

# Lab 06 - Jenkins and Nginx

In this lab, you will learn how modern applications are deployed using automation, containers, and load balancing.

You will use **Jenkins** to automate the deployment of multiple backend services running in Docker containers, and NGINX to distribute incoming traffic across these services using load balancing.

This lab focuses on conceptual understanding, not complex application development.

## What is Load Balancing?

Load balancing is a technique used to distribute incoming network traffic across multiple backend servers. This improves:

- Performance
- Reliability
- Fault tolerance

In this lab, **NGINX** acts as a load balancer and forwards requests to multiple backend containers.

**NOTE: A couple of common errors that may be faced are mentioned at the end of the document. Refer to it.**

## Overview of the Lab:

You will perform the following tasks:

1. Set up Jenkins using Docker
2. Create a backend application that identifies itself
3. Dockerize the backend application
4. Configure NGINX for load balancing
5. Use Jenkins to deploy multiple backend containers
6. Verify load balancing through a web browser

**Before we begin, create a PUBLIC github repo named CC\_Lab-6**

# Task 1: Set Up Jenkins Using Docker

## Aim

To run Jenkins as a Docker container.

## Deliverables

This is what you should have by the end of this task.

Screenshot of Jenkins dashboard running in the browser (SS1)

```
[LF]>
[LF]> *****
[LF]> *****
[LF]> *****
[LF]> *****
[LF]> Jenkins initial setup is required. An admin user has been created and a password generated.
[LF]> Please use the following password to proceed to installation:
[LF]>
[LF]> a85c3fc683ee46278c1b17b491d5a0a4
[LF]> This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
[LF]>
[LF]> *****
[LF]> *****
[LF]> *****
```

- Use the code: `docker logs jenkins` once docker is installed to get this SS.

## Steps

DO THE NEXT STEPS ONLY IF YOU HAVENT DONE IT AS PART OF THE PRE-REQ DOCUMENT:

Restart Jenkins in the lab environment. If Jenkins was already set up earlier (for example, at home or as part of the pre-requisite document), students do not need to pull the image or create a new container again.

Jenkins configuration and data are preserved using a Docker volume

Look for existing Jenkins container: **docker ps -a**

Start the Jenkins container: **docker start jenkins** and go to **http://localhost:8080**

1. Pull the Jenkins image:

```
docker pull jenkins/jenkins:lts
```

2. Build a custom jenkins image, use the file Dockerfile.jenkins provided to you

From inside the folder run:

**docker build -t jenkins-docker -f Dockerfile.jenkins .**

(This will take a while)

```

+] Building 98.8s (6/6) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile.jenkins    0.0s
=> => transferring dockerfile: 178B                             0.0s
=> [internal] load metadata for docker.io/jenkins/jenkins:lts 0.1s
=> [internal] load .dockerignore                               0.0s
=> => transferring context: 2B                                    0.0s
=> [1/2] FROM docker.io/jenkins/jenkins:lts@sha256:d1ea795c6facd7f549a21c40e5e43ffcc5fbc5f48683d9b24750f26e8079d 1.8s
=> => resolve docker.io/jenkins/jenkins:lts@sha256:d1ea795c6facd7f549a21c40e5e43ffcc5fbc5f48683d9b24750f26e8079d 1.6s
=> [2/2] RUN apt-get update && apt-get install -y docker.io make g++ curl 44.6s
=> => exporting to image                                         52.0s
=> => exporting layers                                           36.4s
=> => exporting manifest sha256:b27db89f15f01b8b79f4a3c3db6dab9c31e70ceae34183c721c45a717559701d 0.0s
=> => exporting config sha256:d0b10a5e6aa8aa3b955d87af7c6e214a3cfc97f6f8efe088ee6a9e2ddbe8633b 0.0s
=> => exporting attestation manifest sha256:89ebb2aa7383ddcc6ef7d32f306329eeb6dafb01bc1630d4a3265836d72cc73e 0.0s
=> => exporting manifest list sha256:b6039ad4387c8ce63b3a1e474527dd5bae9251423b28f059b7751361b415fde3 0.0s
=> => naming to docker.io/library/jenkins-docker:latest        0.0s
=> => unpacking to docker.io/library/jenkins-docker:latest     15.4s

view build details: docker-desktop:///dashboard/build/desktop-linux/desktop-linux/oqverh049be01izamr4lo1zz9

```

### 3. Run Jenkins container:

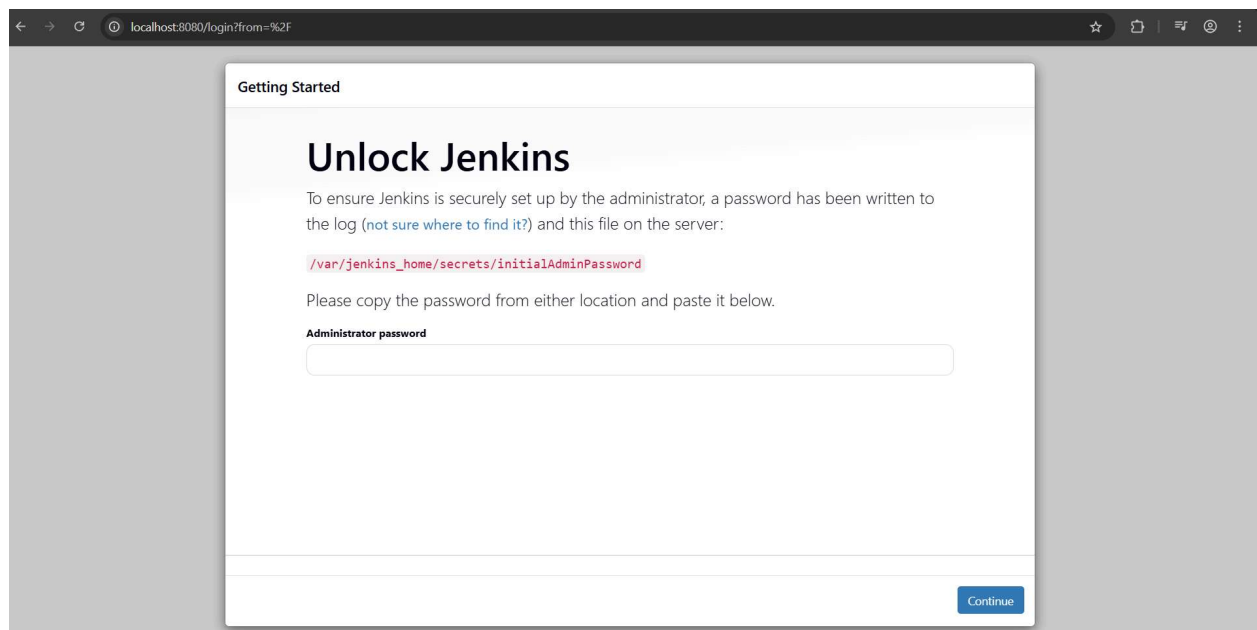
**`docker run -d -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock --name jenkins jenkins-docker`**

This will start Jenkins using the custom image with Docker support and persistent storage. (This might also be the reason your build fails so we would have to rerun as root, please refer to the errors part in the last part of the document if it does fail during Task-2)

The Jenkins container is run with a Docker volume to persist all Jenkins data across restarts.

### 3. Next, open your browser and navigate to:

**`http://localhost:8080`**



### 4. Retrieve the initial admin password:

```
docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

**`docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword`**