

# JupyterLab im LXC-Container installieren & konfigurieren

(mit Python, R, Java, systemd-Dienst)

## Einleitung

JupyterLab ist eine moderne, interaktive Weboberfläche für Programmierung, Datenanalyse und wissenschaftliches Rechnen. Als Nachfolger der klassischen Jupyter Notebooks bietet es Unterstützung für viele Sprachen wie Python, R oder Java sowie zahlreiche Visualisierungs- und Erweiterungsmöglichkeiten.

Diese Anleitung zeigt die Installation und Konfiguration von JupyterLab in einem Ubuntu-basierten LXC-Container, mit:

- Python (in venv)
- R-Kernel (IRkernel)
- Java-Kernel (IJava)
- systemd-Dienstbetrieb
- Passwortschutz
- SSL-Konfiguration (verlinkt)

## Voraussetzungen

Komponente	Voraussetzung
System	LXC-Container mit Ubuntu 22.04 / 24.04
IP-Adresse des Containers	z.B. <b>192.168.137.180</b>
Benutzer mit sudo-Rechten	z.B. <b>pdal</b>
Internetverbindung	Ja

## 1. System aktualisieren

```
sudo apt update && sudo apt upgrade -y
```

 Dies stellt sicher, dass alle Paketquellen aktuell sind und die neuesten Sicherheitsupdates installiert werden. Dies ist eine wichtige Grundregel bei jeder Linux-Installation. ☈ **Ziel:** Das System ist auf dem neuesten Stand – besonders wichtig, bevor neue Software installiert wird.

## 2. Python + Pip + venv installieren

```
sudo apt install python3 python3-pip python3-venv -y
```

```
pdal@jupyterlab180:~$ sudo apt install python3 python3-pip python3-venv -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2).
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 cpp cp
  fakeroot fontconfig-config fonts-dejavu-core fonts-dejavu-mono g++ g++-13 g++-
  gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu gnupg gnupg-110n gnupg-utils gpg g
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libao
```

- 💡 Dies installiert den Python-Interpreter (python3) sowie `pip`, den Paketmanager für Python. Beides wird benötigt, um später JupyterLab und zusätzliche Pakete zu installieren. Das dritte Paket installiert die virtuelle Umgebung für Python. `apt` sorgt dafür, dass alle nötigen Abhängigkeiten aus den offiziellen Ubuntu-Repositories automatisch mit installiert werden. (Venv) steht für virtuelles Environment und wird von Python und Jupyterlab benötigt.

## 3. JupyterLab in virtueller Umgebung installieren

- Q Warum eine virtuelle Umgebung verwenden?** Die Installation von JupyterLab innerhalb einer `venv` stellt sicher, dass alle Python-Pakete unabhängig vom System verwaltet werden. Dies verhindert Versionskonflikte, schützt die Systemumgebung und ermöglicht die einfache Reproduzierbarkeit und Portabilität der Installation – besonders wichtig in Containern und bei mehreren parallelen Python-Projekten.

Virtuelle Umgebungen verhindern Konflikte zwischen globalen und projektspezifischen Python-Paketen und sind Best Practice bei modernen Python-Projekten.

- ## ◊ 3.1 Virtuelle Umgebung erstellen

```
python3 -m venv ~/jupyterlab/venv
```

```
pdal@jupyterlab180:~$ python3 -m venv ~/jupyterlab/venv  
pdal@jupyterlab180:~$
```

- 🔍 Erstellt eine neue virtuelle Umgebung im Verzeichnis `~/jupyterlab/venv`. Diese Umgebung enthält einen eigenen Python-Interpreter sowie eine eigene pip-Installation.

## Verzeichnisstruktur:

~ /jupyterlab/venv/ └── bin/ ← enthält python, pip, jupyterlab usw. └── lib/ ← Python-Bibliotheken (site-packages) └── pyvenv.cfg ← Konfigurationsdatei

- ### ◊ 3.2 Virtuelle Umgebung aktivieren

```
source ~/jupyterlab/venv/bin/activate
```

```
ndal@jupyterlab180:~$ source ~/jupyterlab/venv/bin/activate  
(ndal) pdal@jupyterlab180:~$
```

⌚ Aktiviert die virtuelle Umgebung:

- Die Shell verwendet ab jetzt `python` und `pip` aus der Umgebung, nicht mehr die systemweiten Versionen.
- Der Terminal-Prompt zeigt typischerweise (`venv`) zur Kennzeichnung.
- Alle nachfolgenden Installationen gelten nur innerhalb dieser Umgebung.

◊ 3.3 pip in der virtuellen Umgebung aktualisieren

```
pip install --upgrade pip
```

```
(venv) pdal@jupyterlab180:~$ pip install --upgrade pip
Requirement already satisfied: pip in ./jupyterlab/venv/lib/python3.12/site-packages (24.0)
Collecting pip
  Downloading pip-25.1.1-py3-none-any.whl.metadata (3.6 kB)
  Downloading pip-25.1.1-py3-none-any.whl (1.8 MB)
                                             1.8/1.8 MB 8.3 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.0
    Uninstalling pip-24.0:
      Successfully uninstalled pip-24.0
Successfully installed pip-25.1.1
(venv) pdal@jupyterlab180:~$ █
```

⌚ Aktualisiert den Paketmanager pip innerhalb der virtuellen Umgebung. Dies stellt sicher, dass aktuelle Funktionen und Kompatibilität gewährleistet sind.

◊ 3.4 JupyterLab installieren

```
pip install jupyterlab
```

⌚ Installiert JupyterLab sowie alle dazugehörigen Abhängigkeiten (z.B. notebook, traitlets, tornado) in die virtuelle Umgebung.

⌚ Nach der Installation ist JupyterLab über folgenden Pfad aufrufbar:

```
~/jupyterlab/venv/bin/jupyter-lab
```

JupyterLab kann mit `Ctrl + C` beendet werden; dann mit `ja` bestätigen. Die virtuelle Umgebung kann mit `deactivate` beendet werden.

✓ Ergebnis

- Eine vollständig isolierte, aktuelle und sichere JupyterLab-Installation – unabhängig von der System-Python-Umgebung.
- Geeignet für Server, Container und Umgebungen mit aktivem PEP 668, das globale Python-Installationen schützt.

## 📁 4. Arbeitsverzeichnis für Notebooks anlegen

### ◊ 4.1 Verzeichnis erstellen

```
mkdir -p ~/jupyterlab/projects
```

```
(venv) pdal@jupyterlab180:~/jupyterlab$ mkdir -p ~/jupyterlab/projects  
(venv) pdal@jupyterlab180:~/jupyterlab$ █
```

⌚ Erstellt das Verzeichnis **projects** im Ordner **~/jupyterlab**. Die Option **-p** stellt sicher, dass auch der übergeordnete Ordner (jupyterlab) angelegt wird, falls er noch nicht existiert.

Hinweis: Wechseln sie auf der Web-Oberfläche in das Verzeichnis **/jupyterlab/projects** um ihre Projekte dort zu speichern.

### Ergebnis

Ein sauber strukturierter Ort für eigene Notebook-Dateien, getrennt von der Python-Umgebung (venv). Dies erleichtert Organisation, Backup und spätere Erweiterungen.

## 🔒 5. Passwortschutz aktivieren

### ◊ 5.1 Virtuelle Umgebung aktivieren

```
source ~/jupyterlab/venv/bin/activate
```

⌚ Aktiviert die zuvor eingerichtete virtuelle Umgebung. Alle folgenden Jupyter-Befehle werden damit innerhalb der isolierten Umgebung ausgeführt. falls man sich noch nicht in der **venv** befindet.

### ◊ 5.2 Passwort für JupyterLab setzen

```
jupyter-lab password
```

```
(venv) pdal@jupyterlab180:~/jupyterlab/projects$ jupyter-lab password  
Enter password:  
Verify password:  
[JupyterPasswordApp] Wrote hashed password to /home/pdal/.jupyter/jupyter_server_config.json  
(venv) pdal@jupyterlab180:~/jupyterlab/projects$ █
```

⌚ Startet den interaktiven Dialog zur Vergabe eines Passworts für den Webzugang zu JupyterLab. Das eingegebene Passwort wird gehasht und in der Datei:

```
~/.jupyter/jupyter_server_config.json
```

gespeichert. Dieser Hash wird beim Start des Servers zur Authentifizierung verwendet.

Hinweis: Ein Ordner mit vorangestelltem Punkt (`.jupyter`) ist mit dem `dir`-Befehl nicht sichtbar. Hier müsste man `ls -a -l` verwenden.

## Ergebnis

Beim Öffnen von JupyterLab im Browser wird nun ein Login mit Benutzerpasswort verlangt. Dies schützt den Server vor unberechtigtem Zugriff – insbesondere in Netzwerken ohne SSL.

Optional kann in einem späteren Schritt HTTPS aktiviert werden (siehe Punkt 10). Für produktive Umgebungen ist die Kombination aus Passwort und TLS empfohlen.

## 6. JupyterLab starten (manuell)

### ◊ 6.1 JupyterLab manuell starten

```
~/jupyterlab/venv/bin/jupyter-lab --ip=0.0.0.0 --port=8888 --no-browser
```

```
(venv) pdal@jupyterlab180:~/jupyterlab/projects$ ~/jupyterlab/venv/bin/jupyter-lab --ip=0.0.0.0 --port=8888 --no-browser
[1 2025-07-28 12:33:14.0/2 ServerApp] jupyter_lsp | extension was successfully linked.
[1 2025-07-28 12:33:40.089 ServerApp] jupyter_server_terminals | extension was successfully linked.
[1 2025-07-28 12:33:40.103 ServerApp] jupyterlab | extension was successfully linked.
[1 2025-07-28 12:33:40.107 ServerApp] Writing Jupyter server cookie secret to /home/pdal/.local/share/jupyter/runtime/jupyter_cookie_secret
[1 2025-07-28 12:33:41.154 ServerApp] notebook_shim | extension was successfully linked.
[1 2025-07-28 12:33:41.202 ServerApp] notebook_shim | extension was successfully loaded.
[1 2025-07-28 12:33:41.208 ServerApp] jupyter_lsp | extension was successfully loaded.
[1 2025-07-28 12:33:41.211 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[1 2025-07-28 12:33:41.218 LabApp] JupyterLab extension loaded from /home/pdal/jupyterlab/venv/lib/python3.12/site-packages/jupyterlab
[1 2025-07-28 12:33:41.218 LabApp] JupyterLab application directory is /home/pdal/jupyterlab/venv/share/jupyter/lab
[1 2025-07-28 12:33:41.219 LabApp] Extension Manager is 'pypi'.
[1 2025-07-28 12:33:41.395 ServerApp] jupyterlab | extension was successfully loaded.
[1 2025-07-28 12:33:41.396 ServerApp] Serving notebooks from local directory: /home/pdal/jupyterlab/projects
[1 2025-07-28 12:33:41.397 ServerApp] Jupyter Server 2.16.0 is running at:
[1 2025-07-28 12:33:41.397 ServerApp] http://jupyterlab180:8888/lab
[1 2025-07-28 12:33:41.397 ServerApp] http://127.0.0.1:8888/lab
[1 2025-07-28 12:33:41.397 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation)
[1 2025-07-28 12:33:41.452 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-server, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, pascal-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-language-server
```

⌚ Startet den JupyterLab-Server in der zuvor erstellten virtuellen Umgebung:

- `--ip=0.0.0.0`: JupyterLab lauscht auf allen Netzwerkadressen des Containers.
- `--port=8888`: Der Dienst wird auf Port 8888 bereitgestellt.
- `--no-browser`: Verhindert, dass ein lokaler Browser geöffnet wird (nützlich auf Servern oder Headless-Systemen).
- mit `strg + c` beendet man den Server wieder. Dies muss allerdings mit `y` innerhalb von 5 Sekunden bestätigt werden.

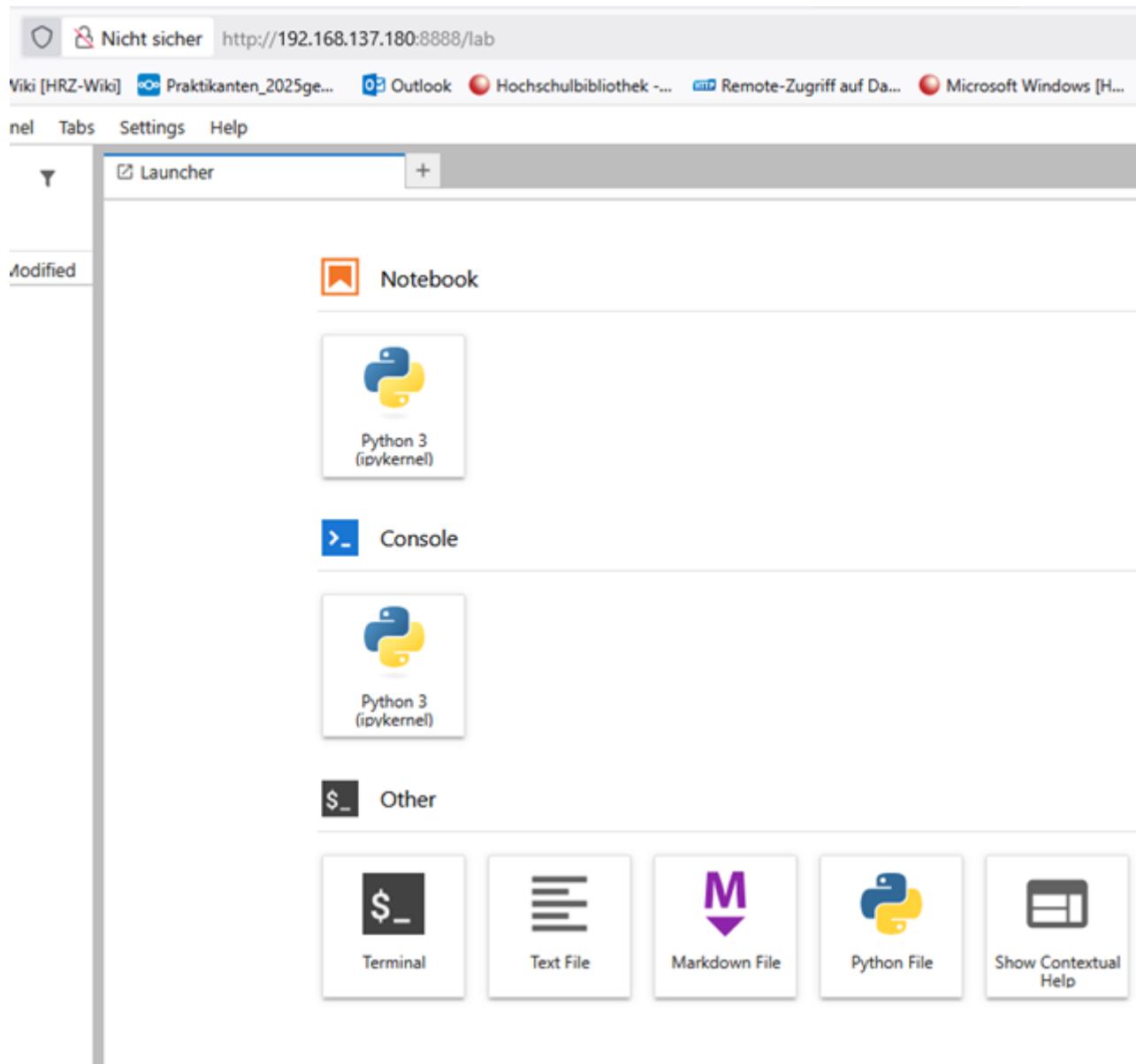
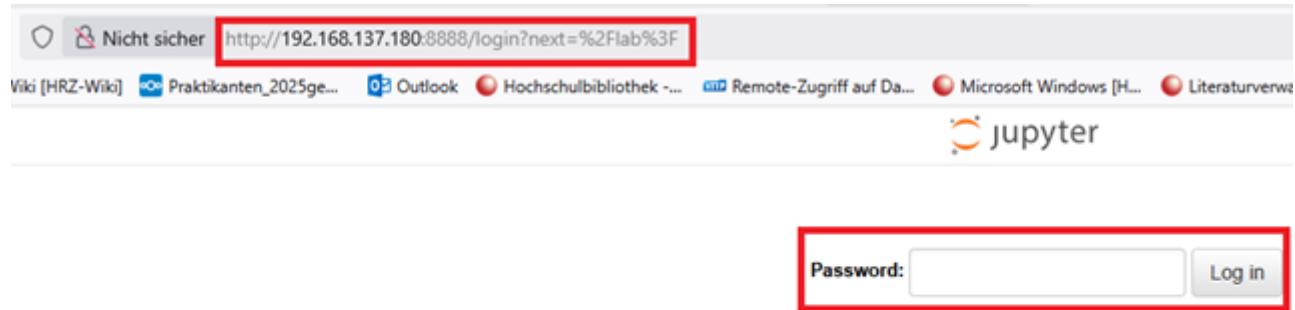
### ◊ 6.2 Webzugriff

🔗 Der Zugriff auf JupyterLab erfolgt über einen Webbrowser auf ihrem Client-PC, z. B.:

```
http://192.168.137.180:8888
```

Hinweis: Achten sie darauf <http://<ihre-ip-adresse>:8888/lab> zu verwenden - <https://<ihre-ip-adresse>:8888/lab> funktioniert noch nicht.

Wenn ein Passwort gesetzt wurde (siehe Punkt 5), erscheint zunächst eine Anmeldemaske.



**Ergebnis**

JupyterLab ist nun aktiv und über das Netzwerk erreichbar. Die Session bleibt aktiv, solange der Terminalprozess läuft (z.B. via tmux, screen oder Hintergrundprozess).

Bei Bedarf kann im nächsten Schritt eine systemweite oder benutzerspezifische Service-Datei für automatischen Start hinterlegt werden.

## ⚙️ 7. systemd-Dienst erstellen

⌚ **Ziel:** JupyterLab automatisch beim Systemstart als Hintergrunddienst ausführen – ohne manuelles Starten im Terminal.

### ◊ 7.1 Dienstdatei anlegen

```
sudo nano /etc/systemd/system/jupyterlab.service
```

⌚ Öffnet einen neuen systemweiten Dienst unter [/etc/systemd/system/](#). Dieser Dienst wird von [systemd](#) verwaltet und kann automatisch gestartet und überwacht werden.

### ◊ 7.2 Inhalt der Dienstdefinition

```
[Unit]
Description=JupyterLab (via systemd)
After=network.target

[Service]
Type=simple
User=pdal
Group=pdal
WorkingDirectory=/home/pdal
Environment="PATH=/home/pdal/jupyterlab/venv/bin:/usr/bin:/bin"
ExecStart=/home/pdal/jupyterlab/venv/bin/jupyter-lab --ip=0.0.0.0 --port=8888 --no-browser
Restart=on-failure

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

```
[Unit]
Description=JupyterLab (via systemd)
After=network.target

[Service]
Type=simple
User=pdal
Group=pdal
WorkingDirectory=/home/pdal
Environment="PATH=/home/pdal/jupyterlab/venv/bin:/usr/bin:/bin"
ExecStart=/home/pdal/jupyterlab/venv/bin/jupyter-lab --ip=0.0.0.0 --port=8888 --no-browser
Restart=on-failure

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

## 🔍 Erklärung der wichtigsten Parameter:

- **User / Group:** Führt den Dienst unter dem Benutzer pdal aus (sicherheitsrelevant).
- **WorkingDirectory:** Startverzeichnis für den Dienst.
- **Environment:** Wichtig: Diese Zeile setzt die PATH-Variable, sodass alle Tools aus der `venv` benutzt werden (z. B. `python`, `pip`, `jupyter-lab`). Ohne das wird ggf. das systemweite Python benutzt.
- **ExecStart:** Startet JupyterLab aus der virtuellen Umgebung.
- **Restart=on-failure:** Automatischer Neustart bei Fehlern oder Abstürzen.
- **WantedBy=multi-user.target:** Aktiviert den Dienst für den regulären Mehrbenutzermodus.

### ◊ 7.3 Dienst aktivieren und starten

```
sudo systemctl daemon-reload  
sudo systemctl enable jupyterlab  
sudo systemctl start jupyterlab
```

```
pdal@jupyterlab180:~/jupyterlab/projects$ sudo systemctl daemon-reload  
sudo systemctl enable jupyterlab  
sudo systemctl start jupyterlab  
Created symlink /etc/systemd/system/multi-user.target.wants/jupyterlab.service → /etc/systemd/system/jupyterlab.service.  
pdal@jupyterlab180:~/jupyterlab/projects$ █
```

## 🔍

- **daemon-reload:** Lädt neue oder geänderte Dienstdateien neu ein.
- **enable:** Aktiviert den automatischen Start beim Booten.
- **start:** Startet den Dienst sofort.

## ☑ Ergebnis

JupyterLab wird nun beim Systemstart automatisch im Hintergrund gestartet und ist über das Netzwerk erreichbar. Status- und Fehlermeldungen können jederzeit mit folgendem Befehl geprüft werden:

```
journalctl -u jupyterlab -f
```

## ⌚ 8. Java installieren & Kernel (nur für JupyterLab + Nutzer) einrichten

⌚ **Ziel:** Java-Laufzeitumgebung bereitstellen und den Java-Kernel (IJava) für den Benutzer in JupyterLab verfügbar machen.

### ◊ 8.1 Java JDK installieren

Sollte die `venv` noch aktiv sein, so sollten sie zunächst mit dem Befehl `deactivate` beendet werden.

```
deactivate
```

```
(venv) pdal@jupyterlab180:~/jupyterlab$ deactivate  
pdal@jupyterlab180:~/jupyterlab$ █
```

Die Deaktivierung dient der Klarheit, da der folgende Befehl (`sudo apt install ...`) eine systemweite Installation ist, die außerhalb des venv-Kontexts liegt.

```
sudo apt install openjdk-21-jdk -y
```

```
pdal@jupyterlab180:~/jupyterlab/projects$ sudo apt install openjdk-21-jdk -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
adwaita-icon-theme alsamixer alsa-topology-conf alsa-ucm-conf at-spi2-common at-spi2-
fonts-dejavu-extra gsettings-desktop-schemas gtk-update-icon-cache hicolor-
libatk-bridge2.0-0t64 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-
libavahi-common3 libcairo-gobject2 libcairo2 libcolord2 libcurl2t64 libdati-
libdrm-radeon1 libdrm2 libepoxy0 libgbm1 libgdk-pixbuf-2.0-0 libgdk-pixbuf2-
libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-1-
libice6 liblcms2-2 libllvml9 libnspr4 libnss3 libpango-1.0-0 libpangocairo-
libpthread-stubs0-dev librsvg2-2 librsvg2-common libsm-dev libsm6 libthai0-
libwayland-egl1 libwayland-server0 libx11-dev libx11-xcb1 libxau-dev libxau-
libxcb-render0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb-
libxkbcommon0 libxkbcommon-x110 libxkbcommon-xcb0 libxkbcommon-xcb1 libxkbcommon-xcb1-0
```



- Installiert das Java Development Kit (JDK) Version 21, das für Java-Anwendungen und die Ausführung des Java-Kernels benötigt wird.
  - Das Paket enthält Compiler, Laufzeitumgebung und weitere Werkzeuge.
    - ◊ 8.2 Java-Kernel installieren (nur für den aktuellen Benutzer)

Es muss unter Umständen noch das Paket `curl` und `unzip` nach installiert werden.

```
sudo apt install curl -y  
sudo apt install unzip -y
```

```
cd ~/jupyterlab  
curl -L -o ijava.zip  
https://github.com/SpencerPark/IJava/releases/download/v1.3.0/ijava-1.3.0.zip
```

```
pdal@jupyterlab180:~/jupyterlab$ curl -L -o ijava.zip https://github.com/SpencerPar  
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current  
                                         Dload  Upload   Total   Spent   Left  Speed  
  0      0    0      0      0       0      0 --:--:-- --:--:-- --:--:--      0  
100  3287k  100  3287k    0      0  3436k      0 --:--:-- --:--:-- --:--:--  3436k  
pdal@jupyterlab180:~/jupyterlab$
```

```
unzip ijava.zip -d ijava-installer
```

```
pdal@jupyterlab180:~/jupyterlab$ unzip ijava.zip -d ijava-installer
Archive:  ijava.zip
  creating: ijava-installer/java/
  inflating: ijava-installer/java/ijava-1.3.0.jar
  inflating: ijava-installer/java/kernel.json
  creating: ijava-installer/java/dependency-licenses/
  creating: ijava-installer/java/dependency-licenses/commons-lang3-3.8.
  creating: ijava-installer/java/dependency-licenses/commons-lang3-3.8.
  inflating: ijava-installer/java/dependency-licenses/commons-lang3-3.8.
  inflating: ijava-installer/java/dependency-licenses/commons-lang3-3.8.
  inflating: ijava-installer/java/dependency-licenses/dependencies.html
  inflating: ijava-installer/java/dependency-licenses/dependencies.json
  creating: ijava-installer/java/dependency-licenses/ivy-2.5.0-rc1.jar/
  creating: ijava-installer/java/dependency-licenses/ivy-2.5.0-rc1.jar/
  inflating: ijava-installer/java/dependency-licenses/ivy-2.5.0-rc1.jar/
  inflating: ijava-installer/java/dependency-licenses/ivy-2.5.0-rc1.jar/
  creating: ijava-installer/java/dependency-licenses/maven-artifact-3.6
  creating: ijava-installer/java/dependency-licenses/maven-artifact-3.6
  inflating: ijava-installer/java/dependency-licenses/maven-artifact-3.6
  inflating: ijava-installer/java/dependency-licenses/maven-artifact-3.6
  creating: ijava-installer/java/dependency-licenses/maven-builder-supp
  creating: ijava-installer/java/dependency-licenses/maven-builder-supp
  inflating: ijava-installer/java/dependency-licenses/maven-builder-supp
```

```
cd ijava-installer
~/jupyterlab/venv/bin/python3 install.py --user
```

```
pdal@jupyterlab180:~/jupyterlab$ cd ijava-installer
pdal@jupyterlab180:~/jupyterlab/ijava-installer$ ~/jupyterlab/venv/bin/python3 install.py --user
/home/pdal/jupyterlab/ijava-installer/install.py:164: DeprecationWarning: replace is ignored. Inst
ition
    install_dest = KernelSpecManager().install_kernel_spec(
Installed java kernel into "/home/pdal/.local/share/jupyter/kernels/java"
pdal@jupyterlab180:~/jupyterlab/ijava-installer$
```



- Lädt die offizielle IJava-Kernel-Version 1.3.0 herunter und entpackt sie.
- Führt das Installationsskript mit dem Python aus der virtuellen JupyterLab-Umgebung aus.
- Die Option `--user` installiert den Kernel nur für den aktuellen Benutzer, ohne Systemänderungen.
- Der Kernel ist danach nur in der JupyterLab-Instanz innerhalb der `venv` für diesen Benutzer sichtbar und nutzbar.

Danach können die zip-Datei und der `ijava-installer` gelöscht werden; `rm -r ijava.zip` und `rm -r ijava-installer/`.

## Ergebnis

Der Java-Kernel steht im JupyterLab nur für den betreffenden Benutzer zur Verfügung und kann in Notebooks zur Ausführung von Java-Code verwendet werden. Es erfolgt keine systemweite Installation oder Änderung.

## 9. (Optional) R installieren & IRkernel einrichten (nur für Nutzer & JupyterLab)

R ist eine Programmiersprache, die hauptsächlich für statistische Analysen, Datenbereinigung, Datenimport und Datenvisualisierung verwendet wird. Sie ist ein mächtiges Werkzeug für Data Scientists und Wissenschaftler.

 **Ziel:** R-Programmiersprache und den IRkernel nur für den Benutzer in JupyterLab verfügbar machen.

Hinweis: Machen Sie das, wenn sie die Sprache **R** benötigen.

### ◊ 9.1 R installieren

```
sudo apt install r-base -y
```

```
pdal@jupyterlab180:~/jupyterlab/ijava-installer$ sudo apt install r-base -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bzip2-doc gfortran gfortran-13 gfortran-13-x86-64-linux-gnu gfortran-x86-64
  libclone-perl libdata-dump-perl libegl-mesa0 libegl1 libencode-locale-perl
  libfile-mimeinfo-perl libfont-afm-perl libgfortran-13-dev libgfortran5 libg
  libhtml-tagset-perl libhtml-tree-perl libhttp-cookies-perl libhttp-daemon-p
  libicu-dev libio-html-perl libio-socket-ssl-perl libio-stringy-perl libipc-
  liblapack-dev liblapack3 liblwp-mediatypes-perl liblwp-protocol-https-perl
  libnet-http-perl libnet-smtp-ssl-perl libnet-ssleay-perl libpaper-utils libp
  libpng-dev libpng-tools libreadline-dev libtcl8.6 libtie-ixhash-perl libtim
  libwww-robotrules-perl libx11-protocol-perl libxml-parser-perl libxml-twig-
  pkgconf pkgconf-bin r-base-core r-base-dev r-base-html r-cran-boot r-cran-c
  r-cran-lattice r-cran-mass r-cran-matrix r-cran-mgcv r-cran-nlme r-cran-nne
  x11-xserver-utils xdg-utils zip zutty
Suggested packages:
  gfortran-multilib gfortran-doc gfortran-13-multilib gfortran-13-doc libcoar
  libbio-compress-brotli-perl icu-doc libcrypt-ssleay-perl liblzma-doc ncurses
  libregexp-ipv6-perl libauthen-ntlm-perl libunicode-map8-perl libunicode-str
  texlive-base texlive-latex-base texlive-plain-generic texlive-fonts-recomm
  texlive-latex-extra texinfo mozilla | www-browser nickle cairo-5c xorg-docs
The following NEW packages will be installed:
  bzip2-doc gfortran gfortran-13 gfortran-13-x86-64-linux-gnu gfortran-x86-64
```



- Installiert die Basis-R-Laufzeitumgebung.
- Ermöglicht die Ausführung von R-Skripten und -Paketen.

### ◊ 9.2 IRkernel installieren (nur für den aktuellen Benutzer und nur für die venv)

```
source ~/jupyterlab/venv/bin/activate
```

R

```
pdal@jupyterlab180:~/jupyterlab$ source ~/jupyterlab/venv/bin/activate
(venv) pdal@jupyterlab180:~/jupyterlab$ R

R version 4.3.3 (2024-02-29) -- "Angel Food Cake"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

In der R-Konsole:

```
install.packages("IRkernel")
```

Antworte mit **yes**.

```
> install.packages("IRkernel")
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
Warning in install.packages("IRkernel") :
  'lib = "/usr/local/lib/R/site-library"' is not writable
Would you like to use a personal library instead? (yes/No/cancel) yes
Would you like to create a personal library
'/home/pdal/R/x86_64-pc-linux-gnu-library/4.3'
to install packages into? (yes/No/cancel) |
```

```
IRkernel::installspec(user = TRUE)
```

```
> IRkernel::installspec(user = TRUE)
> |
```

```
q()
```

```
> q()  
Save workspace image? [y/n/c]: n  
>
```



- Installiert das IRkernel-Paket, das Jupyter die Ausführung von R-Notebooks ermöglicht.
- Die Funktion `installspec(user = TRUE)` registriert den Kernel nur für den aktuellen Benutzer.
- Nach Verlassen der R-Konsole (`q()`) steht der Kernel in JupyterLab nur für den Nutzer bereit.

## Ergebnis

Der R-Kernel ist ausschließlich für den jeweiligen Benutzer in JupyterLab verfügbar. Dies eignet sich besonders für LXC-Container oder Single-User-Umgebungen ohne systemweite Kernel-Installation.

### Abfrage installierter JupyterLab-Kernel

Um alle in JupyterLab verfügbaren Kernel (z.B. [Python](#), [R](#), [Java](#)) anzuzeigen, verwendet man folgenden Terminal-Befehl in der [venv](#):

```
jupyter kernelspec list
```

```
(venv) pdal@jupyterlab180:~/jupyterlab$ jupyter kernelspec list  
Available kernels:  
python3      /home/pdal/jupyterlab/venv/share/jupyter/kernels/python3  
ir          /home/pdal/.local/share/jupyter/kernels/ir  
java        /home/pdal/.local/share/jupyter/kernels/java  
(venv) pdal@jupyterlab180:~/jupyterlab$
```

Dieser Befehl listet alle installierten Kernel auf und zeigt deren Namen sowie die zugehörigen Pfadverzeichnisse an.

### Beispieldaten:

Available kernels:

Kernelname	Pfad
python3	/home/pdal/jupyterlab/venv/share/jupyter/kernels/python3
ir	/home/pdal/.local/share/jupyter/kernels/ir
java	/home/pdal/.local/share/jupyter/kernels/java

Erläuterung:

Kernelname	Beschreibung
python3	Python-Kernel (z. B. aus einer virtuellen Umgebung)
ir	R-Kernel (bereitgestellt durch IRkernel)
java	Java-Kernel (bereitgestellt durch IJava)

Alle aufgelisteten Kernel stehen dir im JupyterLab-Launcher sowie beim Kernel-Wechsel innerhalb von Notebooks zur Auswahl.

## 💡 10 Zusätzliche Python-Bibliotheken für JupyterLab installieren (Beispiel: paho-mqtt)

Zusätzliche Bibliotheken müssen innerhalb der Jupyterlab-Instanz installiert werden.

```
source ~/jupyterlab/venv/bin/activate
pip install paho-mqtt
```

```
(venv) pdal@jupyterlab180:~/jupyterlab$ pip install paho-mqtt
Collecting paho-mqtt
  Using cached paho_mqtt-2.1.0-py3-none-any.whl.metadata (23 kB)
Using cached paho_mqtt-2.1.0-py3-none-any.whl (67 kB)
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-2.1.0
(venv) pdal@jupyterlab180:~/jupyterlab$ █
```



- Aktiviert zuerst die virtuelle Umgebung von JupyterLab.
- Installiert mit pip die Python-Bibliothek paho-mqtt.
- paho-mqtt ist ein Beispiel für eine MQTT-Client-Bibliothek, die in Jupyter-Notebooks verwendet werden kann.
- Weitere Bibliotheken können analog innerhalb der virtuellen Umgebung installiert werden, um Konflikte mit Systempaketen zu vermeiden.

### 💡 Alternative: Installation über JupyterLab-Oberfläche Option 1: Terminal in JupyterLab öffnen

```
Menü → File → New → Terminal
```

Dann dort:

```
pip install <paketname>
```

```
pdal@jupyterlab180:~/jupyterlab/projects$ pip install paho-mqtt
Collecting paho-mqtt
  WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, st
  iled to establish a new connection: [Errno 101] Network is unreachable'): /pa
    Downloading paho_mqtt-2.1.0-py3-none-any.whl.metadata (23 kB)
    Downloading paho_mqtt-2.1.0-py3-none-any.whl (67 kB)
    Installing collected packages: paho-mqtt
    Successfully installed paho-mqtt-2.1.0
pdal@jupyterlab180:~/jupyterlab/projects$
```

## Option 2: Direkt im Notebook installieren

Im Codefeld einer Notebook-Zelle:

```
!pip install paho-mqtt
```

**Hinweis:** Diese Methode funktioniert nur innerhalb der aktiven Kernel-Umgebung (z.B. Python venv) und setzt voraus, dass der Kernel korrekt initialisiert wurde.  **Ergebnis** Installierte Pakete stehen direkt im Notebook zur Verfügung – ideal für dynamisches Arbeiten mit z.B. matplotlib, pandas, numpy, paho-mqtt usw.

## 🔒 10. (Optional) SSL-Verschlüsselung

JupyterLab kann mit TLS/SSL abgesichert betrieben werden.

SSL-Konfiguration siehe offizielle Dokumentation:

[HTTPS in Jupyter aktivieren](#)

Für die Integration eigener Zertifikate (z.B. aus einer lokalen CA):

**Siehe separate Anleitung:** [0650CA-sslmitSANZertifikat.md]

## ❖ Zusammenfassung

Komponente	Status / Wert
Python-Version	3.x
JupyterLab	Aktuell via venv + pip
Java	OpenJDK 21, IJava (nur für Nutzer)

Komponente	Status / Wert
R	r-base, IRkernel (nur für Nutzer)
Startport	<a href="http://&lt;IP&gt;:8888">http://&lt;IP&gt;:8888</a>
Dienstbetrieb	<input checked="" type="checkbox"/> via systemd
SSL	optional, siehe externe Anleitung
Passwortschutz	<input checked="" type="checkbox"/> (via <a href="#">jupyter-lab password</a> )
Systemuser	z. B. <a href="#">pdal</a>

## 📋 Hinweise zu Erweiterungen, Updates & Backup

- Erweiterungen (z. B. jupyterlab-git, jupyterlab-lsp) sind optional und können über den Extension Manager oder pip installiert werden.
- Updates:

```
pip list --outdated
pip install --upgrade <paketname>
```

Backups: Einfach per rsync, z. B.:

```
rsync -a ~/jupyterlab/projects /mnt/backup/jupyterlab/
```

## 📋 Quellen

- „Documentation · IRkernel“. Zugegriffen: 28. Juli 2025. [Online]. Verfügbar unter: [IRkernel](#)
- „Get Started — JupyterLab 4.5.0a1 documentation“ . Zugegriffen: 28. Juli 2025. [Online]. Verfügbar unter: [Jupyterlab getting\\_started](#)
- „Installation — JupyterLab 4.5.0a1 documentation“ . Zugegriffen: 28. Juli 2025. [Online]. Verfügbar unter: [Jupyterlab Installation](#)
- S. Park, SpencerPark/IJava. (27. Juli 2025). Java. Zugegriffen: 28. Juli 2025. [Online]. Verfügbar unter: [Github IJava](#)

## Lizenz

Dieses Werk ist lizenziert unter der **Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz**.

[Zum Lizenztext auf der Creative Commons Webseite](#)