3. RIGHT JOIN (RIGHT OUTER JOIN):

- Returns all rows from right table and matching rows from left table
- NULL values for non-matching rows from left table

4. FULL OUTER JOIN:

- Returns all rows from both tables
- NULL values where there's no match

5. CROSS JOIN:

- Returns Cartesian product of both tables
- Every row from first table combined with every row from second table

6. SELF JOIN:

- Table joined with itself
- Useful for hierarchical data

Q: What is RDBMS and NoSQL?

RDBMS (Relational Database Management System):

- Stores data in structured tables with rows and columns

- Uses SQL for querying

- ACID compliant (Atomicity, Consistency, Isolation, Durability)

- Examples: Oracle, MySQL, PostgreSQL, SQL Server

- Best for: Structured data, complex queries, transactions

NoSQL (Not Only SQL):

- Stores data in non-tabular format

- Types: Document (MongoDB), Key-Value (Redis), Column-family (Cassandra), Graph (Neo4j)

- Flexible schema

- Horizontally scalable

- Best for: Unstructured data, high volume, rapid development

Aggregation Functions:

- COUNT(): Counts number of rows

- SUM(): Calculates sum of numeric column

- AVG(): Calculates average value

- MAX(): Returns maximum value

- MIN(): Returns minimum value

- GROUP BY: Groups rows sharing a property

Date and String Functions:

Date Functions: SYSDATE, TO_DATE, TO_CHAR, ADD_MONTHS, MONTHS_BETWEEN, TRUNC, EXTRACT

String Functions: CONCAT, SUBSTR, LENGTH, UPPER, LOWER, TRIM, REPLACE, INSTR

Constraints:

- PRIMARY KEY: Uniquely identifies each row

- FOREIGN KEY: Links tables together

- NOT NULL: Column cannot have NULL values

- UNIQUE: All values must be unique

- CHECK: Validates values meet specific condition

- DEFAULT: Sets default value

Indexes:

- Database objects that improve query performance

- Created on columns frequently used in WHERE clauses

- Trade-off: Faster reads, slower writes

- Types: B-tree (default), Bitmap, Unique, Composite

Q1: Create Tables with Constraints

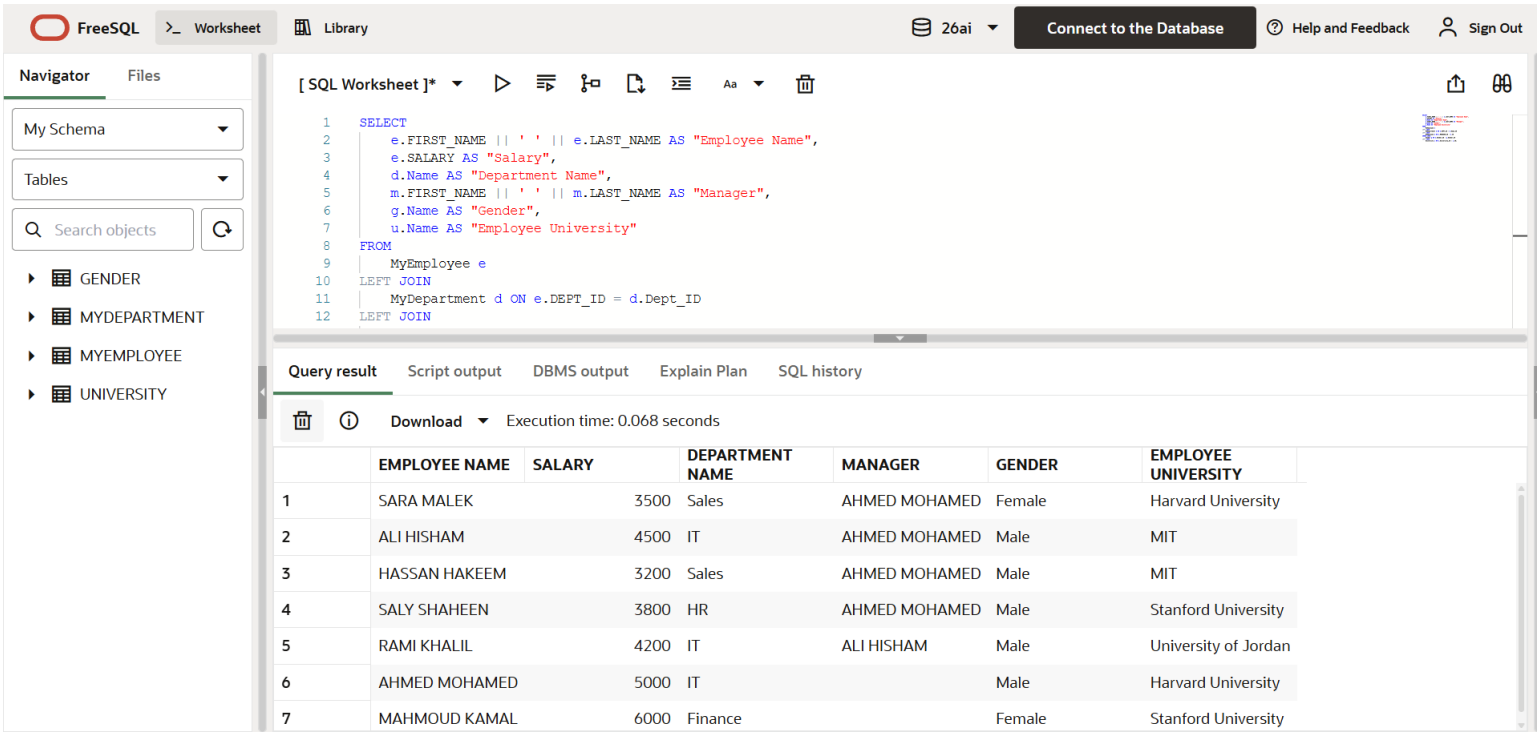
```sql
-- Create MyDepartment table
CREATE TABLE MyDepartment (
    Dept_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100) NOT NULL
);

-- Create Gender table
CREATE TABLE Gender (
    Gender_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(20) NOT NULL,
    CONSTRAINT chk_gender_name CHECK (Name IN ('Male', 'Female'))
);

-- Create University table
CREATE TABLE University (
    ID NUMBER PRIMARY KEY,
    Name VARCHAR2(200) NOT NULL
);
```

```sql
-- Create MyEmployee table
CREATE TABLE MyEmployee (
    ID NUMBER PRIMARY KEY,
    LAST_NAME VARCHAR2(50) NOT NULL,
    FIRST_NAME VARCHAR2(50) NOT NULL,
    HIRE_DATE DATE NOT NULL,
    USERID NUMBER UNIQUE NOT NULL,
    SALARY NUMBER NOT NULL,
    DEPT_ID NUMBER,
    Gender_ID NUMBER NOT NULL,
    University_ID NUMBER,
    EMP_IMAGE BLOB,
    MANAGER_ID NUMBER,
    JOB_TITLE VARCHAR2(50),
    CONSTRAINT fk_dept FOREIGN KEY (DEPT_ID) REFERENCES MyDepartment(Dept_ID),
    CONSTRAINT fk_gender FOREIGN KEY (Gender_ID) REFERENCES Gender(Gender_ID),
    CONSTRAINT fk_university FOREIGN KEY (University_ID) REFERENCES University(ID),
    CONSTRAINT fk_manager FOREIGN KEY (MANAGER_ID) REFERENCES MyEmployee(ID),
    CONSTRAINT chk_salary CHECK (SALARY > 0)
);
```

Note: Added for manager relationship , added job_title for Q3

Q2: Write a SQL query to retrieve Employee Data with joins

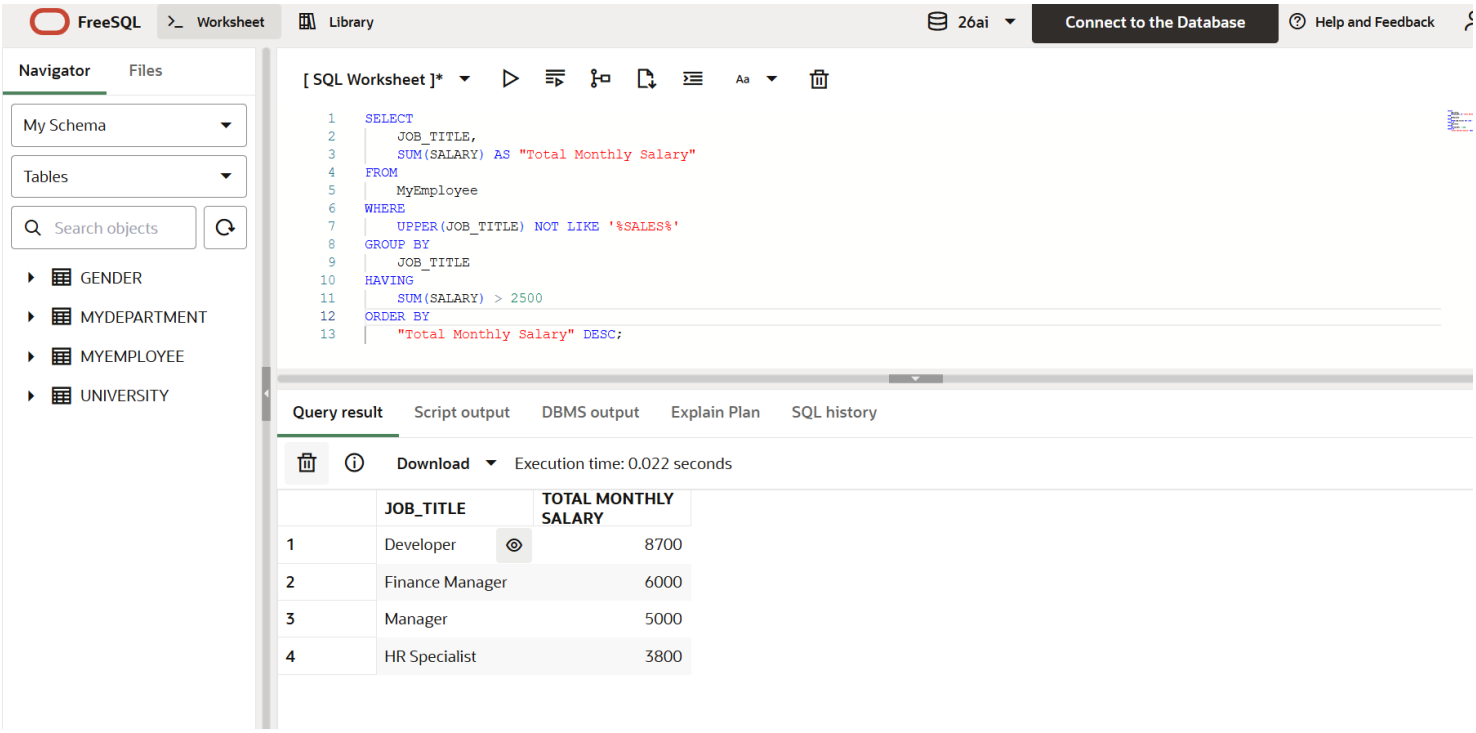
```sql
SELECT
    e.FIRST_NAME || ' ' || e.LAST_NAME AS "Employee Name",
    e.SALARY AS "Salary",
    d.Name AS "Department Name",
    m.FIRST_NAME || ' ' || m.LAST_NAME AS "Manager",
    g.Name AS "Gender",
    u.Name AS "Employee University"
FROM
    MyEmployee e
LEFT JOIN
    MyDepartment d ON e.DEPT_ID = d.Dept_ID
LEFT JOIN
    MyEmployee m ON e.MANAGER_ID = m.ID
INNER JOIN
    Gender g ON e.Gender_ID = g.Gender_ID
LEFT JOIN
    University u ON e.University_ID = u.ID;
```



	EMPLOYEE NAME	SALARY	DEPARTMENT NAME	MANAGER	GENDER	EMPLOYEE UNIVERSITY
1	SARA MALEK	3500	Sales	AHMED MOHAMED	Female	Harvard University
2	ALI HISHAM	4500	IT	AHMED MOHAMED	Male	MIT
3	HASSAN HAKEEM	3200	Sales	AHMED MOHAMED	Male	MIT
4	SALY SHAHEEN	3800	HR	AHMED MOHAMED	Male	Stanford University
5	RAMI KHALIL	4200	IT	ALI HISHAM	Male	University of Jordan
6	AHMED MOHAMED	5000	IT		Male	Harvard University
7	MAHMOUD KAMAL	6000	Finance		Female	Stanford University

Q3: Display all job titles and the total monthly salary for each job title, where the total payroll exceeds $2500, excluding sales employees.

```sql
SELECT
    JOB_TITLE,
    SUM(SALARY) AS "Total Monthly Salary"
FROM
    MyEmployee
WHERE
    UPPER(JOB_TITLE) NOT LIKE '%SALES%'
GROUP BY
    JOB_TITLE
HAVING
    SUM(SALARY) > 2500
ORDER BY
    "Total Monthly Salary" DESC;
```



Q4: There are four coding errors in the following statement. Identify them:
SELECT empno, ename,
salary x 12 ANNUAL SALARY; FROM emp;

❌ Errors identified

1. Missing multiplication operator (*)

- "salary x 12" should be "salary * 12"

2. Column alias needs quotes or underscore

- "ANNUAL SALARY" should be "ANNUAL_SALARY" or "ANNUAL SALARY" (in quotes)

3. Semicolon in wrong position

- Semicolon should be at the end, not before FROM

4. Incorrect statement structure

- FROM clause should come before the semicolon

✅ Correct statement structure

```sql
SELECT
    empno,
    ename,
    salary * 12 AS ANNUAL_SALARY
FROM
    emp;

-- Alternative with quoted alias:
SELECT
    empno,
    ename,
    salary * 12 AS "ANNUAL SALARY"
FROM
    emp;
```

Q5: Oracle function – F_HR_QUERY

```
CREATE OR REPLACE FUNCTION F_HR_QUERY
RETURN SYS_REFCURSOR
IS
    v_scott_hire_date DATE;
    result_cursor SYS_REFCURSOR;
BEGIN
    -- Get SCOTT's hire date
    SELECT HIRE_DATE
    INTO v_scott_hire_date
    FROM MyEmployee
    WHERE UPPER(LAST_NAME) = 'SCOTT';

    -- Open cursor with employees hired after SCOTT
    OPEN result_cursor FOR
        SELECT
            FIRST_NAME || ' ' || LAST_NAME AS Employee_Name,
            HIRE_DATE
        FROM
            MyEmployee
        WHERE
            HIRE_DATE > v_scott_hire_date
        ORDER BY
            HIRE_DATE;

    RETURN result_cursor;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee SCOTT not found');
        RETURN NULL;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        RETURN NULL;
END F_HR_QUERY;
/
```

Q6: Oracle function – P_COPY_EMPLOYEE

Statement details

```
CREATE OR REPLACE PROCEDURE P_COPY_EMPLOYEE
IS
    v_count NUMBER;
BEGIN
    -- Clear existing data in target table
    DELETE FROM MyEmployee_update;

    -- Copy all data from MyEmployee to MyEmployee_update
    INSERT INTO MyEmployee_update
    SELECT * FROM MyEmployee;

    -- Get count of inserted records
    v_count := SQL%ROWCOUNT;

    -- Commit the transaction
    COMMIT;

    -- Display success message
    DBMS_OUTPUT.PUT_LINE('Successfully copied ' || v_count ||
 ' records from MyEmployee to MyEmployee_update');

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
        RAISE;
END P_COPY_EMPLOYEE;
```

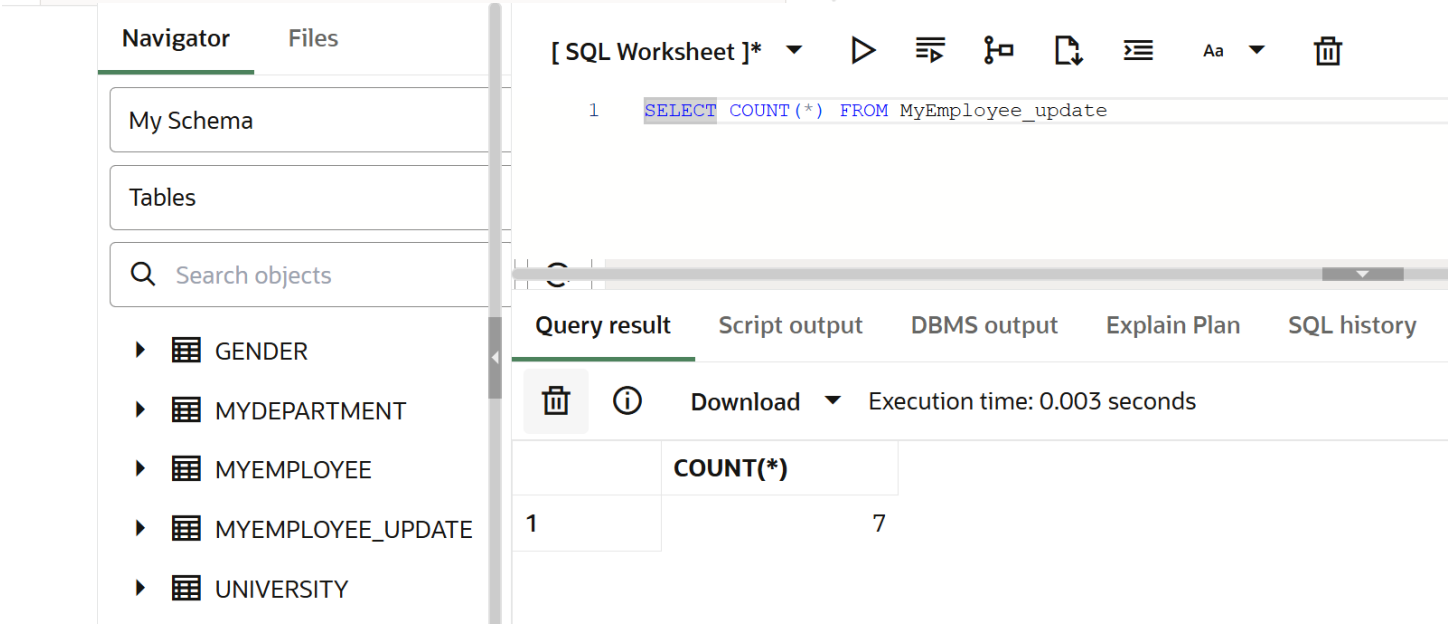
Insert Into Editor

Result

```
Procedure P_COPY_EMPLOYEE compiled

Elapsed: 00:00:00.044
```

Execute:

EXEC P_COPY_EMPLOYEE



Tomcat Section:

1. In your own words, explain what JVM means ?

The **JVM (Java Virtual Machine)** is a runtime environment that executes Java bytecode.
It allows Java applications to run on any operating system without modification by abstracting the underlying hardware and OS.

Responsibilities of JVM:

- Loads and verifies bytecode
- Manages memory (Heap, Stack, Garbage Collection)
- Executes Java instructions
- Provides platform independence ("Write once, run anywhere")

What is an Application Server?

An **application server** is software that provides an environment to run backend applications and manage:

- Request handling
- Session management
- Security
- Thread management
- Resource pooling (DB connections)

Examples:

Apache Tomcat (Servlet container), JBoss / WildFly, WebLogic, WebSphere

Tomcat is a lightweight application server, mainly for Java web apps (Servlets & JSP).

Q3: What is a WAR file? How Tomcat handles it?

⇨ A **WAR (Web Application Archive)** file is a packaged Java web application.

It contains:

- Compiled Java classes
- JSP files
- Configuration files (web.xml)
- Libraries (WEB-INF/lib)

How Tomcat handles a WAR?

- When a WAR file is placed in Tomcat's webapps/ directory
- Tomcat automatically extracts it
- Deploys it as a web application

Deployment location: **$TOMCAT_HOME/webapps/**

Practice:

1. Download and install Tomcat as a WebApp container. Set up a reverse proxy with Nginx and connect both so that the WebApp can be accessed through a DNS name over port 80.

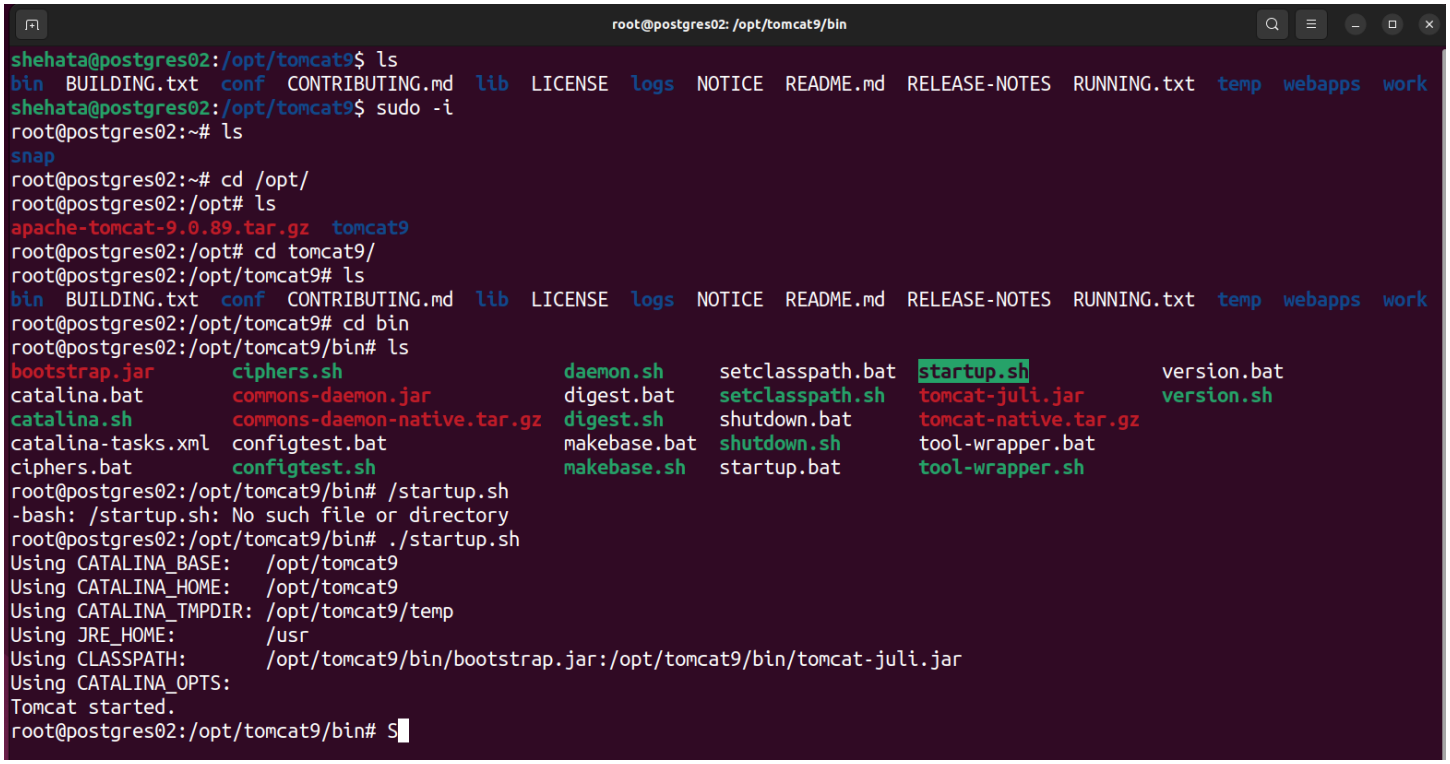
```
sudo apt update
sudo apt install openjdk-11-jdk -y
```

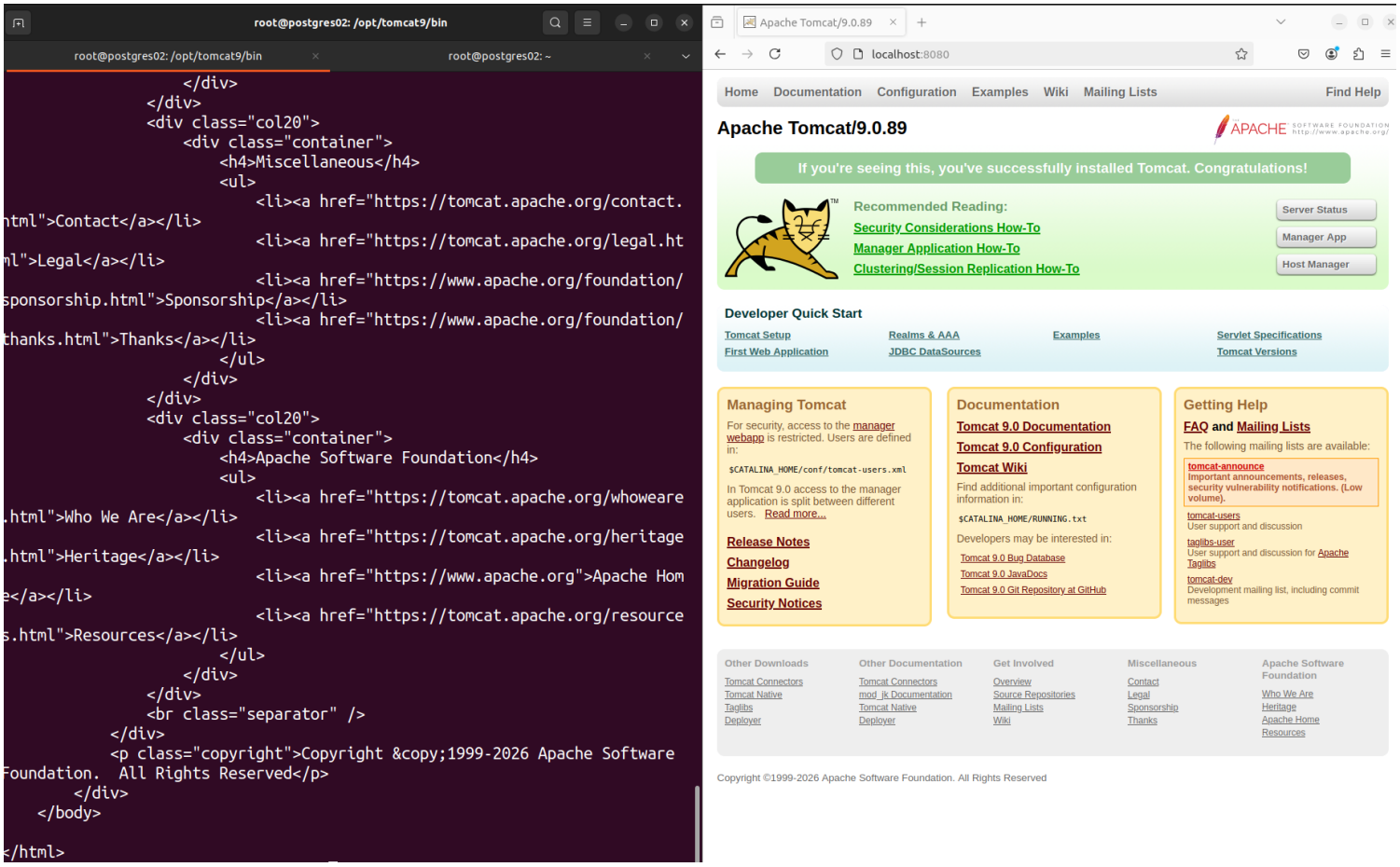
➢ Verify : java -version

- Download & Install Tomcat 9

```

cd /opt

sudo wget https://archive.apache.org/dist/tomcat/tomcat9/v9.0.89/bin/apache-tomcat-9.0.89.tar.gz

sudo tar -xvf apache-tomcat-9.0.89.tar.gz

sudo mv apache-tomcat-9.0.89 tomcat9

```

- Start Tomcat:
  /opt/tomcat9/bin/startup.sh
- Access:
  http://VM_IP:8080





2. Change the Tomcat port to 7070.

- First need to edit server.xml file

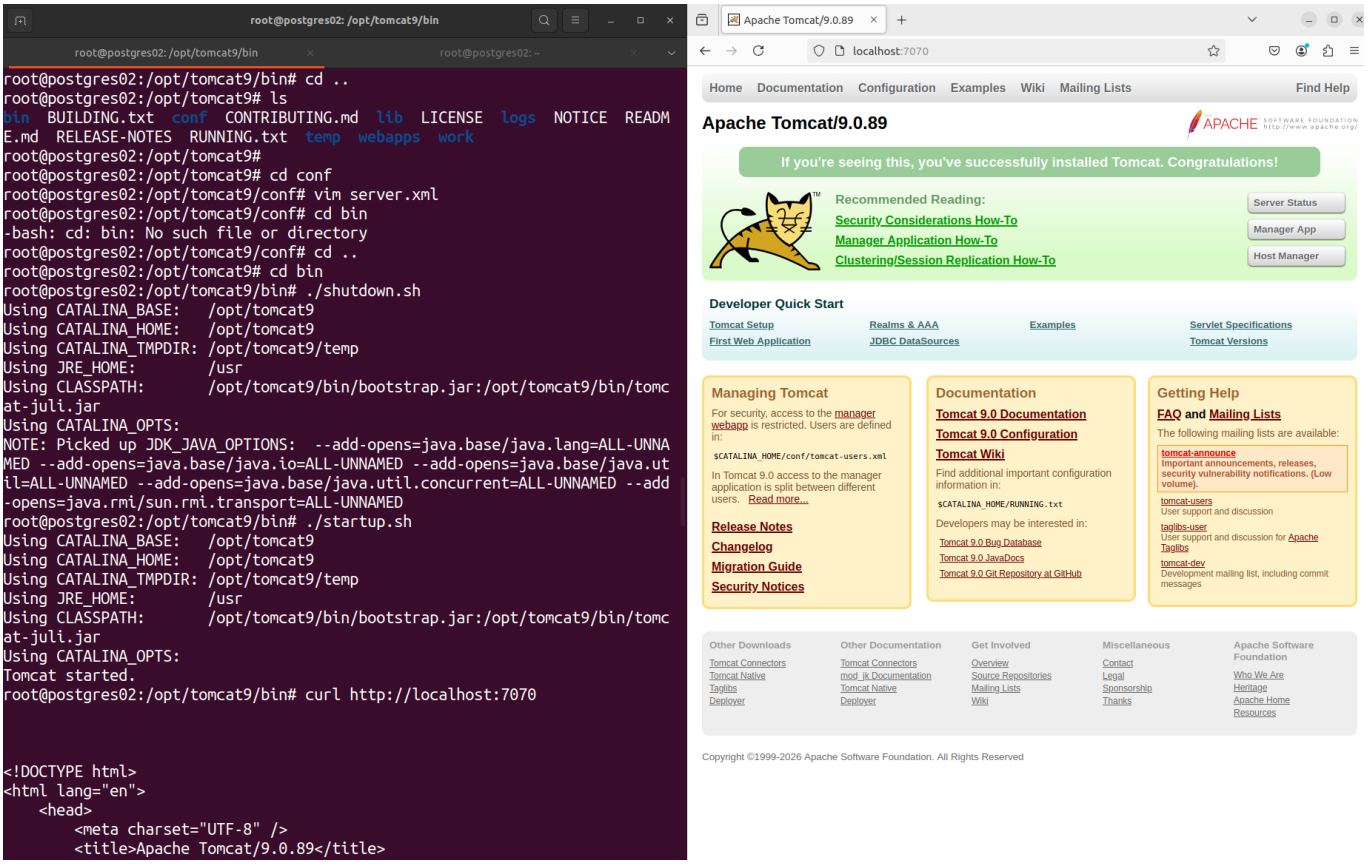
```
sudo vim /opt/tomcat9/conf/server.xml
```

- Second Change this line to :

<Connector port="7070"

- Third restart tomcat

/opt/tomcat9/bin/shutdown.sh
/opt/tomcat9/bin/startup.sh



Nginx Reverse Proxy:

- **Install Nginx**

  sudo apt install nginx -y

- Configure Nginx

  sudo nano /etc/nginx/sites-available/tomcat

```
server {
    listen 80;
    server_name tomcat.local;

    location / {
        proxy_pass http://localhost:7070;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }}
```

- **Enable page or site:**

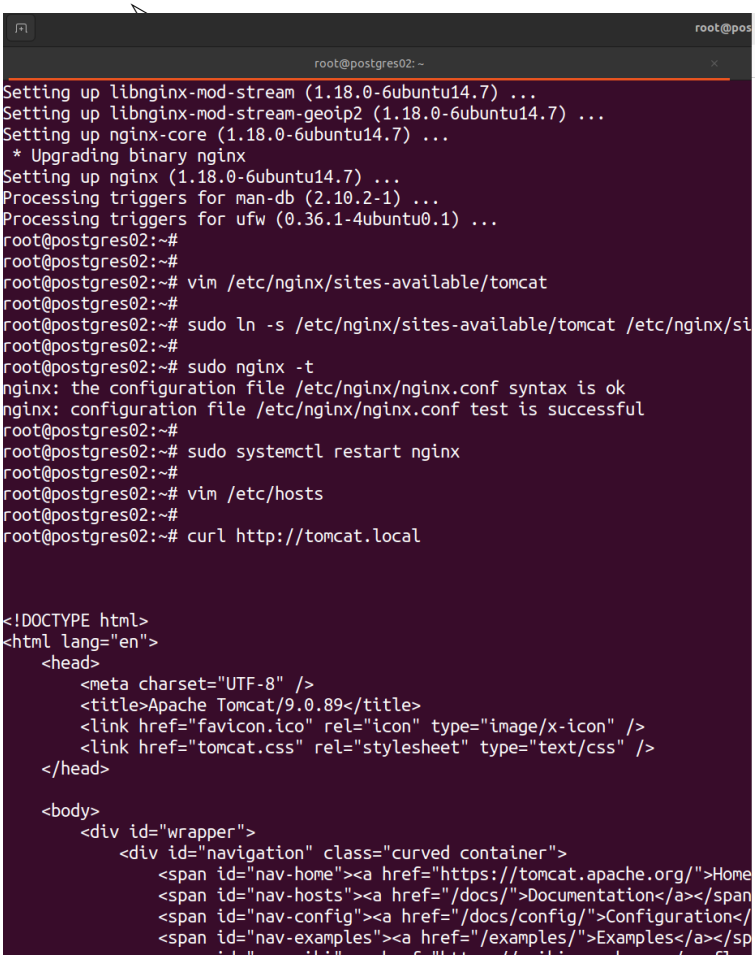
  sudo ln -s /etc/nginx/sites-available/tomcat /etc/nginx/sites-enabled/
  sudo nginx -t
  sudo systemctl restart nginx

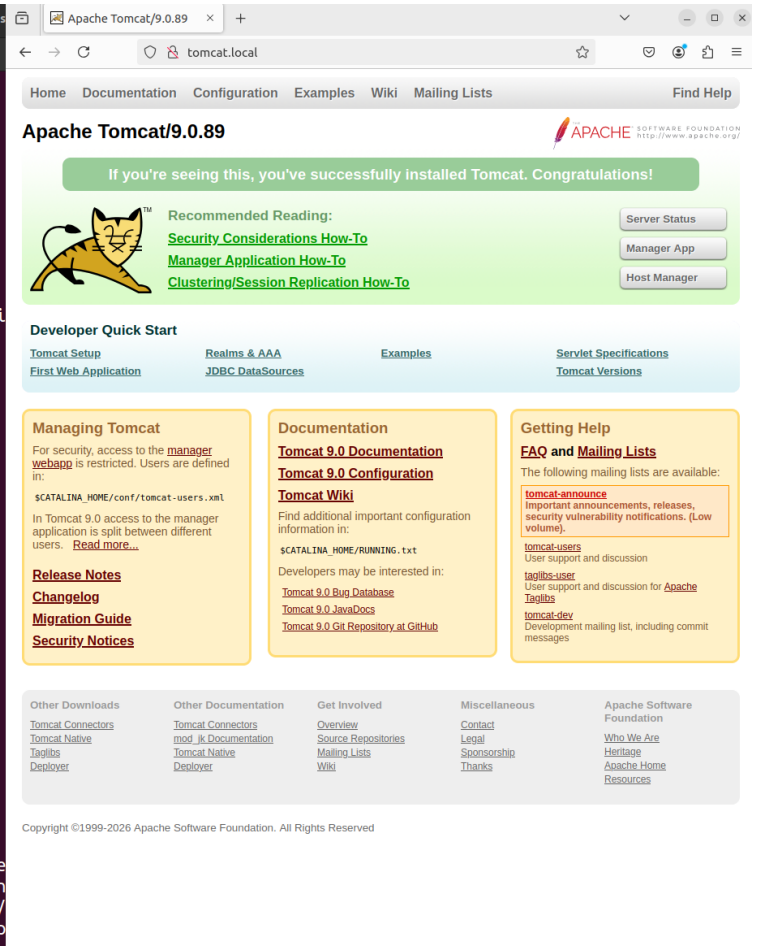
- **To add DNS entry:**

  sudo vim /etc/hosts

  127.0.0.1 tomcat.local

- **Then Access it**
  http://tomcat.local





3. Create a Vagrantfile to perform the following:

Requirements:

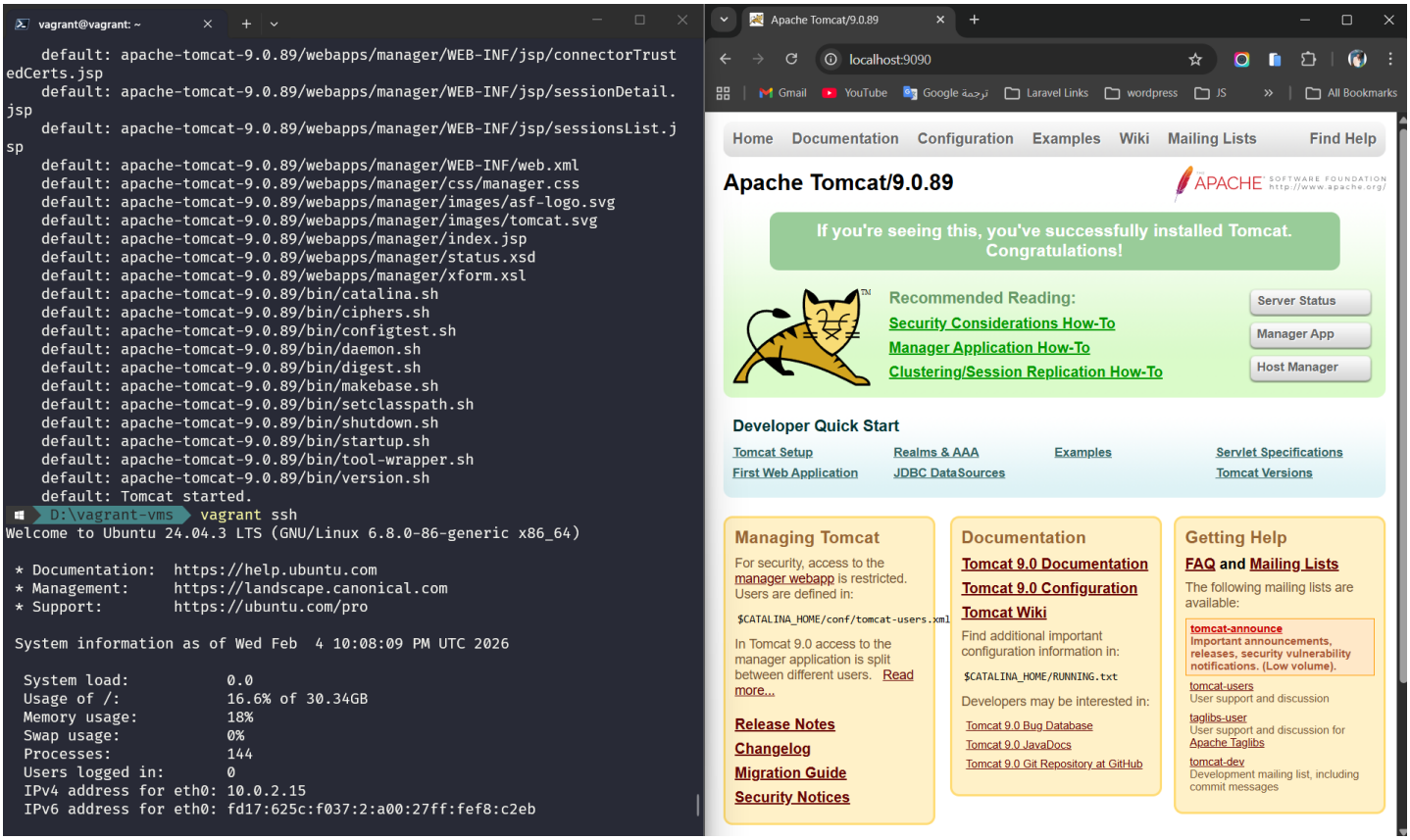
- Image: bento/ubuntu-24.04
- Guest port 7070 → Host port 9090
- Memory: 2GB
- Java 8
- Tomcat 9

Vagrantfile NOTE： I used shell script as inline:

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-24.04"

  config.vm.network "forwarded_port", guest: 7070, host: 9090

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "2048"
  end

  config.vm.provision "shell", inline: <<-SHELL
    apt update
    apt install -y openjdk-8-jdk wget

    cd /opt
    wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.89/bin/apache-tomcat-9.0.89.tar.gz
    tar -xvf apache-tomcat-9.0.89.tar.gz
    mv apache-tomcat-9.0.89 tomcat9

    sed -i 's/port="8080"/port="7070"/' /opt/tomcat9/conf/server.xml
    /opt/tomcat9/bin/startup.sh
  SHELL
end

```

- **Run Vagrantfile**
  `vagrant up` & to login vm use: `vagrant ssh`

- **Access from host:**
  http://localhost:9090



DevOps Section:

1. Read about Docker: What it is, use cases, and basic commands?

⇨ Docker is a containerization platform that allows applications to be packaged with all their dependencies and run consistently across different environments.

Why Docker is used:

- ❖ Faster deployments
- ❖ Environment consistency
- ❖ Lightweight compared to virtual machines
- ❖ Easy scaling and portability

Common use cases:

- ❖ Microservices architecture
- ❖ CI/CD pipelines
- ❖ Application isolation
- ❖ Dev/Test environments

Basic Docker Commands:

docker –version

docker login

docker pull <image_name>

docker push <image_name>

docker images

docker ps

docker ps -a

docker build -t myapp .

docker run -d -p 80:80 nginx

docker stop <container_id>

docker rm <container_id>

docker rmi <image_id>

docker logs container_name

docker network ls

2. Create a Dockerfile for a Tomcat Docker image, deploy a sample WAR in it, and push it to a Docker registry?

Dockerfile

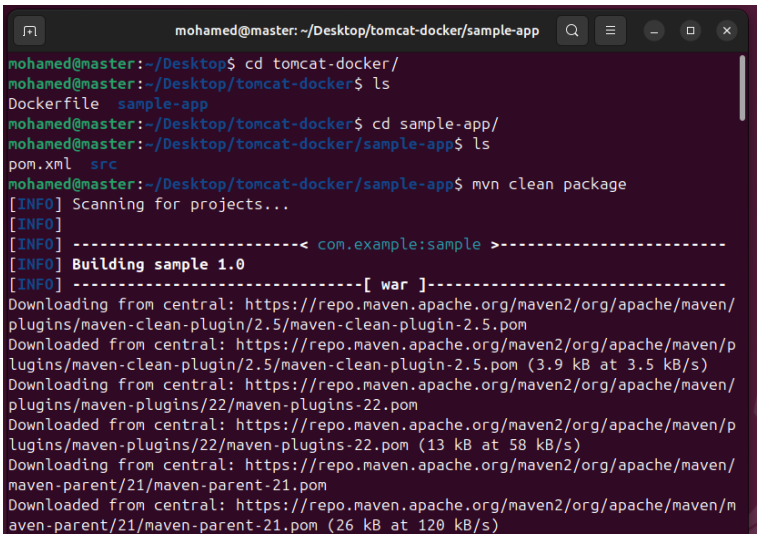
FROM tomcat:9-jdk8

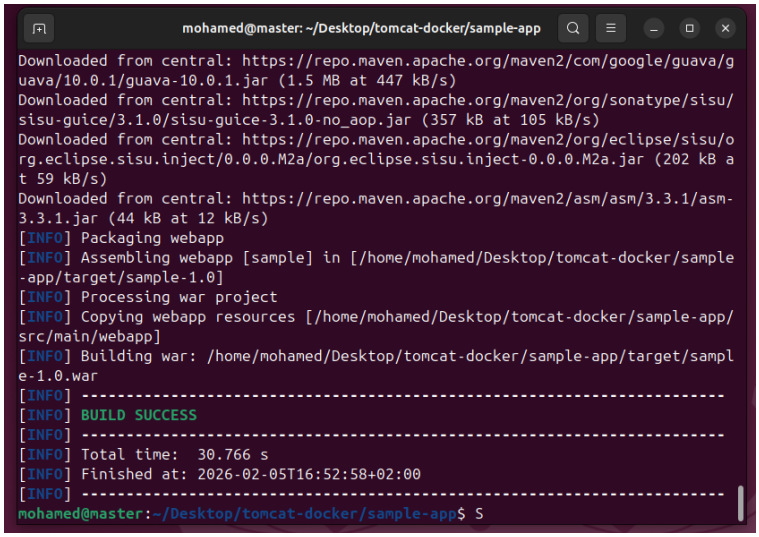
RUN rm -rf /usr/local/tomcat/webapps/*

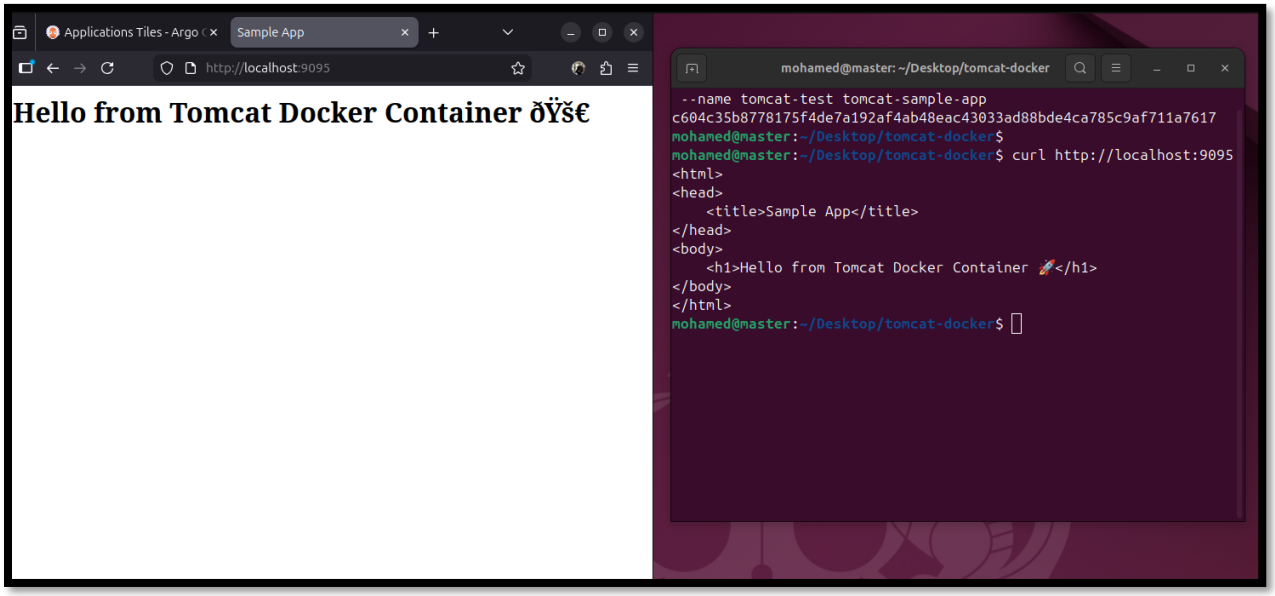
COPY sample.war /usr/local/tomcat/webapps/ROOT.war

EXPOSE 8080

CMD ["catalina.sh", "run"]





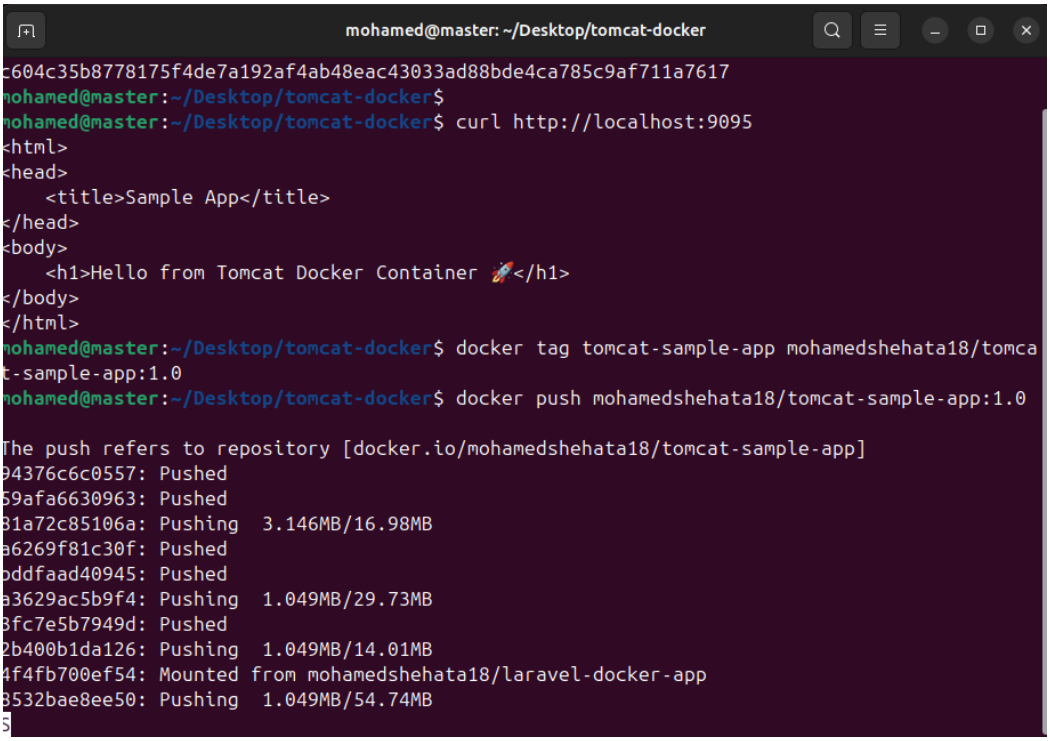


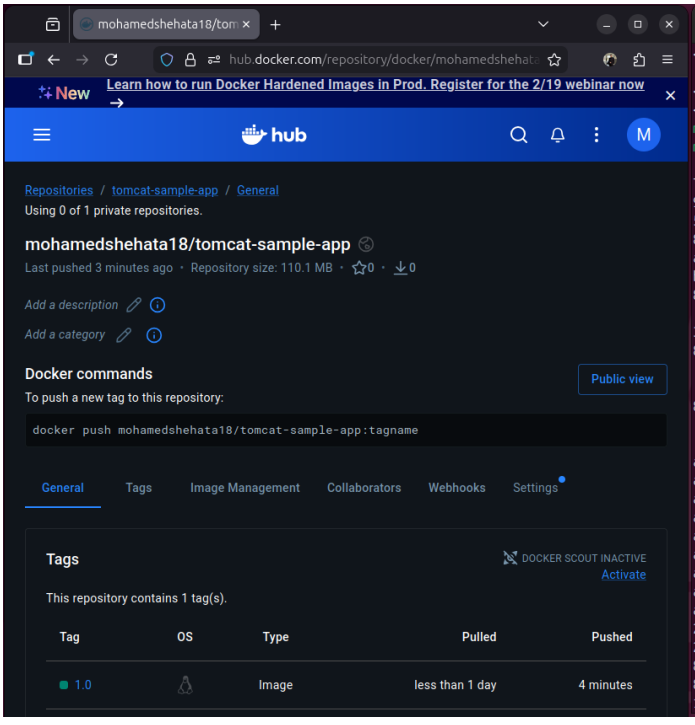
Push image to Docker hub registry

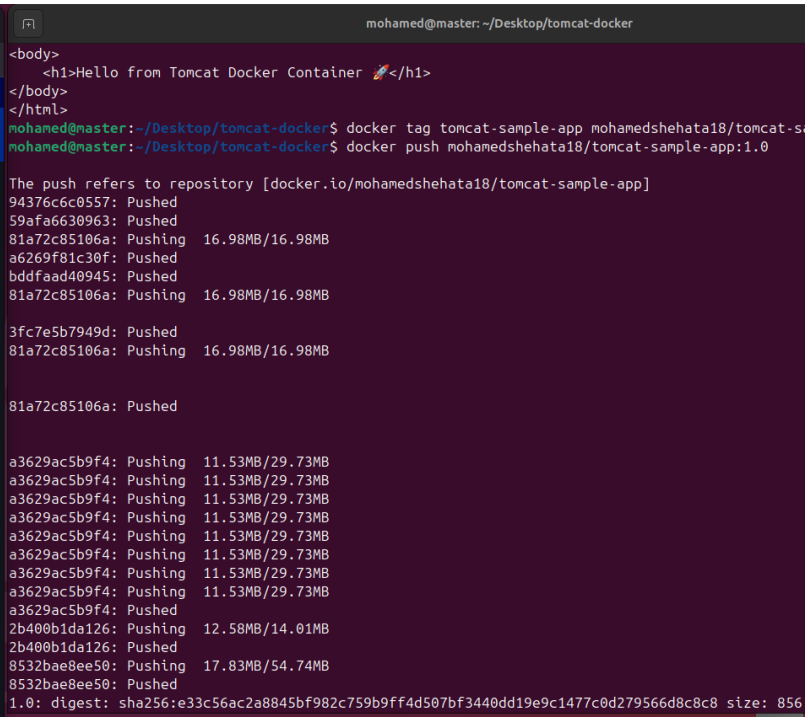
docker login

docker tag my-tomcat-app mohamedshehata18/my-tomcat-app

docker push mohamedshehata18/my-tomcat-app

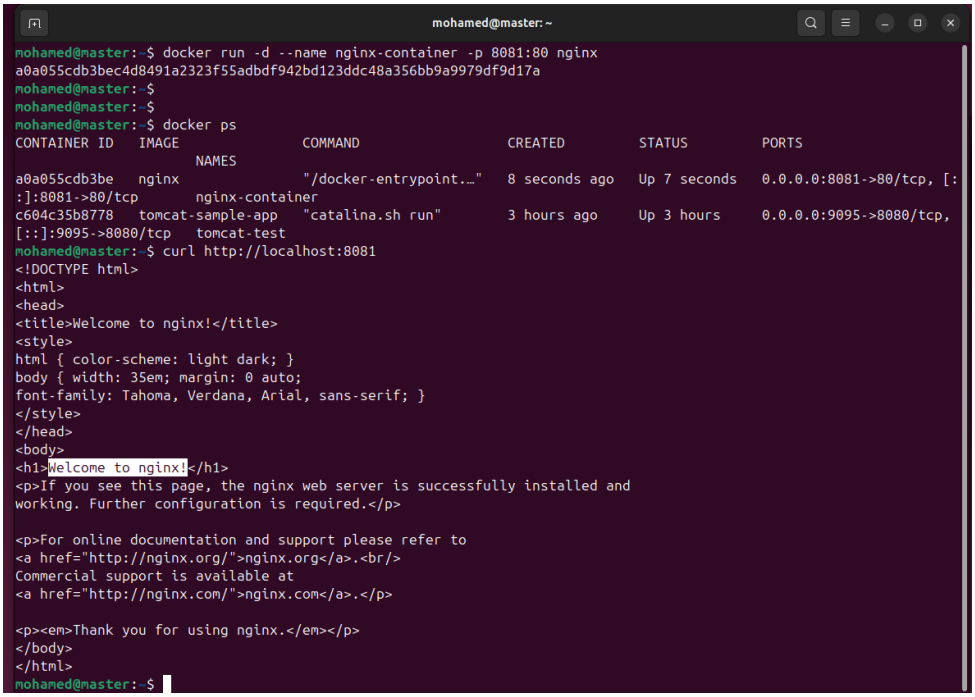






3- Run Nginx & PostgreSQL Containers

docker run -d --name nginx-container -p 8081:80 nginx



docker run -d \

--name postgres-container \

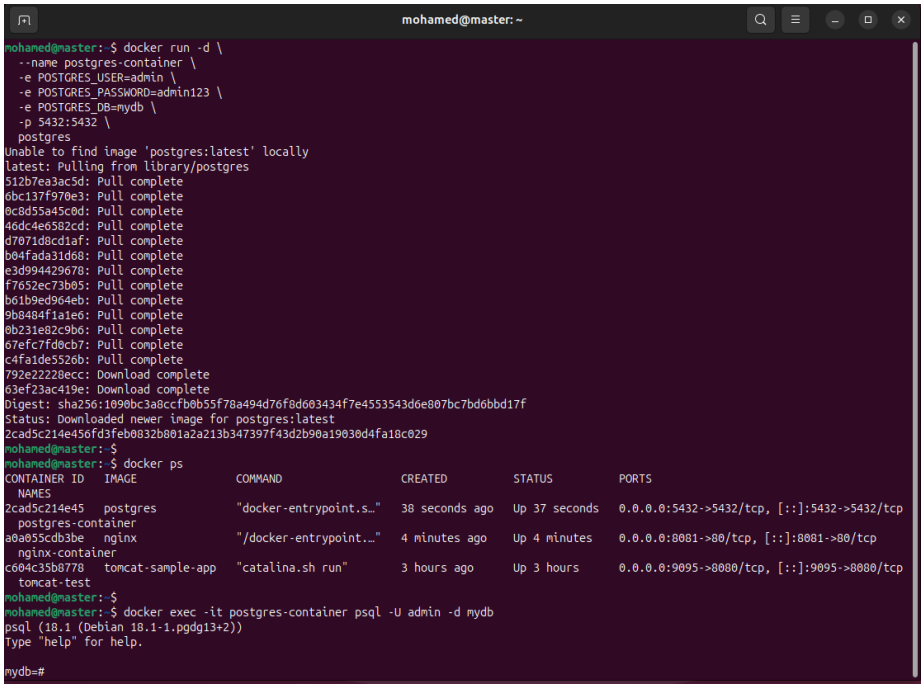
-e POSTGRES_USER=admin \

-e POSTGRES_PASSWORD=admin123 \

-e POSTGRES_DB=mydb \

-p 5432:5432 \

postgres



Kubernetes Section:

Kubernetes (K8s) is a container orchestration platform used to manage, scale, and deploy containerized applications automatically.

It handles:

- Pod scheduling
- Service discovery
- Scaling
- Self-healing
- Load balancing

---

2. Explain the purpose of master and worker nodes, and how to identify whether a node is master or worker.

Master Node (Control Plane)
The master node is the brain of the Kubernetes cluster. It manages, controls, and makes decisions about the cluster.

Worker Node
Worker nodes run containerized applications and are managed by the master node.

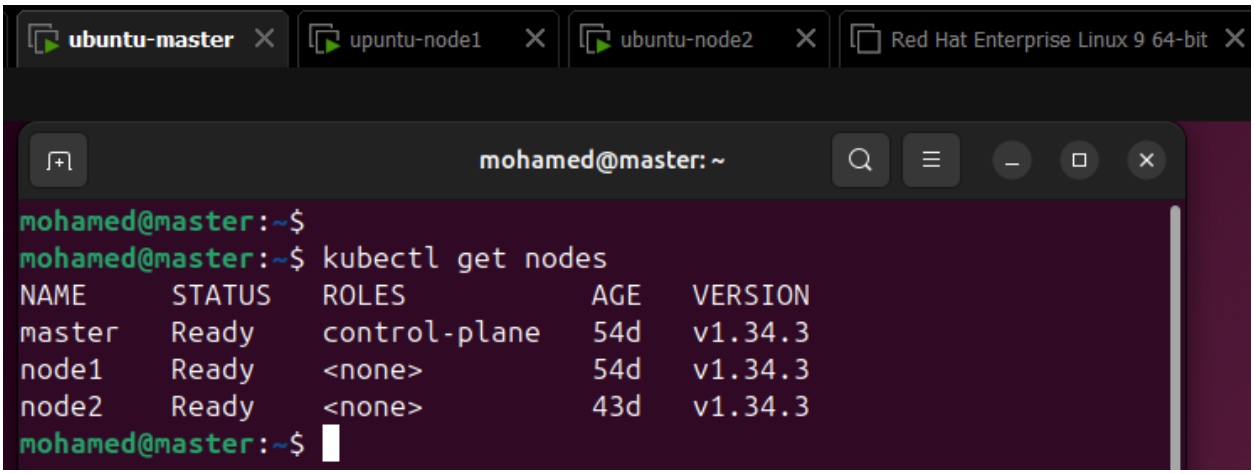
Key Components (run on master)

- kube-apiserver → Entry point for all cluster operations
- etcd → Stores cluster state & configuration
- kube-scheduler → Decides where pods run
- kube-controller-manager → Ensures desired state

Key Components (run on Worker)

- Kubelet: Agent that runs on each node, ensures containers are running in pods
- Kube-proxy: Network proxy that maintains network rules for pod communication
- Container Runtime: Software responsible for running containers

Q: How to Identify Master vs Worker Nodes ?

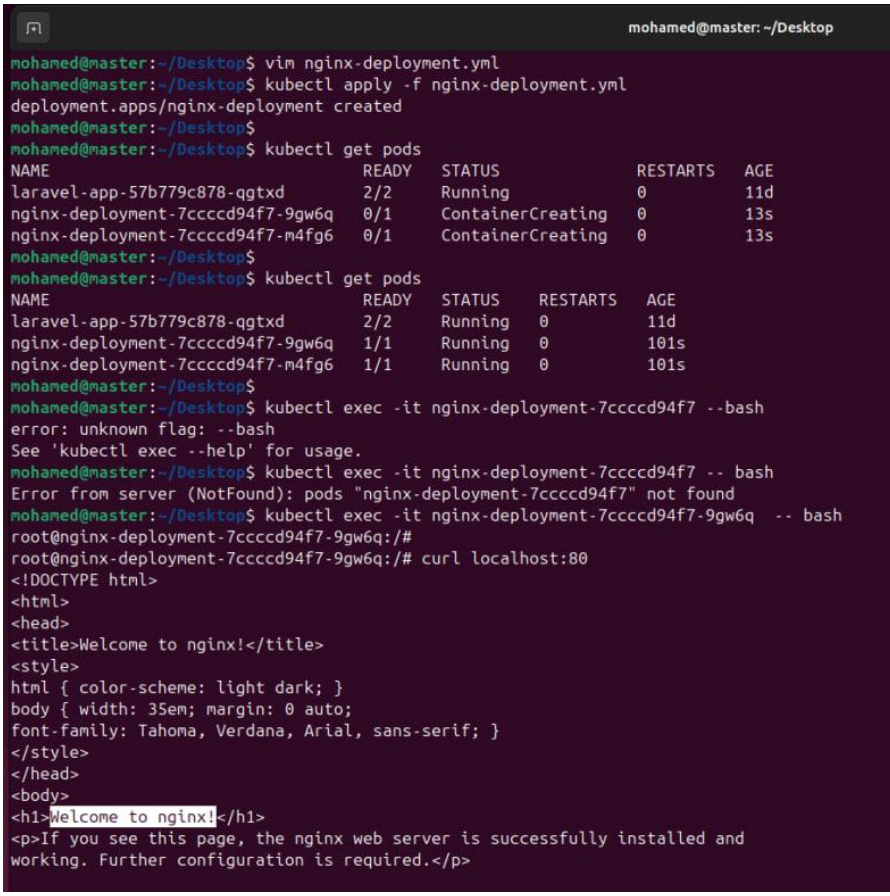


3. Try installing K8s, K3s, or MicroK8s and perform a sample deployment.

I install K8S using Kubeadm on vmware used 1 Master node, 2 Worker nodes

Sample Kubernetes Deployment:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```



Key Concepts:

1. RAID (Redundant Array of Independent Disks)

RAID is a technology used to combine multiple physical hard disks into one logical unit to improve performance, fault tolerance, or both.

Why RAID is used

- Protect data from disk failure
- Increase read/write performance
- Improve system availability

Use cases

- Databases (Oracle, PostgreSQL)
- File servers
- Critical production systems

Common RAID Levels

- **RAID 0 (Striping)**
  - Splits data across disks
  - High performance
  - No fault tolerance (if one disk fails → data lost)
- **RAID 1 (Mirroring)**
  - Data is duplicated on two disks
  - High data protection
  - Uses more disk space
- **RAID 5 (Striping + Parity)**
  - Data + parity distributed across disks
  - Can tolerate **one disk failure**
  - Good balance between performance and safety
- **RAID 10 (1+0)**
  - Combination of RAID 1 and RAID 0
  - High performance + high availability
  - Used in databases and production systems

2. DevOps

DevOps is a culture and set of practices that aims to **bridge the gap between Development and Operations** to deliver software faster, more reliably, and with better quality.

Key goals of DevOps

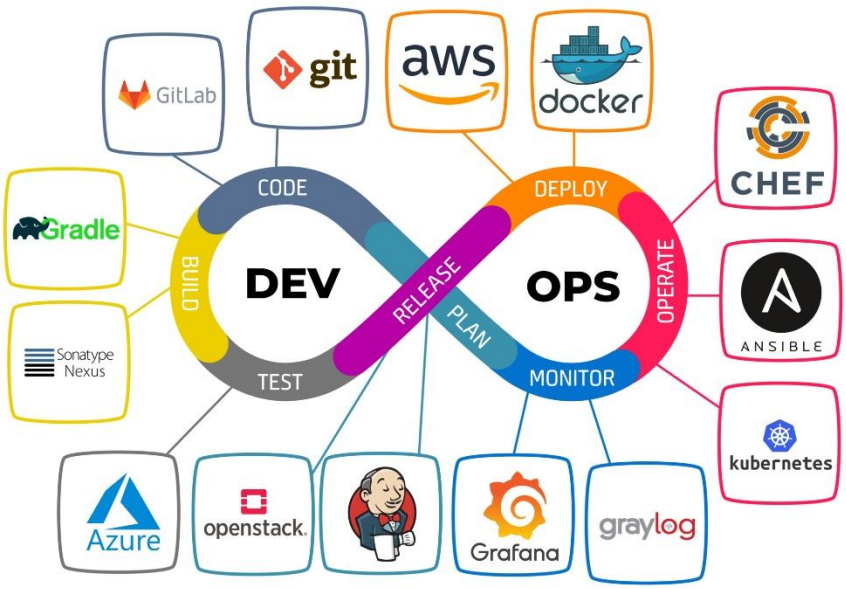
- Faster deployments
- Automation of repetitive tasks
- Continuous delivery
- High system reliability

Benefits

- Reduced human errors
- Faster release cycles
- Better collaboration between teams
- Improved system stability

Core DevOps practices

- **CI/CD** (Continuous Integration & Continuous Deployment)
- **Infrastructure as Code (IaC)** – Terraform, Ansible
- **Containerization** – Docker
- **Orchestration** – Kubernetes
- **Monitoring & Logging** – Prometheus, Grafana, ELK



3. High Availability (HA) and Disaster Recovery (DR) – what they are, and technologies that can be used for them (examples: Oracle, PostgreSQL, or Tomcat)

- High Availability (HA)

HA ensures that a system remains operational with minimal downtime, even if a component fails.

HA techniques

- Load balancing
- Active/Active or Active/Passive setups
- Failover clustering
- Replication

Examples

- PostgreSQL: Streaming replication, Patroni
- Oracle: Oracle RAC
- Tomcat: Multiple Tomcat instances behind Nginx/HAProxy

Disaster Recovery (DR)

DR focuses on recovering systems and data after a major failure, such as:

- Data center outage
- Natural disasters
- Cyberattacks

DR strategies

- Regular backups
- Offsite backups
- Database snapshots
- Standby environments

Key metrics

- RTO (Recovery Time Objective) – how fast recovery happens
- RPO (Recovery Point Objective) – how much data loss is acceptable

4. Cloud Computing

Cloud Computing is the delivery of computing services (servers, storage, databases, networking, software) over the internet.

Cloud deployment models

- **Public Cloud** – AWS, Azure, GCP
- **Private Cloud** – On-prem or private data center
- **Hybrid Cloud** – Mix of public + private

Cloud Service Models

IaaS (Infrastructure as a Service)

- You manage: OS, runtime, apps
- Provider manages: hardware, network
- Examples:
  - AWS EC2
  - Azure Virtual Machines

PaaS (Platform as a Service)

- You manage: application & data
- Provider manages: OS, runtime, infrastructure
- Examples:
  - AWS Elastic Beanstalk

SaaS (Software as a Service)

- Fully managed software
- No infrastructure management
- Examples:
  - Gmail
  - Google Drive
  - Salesforce

5. DNS and Load Balancers

DNS (Domain Name System)

DNS translates human-readable domain names into IP addresses.

Example

www.example.com → 192.168.1.1

Load Balancer

A Load Balancer distributes incoming traffic across multiple servers to:

- **Improve performance**
- **Prevent server overload**
- **Increase availability**

Types of Load Balancers

- Layer 4 (Transport level) – TCP/UDP
    - Example: HAProxy (TCP mode)
- Layer 7 (Application level) – HTTP/HTTPS
    - Example: Nginx, AWS ALB

Common tools

- Nginx
- HAProxy
- AWS ELB / ALB
- Kubernetes Service & Ingress