Using docker pull tomcat in docker on ubuntu 22.04

Q1 >>

how to using docker pull tomcat in docker on ubuntu 22.04 how to deploy java Application with simple program and code using httpd with command and example? docker pull tomcat

Answers ➔ To deploy a Java application using Apache Tomcat and HTTPD on Docker in Ubuntu 22.04, follow these steps:

Prerequisites

- Docker installed on Ubuntu 22.04.
- Basic understanding of Docker and Docker Compose.

Step-by-Step Guide

1. Install Docker

If Docker is not already installed on your Ubuntu machine, you can install it using the following commands:

bash

sudo apt update

sudo apt install -y docker.io

sudo systemctl start docker

sudo systemctl enable docker

2. Pull the Tomcat Docker Image

Pull the official Tomcat Docker image from Docker Hub:

Bash➔ docker pull tomcat

3. Create a Simple Java Web Application

Create a simple Java Servlet application. Here is an example structure and code for the web application:

Directory Structure:

```
css
myapp/
├── Dockerfile
├── webapps/
│   └── myapp.war
└── src/
    └── HelloWorld.java
```

HelloWorld.java:

```
java
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


public class HelloWorld extends HttpServlet {
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.getWriter().println("Hello, World!");
   }
}
```

Compile the Java servlet and package it into a WAR file (Web Archive). You can use a build tool like Maven or manually create the WAR file.

Using Maven:

- Create a pom.xml file for the project.
- Package the application using mvn package.

This will generate myapp.war in the target directory.

4. Dockerfile for Tomcat

Create a Dockerfile to deploy the WAR file to Tomcat:

Dockerfile:

Dockerfile

Copy code

```
FROM tomcat:latest

COPY webapps/myapp.war /usr/local/tomcat/webapps/
```

5. Build the Docker Image

Navigate to the directory containing the Dockerfile and build the Docker image:

Bash➔ docker build -t myapp-tomcat .

6. Run the Tomcat Container

Run the container with the custom image:

Bash➔ docker run -d -p 8080:8080 myapp-tomcat

Your Java application should now be accessible at http://localhost:8080/myapp.

7. Set Up Apache HTTPD with Docker

To use Apache HTTPD as a reverse proxy for Tomcat, follow these steps:

Create a Dockerfile for HTTPD:

Dockerfile

```
FROM httpd:latest

COPY ./httpd.conf /usr/local/apache2/conf/httpd.conf
```

httpd.conf: Make sure the httpd.conf is configured to proxy requests to the Tomcat server. Here is an example configuration:

conf

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so

<VirtualHost *:80>
    ProxyPreserveHost On
    ProxyPass / http://tomcat:8080/
    ProxyPassReverse / http://tomcat:8080/
</VirtualHost>
```

8. Docker Compose for Running Both Containers

Create a docker-compose.yml file to run both Tomcat and HTTPD:

```
yaml
version: '3'
services:
  tomcat:
    image: myapp-tomcat
    ports:
      - "8080:8080"

  httpd:
    build: ./httpd
    ports:
      - "80:80"
    depends_on:
      - tomcat
```

Place this file in the root directory of your project.

9. Start the Containers

Use Docker Compose to start both containers:

bashdocker-compose up -d

Now, your Java application should be accessible through Apache HTTPD at http://localhost.

Summary

This guide demonstrates how to:

1. Pull and set up a Tomcat Docker image.
2. Create and deploy a simple Java web application.
3. Use Apache HTTPD as a reverse proxy for Tomcat using Docker Compose.

This setup ensures your Java application is served via HTTPD and managed efficiently with Docker.

4o