MySQL Configuration Changes & Commands

Change MySQL Configuration to Allow Remote Connections

Commands:

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

bind-address = 127.0.0.1

bind-address = 0.0.0.0

sudo systemctl restart mysql

Database and Table Creation (event.sql)

Command to execute event.sql:

mysql -u root -p < event.sql

Key Contents of event.sql:

-- Create user with remote access

CREATE USER IF NOT EXISTS 'mina'@'%' IDENTIFIED BY 'password123';

-- Grant necessary permissions to the user

GRANT SELECT, INSERT, UPDATE, DELETE ON xperienceDB.* TO 'mina'@'%';

-- Ensure remote access authentication

ALTER USER 'mina'@'%' IDENTIFIED WITH mysql_native_password BY 'password123';

-- Apply privileges

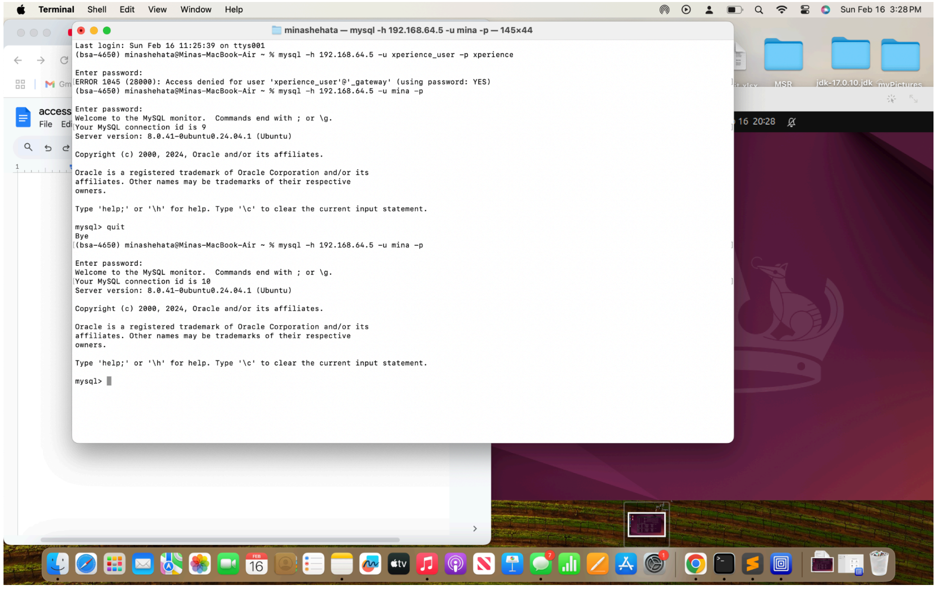
FLUSH PRIVILEGES;

Check MySQL is Listening on All Interfaces (for Verification)

ss -ltn

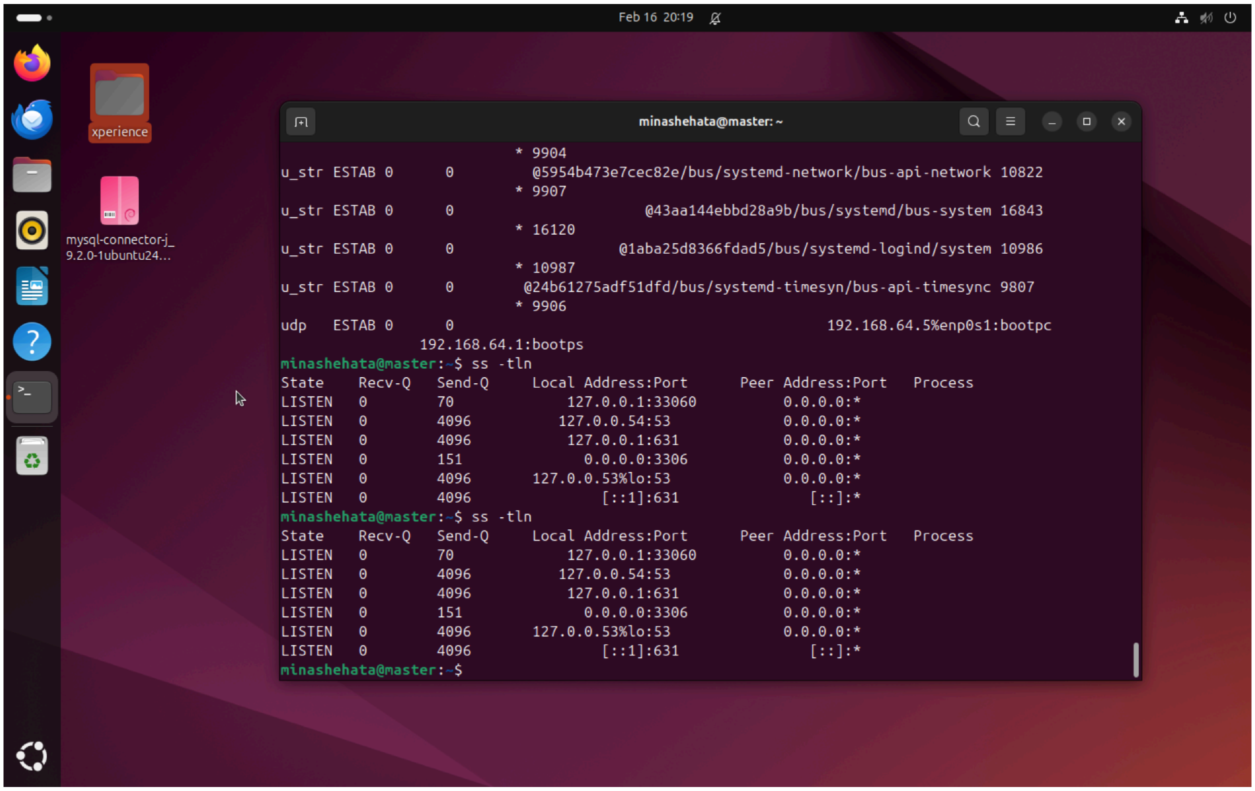
Command to run from host:

mysql -h 192.168.64.5 -u mina -p



```
Last login: Sun Feb 16 11:25:39 on ttys001
(bsa-4650) minashehata@Minas-MacBook-Air ~ % mysql -h 192.168.64.5 -u xperience_user -p xperience

Enter password:
ERROR 1045 (28000): Access denied for user 'xperience_user'@'_gateway' (using password: YES)
(bsa-4650) minashehata@Minas-MacBook-Air ~ % mysql -h 192.168.64.5 -u mina -p

Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.41-0ubuntu0.24.04.1 (Ubuntu)

Copyright (c) 2000, 2024, 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> quit
Bye
(bsa-4650) minashehata@Minas-MacBook-Air ~ % mysql -h 192.168.64.5 -u mina -p

Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.41-0ubuntu0.24.04.1 (Ubuntu)

Copyright (c) 2000, 2024, 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> 
```

Sockets:



```
                           * 9904
u_str ESTAB 0      0        @5954b473e7cec82e/bus/systemd-network/bus-api-network 10822
                           * 9907
u_str ESTAB 0      0                     @43aa144ebbd28a9b/bus/systemd/bus-system 16843
                           * 16120
u_str ESTAB 0      0                  @1aba25d8366fdad5/bus/systemd-logind/system 10986
                           * 10987
u_str ESTAB 0      0       @24b61275adf51dfd/bus/systemd-timesyn/bus-api-timesync 9807
                           * 9906
udp   ESTAB 0      0                                     192.168.64.5%enp0s1:bootpc
                192.168.64.1:bootps
minashehata@master:~$ ss -tln
State     Recv-Q    Send-Q     Local Address:Port      Peer Address:Port   Process
LISTEN    0         70             127.0.0.1:33060          0.0.0.0:*
LISTEN    0         4096          127.0.0.54:53             0.0.0.0:*
LISTEN    0         4096           127.0.0.1:631            0.0.0.0:*
LISTEN    0         151              0.0.0.0:3306           0.0.0.0:*
LISTEN    0         4096       127.0.0.53%lo:53             0.0.0.0:*
LISTEN    0         4096               [::1]:631               [::]:*
minashehata@master:~$ ss -tln
State     Recv-Q    Send-Q     Local Address:Port      Peer Address:Port   Process
LISTEN    0         70             127.0.0.1:33060          0.0.0.0:*
LISTEN    0         4096          127.0.0.54:53             0.0.0.0:*
LISTEN    0         4096           127.0.0.1:631            0.0.0.0:*
LISTEN    0         151              0.0.0.0:3306           0.0.0.0:*
LISTEN    0         4096       127.0.0.53%lo:53             0.0.0.0:*
LISTEN    0         4096               [::1]:631               [::]:*
minashehata@master:~$
```

Documentation for Two Docker Images with Jenkins:

Step 1: Setup Jenkins Pipeline

1.1 Create a New Pipeline in Jenkins:

- Open Jenkins in your web browser.
- Click on **New Item** in the Jenkins dashboard.
- Choose **Pipeline**, then name it (e.g., `docker-build-pipeline`).
- Click **OK**.

1.2 Configure Pipeline Script:

- In the pipeline configuration, scroll to **Pipeline** section.
- Set **Definition** to **Pipeline script** and enter your pipeline script in the text box.

Step 2: Write Jenkins Pipeline Script

2.1 Example Pipeline Script:

Here’s a basic example of a Jenkins pipeline that creates and runs two Docker images:

```
pipeline {
    agent any

    environment {
        DOCKER_IMAGE_1 = 'your-image-name-1'
        DOCKER_IMAGE_2 = 'your-image-name-2'
    }

    stages {
        stage('Checkout Code') {
            steps {
                git
'https://github.com/your-repo/your-project.git'
            }
        }

        stage('Build Docker Image 1') {
            steps {
                script {
                    // Build the first Docker image
                    sh 'docker build -t $DOCKER_IMAGE_1
./path-to-dockerfile-1'
                }
            }
        }
        stage('Build Docker Image 2') {
            steps {
                script {
                    // Build the second Docker image
                    sh 'docker build -t $DOCKER_IMAGE_2
./path-to-dockerfile-2'
                }
            }
        }

        stage('Run Docker Images') {
            steps {
                script {
                    // Stop any running containers of the same
image
                    sh 'docker stop $DOCKER_IMAGE_1 || true'
                    sh 'docker stop $DOCKER_IMAGE_2 || true'
                    sh 'docker rm $DOCKER_IMAGE_1 || true'
                    sh 'docker rm $DOCKER_IMAGE_2 || true'

                    // Run the Docker containers
                    sh 'docker run -d --name $DOCKER_IMAGE_1
$DOCKER_IMAGE_1'
                    sh 'docker run -d --name $DOCKER_IMAGE_2
$DOCKER_IMAGE_2'
                }
            }
        }

        stage('Post Build Cleanup') {
            steps {
                script {
                    // Optionally, clean up the Docker images
                    sh 'docker system prune -f'
                }
            }
        }
    }
}
```

Explanation of Each Stage:

- **Checkout Code:** Fetches the code from your Git repository.
- **Build Docker Image 1 & 2:** Build two Docker images using the Dockerfiles located in your project directories.
- **Run Docker Images:** Stops and removes any existing containers using the same names, then starts new containers using the newly built images.
- **Post Build Cleanup:** Cleans up unused Docker resources (optional).

Step 3: Run the Jenkins Pipeline

1. Once your Jenkins pipeline is set up, click **Build Now** in the pipeline project.
2. Jenkins will start executing the pipeline:
    - It will pull the latest code from your Git repository.
    - Build the Docker images.
    - Run the Docker containers.
3. You can monitor the build output in the Jenkins console output.

Step 4: Verify Docker Containers

After the pipeline runs, you can verify that the Docker containers are running by executing the following command on your terminal:

```
docker ps
```

This will show all the currently running containers, including the ones created by Jenkins.

Running the Tests from the Host Machine

The tests are designed to interact with the servers you have deployed. To run the tests from the **host machine**, you need to use a **test client** that connects to the servers running in Docker containers on your **VM**.

Ensure the Professor’s Test File is on Your Host Machine

1. T**est file** (e.g., XPerienceTests.jar) is available on your **host machine**.

Run the Test Using nc (Netcat)

2. If the tests or your own tests use **TCP communication** (via Netcat), you can use nc to send events directly to the XPerience servers running in Docker containers. Here's how you can interact with the servers:

Example: Sending an Event to the Memory Server

If the **memory server** is running on port **8000** in the Docker container, you can use nc to send event data directly to the server.

Connect to the server using nc (Netcat):

```
echo -n "name#2025-04-09#14:00#Sample Event#password123" | nc <VM-IP> 8000
```

1.
   - This command sends a string (representing an event) to the **memory server** via port **8000** on your **VM**.

    - <VM-IP> should be replaced with the IP address of your **VM** where the Docker container is running.

    - The event data consists of: name, date, time, description, and password (as shown in the example).

For the DB server, use a similar command to interact with the server on port **9000**:

```
echo -n "name#2025-04-09#14:00#Sample Event#password123" | nc <VM-IP> 9000
```

Test Interaction

After running the tests using nc, the event data will be sent to the **XPerience server** (either memory or DB server), and the server will process the data and send a response.

You can verify the results by checking the responses printed by nc (or any other output configured by the test script).