

[Deploy applications](#)[Data Science catalog](#)[Jupyter Notebooks](#)

Jupyter Notebooks



Start JupyterLab

Start a JupyterLab container based on the [official Jupyter docker stacks](#) (debian), with `sudo` privileges to install anything you need (e.g. pip or apt packages)

You can start a container using the **JupyterLab** template in the [Catalog web UI](#) (make sure the **Templates** checkbox is checked)

When instantiating the template you can provide a few parameters, such as:

- **Password** to access the notebook
- Optionally you can provide a **git repository** to be automatically cloned in the JupyterLab (if there is a `requirements.txt` packages will be automatically installed with `pip`)
- **Docker image** to use for the notebook (see below for more details on customizing the docker image)
- Your **git username and email** to automatically configure git

The DSRI will automatically create a persistent volume to store data you will put in the `/home/jovyan/work` folder (the folder used by the notebook interface). You can find the persistent volumes in the DSRI web UI, go to the **Administrator** view > **Storage** > **Persistent Volume Claims**.

Developer Catalog

Add shared apps, services, or source-to-image builders to your project from the Developer Catalog. CLI automatically.

All Items

Languages

Databases

Middleware

CI/CD

Other

Type

☐ Operator Backed (0)

☐ Helm Charts (0)


☐ Builder Image (0)

☐ Template (1)

☐ Service Class (0)

All Items

Group By: None ▼

Template

JupyterLab with root user
(Persistent)
provided by Institute of Data
Science, UM

Start JupyterLab with the 'root'
user using the image defined at
<https://github.com/vemonet/Ju...>

With this template you can use any image based on the official Jupyter docker stack:

<https://github.com/jupyter/docker-stacks>

- `ghcr.io/maastrichtu-ids/jupyterlab:latest`: custom image for data science on the DSRI, with additional kernels (Java), conda integration, VisualStudio Code, and autocomplete for Python
- `ghcr.io/maastrichtu-ids/jupyterlab:knowledge-graph`: custom image for working with knowledge graph on the DSRI, with SPARQL kernel and OpenRefine
- `jupyter/scipy-notebook`: some packages for science are preinstalled
- `jupyter/datascience-notebook`: with Julia kernel
- `jupyter/tensorflow-notebook`: with tensorflow package pre-installed
- `jupyter/r-notebook`: to work with R
- `jupyter/pyspark-notebook`: if you want to connect to a Spark cluster
- `jupyter/all-spark-notebook`: if you want to run Spark locally in the notebook

You can also build your own image, we recommend to use this repository as example to extend a JupyterLab image: <https://github.com/MaastrichtU-IDS/jupyterlab>



Manage dependencies with Conda

With the `ghcr.io/maastrichtu-ids/jupyterlab:latest` image, you can easily start notebooks from the JupyterLab Launcher page using installed conda environments, at the condition `nb_conda_kernels` and `ipykernel` are installed in those environments.

- You can pass a Git repository URL which contains an `environment.yml` file in the root folder when starting JupyterLab, the conda environment will automatically be installed at the start of your container, and available in the JupyterLab Launcher page. You can use this repository as example: <https://github.com/MaastrichtU-IDS/dsri-demo>
- Or you can install it directly in a running JupyterLab (we use `mamba` which is like `conda` but faster):

```
mamba env create -f environment.yml
```

You'll need to wait for 1 or 2 minutes before the new conda environment becomes available on the JupyterLab Launcher page.

You can easily install an environment with a different version of Python if you need it. Here is an example of an `environment.yml` file to create an environment with Python 3.9, install the minimal dependencies required to easily starts notebooks in this environment with `conda`, and install a `pip` package:

```
name: custom-env
channels:
  - defaults
  - conda-forge
  - anaconda
dependencies:
  - python=3.9
  - ipykernel
  - nb_conda_kernels
  - pip
  - pip:
    - matplotlib
```

⚠ You cannot use `conda activate` in a Docker container, so you will need to either open a notebook using the kernel for your conda env, or use `conda run` to run scripts in the new environment:

```
conda run -n custom-env python --version
```

Use git in JupyterLab

You can always use `git` from the terminal.

CONFIGURE USERNAME

Before pushing back to GitHub or GitLab, you will need to **configure you username and email** in VSCode terminal:

```
git config --global user.name "Jean Dupont"
git config --global user.email jeandupont@gmail.com
```

SAVE YOUR PASSWORD

You can run this command to ask git to save your password for 15min:

```
git config credential.helper cache
```

Or store the password in a plain text file:

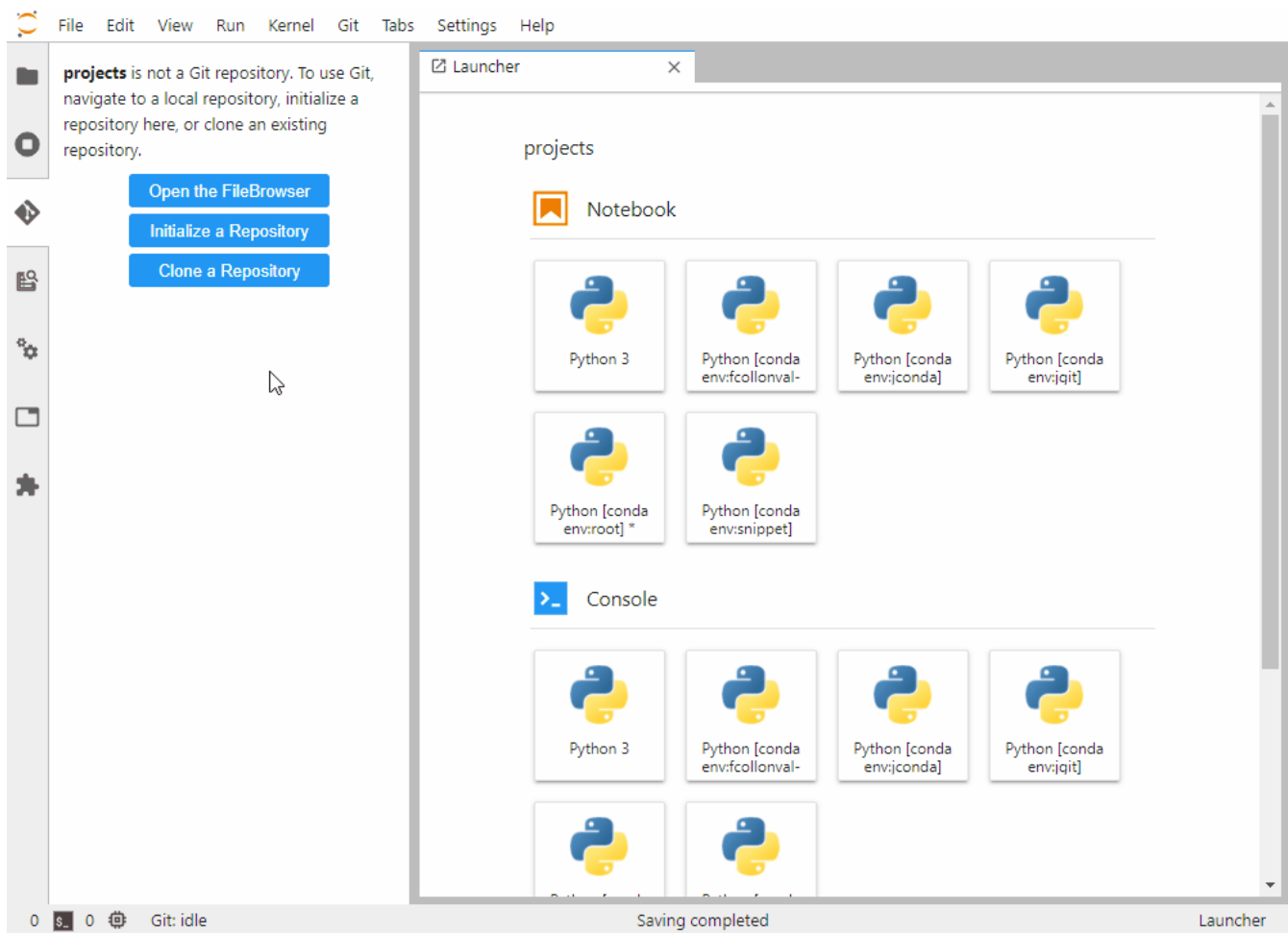
```
git config --global credential.helper 'store --file ~/.git-credentials'
```


GIT TIP

We recommend to use SSH instead of HTTPS connection when possible, checkout [here](#) how to generate SSH keys and use them with your GitHub account.

You can also enable and use the **JupyterLab Git extension** to clone and manage your `git` repositories.

It will prompt you for a username and password if the repository is private.



 [Edit this page](#)

Last updated on **Dec 6, 2022** by **Vincent Emonet**