

# How to Deploy JupyterLab – Interactive Notebook IDE

Run JupyterLab interactively on Ubuntu with Docker Compose, persistent storage, and HTTPS.

---

# Contents

01	Introduction	3
02	Prerequisites	3
03	Set Up the Directory Structure and Environment Variables	3
04	Deploy with Docker Compose	5
05	Access JupyterLab	8
06	Conclusion	9

# Introduction

---

JupyterLab is an open-source interactive development environment for working with notebooks, code, and data. It provides a flexible interface for data science, machine learning, and scientific computing with support for multiple programming languages including Python, R, and Julia. JupyterLab combines code execution, rich text documentation, and visualization capabilities in a browser-based workspace.

In this article, you will deploy JupyterLab using Docker Compose, configure persistent storage for notebooks and data, and set up Traefik as a reverse proxy to securely access your JupyterLab instance.

## Prerequisites

---

Before you begin, you need to:

- Have access to an [Ubuntu 24.04-based server](#) as a [non-root user with `sudo` privileges](#).
- Install [Docker and Docker Compose](#).
- Configure a [domain A record](#) pointing to your server's IP address (for example, `jupyter.example.com`).

## Set Up the Directory Structure and Environment Variables

---

In this section, you prepare the required directory structure for JupyterLab and define environment variables in a `.env` file.

1. Create the directory structure for JupyterLab.

CONSOLE

```
$ mkdir -p ~/jupyter-lab/{notebooks,data}
```

These directories serve different purposes:

- **notebooks**: Stores Jupyter notebook files and Python scripts.
- **data**: Contains datasets and project files for analysis.

2. Navigate into the `jupyter-lab` directory.

CONSOLE

```
$ cd ~/jupyter-lab
```

3. Set proper ownership for the JupyterLab directories. JupyterLab runs as the `joyvan` [user \(UID 1000\)](#) inside the container.

CONSOLE

```
$ sudo chown -R 1000:1000 notebooks data
```

4. Create a `.env` file.

CONSOLE

```
$ nano .env
```

Add the following variables:

INI

```
DOMAIN=jupyter.example.com  
LETCRYPT_EMAIL=admin@example.com
```

Replace:

- `jupyter.example.com` with your domain.
- `admin@example.com` with your email.

Save and close the file.

# Deploy with Docker Compose

In this section, you create and deploy the Docker Compose stack that runs JupyterLab behind Traefik. Docker Compose manages both containers, applies the environment variables from your `.env` file, and automatically configures HTTPS routing through Traefik.

1. Create a new Docker Compose manifest.

CONSOLE

```
$ nano docker-compose.yaml
```

2. Add the following content.

YAML

```
services:
  traefik:
    image: traefik:v3.6
    container_name: traefik
    command:
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.websecure.address=:443"
      - "--
entrypoints.web.http.redirections.entrypoint.to=websecure"
      - "--
entrypoints.web.http.redirections.entrypoint.scheme=https"
      - "--
certificatesresolvers.letsencrypt.acme.httpchallenge=true"
      - "--
certificatesresolvers.letsencrypt.acme.httpchallenge.entrypoint=web"
      - "--certificatesresolvers.letsencrypt.acme.email=${LETSENCRYPT_EMAIL}"
      - "--certificatesresolvers.letsencrypt.acme.storage=/letsencrypt/acme.json"
    ports:
      - "80:80"
```

```
- "443:443"
volumes:
- "letsencrypt:/letsencrypt"
- "/var/run/docker.sock:/var/run/docker.sock:ro"
restart: unless-stopped

jupyter:
image: jupyter/scipy-notebook:latest
container_name: jupyter
hostname: jupyter
expose:
- "8888"
volumes:
- "./notebooks:/home/jovyan/work"
- "./data:/home/jovyan/data"
environment:
- JUPYTER_ENABLE_LAB=yes
command: start-notebook.sh --NotebookApp.token='' --
NotebookApp.password=''
labels:
- "traefik.enable=true"
- "traefik.http.routers.jupyter.rule=Host(`${DOMAIN}`)"
- "traefik.http.routers.jupyter.entrypoints=websecure"
-
"traefik.http.routers.jupyter.tls.certresolver=letsencrypt"
-
"traefik.http.services.jupyter.loadbalancer.server.port=8888"
restart: unless-stopped

volumes:
letsencrypt:
```

Save and close the file.

This configuration establishes:

- **services:** Two containers form the computing environment:
  - **traefik:** Handles incoming requests, obtains SSL certificates, and routes traffic to JupyterLab.
  - **jupyter:** Runs the interactive notebook server with scientific Python libraries pre-installed.
- **image:** The `scipy-notebook` image includes NumPy, Pandas, Matplotlib, and other data science packages.

- **container\_name**: Fixed naming simplifies container operations and troubleshooting.
- **command (Traefik)**: Activates Docker provider integration, opens HTTP (80) and HTTPS (443) ports, forces HTTPS redirects, and configures Let's Encrypt ACME client.
- **ports (Traefik)**: Exposes web service ports to the host for public connectivity.
- **expose (JupyterLab)**: Makes port 8888 available within the Docker network for Traefik routing.
- **volumes**:
  - Bind mounts ( `./notebooks` , `./data` ) preserve your work and datasets across container restarts.
  - Named volume `letsencrypt` stores SSL certificates independently.
  - Docker socket access permits Traefik to discover running services.
- **environment (JupyterLab)**: Enables the JupyterLab interface instead of classic Jupyter Notebook.
- **command (JupyterLab)**: Disables token authentication since HTTPS provides transport security.
- **labels (JupyterLab)**: Traefik annotations enabling proxy functionality, hostname routing, SSL certificates, and backend port specification.
- **restart: unless-stopped**: Ensures automatic service recovery after system events.

### 3. Create and start the services.

CONSOLE

```
$ docker compose up -d
```

### 4. Verify that the services are running.

CONSOLE

```
$ docker compose ps
```

Output:

NAME	IMAGE	COMMAND	SERVICE
CREATED	STATUS	PORTS	
jupyter	jupyter/scipy-notebook:latest	"tini -g -- start-no..."	jupyter 32
seconds ago	Up 31 seconds	8888/tcp	
traefik	traefik:v3.6	"/entrypoint.sh --pr..."	traefik 32
seconds ago	Up 31 seconds	0.0.0.0:80->80/tcp, [::]:80->80/tcp, 0.0.0.0:443->443/tcp, [::]:443->443/tcp	

Both containers are operational. JupyterLab serves notebooks while Traefik manages connections on ports `80` and `443`.

#### 5. View the logs of the services.

##### CONSOLE

```
$ docker compose logs
```

For more information on managing a Docker Compose stack, see the [How To Use Docker Compose](#) article.

## Access JupyterLab

This section shows how to access the JupyterLab interface and verify your installation by creating a test notebook.

#### 1. Open the JupyterLab interface in your browser.

```
https://jupyter.example.com
```

#### 2. The JupyterLab launcher displays. The file browser on the left shows the `work` and `data` directories you mounted.

#### 3. Explore the main interface components:

- Click **Python 3** under **Notebook** to create a new Jupyter notebook.
- Click **Terminal** to open a command-line shell inside the container.
- Click **Text File** to create plain text documents or scripts.

4. Verify JupyterLab is functioning by running a test cell:

- In a new notebook, enter the following code in the first cell:

PYTHON

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
plt.title('Test Plot')
plt.show()
```

- Press `Shift + Enter` to execute. A sine wave plot should display below the cell.

5. Save your notebook using **File > Save Notebook**. The file persists in the `notebooks` directory on your host.

## Conclusion

You have successfully deployed JupyterLab for interactive computing with HTTPS encryption. The Docker Compose setup combines the notebook server with automated SSL management, while bind-mounted directories preserve your notebooks and datasets through container updates. Traefik handles certificate provisioning and secure traffic routing. Your JupyterLab instance is ready for data exploration, machine learning experiments, and collaborative scientific computing with full Python data science stack support.



VULTR

