Best practices for managing JupyterLab-based data science projects using Conda (+pip)

Dr. David R. Pugh

Staff Scientist, KAUST Visualization Core Lab
Certified Instructor, The Carpentries

Outline

- "System-wide" JupyterLab installation
- "Project-based" JupyterLab installation
- Provide some "best-practices"
- Discuss the relevant tradeoffs

"System-wide" JupyterLab

Conda (+pip) manage a JupyterLab installation shared across all projects.

- Common set of JupyterLab extensions simplifies user interface (UI) and user experience (UX).
- Allows for quicker start of new projects as no need to install (and build!) JupyterLab.
- Easy low-level configuration of JupyterLab via files inside the `~/.jupyter` directory in your user home directory.

`environment.yml` for a "system-wide" install

```yaml
name: jupyterlab-base-env

channels:
  - conda-forge
  - defaults

dependencies:
  - jupyterlab
  - jupyterlab-git # provides git support
  - nodejs # required for building (some) extensions
  - pip
  - pip:
    - -r file:requirements.txt # extensions available via pip go here
  - python
  - xeus-python
```

"System-wide" JupyterLab best practices

`jupyterlab-base-env` should *only* contain JupyterLab and required extensions (+deps).

- Automate environment build with Bash script.
- Projects should have separate Conda (+pip) environments.
- Create custom Jupyter kernels for project Conda (+pip) environments.

Automate environment build with Bash script

```
#!/bin/bash --login
set -e

conda env create \
    --name jupyterlab-base-env \
    --file environment.yml \
    --force
conda activate jupyterlab-base-env
source postBuild # put jupyter labextension install commands here
```

Example

https://github.com/davidrpugh/jupytercon-2020-talk/tree/jupyterlab-base-env

Creating Jupyter kernels for Conda environments

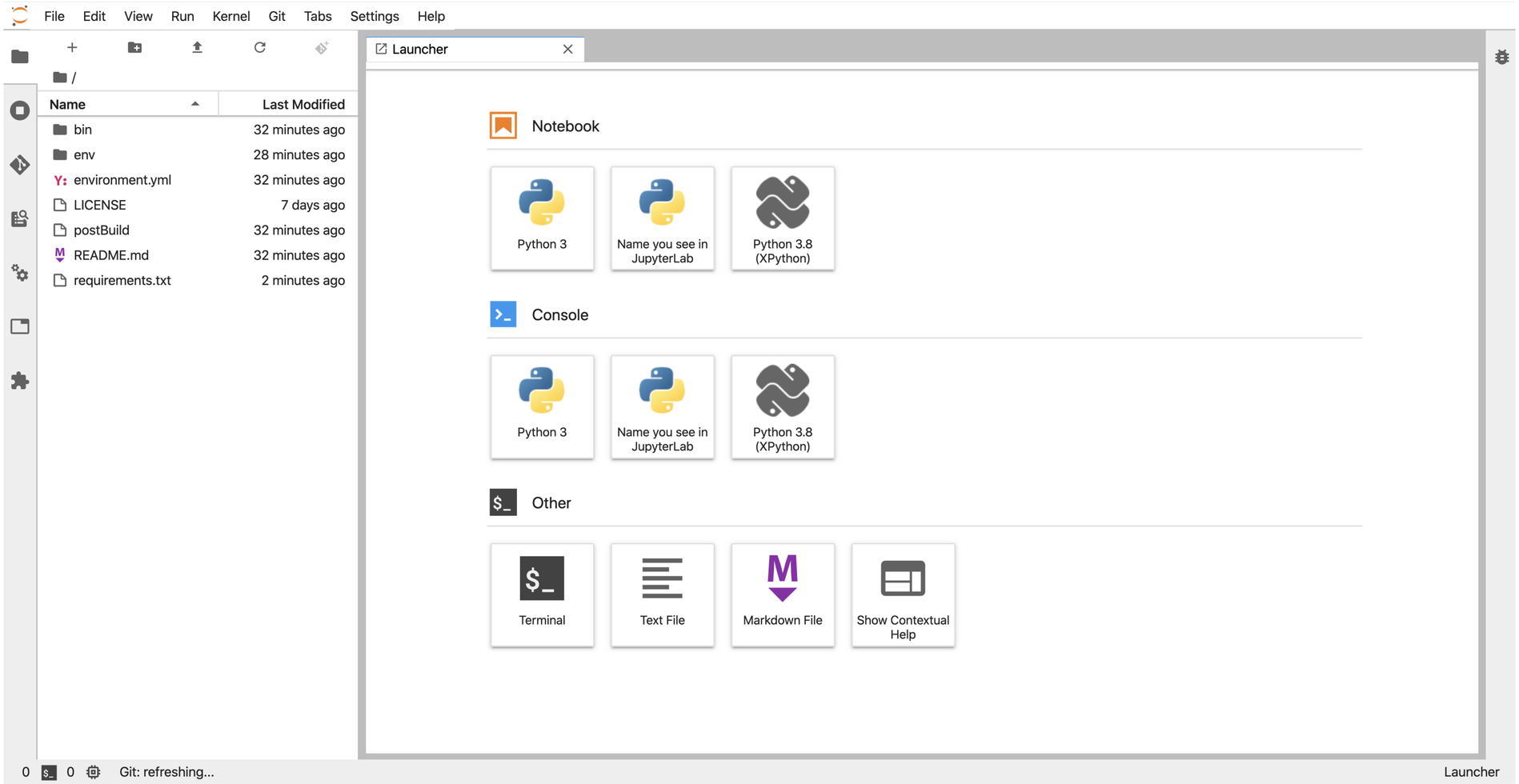
Allows you to launch Jupyter Notebooks and IPython consoles for different Conda (+pip) environments within a common JupyterLab installation.

- Can automate process for all Conda (+pip) envs using `jupyter-conda` extension.
- Can manually create custom Jupyter kernel for particular Conda (+pip) envs.

How to manually create custom Jupyter kernel

```
conda activate $PROJECT_DIR/env # don't forget to activate env first!
python -m ipykernel install \ # requires ipykernel installed in the env
    --user \
    --name $PROJECT_NAME-kernel \ # for internal use only!
    --display-name "Name you will see in JupyterLab"
```

Example


File Edit View Run Kernel Git Tabs Settings Help
/
Name
Last Modified
bin 32 minutes ago
env 28 minutes ago
environment.yml 32 minutes ago
LICENSE 7 days ago
postBuild 32 minutes ago
README.md 32 minutes ago
requirements.txt 2 minutes ago
Launcher
Notebook
Python 3
Name you see in JupyterLab
Python 3.8 (XPython)
Console
Python 3
Name you see in JupyterLab
Python 3.8 (XPython)
Other
Terminal
Text File
Markdown File
Show Contextual Help
0 0 Git: refreshing...
Launcher

`%conda` and `%pip` magic commands

Built-in IPython magic commands for installing packages into the *active* kernel via Conda (`%conda`) or Pip (`%pip`).

- Both commands can be used from within Jupyter Notebooks or IPython consoles.
- Both `%conda` and `%pip` are mostly useful for prototyping new projects.
- For "production", prefer adding new packages to `environment.yml`/`requirements.txt` (and rebuilding environment).

"Project-based" JupyterLab

Conda (+pip) manage separate JupyterLab installations for each project.

- More flexible UI/UX as JupyterLab version and extensions can customized for each project.
- Easier experimentation with bleeding edge features of JupyterLab.
- Automatically makes a data science project repo "Binder-ready".

`environment.yml` for a "project-based" install

```yaml
name: null

channels:
  - conda-forge
  - defaults

dependencies:
  - jupyterlab
  - jupyterlab-git # extensions available via conda go here
  - nodejs # required for building (some) extensions
  - pandas # now project deps go here!
  - pip
  - pip:
    - -r file:requirements.txt # packages available via pip go here
  - python
  - scikit-learn
```

Automate environment creation with Bash script

```
#!/bin/bash --login
set -e

export ENV_PREFIX=$PROJECT_DIR/env # directory included in .gitignore
conda env create \
    --prefix $ENV_PREFIX
    --file environment.yml \
    --force
conda activate $ENV_PREFIX
source postBuild # put jupyter labextension install commands here
```

Examples of "project-based" JupyterLab installs

- JupyterLab + Scikit Learn + friends
- JupyterLab + PyTorch + friends
- JupyterLab + NVIDIA RAPIDS + BlazingSQL + Dask

"System-wide" or "project-based"?

- Prefer "project-based" for its greater flexibility with minimal additional overhead.
- Prefer "project-based" if only some of your projects use GPUs.
- Prefer "system-wide" when all projects use a common set of extensions.

Thanks!

https://github.com/davidrpugh/jupytercon-2020-talk

TheSandyCoder

davidrpugh

davidrpugh