

# [CS304] Tutorial 12 - Docker

---

## Docker overview

Docker is an open-source engine that automates the deployment of applications into containers.

It is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

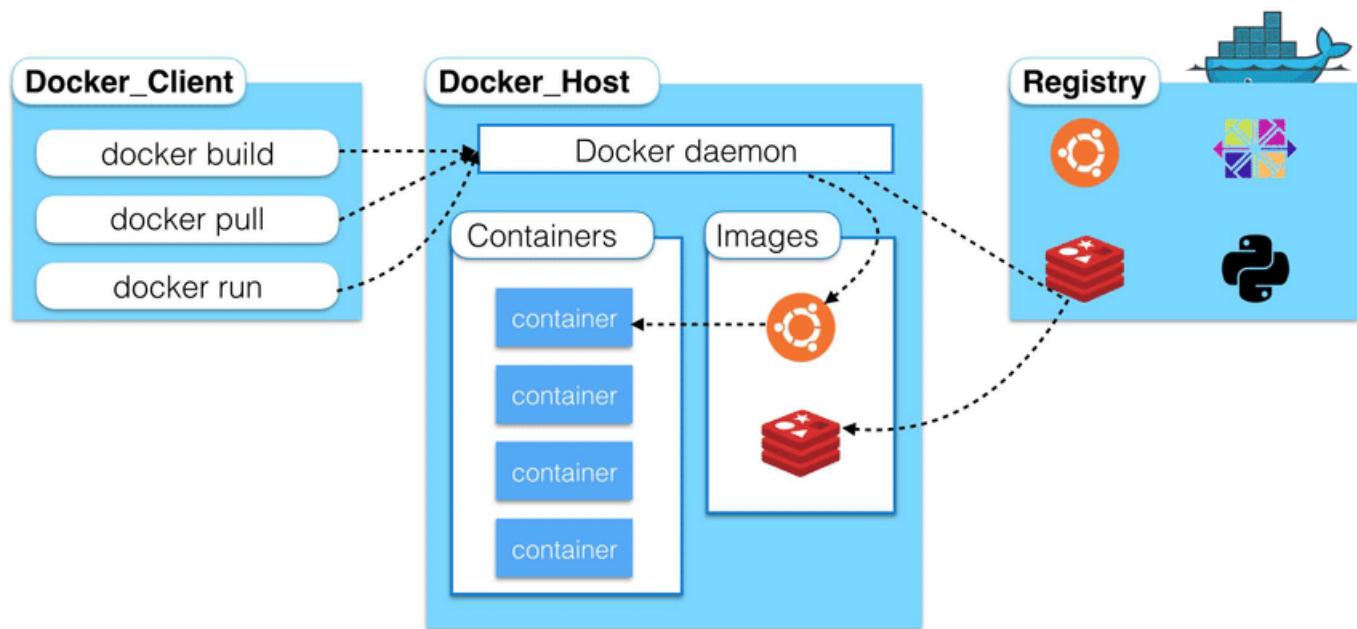
Official Resources:

- Docker : <http://www.docker.com>
- Docker Hub : <https://hub.docker.com>

## **What can I use Docker for:**

- Fast, consistent delivery of your applications.
- Responsive deployment and scaling.
- Running more workloads on the same hardware.

## Docker components



## Images:

When running a container, it uses an isolated filesystem. This custom filesystem is provided by a container image. Since the image contains the container's filesystem, it must contain everything needed to run an application - all dependencies, configurations, scripts, binaries, etc. The image also contains other configuration for the container, such as environment variables, a default command to run, and other metadata.

## Containers:

Simply put, a container is a sandboxed process on your machine that is isolated from all other processes on the host machine. That isolation leverages kernel namespaces and cgroups, features that have been in Linux for a long time. Docker has worked to make these capabilities approachable and easy to use. To summarize, a container:

- Is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI.
- Can be run on local machines, virtual machines or deployed to the cloud.
- Is portable (can be run on any OS).
- Is isolated from other containers and runs its own software, binaries, and configurations.

**Registries:** Docker Hub is the world's largest library and community for container images (from its website)

---

## Part I: Installing Docker

Please refer below to install docker in Ubuntu:

<https://docs.docker.com/engine/install/ubuntu/#installation-methods>

If you want manage Docker as a non-root user:

<https://docs.docker.com/engine/install/linux-postinstall/#manage-docker-as-a-non-root-user>

Verify that the Docker Engine installation is successful by running the `hello-world` image.

```
# Enable non-root access
sudo usermod -aG docker $USER
newgrp docker

# Verify installation
docker run hello-world
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

<https://hub.docker.com/>

For more examples and ideas, visit:

<https://docs.docker.com/get-started/>

## Part II: Run Teedy in Docker manually

Before using Jenkins, let's build a Docker image manually to understand the process.

### 2.1 Build Image

#### 1. Dockerfile (Example)

This is the Dockerfile used to build the Teedy project image. Each section installs required dependencies and configures Jetty and OCR tools.

```
# Using a custom image that already includes Ubuntu 22.04
FROM ubuntu:22.04

# Using the LABEL instruction to add metadata to the image, sets the maintainer's
# email #address for the image
LABEL maintainer="b.gamard@sismics.com"

# Run Debian in non interactive mode
ENV DEBIAN_FRONTEND noninteractive

# Configure environment variables
ENV LANG C.UTF-8
ENV LC_ALL C.UTF-8
ENV JAVA_HOME /usr/lib/jvm/java-11-openjdk-amd64/
ENV JAVA_OPTIONS -Dfile.encoding=UTF-8 -Xmx1g
ENV JETTY_VERSION 11.0.20
ENV JETTY_HOME /opt/jetty

# clean up unnecessary files after installation to reduce the size of the image.
# Install necessary packages and OCR languages
RUN apt-get update && \
    apt-get -y -q --no-install-recommends install \
    vim less procps unzip wget tzdata openjdk-11-jdk \
    ffmpeg \
```

```
mediainfo \
tesseract-ocr \
tesseract-ocr-ara \
tesseract-ocr-ces \
tesseract-ocr-chi-sim \
tesseract-ocr-chi-tra \
tesseract-ocr-dan \
tesseract-ocr-deu \
tesseract-ocr-fin \
tesseract-ocr-fra \
tesseract-ocr-heb \
tesseract-ocr-hin \
tesseract-ocr-hun \
tesseract-ocr-ita \
tesseract-ocr-jpn \
tesseract-ocr-kor \
tesseract-ocr-lav \
tesseract-ocr-nld \
tesseract-ocr-nor \
tesseract-ocr-pol \
tesseract-ocr-por \
tesseract-ocr-rus \
tesseract-ocr-spa \
tesseract-ocr-swe \
tesseract-ocr-tha \
tesseract-ocr-tur \
tesseract-ocr-ukr \
tesseract-ocr-vie \
tesseract-ocr-sqi \
&& apt-get clean && \
rm -rf /var/lib/apt/lists/*
RUN dpkg-reconfigure -f noninteractive tzdata

# Install Jetty server
RUN wget -nv -O /tmp/jetty.tar.gz \
"https://repo1.maven.org/maven2/org/eclipse/jetty/jetty-home/\${JETTY\_VERSION}/jetty-home-\${JETTY\_VERSION}.tar.gz" \
&& tar xzf /tmp/jetty.tar.gz -C /opt \
&& mv /opt/jetty* /opt/jetty \
&& useradd jetty -U -s /bin/false \
&& chown -R jetty:jetty /opt/jetty \
&& mkdir /opt/jetty/webapps \
&& chmod +x /opt/jetty/bin/jetty.sh

# informs Docker that the container will listen on port 8080 at runtime
EXPOSE 8080

# Install the application
RUN mkdir /app && \
cd /app && \
java -jar /opt/jetty/start.jar --add-modules=server,http,webapp,deploy

# Add the local file docs.xml and the built WAR file docs-web-*.war to the
# container's Jetty web applications directory. Allows Jetty to load these web
```

```
applications at startup.  
ADD docs.xml /app/webapps/docs.xml  
ADD docs-web/target/docs-web-*.war /app/webapps/docs.war  
  
# sets the working directory for any RUN, CMD, ENTRYPOINT, COPY, and ADD  
instructions that follow it  
WORKDIR /app  
  
# sets the default command that will run when a container is started from the  
image.  
# Here it tells the container to run the Jetty web server by executing the  
start.jar file with Java  
CMD ["java", "-jar", "/opt/jetty/start.jar"]
```

## 2. Build Image:

```
docker build -t teedy2025_manual .
```

```
[ao_hao@DESKTOP-4JVIQPU:/mnt/d/github/Teedy-doc$ sudo docker build -t teedy2025_manual .  
[+] Building 264.5s (5/12) docker:default  
=> [internal] load build definition from Dockerfile 0.1s  
=> => transferring dockerfile: 2.10kB 0.0s  
=> [internal] load metadata for docker.io/library/ubuntu:22.04 3.6s  
=> [internal] load .dockerignore 0.1s  
=> => transferring context: 2B 0.0s  
=> [1/8] FROM docker.io/library/ubuntu:22.04@sha256:d80997daaa3811b175119350d84305e1ec9129e1799bba0bd1e3120da3ff 6.4s  
=> => resolve docker.io/library/ubuntu:22.04@sha256:d80997daaa3811b175119350d84305e1ec9129e1799bba0bd1e3120da3ff 0.0s  
=> => sha256:d80997daaa3811b175119350d84305e1ec9129e1799bba0bd1e3120da3ff52c3 6.69kB / 6.69kB 0.0s  
=> => sha256:a76d0e9d99f0e91640e35824a6259c93156f0f07b7778ba05808c750e7fa6e68 424B / 424B 0.0s  
=> => sha256:cc934a90cd99a939f3922f858ac8f055427300ee3ee4dfcd303c53e571d0aeab 2.30kB / 2.30kB 0.0s  
=> => sha256:30a9c22ae099393b0131322d7f50d8a9d7cd06c5e518cd27a19ac960a4d0aba3 29.53MB / 29.53MB 3.2s  
=> => extracting sha256:30a9c22ae099393b0131322d7f50d8a9d7cd06c5e518cd27a19ac960a4d0aba3 2.8s  
=> [internal] load build context 132.0s  
=> => transferring context: 88.45MB 131.9s  
=> [2/8] RUN apt-get update && apt-get -y -q --no-install-recommends install vim less procps unzip wge 254.1s  
=> => # Selecting previously unselected package libzvbi0:amd64.  
=> => # Preparing to unpack .../libzvbi0_0.2.35-19_amd64.deb ...  
=> => # Unpacking libzvbi0:amd64 (0.2.35-19) ...  
=> => # Selecting previously unselected package libavcodec58:amd64.  
=> => # Preparing to unpack .../libavcodec58_7%3a4.4.2-0ubuntu0.22.04.1_amd64.deb ...  
=> => # Unpacking libavcodec58:amd64 (7%3a4.4.2-0ubuntu0.22.04.1) ...
```

```
ao_hao@DESKTOP-4JVIQPU: / + - x
=> => sha256:d80997daaa3811b175119350d84305e1ec9129e1799bba0bd1e3120da3ff52c3 6.69kB / 6.69kB 0.0s
=> => sha256:a76d0e9d99f0e91640e35824a6259c93156f0f07b7778ba05808c750e7fa6e68 424B / 424B 0.0s
=> => sha256:cc934a90cd99a939f3922f858ac8f055427300ee3ee4dfcd303c53e571d0aeab 2.30kB / 2.30kB 0.0s
=> => sha256:30a9c22ae099393b0131322d7f50d8a9d7cd06c5e518cd27a19ac960a4d0aba3 29.53MB / 29.53MB 3.2s
=> => extracting sha256:30a9c22ae099393b0131322d7f50d8a9d7cd06c5e518cd27a19ac960a4d0aba3 2.8s
=> [internal] load build context 132.0s
=> => transferring context: 88.45MB 131.9s
=> [2/8] RUN apt-get update && apt-get -y -q --no-install-recommends install vim less procps unzip wge 303.1s
=> [3/8] RUN dpkg-reconfigure -f noninteractive tzdata 0.9s
=> [4/8] RUN wget -nv -O /tmp/jetty.tar.gz "https://repo1.maven.org/maven2/org/eclipse/jetty/jetty-home/1 1607.8s
=> [5/8] RUN mkdir /app && cd /app && java -jar /opt/jetty/start.jar --add-modules=server,http,webapp,de 2.3s
=> [6/8] ADD docs.xml /app/webapps/docs.xml 0.1s
=> [7/8] ADD docs-web/target/docs-web-*.war /app/webapps/docs.war 0.7s
=> [8/8] WORKDIR /app 0.1s
=> exporting to image 24.8s
=> => exporting layers 24.8s
=> => writing image sha256:bc2ac81973ae9d775c25cb53e25c0baccc4f12d8741c8a90ca15b3099f816459 0.0s
=> => naming to docker.io/library/teedy2025_manual 0.0s

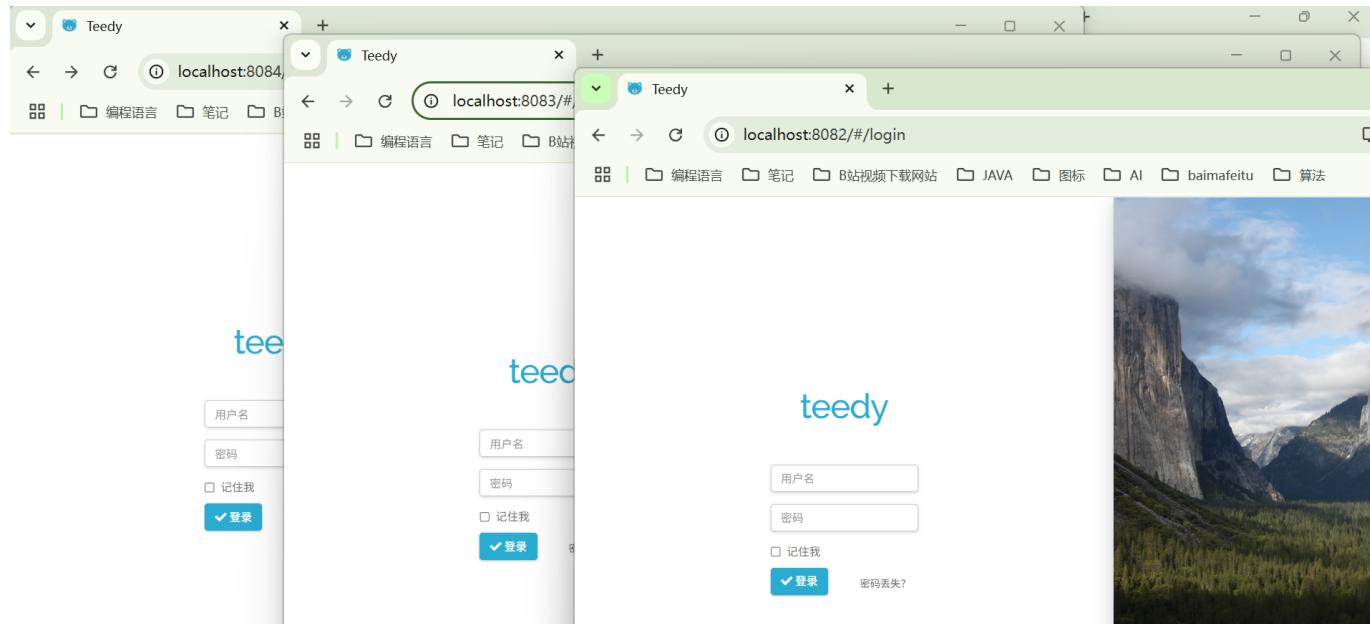
7 warnings found (use docker --debug to expand):
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 13)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 5)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 8)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 9)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 10)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 11)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 12)
```

### 3. Run Containers:

```
docker run -d -p 8084:8080 --name teedy_manual01 teedy2025_manual
docker run -d -p 8083:8080 --name teedy_manual02 teedy2025_manual
docker run -d -p 8082:8080 --name teedy_manual03 teedy2025_manual
```

**Tip:** `-d` flag runs the container in background. `-p` maps host port to container port.

```
ao_hao@DESKTOP-4JVIQPU:/mnt/d/github/Teedy-doc$ docker run -d -p 8084:8080 --name teedy_manual01 teedy2025_manual
docker run -d -p 8083:8080 --name teedy_manual02 teedy2025_manual
docker run -d -p 8082:8080 --name teedy_manual03 teedy2025_manual
c08c6d29fe71f8c128218176555986a6b077052f23ba36b9fefc23c832aa020e
64437e505e8a44c1f03de6fd3a20b8f2ffabb5419692c410b928e7e4291798a
2eb4a88a3ae04b1994e287c1137e5a786d24aa7d71789d9bb314e1e0017953e7
```



## Manage Docker Containers (Optional Section)

If you need to stop, restart, or delete a Docker container manually:

- **Stop a running container:**

```
sudo docker stop teedy_manual01
```

- **Start a stopped container:**

```
sudo docker start teedy_manual01
```

- **Force stop a container:**

```
sudo docker kill teedy_manual01
```

- Remove a stopped container:

```
sudo docker rm teedy_manual01
```

- Force remove a running container:

```
sudo docker rm -f teedy_manual01
```

- Check status of containers:

```
docker ps -a
```

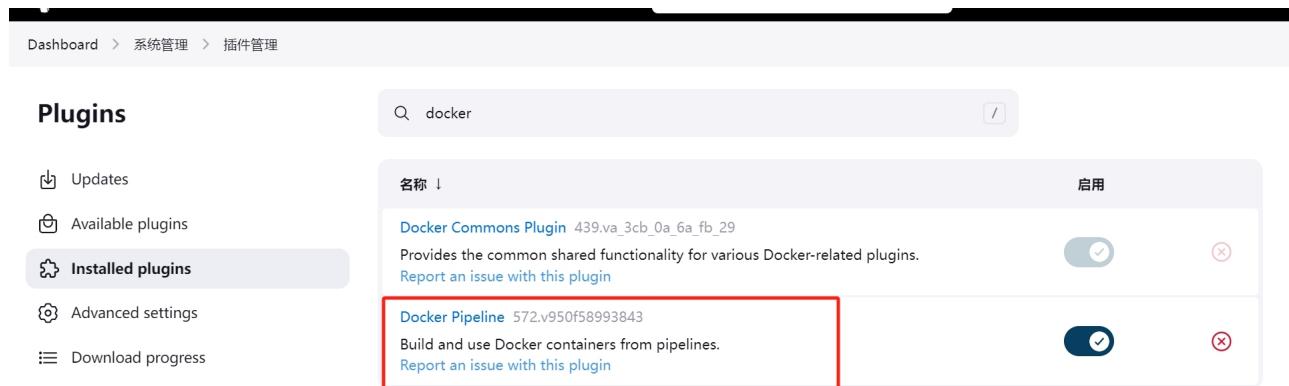
- View logs of a container:

```
docker logs teedy_manual01
```

## Part III Automated Deployment with Jenkins

### 1. Configure jenkins:

- Install "Docker Pipeline" in your Jenkins. Open "Manage Plugins" and find it under "Available Plugins" and install it:



- **Configure Docker Hub account in Jenkins:** Go to “Manage Jenkins” → “Manage Credentials” → “System” → “New Domain” → “Add Credentials” to add your Docker Hub credentials. If you don’t have a Docker Hub account yet, sign up at <https://hub.docker.com/> and create a new repository (e.g., `xx/teedy`).

**凭据**

类型	提供者	存储	域	唯一标识	名称
System	System	全局			Stores scoped to Jenkins

**New domain**

域名 ? Docker-Hub

描述 ? Docker-Hub

规范 ? 新增

**Create**

**New credentials**

类型: Username with password

范围: 全局 (Jenkins, nodes, items, all child items, etc)

用户名: Dockerhub account

ID: set id

描述:

**Create**

**全局凭据 (unrestricted)**

Credentials that should be available irrespective of domain specification to requirements matching.

ID	名称	类型	描述
dockerhub_credentials	traccytian/******** (login dockerhub)	Username with password	login dockerhub

## 2. Jenkinsfile Pipeline Example:

```
pipeline {
    agent any

    environment {
        // define environment variable
        // Jenkins credentials configuration
        DOCKER_HUB_CREDENTIALS = credentials('dockerhub_credentials') // Docker Hub credentials ID store in Jenkins
        // Docker Hub Repository's name
        DOCKER_IMAGE = 'xx/teedy-app' // your Docker Hub user name and Repository's name
        DOCKER_TAG = "${env.BUILD_NUMBER}" // use build number as tag
    }

    stages {
        stage('Build') {
            steps {
                checkout scmGit(
                    branches: [[name: '*/master']],
                    extensions: [],
                    userRemoteConfigs: [[url: 'https://github.com/xx/Teedy.git']]]
                // your github Repository
                )
                sh 'mvn -B -DskipTests clean package'
            }
        }

        // Building Docker images
        stage('Building image') {
            steps {
                script {
                    // assume Dockerfile locate at root
                    docker.build("${env.DOCKER_IMAGE}:${env.DOCKER_TAG}")
                }
            }
        }

        // Uploading Docker images into Docker Hub
        stage('Upload image') {
            steps {
                script {
                    // sign in Docker Hub
                    docker.withRegistry('https://registry.hub.docker.com',
'DOCKER_HUB_CREDENTIALS') {
                        // push image

                        docker.image("${env.DOCKER_IMAGE}:${env.DOCKER_TAG}").push()

                            // :optional: label latest

                        docker.image("${env.DOCKER_IMAGE}:${env.DOCKER_TAG}").push('latest')
                    }
                }
            }
        }
    }
}
```

```
        }

    }

    // Running Docker container
    stage('Run containers') {
        steps {
            script {
                // stop then remove containers if exists
                sh 'docker stop teedy-container-8081 || true'
                sh 'docker rm teedy-container-8081 || true'

                // run Container
                docker.image("${env.DOCKER_IMAGE}:${env.DOCKER_TAG}").run(
                    '--name teedy-container-8081 -d -p 8081:8080'
                )
            }

            // Optional: list all teedy-containers
            sh 'docker ps --filter "name=teedy-container"'

        }
    }
}
```

### 3. Build the Docker Pipeline in Jenkins

Click "Build Now" and wait for results:

The screenshots show the Jenkins interface for a Docker task. The first screenshot shows the pipeline log with commands like `docker run` and `docker ps`. The second screenshot shows the build details for job #6, including the start time, duration, and commit information. The third screenshot shows the pipeline overview, which includes a summary of the stages: Start, Tool Install, Package, Building image, Upload image, Run containers, and End.

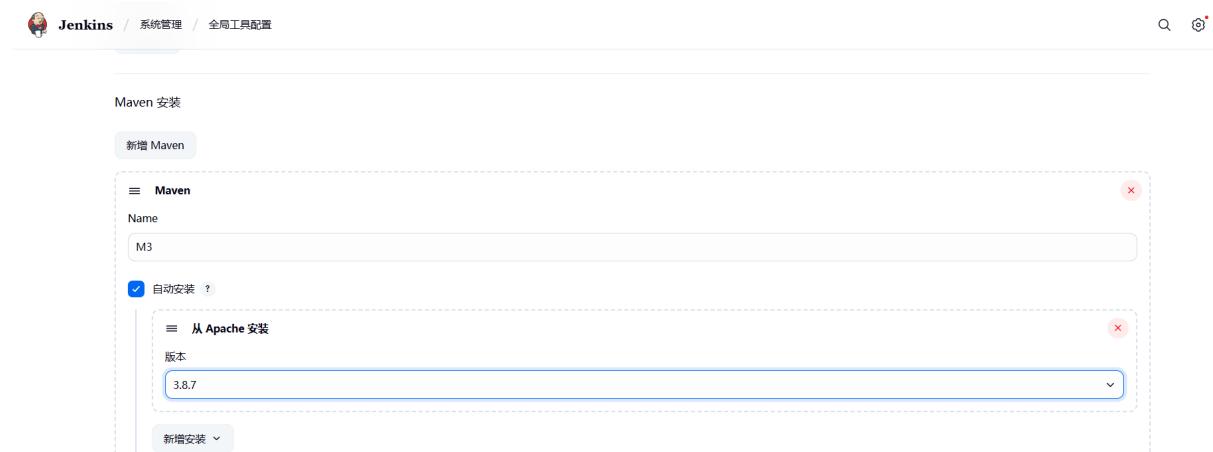
If a "Maven not found" error occurs during the build process, try configuring Maven in Jenkins Global Tools as follows:

### 1. Go to Jenkins Dashboard → System Manage → Global Tool Configuration

The screenshot shows the Jenkins Global Tool Configuration page. It includes sections for Maven (with a 'New Maven' button), Ant (with a 'New Ant' button), Gradle (with a 'New Gradle' button), and Docker (with a 'New Docker' button).

### 2. In the **Maven** section:

- **Name:** Enter a descriptive name (e.g., M3)
- **Install automatically:** Check this box to let Jenkins install Maven automatically
- **Version:** Select the desired version from the dropdown (e.g., 3.8.7)



3. Click **Save** to apply the changes.

4. In your `Jenkinsfile`, use the `tools` directive to reference the Maven installation configured above.

```
pipeline {
    agent any
    tools {
        // Use the name defined in Global Tool Configuration
        maven 'M3'
    }
    stages {
        stage('Build') {
            steps {
                sh 'mvn --version' // Verify Maven is available
                sh 'mvn -B -DskipTests clean package'
            }
        }
    }
}
```

## Part IV: Use GitHub Actions to Build and Push Docker Image

GitHub Actions provides CI/CD capabilities directly from GitHub. Here's how to use it to build and push Docker images.

Github Actions Build and push Docker images guide:

<https://github.com/marketplace/actions/build-and-push-docker-images>

Step 1: Prepare Secrets in GitHub Repo

In your repository, go to **Settings > Secrets and variables > Actions > New repository secret** and add:

- DOCKERHUB\_USERNAME
- DOCKERHUB\_TOKEN

The screenshot shows the 'Repository secrets' page in GitHub. On the left, there's a sidebar with 'Actions' highlighted by a red box labeled '2'. In the main area, two secrets are listed: 'DOCKERHUB\_TOKEN' and 'DOCKERHUB\_USERNAME', both highlighted by a blue box. A red box labeled '3' is at the top right, pointing to a 'New repository secret' button.

## Step 2: Config GitHub Workflow File

The screenshot shows the GitHub repository settings page with the 'Actions' tab selected, highlighted by a red box. Other tabs like 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings' are visible.

### Choose a workflow

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

[Skip this and set up a workflow yourself →](#)

Search workflows

#### Suggested for this repository

A workflow card for 'Docker image' by GitHub Actions. It says: 'Build a Docker image to deploy, run, or push to a registry.' A red box highlights the 'Configure' button. Below it is a 'Dockerfile' link.

A workflow card for 'Grunt' by GitHub Actions. It says: 'Build a NodeJS project with npm and grunt.' A red box highlights the 'Configure' button. Below it is a 'JavaScript' link.

A workflow card for 'Publish Java Package with Gradle' by GitHub Actions. It says: 'Build a Java Package using Gradle and publish to GitHub Packages.' A red box highlights the 'Configure' button. Below it is a 'Java' link.

The screenshot shows the GitHub repository settings page with the 'Actions' tab selected. Other tabs like 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings' are visible.

[Teedy-doc/.github/workflows/docker-image.yml](#) in [master](#)

The screenshot shows the GitHub workflow file editor with the 'Edit' tab selected. The code is as follows:

```

1 name: Docker Image CI
2
3 on:
4   push:
5     branches: [ "master" ]
6   pull_request:
7     branches: [ "master" ]
8
9 jobs:
10
11   build:
12
13     runs-on: ubuntu-latest
14
15     steps:
16       - uses: actions/checkout@v4
17       - name: Build the Docker image
18         run: docker build --file Dockerfile --tag my-image-name:$(date +%s)
19

```

A red box highlights the entire 'on' section of the YAML file. The word 'rewrite' is written in red next to the 'on' keyword.

Copy the following to the [docker-image.yml](#):

```
name: Maven CI/CD
```

```
on:
  push:
    branches: [master]
    tags: [v*]
  workflow_dispatch:

jobs:
  build_and_publish:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
      - name: Set up JDK 11
        uses: actions/setup-java@v4
        with:
          java-version: "11"
          distribution: "temurin"
          cache: maven
      - name: Install test dependencies
        run: sudo apt-get update && sudo apt-get -y -q --no-install-recommends install ffmpeg mediainfo tesseract-ocr tesseract-ocr-deu
      - name: Build with Maven
        run: mvn --batch-mode -Pprod clean install
      - name: Upload war artifact
        uses: actions/upload-artifact@v4
        with:
          name: docs-web-ci.war
          path: docs-web/target/docs*.war

build_docker_image:
  name: Publish to Docker Hub
  runs-on: ubuntu-latest
  needs: [build_and_publish]

  steps:
    -
      name: Checkout
      uses: actions/checkout@v2
    -
      name: Download war artifact
      uses: actions/download-artifact@v4
      with:
        name: docs-web-ci.war
        path: docs-web/target
    -
      name: Setup up Docker Buildx
      uses: docker/setup-buildx-action@v1
    -
      name: Login to DockerHub
      if: github.event_name != 'pull_request'
      uses: docker/login-action@v1
      with:
        username: ${{ secrets.DOCKERHUB_USERNAME }}
        password: ${{ secrets.DOCKERHUB_TOKEN }}
```

```

    name: Populate Docker metadata
    id: metadata
    uses: docker/metadata-action@v3
    with:
      images: xx/Teedy # Docker Hub repository
      flavor: |
        latest=false
      tags: |
        type=ref,event=tag
        type=raw,value=latest,enable=${{ github.ref_type != 'tag' }}
      labels: |
        org.opencontainers.image.title = Teedy
        org.opencontainers.image.description = Teedy is an open source,
lightweight document management system for individuals and businesses.
        org.opencontainers.image.created = ${{ github.event_created_at }}
        org.opencontainers.image.author = Sismics
        org.opencontainers.image.url = https://teedy.io/
        org.opencontainers.image.vendor = Sismics
        org.opencontainers.image.license = GPLv2
        org.opencontainers.image.version = ${{ github.event_head_commit.id }}

    name: Build and push
    id: docker_build
    uses: docker/build-push-action@v4
    with:
      context: .
      push: ${{ github.event_name != 'pull_request' }}
      tags: ${{ steps.metadata.outputs.tags }}
      labels: ${{ steps.metadata.outputs.labels }}

```

Line 61, Change images to your own dockerhub image.

The screenshot shows the Docker Hub 'My Hub' interface. On the left, there's a sidebar with navigation links: 'Repositories' (which is selected and highlighted in grey), 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main content area displays the 'zhaoy6/teedy-app' repository under the 'General' tab. The repository details include: 'Last pushed 2 minutes ago • Repository size: 1 GB'. There are buttons to 'Add a description' and 'Add a category'. Below this, the 'Tags' section shows three tags: 'latest' (OS: Image, Type: Image, Pulled: less than 1 day, Pushed: 3 minutes). A note at the top right says 'Using C' and 'Docker ci' with a link 'To push a n repository: docker p ame'. A 'build' button is also visible.

You can compare the above with Teedy's original [build-deploy.yml](#).

## Step 3: Commit and Push

Committing changes will trigger the GitHub Actions workflow, which will:

- Build the Docker image from the Dockerfile
- Push the image to Docker Hub

The screenshots show the GitHub Actions interface for the 'Update docker-image.yml' workflow across three different commits:

- Screenshot 1 (Top):** Shows the 'Actions' tab with the 'All workflows' section. It lists four workflow runs for 'Update docker-image.yml'. The first run (Commit 4bd6afb) is 'In progress'. The second (Commit 8cf075c), third (Commit 7229ac0), and fourth (Commit 4bd6afb) are completed successfully.
- Screenshot 2 (Middle):** Shows the details for the most recent workflow run (Commit 4bd6afb). The 'Summary' tab shows the status as 'In progress'. The 'docker-image.yml' job is shown with a duration of 1m 7s, followed by a step to 'Publish to Docker Hub'.
- Screenshot 3 (Bottom):** Shows the details for the previous commit (Commit 8cf075c). The 'Summary' tab shows the status as 'In progress'. The 'docker-image.yml' job is shown with a duration of 2m 40s, followed by a step to 'Publish to Docker Hub'.

The screenshot shows a GitHub Actions pipeline named 'Update docker-image.yml #5'. It triggered via a push 4 minutes ago from the 'master' branch. The status is 'Success' with a total duration of 4m 41s. There is one artifact. The workflow file is 'docker-image.yml' with an 'on: push' trigger. The pipeline consists of two steps: 'build\_and\_publish' (2m 40s) and 'Publish to Docker Hub' (1m 53s), both of which are green (successful).

If account fail, you can click the corresponds button to see detail error:

The screenshot shows a GitHub Actions pipeline named 'Update docker-image.yml #4'. It failed 9 minutes ago. The 'Annotations' section shows 1 error. The specific error is in the 'Publish to Docker Hub' step, which failed with the message 'Env: Missing download info for actions/download-artifact@v2'. The pipeline includes 'build\_and\_publish' and 'Publish to Docker Hub' steps.

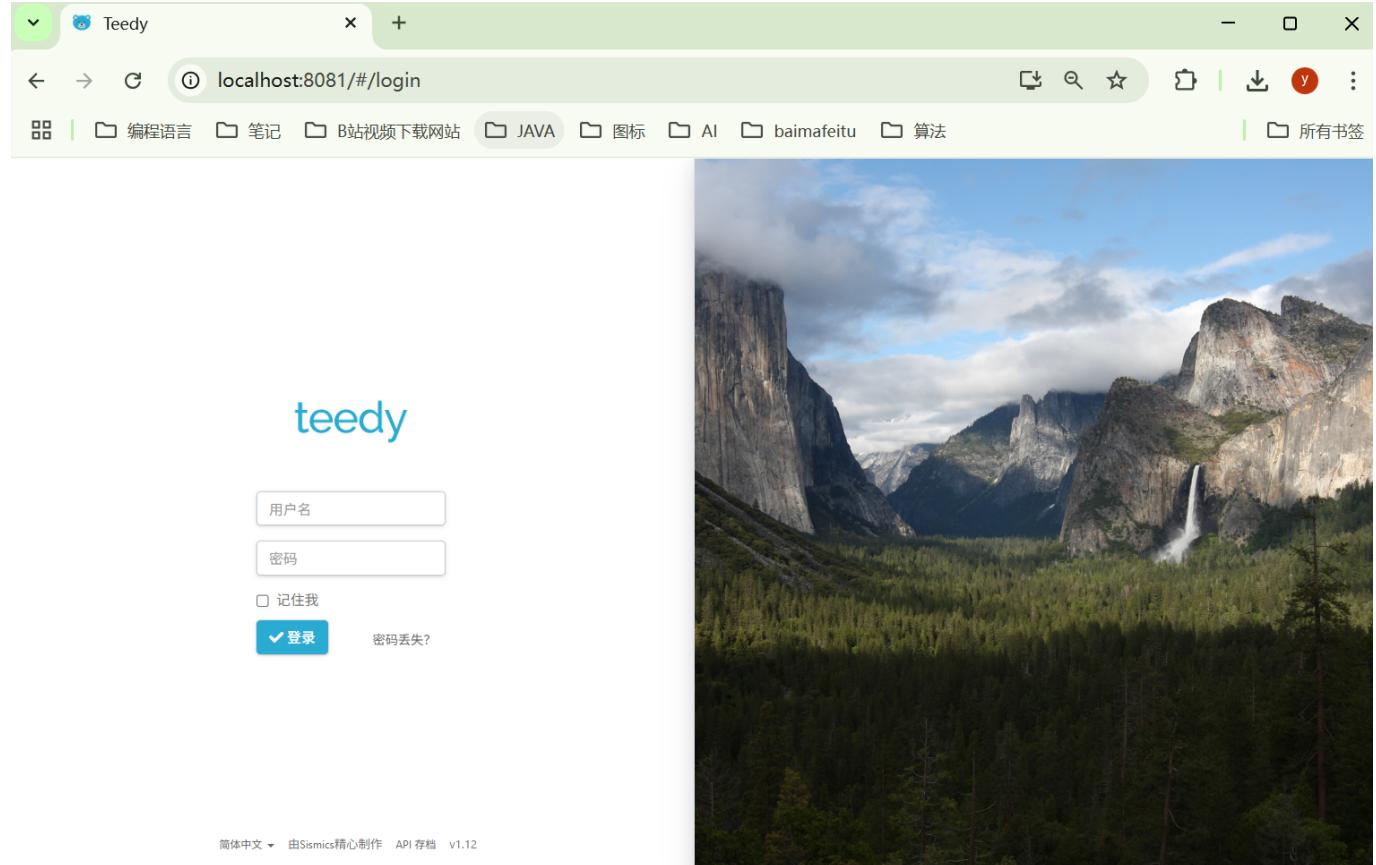
## Step 4: Pull and Run Locally

```
docker pull xx/Teedy:tag
docker run -d -p 8081:8080 --name myteedy01 xx/Teedy
```

```
ao_hao@DESKTOP-4JVIQPU:/mnt/d/github/Teedy-doc$ docker pull zhaoy6/teedy-app
Using default tag: latest
latest: Pulling from zhaoy6/teedy-app
30a9c22ae099: Already exists
e73db030b645: Pull complete
3e45d5ec32b4: Pull complete
cfbcf5f16946: Pull complete
04f0ed814a70: Pull complete
f083b07ba6cb: Pull complete
4f2fb00f71de: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:a8ab4ba287583196ecdcaaf16ee1ad55d1b7f825135adeddd064fea2119faedd
Status: Downloaded newer image for zhaoy6/teedy-app:latest
docker.io/zhaoy6/teedy-app:latest
```

```
ao_hao@DESKTOP-4JVIQPU:/mnt/d/github/Teedy-doc$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
zhaoy6/teedy-app    latest   3fd40a8d21ab  8 minutes ago  1.04GB
zhaoy6/teedy-app    7        72bf368374f9  22 hours ago  1.04GB
registry.hub.docker.com/zhaoy6/teedy-app 7        72bf368374f9  22 hours ago  1.04GB
registry.hub.docker.com/zhaoy6/teedy-app  latest   72bf368374f9  22 hours ago  1.04GB
zhaoy6/teedy-app    6        cd9f50823ed6  22 hours ago  1.04GB
registry.hub.docker.com/zhaoy6/teedy-app  6        cd9f50823ed6  22 hours ago  1.04GB
teedy2025_manual    latest   bc2ac81973ae  41 hours ago  1.04GB
docker              dind     721f8c2d3591  13 days ago   395MB
hello-world         latest   74cc54e27dc4  3 months ago  10.1kB
```

```
ao_hao@DESKTOP-4JVIQPU:/mnt/d/github/Teedy-doc$ docker run -d -p 8081:8080 --name mytedy01 zhaoy6/teedy-app:latest
47d2ddf5a1f03ca9179b66e06ae7749ca90b8d329f03ace142b8bd5cbc61210c
```



You can also use official image:

```
docker pull sismics/docs:latest
```

```
root@LAPTOP-H9VR9D4P:~# docker pull sismics/docs:latest
latest: Pulling from sismics/docs
bccd10f490ab: Pull complete
5d890301af4d: Pull complete
eefa7a0fbceb: Pull complete
ee3489578d09: Pull complete
069ef754ac50: Pull complete
3018db1b72ec: Pull complete
2ba6d61a4372: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:deb9eb2a138efc52131bcda0a3cc28e455888c2c55171cfea47c26a9a7a6e3a
Status: Downloaded newer image for sismics/docs:latest
docker.io/sismics/docs:latest
root@LAPTOP-H9VR9D4P:~# docker images
REPOSITORY          TAG        IMAGE ID      CREATED       SIZE
sismics/docs        latest     d3b3dfb621fb    3 weeks ago   1.04GB
hello-world         latest     d2c94e258dcb   12 months ago  13.3kB
root@LAPTOP-H9VR9D4P:~# docker run -d -p 8080:8080 --name mytedy sismics/docs
4a65908b544b1205311bdd03a11000c0d105ebc7cf2d2ce122bc02d1c59507e
root@LAPTOP-H9VR9D4P:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED          STATUS           PORTS          NAMES
4a65908b544b        sismics/docs      "java -jar /opt/jett..."   6 seconds ago   Up 4 seconds   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   mytedy
root@LAPTOP-H9VR9D4P:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED          STATUS           PORTS          NAMES
lab22cdb75ad        traccytian/teedytest2024  "bin/jetty.sh run"   4 seconds ago   Up 3 seconds   0.0.0.0:8081->8080/tcp, :::8081->8080/tcp   mytedy01
4a65908b544b        sismics/docs      "java -jar /opt/jett..."   2 minutes ago   Up 2 minutes   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   mytedy
root@LAPTOP-H9VR9D4P:~#
```

