Environment management

micromamba

Valentin Pestel
LPC Caen Bootcamp - 04/11/2024

What are we trying to solve

Living by the scientific method:

- I want my analysis to be reproducible
  - ➢ Required software environment should be easily reproducible as well

Avoiding “Dependency hell”:

- I need Software *B* and *C*, both relying on a different version of Software *A*
  - ➢ I need 2 different version of *A*, and the ability to easily swap between both setup

Make deployment easy:

- Analysis is growing, need to move to a bigger data center to access more computing power
  - ➢ I need something to deploy the required softwares on the accessible machines

But what is versioning ?

From *semantic versioning* *(https://semver.org/)* :

Given a version number MAJOR.MINOR.PATCH, increment the:

- MAJOR version when you make incompatible API changes
- MINOR version when you add functionality in a backward compatible manner
- PATCH version when you make backward compatible bug fixes

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Many solution in the wild

Package manager:

- `apt` on Debian-based system
- Manage package installation on your machine
  - Typically needs root privilege. Only one environment : your machine

Python, venv and `pip`:

- Create a “virtual environment”, where you can install python package with `pip`
  - No root privilege, can handle multiple contexts with multiple environments in parallel
  - Easy export through a text files containing the packages to install
  - Handle only python packages

Conda environment:

- Create and manage different environments
  - No root privilege, can handle multiple contexts with multiple environments in parallel
  - Easy export through a text files containing the packages to install
  - Solving dependencies is terribly slow

`mamba` and `micromamba`

`mamba` is a C++ reimplementation ++ of `conda`:

- All the good from conda, with blazing fast dependencies resolution
- Command line interface is exactly the same than conda
- Reference: official documentation

`micromamba` is the easy-to-deploy version of `mamba`

- Installation instructions:
  - `"${SHELL}" <(curl -L micro.mamba.pm/install.sh)`
  - Prompt a series of question (see details in next slide)
- Reference:
  - LPC-dev example (<- base material for this bootcamp)
  - official documentation

Micromamba configuration

"${SHELL}" <(curl -L micro.mamba.pm/install.sh)

- ➢ `Micromamba binary folder? [~/.local/bin]`
  - Where the micromamba binary will be stored. Default works well on your local machine as well as on data centers.
- ➢ `Init shell (bash)? [Y/n]`
  - Automatically update the `bashrc` to define the environment variables. Same as running manually `micromamba shell init`
- ➢ `Configure conda-forge? [Y/n]`
  - Configure conda-forge as default package provider. Probably the relevant choice in 99.9% of cases.
- ➢ `Prefix location? [~/micromamba]`
  - Define where the environments will be stored per default. Environment can get pretty large (few hundreds MB to few GB). Preferably set to a place with disk space available (e.g. in data centers, maybe not in the `Home` folder). If not sure, ask around ;)

DONE ! (just need to source the bashrc)

Creating my first environment

First, let's `my_first_env` and activate it:

```
➢  micromamba env create -n my_first_env
➢  micromamba activate my_first_env
```

Ok, let's install something now: python v3.9

```
➢  micromamba install python==3.9
```

Ho yeah, I need Snakemake as well, but version >= 8:

```
➢  micromamba install -c conda-forge -c bioconda "snakemake>=8"
```

Does it work ? Why ?

Another detour: the channels

Channels are remote package repository. They contains package names, versions and associated dependencies.

- Most common one by far is `conda-forge`, but other exists (`bioconda`)
- In the past, `anaconda` was a very popular one, but the term of condition have changed in 2024 and makes it not entirely free for academia (link)

Environment spec files

The definition of an environment can be provided as a spec files

- ➢ Can be provided along the analysis code
- ➢ Contains all the relevant information

```
micromamba env create -f <spec_file.yaml>
```

```yaml
name: root_py11

channels:
  - conda-forge
  - bioconda # for Snakemake

dependencies:
  - python=3.11
  - root>=6.32
  - apptainer
  - git-lfs
  - git
  - uproot
  - pip
  - snakemake>=8
  - pip:
      - graphviz
```

Spec files for the bootcamp, git repository

Some useful command

```
micromamba env list
```

➢ List available environments

```
micromamba list
```

➢ List installed package in the current environment

```
micromamba env remove -n <env name>
```

➢ Delete environment “env name”

```
micromamba -h | micromamba <subcommand> -h
```

➢ List available subcommands, or help for given subcommand