> Note: For Multipliers (Trainers/Instructors) only

JupyterHub in System-Wide venv with Multi-User Operation, Admin Access, and System-Wide MQTT and GSON Libraries

This guide serves as a **blueprint** for deploying a central **JupyterLab Server for student groups**. It outlines the basic setup of **JupyterHub for multiple users** (PAM authentication) and designates the instructor (`pdal`) as the **Administrator**. This Admin role is crucial for support, as it allows access to the learners' sessions. Additionally, system-wide kernels (Java, R) and IoT libraries (Paho-MQTT, GSON) are made available to all users.

---

1. Install Prerequisites

Create a standard LXC container running **Ubuntu 24.04**. Switch to user `pdal`, then perform an update and upgrade.

```
sudo apt install -y python3 python3-venv python3-pip nodejs npm default-jdk git
curl unzip r-base
```

Explanation: Installs the necessary core packages for Python (JupyterHub), Node.js (Proxy), Java (Java kernel), R (R kernel), Git (for repos), curl/unzip (for downloading and unpacking). Individual packages can also be installed sequentially instead of in one command line.

2. Create Central Library Directory

```
sudo mkdir -p /opt/jupyterhub-libs/{python,java,r}
sudo chmod -R a+rx /opt/jupyterhub-libs
```

Explanation: Creates a central directory that can be used by all users—e.g., for shared libraries, example scripts, or wrappers.

3. Set up System-Wide venv for JupyterHub

```
sudo mkdir -p /opt/jupyterhub-venv
sudo chown root:root /opt/jupyterhub-venv
sudo python3 -m venv /opt/jupyterhub-venv
```

Command	Explanation

Command	Explanation
`sudo mkdir -p /opt/jupyterhub-venv`	Creates the target directory for the virtual Python environment under `/opt`. The `-p` option ensures parent directories are also created if they don't exist yet.
`sudo chown root:root /opt/jupyterhub-venv`	Sets the directory owner to `root`, so only administrative users can make changes to it.
`sudo python3 -m venv /opt/jupyterhub-venv`	Creates the virtual environment (venv) in the specified directory. This isolated Python environment allows JupyterHub and its dependencies to be installed without altering the global system Python.

Explanation: Creates an isolated Python environment where all JupyterHub components are managed independently of the system Python.

4. Install JupyterHub and Lab

```
source /opt/jupyterhub-venv/bin/activate
sudo /opt/jupyterhub-venv/bin/python -m pip install --upgrade pip
sudo /opt/jupyterhub-venv/bin/python -m pip install jupyterhub notebook jupyterlab
deactivate
sudo npm install -g configurable-http-proxy
```

Explanation: Installs all Jupyter components within the `venv` as well as the HTTP proxy (required for JupyterHub). JupyterLab is also installed.

5. Generate JupyterHub Configuration

```
sudo mkdir -p /etc/jupyterhub
cd /etc/jupyterhub
sudo /opt/jupyterhub-venv/bin/jupyterhub --generate-config
```

Explanation: Generates the standard configuration file `jupyterhub_config.py`, which will be customized later to define users, spawners, and admin access.

6. Customize Configuration (incl. Admin User pdal)

Edit `/etc/jupyterhub/jupyterhub_config.py`:

```
sudo nano /etc/jupyterhub/jupyterhub_config.py
```

```python
c = get_config()

# Authenticate via system users
c.JupyterHub.authenticator_class = 'jupyterhub.auth.PAMAuthenticator'
c.Authenticator.allow_existing_users = True
# c.Authenticator.allow_all = True

# Start directly in JupyterLab
c.Spawner.default_url = '/lab'
# Function to create the 'notebooks' directory
import pwd
import os


def create_notebook_dir(spawner):
    username = spawner.user.name
    notebook_dir = f"/home/{username}/notebooks"
    os.makedirs(notebook_dir, exist_ok=True)
    uid = pwd.getpwnam(username).pw_uid
    gid = pwd.getpwnam(username).pw_gid
    os.chown(notebook_dir, uid, gid)
    # Important: Dynamically set `Spawner.notebook_dir`
    spawner.notebook_dir = notebook_dir

c.Spawner.pre_spawn_hook = create_notebook_dir
# Admin rights for user 'pdal'
c.Authenticator.admin_users = {'pdal'}
c.JupyterHub.admin_access = True

# Additional environment variables for library path
import os
c.Spawner.environment = {
    'CLASSPATH': '/opt/jupyterhub-libs/java/*',
    'R_LIBS_USER': '/opt/jupyterhub-libs/r',
    'PYTHONPATH': '/opt/jupyterhub-libs/python'
}
```

Explanation:

- Uses system users (PAM) for **authentication** – no separate JupyterHub accounts are needed.
- Starts directly in **JupyterLab** (`/lab`) instead of the classic Jupyter Notebook interface.
- When a session starts, it automatically checks if a personal directory **`~/notebooks`** exists – if not, it is automatically created and correctly permissioned for the respective user.
- The **`pre_spawn_hook`** configuration dynamically sets the path to the notebooks per user, so everyone lands in the `/home/username/notebooks` directory by default.
- The user `pdal` receives **Admin rights** – they can, among other things, access other users' sessions (e.g., for support).

- Sets central **environment variables** for:

* Java libraries in the `/opt/jupyterhub-libs/java/*` directory (`CLASSPATH`)

* R libraries in the `/opt/jupyterhub-libs/r` directory (`R_LIBS_USER`), which can be used system-wide by all users.

7. Create System Users

```
sudo adduser alice
sudo adduser bob
# pdal already exists
```

Explanation: Creates users who can log in to JupyterHub via **PAM**. Without system users, no login is possible.

> Note: Before the created system users can log in to JupyterHub, the admin must first add these users in the **Admin Panel** in JupyterHub, as they otherwise do not have permission to access JupyterHub.

8. Set up systemd Service for JupyterHub

Create and edit the file `/etc/systemd/system/jupyterhub.service` with `sudo nano /etc/systemd/system/jupyterhub.service`:

```
[Unit]
Description=JupyterHub (venv)
After=network.target

[Service]
User=root
WorkingDirectory=/etc/jupyterhub
Environment="PATH=/opt/jupyterhub-venv/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
Environment="CLASSPATH=/opt/jupyterhub-libs/java/*"
Environment="R_LIBS_USER=/opt/jupyterhub-libs/r"
Environment="PYTHONPATH=/opt/jupyterhub-libs/python"
ExecStart=/opt/jupyterhub-venv/bin/jupyterhub
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Section	Directive	Description
[Unit]	Description	Short description of the service.
[Unit]	After	Specifies the dependency; here, the service should only start after the network.
[Service]	User	User under which the service runs (here `root`, but can be any other user).

Section	Directive	Description
[Service]	WorkingDirectory	Working directory from which the service starts (here, the configuration folder).
[Service]	Environment	Sets environment variables, e.g., PATH, CLASSPATH, R_LIBS_USER for the service process.
[Service]	ExecStart	Command executed to start the service (here, JupyterHub in the venv).
[Service]	Restart	Behavior in case of crash or error — here, restart on failure.
[Install]	WantedBy	Determines when the service is started (here, at multi-user runlevel, i.e., normal boot).

```
sudo systemctl daemon-reexec
sudo systemctl enable --now jupyterhub
```

Explanation: Creates a service that automatically starts **JupyterHub** on boot and reloads it upon failure. Important for continuous operation.

9. Access JupyterHub in the Browser

```
http://<server-ip>:8000
```

Log in with user `pdal`. Under the menu item "File" -> "Hub Control Panel" -> "Admin" -> "add Users", add the users **"bob"** and **"alice"**. Only then can these users log in. For additional users, first create a system user (`adduser xyz`) and then add them in the "Hub Control Panel".

As an Admin, you can log into running sessions of other users, stop servers, or start them. This allows you to **support the learners**.

10. Install IJava Kernel (Java)

```
cd /opt
sudo wget https://github.com/SpencerPark/IJava/releases/download/v1.3.0/ijava-1.3.0.zip
sudo unzip ijava-1.3.0.zip
```

To install the kernel system-wide, we still need the `jupyter-client` package, which we install with:

```
sudo apt install python3-jupyter-client
```

Now we can execute the following commands:

```
sudo python3 install.py --sys-prefix
```

Explanation: Installs a **Java kernel** for Jupyter system-wide. Notebooks with Java code are now possible, e.g., for software development or computer science classes.

11. Install IRKernel (R)

> Note: Only install the R kernel if you really need it; it's large and takes a long time.

First, switch to the R Console:

```
sudo R
```

Execute in the R Console:

```
install.packages("IRkernel")
Sys.setenv(PATH = paste(Sys.getenv("PATH"), "/opt/jupyterhub-venv/bin", sep =
":"))
IRkernel::installspec(user = FALSE)
q()
```

Then exit the R Console with `q()`.

Explanation: Adds an **R kernel** system-wide – usable for statistics, data science, or research. This temporarily ensures the Jupyter command is available in the R environment.

12. Install Paho-MQTT for Python

Switch to the `root` user. Start the virtual environment with:

```
source /opt/jupyterhub-venv/bin/activate
```

Then install the libraries – ensure the "venv" is running (`(jupyterhub-venv) root@jupyterhub:`).

```
pip install paho-mqtt
deactivate
```

Explanation: Installs the **MQTT library for Python** and places example code centrally. All users can thus process MQTT data, e.g., in IoT projects.

13. Add Paho-MQTT for Java + GSON

```
cd /opt/jupyterhub-libs/java
# MQTT
sudo curl -LO https://repo.eclipse.org/content/repositories/paho-releases/org/eclipse/paho/org.eclipse.paho.client.mqttv3/1.2.5/org.eclipse.paho.client.mqttv3-1.2.5.jar
# GSON
sudo curl -LO
https://repo1.maven.org/maven2/com/google/code/gson/gson/2.10.1/gson-2.10.1.jar
```

This makes **PahoMQTT** and **GSON** available system-wide. The libraries can be used in Jupyter Notebooks (Java) as follows:

```
// 1. Load MQTT library
%classpath add jar /opt/jupyterhub-libs/java/org.eclipse.paho.client.mqttv3-1.2.5.jar

// 2. Load GSON library (if also used)
%classpath add jar /opt/jupyterhub-libs/java/gson-2.10.1.jar
import org.eclipse.paho.client.mqttv3.*;
import com.google.gson.*;
```

Explanation: The Java MQTT client and the GSON library (for JSON parsing) are centrally stored as `.jar` files. This allows all Java notebooks to use both libraries directly.

14. Access Rights for Central Libraries

```
sudo chmod -R a+rx /opt/jupyterhub-libs
```

Explanation: All users receive read and execute rights for central resources – e.g., example notebooks or shared libraries.

🗀 Directory Overview

Path	Content
`/opt/jupyterhub-venv`	Python Environment with JupyterHub
`/etc/jupyterhub`	JupyterHub Configuration
`/opt/IJava`	Java Kernel (IJava)
`/opt/jupyterhub-libs/python/`	Shared Python Scripts and Example Code
`/opt/jupyterhub-libs/java/`	Paho MQTT + GSON Java Libraries (.jar)
`/opt/jupyterhub-libs/r/`	Shared R Scripts (MQTT/Wrapper/Examples)
`/usr/local/share/jupyter/`	System-Wide Registered Jupyter Kernels

🔒 Security Note

For production use: Securing via **HTTPS** or a **Reverse Proxy** (e.g., with NGINX or Apache2) is strongly recommended. This prevents passwords and notebook content from being transmitted in plain text. The configuration for SSL encryption can be read in the official JupyterHub documentation: https://jupyterhub.readthedocs.io/en/stable/tutorial/getting-started/security-basics.html

The integration of certificates is described in another document found here: 2050 CA-sslmitSANZertifikat.

Sources

- "Documentation · IRkernel". Accessed: July 28, 2025. [Online]. Available at: https://irkernel.github.io/docs/
- "Installation Basics", JupyterHub. Accessed: July 31, 2025. [Online]. Available at: https://jupyterhub.readthedocs.io/en/stable/tutorial/installation-basics.html
- "Security settings", JupyterHub. Accessed: July 31, 2025. [Online]. Available at: https://jupyterhub.readthedocs.io/en/stable/tutorial/getting-started/security-basics.html

---

License