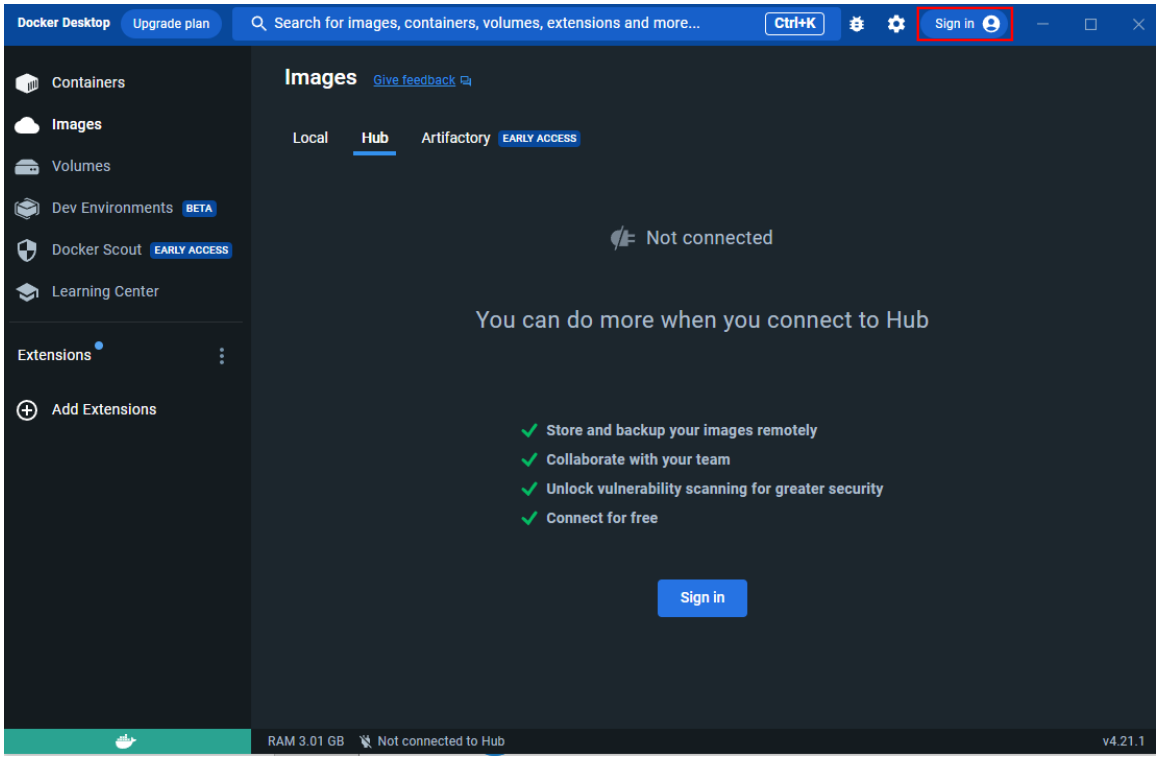
1. Download and install the Docker Desktop. https://docs.docker.com/get-docker/

2. Now there should be an app as shown in figure. We can register an account in DockerHub by clicking the Sign in figure here. Then we will have the access to the public Docker images.



3. Now we can start to use Docker for our war file deployment! Let us open the command line. Yes. Although we have the Docker desktop, we still need to use command line to perform the operations. It is a better way to learn. First we need the tomcat server to deploy the war file.

```
$ docker pull tomcat:9.0-jdk21-openjdk
```

```
F:\>docker pull tomcat:9.0-jdk21-openjdk
9.0-jdk21-openjdk: Pulling from library/tomcat
785ef8b9b236: Pull complete
5a6dad8f55ae: Pull complete
bd36c7bfe5f4: Pull complete
23081abc6d37: Pull complete
e8f504008fce: Pull complete
ec44ab22a9c1: Pull complete
06c487a0cd8e: Pull complete
bf5c3f678ded: Pull complete
Digest: sha256:303a39dca794a5033ea29b54e4e570899c075332476f44e0db7d9923ae0b3525
Status: Downloaded newer image for tomcat:9.0-jdk21-openjdk
docker.io/library/tomcat:9.0-jdk21-openjdk
```

This command will pull the tomcat docker file to our local machine.

(Please note that: tomcat:9.0-jdk21-openjdk can be any version you want. I just take a late version. Check https://hub.docker.com/_/tomcat/tags for more information)

We can play with the tomcat server by starting it with the following command line:

```
$ docker container run -d -p 8080:8080 tomcat:9.0-jdk21-openjdk
```

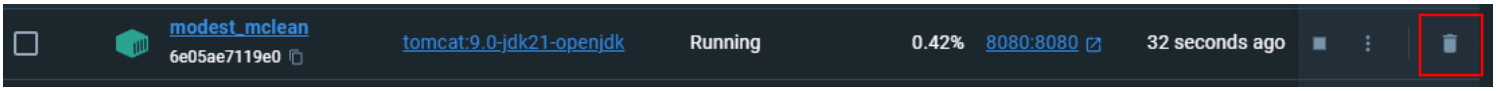
After start it we can check the status of container by:

```
$ docker ps
```

```
F:\>docker container run -d -p 8080:8080 tomcat:9.0-jdk21-openjdk
6e05ae7119e0ff889f2808d1717c70108f56d2e4e172074b22d3f03ad0bd568f

F:\>docker ps
CONTAINER ID   IMAGE                      COMMAND             CREATED         STATUS         PORTS                    NAMES
6e05ae7119e0   tomcat:9.0-jdk21-openjdk   "catalina.sh run"   7 seconds ago   Up 5 seconds   0.0.0.0:8080->8080/tcp   modest_mclean
```

Now we know that the tomcat server can start the server in docker. We will delete it for now, because we need the port 8080 later. You can delete it from either **docker desktop container** page



or from **command line**:

```
$ docker stop "container-name" && docker rm "container-name"
```

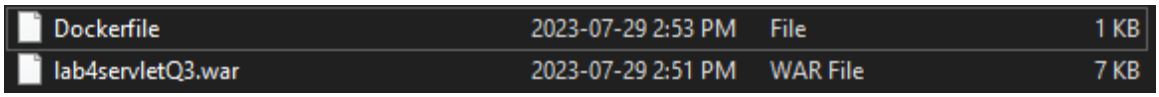
The “container-name” in this case is **modest_mclean.** (We see this names column with *docker ps*.)

4. Now let's start to build our own docker image. First we need to make a **Dockerfile** as script for docker to build new image. Here we use the Osap lab as our war file example.

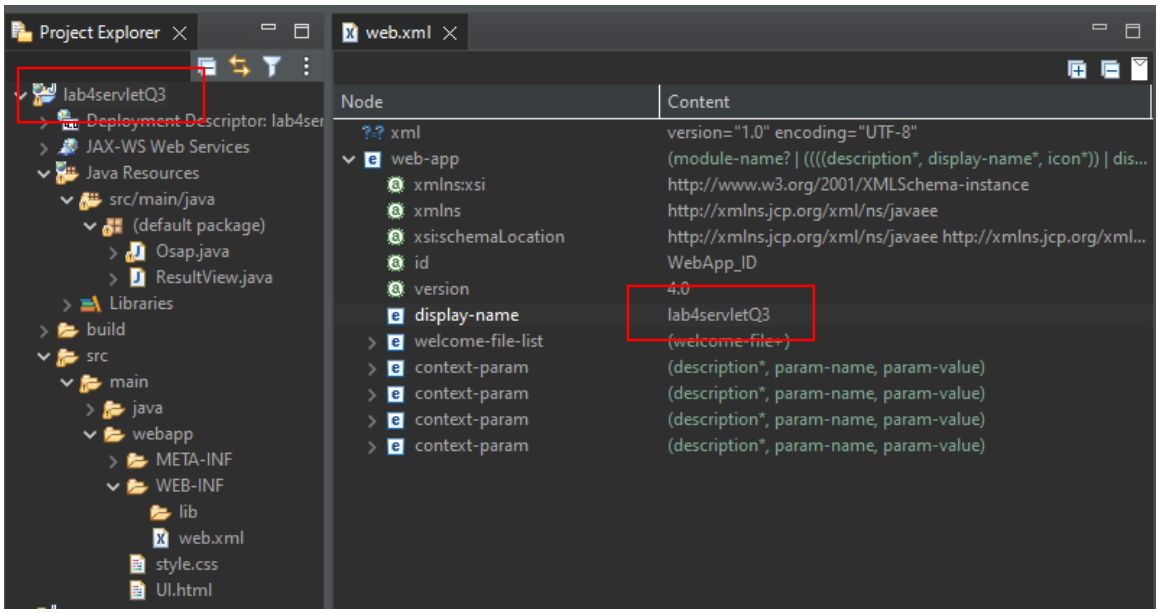
```
FROM tomcat:9.0-jdk21-openjdk
COPY ./lab4servletQ3.war /usr/local/tomcat/webapps
```

There are a few points to mentioned here:

1. The ***Dockerfile*** & ***lab4servletQ3.war*** should be in the same directory.



2. In our java web project, make sure your project name is same as the property "display-name" in web.xml file.



This Dockerfile means that we use tomcat:9.0 as the base and copy our war file into the tomcat server. So when this docker image is running as a container, it will unzip the war file and deploy it as a web service.

5. Now input the following command:

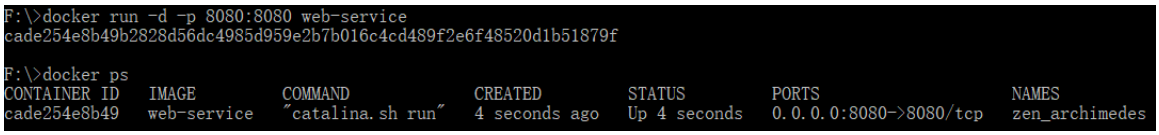
```
$ docker build -t web-service .
```

Now we can check the image we just built with:

```
$ docker image ls
```

Then run it as a container like:

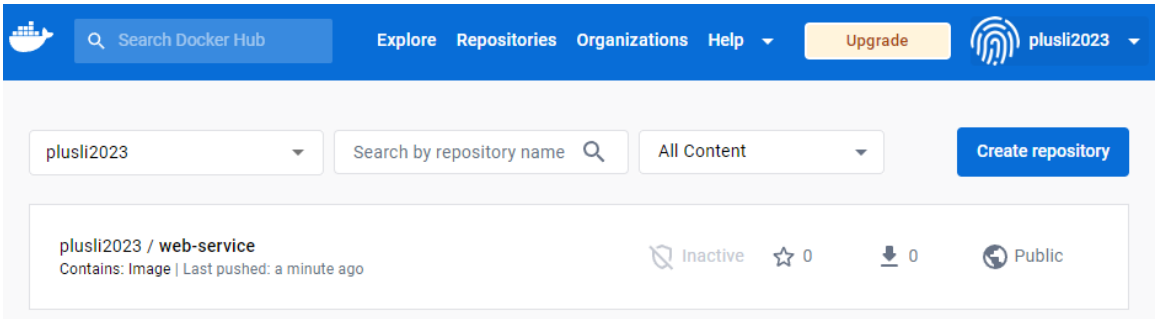
```
$ docker run -d -p 8080:8080 web-service
```



We can see it is already running in the backend. We can access our web page by using curl command or just visit http://localhost:8080 in your browser.

```
$ curl http://localhost:8080/lab4servletQ3/UI.html
```

6. Now login to your docker hub. **Create repository**. Name it with a repository Name (**web-service**). And make it **PUBLIC**. Now we have a repository in DockerHub.



Now go back to our Docker Desktop **image** page. We need the Id of our new created image to tag it. We only need a few character from it, there is no need to copy the full id.


```
$ docker tag 28b965c7 user-name/repo-name:1
```

Note that: “user-name” is **your DockerHub username**. “repo-name“ is your **repository name.** The number “1” represent the version of your program.

Now we can push it to the hub.

```
$ docker push user-name/repo-name:1
```

If you wan to share it. Go to the DockerHub and find the repository Tags page. Share the command in the red box. The public will have the access to your repository.