Docker-Jenkins-Pipeline

Objective:

Demonstrate the continuous integration and delivery by building a Docker Jenkins Pipeline.

Used Technology:

Name
Jenkins
GitHub
DockerHub

---

1) Start the lab and log into Jenkins

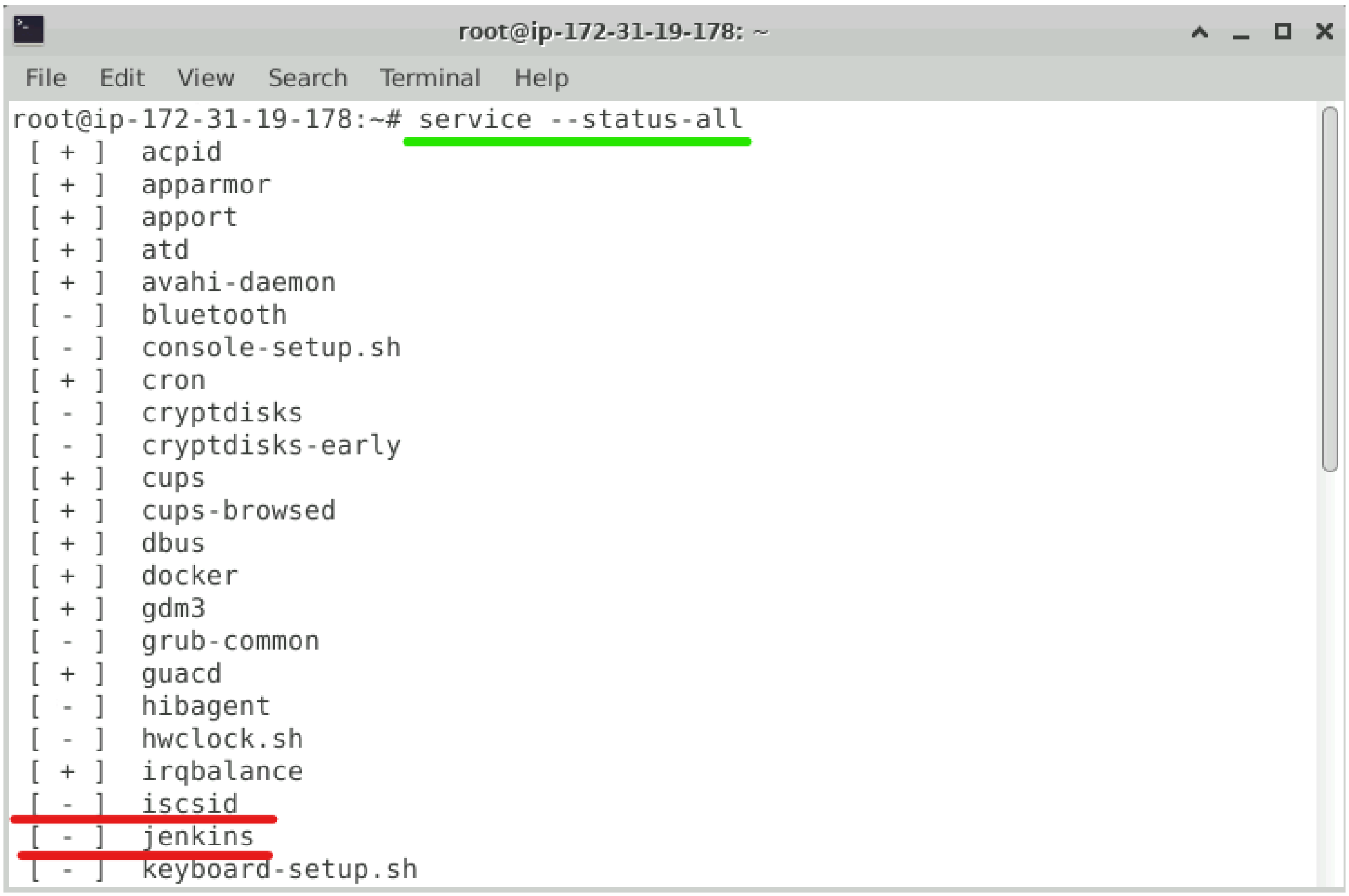
First, we check services. If **Jenkins** is down, we start it.

> Use **root** as needed and give permisions to **jenkins** to execute **docker** commands.

```
chmod 777 /var/run/docker.sock
```

Check Jenkins

```
service --status-all
```



```
root@ip-172-31-19-178:~# service --status-all
 [ + ]  acpid
 [ + ]  apparmor
 [ + ]  apport
 [ + ]  atd
 [ + ]  avahi-daemon
 [ - ]  bluetooth
 [ - ]  console-setup.sh
 [ + ]  cron
 [ - ]  cryptdisks
 [ - ]  cryptdisks-early
 [ + ]  cups
 [ + ]  cups-browsed
 [ + ]  dbus
 [ + ]  docker
 [ + ]  gdm3
 [ - ]  grub-common
 [ + ]  guacd
 [ - ]  hibagent
 [ - ]  hwclock.sh
 [ + ]  irqbalance
 [ - ]  iscsid
 [ - ]  jenkins
 [ - ]  keyboard-setup.sh
```

As we can see, **Jenkins** is down. We need to start the service and see where is running.

Start Jenkins

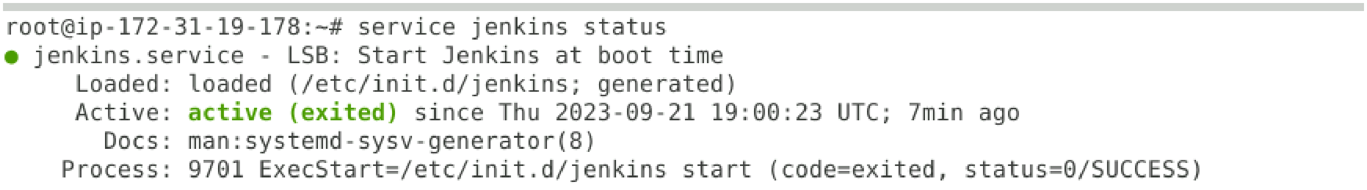
We can inicialize it with the *service* cmd utility.

```
service jenkins start
```

Check Jenkins Service Status

Next we can explore its basic running status.

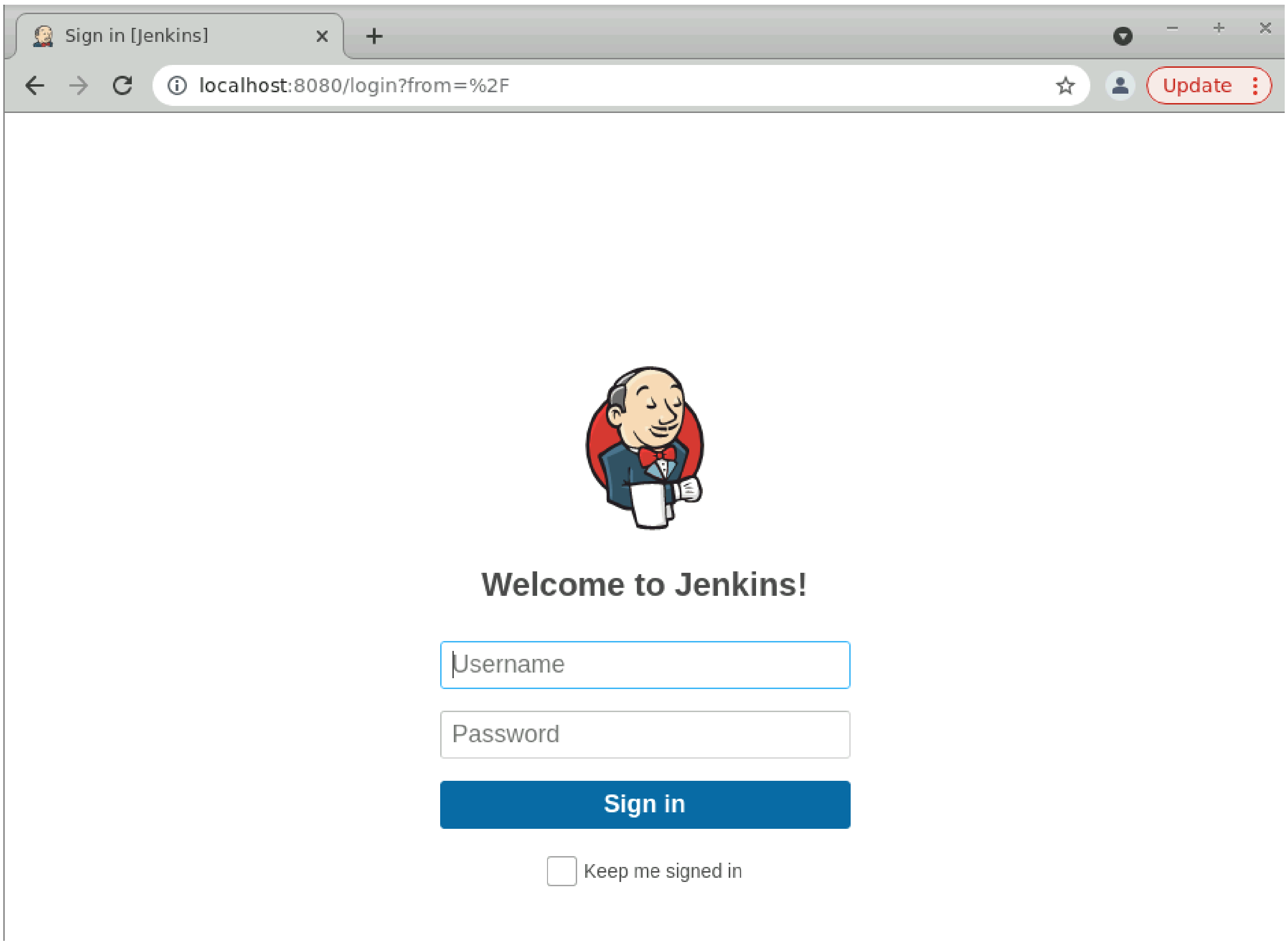
```
service jenkins status
```



```
root@ip-172-31-19-178:~# service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
     Loaded: loaded (/etc/init.d/jenkins; generated)
     Active: active (exited) since Thu 2023-09-21 19:00:23 UTC; 7min ago
       Docs: man:systemd-sysv-generator(8)
    Process: 9701 ExecStart=/etc/init.d/jenkins start (code=exited, status=0/SUCCESS)
```

> We can then access it locally on port **8080**

Open Jenkins in the browser



2) Create the pipeline

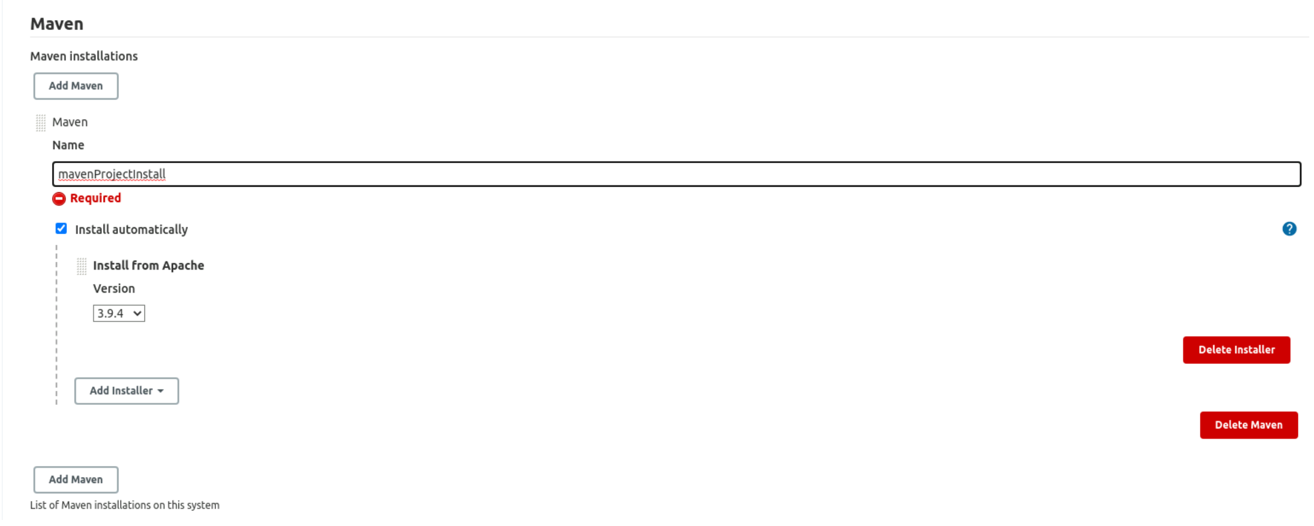
> Since we want to create a Pipeline that both has ***CI*** and ***CD*** integrations we need to install configuration tools needed.

- In this case we will ***Maven*** as the build tool with a repo own by ***Sonal0409***

We enter Jenkins Dashboard directly to Manage Jenkins and Global Tool Configuration

- ***Dashboard*** > ***Global Tool Configuration***

Scroll down to ***Maven*** and install it.



Maven

Maven installations

Add Maven

Maven

Name

mavenProjectInstall

Required

☑ Install automatically

Install from Apache

Version

3.9.4

Delete Installer

Add Installer

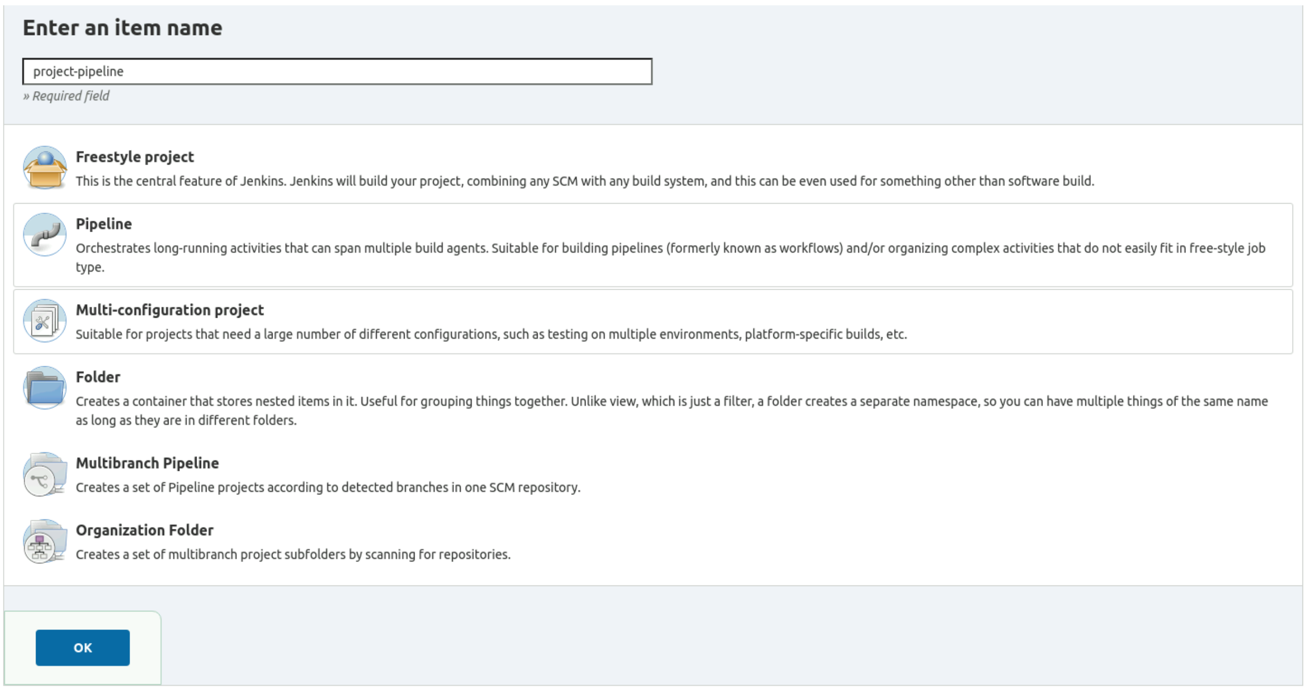
Delete Maven

Add Maven

List of Maven installations on this system

Create Job

Since we need a ***CI/CD*** project we will choose the pipeline option.



Enter an item name

project-pipeline

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

OK

Since we have some requisites, we will go with a jenkins script, located on ***/src/jenkinsfile*** on the project folder.

```
pipeline {

    tools{

    }
    agent any

    stages {
    }
}
```

We will also be using the simple ***dockerfile*** in the GitHub repository.

```
FROM tomcat:9
ADD addressbook.war /usr/local/tomcat/webapps
CMD ["catalina.sh", "run"]
EXPOSE 8080
```

Since we need to complete these next requirements:

- Create a Docker Jenkins Pipeline that will create a Docker image from the Dockerfile and host it on Docker Hub
- It should also pull the Docker image and run it as a Docker container
- Build the Docker Jenkins Pipeline to demonstrate the continuous integration and continuous delivery workflow

We will add some stages in our Jenkins Pipeline script

```
            stage('Clone Github repository')
            {
                steps{
                    echo 'Cloning repository...'
                    git 'https://github.com/Sonal0409/DevOpsClassCodes.git'
                }
            }
            stage('Build Project'){
                steps{
                    echo 'Building the project'
                    sh 'mvn package'
                }
            }
            stage('Build image'){
                steps{
                    echo 'Copying artifact to workspace'
                    sh 'cp /var/lib/jenkins/workspace/project-
pipeline/target/addressbook.war .'
                    echo 'Building image from docker file'
                    sh 'docker build -t addressbook:$BUILD_NUMBER .'
                }
            }

            stage('Push image to DockerHub'){
            steps{

                sh 'docker login -u "fpedrazav02" -p "" docker.io'
                sh 'docker tag addressbook:$BUILD_NUMBER
fpedrazav02/myaddressbook'
                sh 'docker push fpedrazav02/myaddressbook'
                sh 'docker rmi fpedrazav02/myaddressbook'
            }
        }

            stage('Deploy container'){
                steps{
                    echo 'Deploying container'
                    sh 'docker run -d -P fpedrazav02/myaddressbook'
                }
            }
    }
```

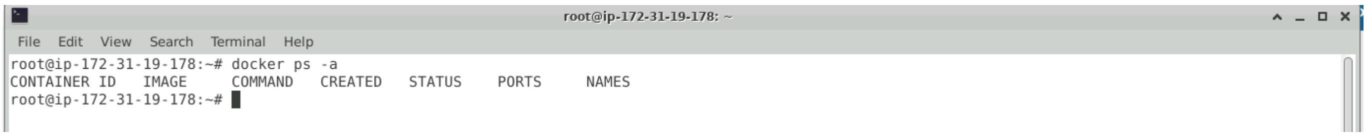
We aim for an image with a build Jenkins build number. This image is then tagged as our final image name, which we will then deploy to Docker Hub.

Finally, we can check if this works deploying the container.

3) Run the Pipeline

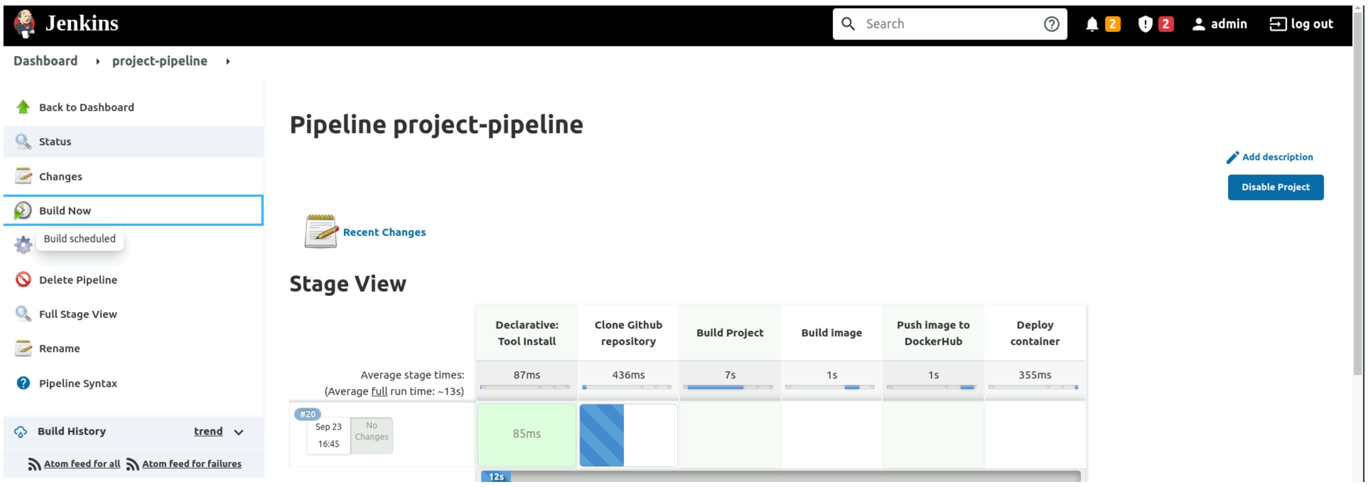
We first check we have nothing running.

```
docker ps -a
```



```
root@ip-172-31-19-178:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-19-178:~# 
```

We then build the JOB

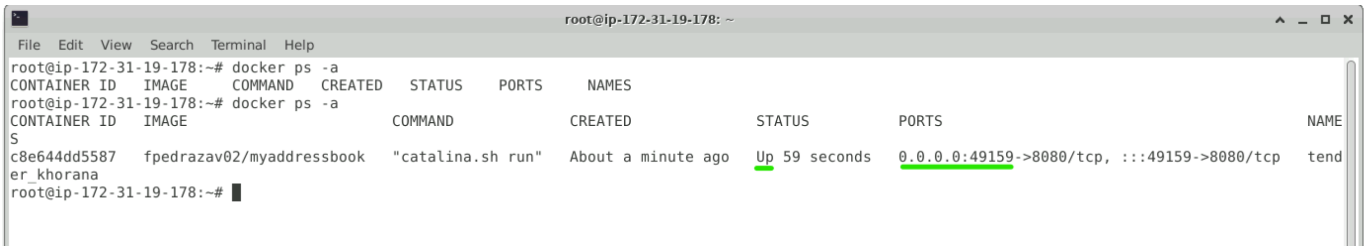


Pipeline project-pipeline

Stage View

	Declarative: Tool Install	Clone Github repository	Build Project	Build image	Push image to DockerHub	Deploy container
Average stage times: (Average full run time: ~13s)	87ms	436ms	7s	1s	1s	355ms
#20 Sep 23 16:45 No Changes	85ms					

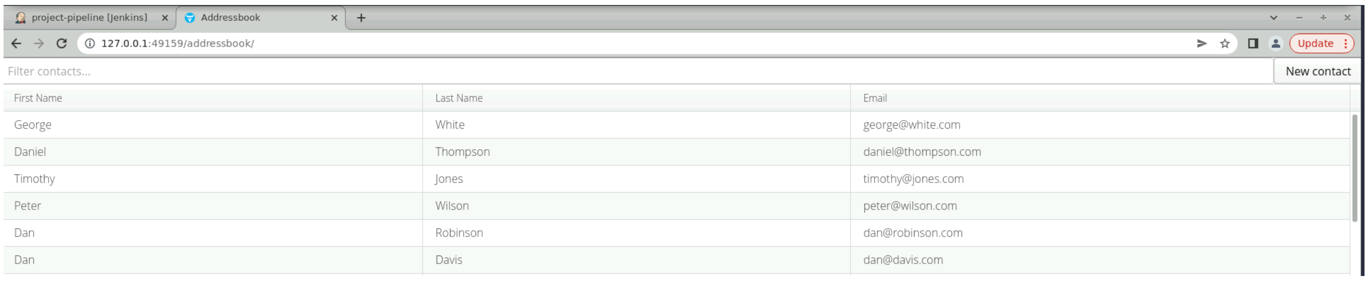
If everythin has compiled correctly, we can see our image running with an open port.



```
root@ip-172-31-19-178:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-19-178:~# docker ps -a
CONTAINER ID   IMAGE                      COMMAND             CREATED              STATUS          PORTS                                         NAME
S
c8e644dd5587   fpedrazav02/myaddressbook   "catalina.sh run"   About a minute ago   Up 59 seconds   0.0.0.0:49159->8080/tcp, :::49159->8080/tcp   tend
er_khorana
root@ip-172-31-19-178:~# 
```

We can also access it via our localhost on the port mentioned.

> **http://127.0.0.1:49159/addressbook**



First Name	Last Name	Email
George	White	george@white.com
Daniel	Thompson	daniel@thompson.com
Timothy	Jones	timothy@jones.com
Peter	Wilson	peter@wilson.com
Dan	Robinson	dan@robinson.com
Dan	Davis	dan@davis.com

The app is working as expected.