

# Setting up OpenOLAT

---

## OpenOLAT Deployment on a Virtual Machine (VM)

---

### 1. System Requirements

#### 1.1. Operating System

- Ensure you're using a Debian-based system (Debian 10 or 11 recommended).
  - **Command to check OS version:**

```
lsb_release -a
```

#### 1.2. RAM

- Minimum: 2 GB (sufficient for testing or low-usage scenarios).
- Recommended: 4 GB or more for production environments.

#### 1.3. Disk Space

- Minimum of 10 GB required.
  - **Tip:** If you're expecting high traffic or large datasets, allocate more disk space based on projected usage.

#### 1.4. CPU

- Multi-core processor recommended for better handling of concurrent tasks.
  - **Tip:** Virtual CPUs should be scaled based on anticipated user load.

---

## 2. Software Requirements

### 2.1. Java 17 (OpenJDK)

- OpenOLAT requires Java 17. Use OpenJDK 17.
  - **Command to install Java 17.0.13+11-jre:**

```
sudo apt install openjdk-17-jdk -y
```

- **Command to verify installation:**

```
java -version
```

### 2.2. Apache Tomcat 10.1.31

- Apache Tomcat serves as the application server for OpenOLAT.
  - **Steps to install Tomcat 10.1.31:**

### 1. Download Tomcat 10.1.31:

```
wget https://dlcdn.apache.org/tomcat/tomcat-10/v10.1.31/bin/apache-tomcat-10.1.31.tar.gz
```

### 2. Extract the tar file:

```
tar xvf apache-tomcat-10.1.31.tar.gz
```

### 3. Create a symbolic link for easier version management:

```
ln -s apache-tomcat-10.1.31 tomcat
```

## 2.3. PostgreSQL (Version 12 or Higher)

- PostgreSQL is the database for OpenOLAT. Ensure you install version 12 or later.
  - **Command to install PostgreSQL:**

```
sudo apt install postgresql postgresql-contrib -y
```

- **Start and enable the service:**

```
sudo systemctl start postgresql  
sudo systemctl enable postgresql
```

## 2.4. Optional: Eclipse IDE

- You may use the Eclipse IDE for deployment management, though it is optional for those who prefer a graphical interface for managing Tomcat and debugging.

---

## 3. Network Requirements

### 3.1. Network Stability

- Ensure your VM has a stable and reliable network connection, particularly for external access.

### 3.2. Firewall Configuration

- **Tomcat Default Port:** 8080
- **PostgreSQL Default Port:** 5432

#### note

Install ufw using

```
sudo apt install ufw
```

- **Command to open ports:**

```
sudo ufw allow 8080/tcp  
sudo ufw allow 5432/tcp  
sudo ufw allow 8088/tcp
```

---

## 4. User Privileges(Optional)

### 4.1. Sudo Access

- Ensure that you have `sudo` privileges to install necessary packages and make system changes.
  - **Command to check if you have sudo access:**

```
sudo -v
```

### 4.2. Create the OpenOLAT User

- Create a dedicated user `openolat` for running the application.
  - **Command to create the user:**

```
sudo useradd -m -s /bin/bash openolat
```

- **Switch to the new user:**

```
su - openolat
```

---

## 5. Database Setup (PostgreSQL)

### 5.1. Login as PostgreSQL Superuser

- Switch to the `postgres` user to create databases and users.
  - **Command to switch to postgres user:**

```
sudo su - postgres
```

### 5.2. Create PostgreSQL User and Database

1. Access the PostgreSQL shell:

```
psql
```

2. Create a new user (`himanshu_mahajan`) with a password:

```
CREATE USER himanshu_mahajan WITH PASSWORD ' ';
```

3. Create a new database (`oodb`) owned by the user:

```
CREATE DATABASE oodb WITH OWNER himanshu_mahajan;
```

### 5.3. Verify the Database Connection

- Exit the PostgreSQL shell and test the connection as the `openolat` user:

```
psql oodb -U himanshu_mahajan -h localhost
```

---

## 6. Backup Strategy

- Set up regular backups for both your database and OpenOLAT data.
  - **Database backup command:**

```
pg_dump oodb > backup_oodb.sql
```

---

## 7. Java Environment Variables

- Set `JAVA_HOME` and update the `PATH` variable.

To directly set `JAVA_HOME` and update the `PATH` from the command line (and automatically append these changes to your `.bashrc` file), you can run the following command:

```
echo 'export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64' >> ~/.bashrc
echo 'export PATH=$PATH:$JAVA_HOME/bin' >> ~/.bashrc
```

- **Activate changes:**

```
source ~/.bashrc
```

---

## 8. Download OpenOLAT

1. **Create a downloads directory:(Optional/not recommended)**

```
mkdir ~/downloads && cd ~/downloads
```

2. **Download the latest OpenOLAT `.war` file (OpenOLAT 18.2.11):**

```
wget https://www.openolat.com/releases/openolat_18211.war
```

3. **Unzip the `.war` file:**

```
unzip openolat_18211.war -d ~/openolat-18.2.11
```

---

Do not forget to change directories

## 9. Apache Tomcat Configuration for OpenOLAT

### 9.1. Create Tomcat Directories

1. **Set up the required directories:**

```
mkdir ~/bin ~/conf ~/lib ~/logs ~/run
```

#### 9.1.1 Create some other links

```
cd ~/conf
ln -s home/himanshu_mahajan/tomcat/conf/web.xml web.xml
```

```
cd ~/bin
ln -s home/himanshu_mahajan/tomcat/bin/catalina.sh catalina.sh
cd
ln -s tomcat/bin/startup.sh start
ln -s tomcat/bin/shutdown.sh stop
```

## 9.2. Configure Tomcat Environment

1. Create a `setenv.sh` script to define environment variables:

```
cat << EOF > ~/bin/setenv.sh
CATALINA_HOME=~ /tomcat
CATALINA_BASE=~ # If your Tomcat base is the home directory; change this if needed
JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64 # Correct path for OpenJDK 17
CATALINA_PID=~ /run/openolat.pid
CATALINA_OPTS="-Xmx1024m -Xms512m -XX:MaxMetaspaceSize=512m -Duser.name=openolat"
EOF
```

2. Make `setenv.sh` executable:

```
chmod +x ~/bin/setenv.sh
```

## 9.3. Configure `server.xml`

- Create the Tomcat server configuration file at `~/conf/server.xml`:

```
cat << EOF > ~/conf/server.xml
<Server port="8085" shutdown="SHUTDOWN">
  <Service name="Catalina">
    <Connector port="8088" protocol="HTTP/1.1"/>
    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps"/>
    </Engine>
  </Service>
</Server>
EOF
```

## 9.4. Start Tomcat

1. Start the Tomcat server:

```
./start
```

2. Check the logs for successful startup:

```
tail -f ~/logs/catalina.out
```

---

## 10. OpenOLAT Configuration

### 10.1. Configure Database Access

1. Create a file `olat.local.properties` in `~/lib/` with the following content:

```
cat << EOF > ~/lib/olat.local.properties
db.source=jndi
db.jndi=java:comp/env/jdbc/openolatDS
db.vendor=postgresql
server.contextpath=/himanshu_mahajan
server.domainname=localhost
server.port=8088
EOF
```

### 10.2. Configure Application Context

1. Create the application context directory and file:

```
mkdir -p ~/conf/Catalina/localhost/
```

2. Create the `ROOT.xml` file in this directory:

```
cat << EOF ><Context path="" docBase="/home/himanshu_mahajan/webapp"
debug="0" reloadable="false" allowLinking="true">
  <Resource name="jdbc/openolatDS" auth="Container"
type="javax.sql.DataSource"
  maxTotal="16" maxIdle="4" maxWaitMillis="60000"
  username="himanshu_mahajan" password=" "
  driverClassName="org.postgresql.Driver"
  validationQuery="SELECT 1"
  url="jdbc:postgresql://localhost:5432/oodb"/>
</Context> EOF
```

### 3. Configure `log4j2`

Create the file `~/lib/log4j2.xml` containing

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
  <Appenders>
    <RollingFile name="RollingFile"
fileName="/home/himanshu_mahajan/logs/olat.log"
    filePattern="/home/himanshu_mahajan/logs/olat.log.%d{yyyy-MM-
dd}">
    <PatternLayout
    pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %-5level
```

```

%marker %c{1} ^%%^ I%X{ref}-J%sn ^%%^ %logger{36} ^%%^ %X{identityKey} ^%%^
%X{ip} ^%%^ %X{referer} ^%%^ %X{userAgent} ^%%^ %msg%ex{full,separator(
)}}%n" />
    <Policies>
        <TimeBasedTriggeringPolicy interval="1" />
    </Policies>
</RollingFile>
</Appenders>
<Loggers>
    <Logger name="org.apache.commons.httpclient" additivity="false"
level="warn">
        <AppenderRef ref="RollingFile" />
    </Logger>
    <Logger name="org.apache.pdfbox" additivity="false" level="fatal">
        <AppenderRef ref="RollingFile" />
    </Logger>
    <Logger name="org.apache.fontbox" additivity="false" level="fatal">
        <AppenderRef ref="RollingFile" />
    </Logger>
    <Logger
name="org.hibernate.engine.internal.StatisticalLoggingSessionEventListener"
additivity="false" level="fatal">
        <AppenderRef ref="RollingFile" />
    </Logger>
    <!-- Change the level to debug to see the SQL statements generated by
Hibernate -->
    <Logger name="org.hibernate.SQL" additivity="false" level="fatal">
        <AppenderRef ref="RollingFile" />
    </Logger>
    <Logger name="org.hibernate.type.descriptor.sql.BasicBinder"
additivity="false" level="fatal">
        <AppenderRef ref="RollingFile" />
    </Logger>
    <Logger name="org.apache.activemq.audit" additivity="false"
level="warn">
        <AppenderRef ref="RollingFile" />
    </Logger>
    <Root level="info">
        <AppenderRef ref="RollingFile" />
    </Root>
</Loggers>
</Configuration>

```

# 11. Testing OpenOLAT

## 1. Access the application in your browser:

```
http://localhost:8088
```

## 2. Default login credentials:

- **Username:** administrator
- **Password:** openolat

## Errors debugging

1. cannot open `'/home/himanshu_mahajan/logs/catalina.out'` for reading: No such file or directory

Solution: The error message indicates that the file `catalina.out` does not exist in the `~/logs/` directory. This could be due to several reasons, including that the logs are stored in a different location or that Tomcat hasn't started yet to generate any logs.

## Steps to Resolve

### 1. Check the Logs Directory

First, verify that the `~/logs/` directory exists and check where Tomcat is configured to store logs. By default, the logs are typically located in the `logs` directory under the Tomcat installation folder. Navigate to the Tomcat installation directory and check if the `logs` folder exists:

```
cd ~/tomcat
ls -l logs
```

### 2. Start Tomcat

If the logs directory does not contain `catalina.out`, it may be because Tomcat has not started or there was an error during startup. Start Tomcat using:

```
cd ~/tomcat/bin
./catalina.sh start
```

### 3. Check for Errors

If Tomcat fails to start, you can check the other log files in the `logs` directory (like `catalina.yyyy-MM-dd.log`) for any error messages that might indicate what went wrong.

### 4. Alternative Log Location

If you configured the logging to point to a different location, check your `server.xml` or logging configuration files in the `~/tomcat/conf/` directory to see if the logs are directed elsewhere.

### 5. Using Tail on the Correct Log File

Once you confirm the location of `catalina.out` or the relevant log file, you can use the `tail` command to monitor it:

```
tail -f ~/tomcat/logs/catalina.out
```

or, if you found an alternative log file:

```
tail -f ~/tomcat/logs/catalina.yyyy-MM-dd.log
```