Conda (mamba)
Software Engineering for Scientists

Conda is a dependency manager for Python. It began as part of the Anaconda Python distribution, which was geared towards data scientists and included hundreds of useful libraries. The conda part of Anaconda was quickly adopted by the Python community and became its own open source project. Why? Many software libraries depend on other libraries, and those libraries depend on others and so on. Finding and installing these can be tedious. Dependencies can even be version specific, meaning that if you accidentally got a newer or older version of a dependency installed than what was expected, the library may not work. Conda takes care of these issues by comparing the software installed locally with a given set of dependencies (specified by the library developers) and installing any missing or out of data libraries. To account for conflicting dependencies between projects, conda allows users to create and name different environments.

Conda has been very successful, but as its size and complexity grew conda became painfully slow. Mamba (https://github.com/mamba-org/mamba) is the solution. Mamba is a transparent replacement for the conda command, so anywhere you see `$ conda` you can safely replace it with `$ mamba`.

Installing Mamba in your environment if you don't have it (`$ mamba` gives an error)

This is confusing. To get **conda** you need to install **anaconda**. Anaconda is a python distribution that is meant to make data science easier. Anaconda provides conda. Conda has become popular independent of anaconda, which leads to **miniconda**, which cuts out all of the extra stuff in anaconda. **Conda is slow**, which leads to the more efficient **mamba** (https://github.com/mamba-org/mamba). We will use mamba in this class. In the beginning, we installed conda, then used conda to install mamba. This method is not recommended. Instead use the Mambaforge distribution (https://github.com/conda-forge/miniforge#mambaforge-pypy3).

```
$ cd $HOME
$ curl -L -O
https://github.com/conda-forge/miniforge/releases/latest/download/Mambaforge-pypy3-Linux-
x86_64.sh
$ bash Mambaforge-pypy3-Linux-x86_64.sh.sh
```

NOTE: The second command (`$ curl -L -O <url>`) is long and wraps across lines 2-4. Coping and pasting into your terminal may result in 2 commands, first `$ curl -L -O` then `$ https://....` To fix this copy and paste in two steps. First `curl -L -O` then the URL into the same line.

Once mamba is installed, please update it

```
$ mamba update --yes mamba
```

Create an environment just once per environment

```
$ mamba create --yes -n swe4s
```

Entering your evnironment do this every time you log in

```
$ conda activate swe4s
(swe4s) $
```

Installing software/libraries

With Conda

```
(swe4s) $ mamba install -y pycodestyle
Collecting package metadata: done
Solving environment: done
...
```

With pip

```
(swe4s) $ pip install pycodestyle
Collecting pycodestyle
...
```

If pip is not installed:

```
$ sudo easy_install pip
$ sudo pip install --upgrade pip
```

Saving your environment

Once you have all of your dependencies installed, you can create a file that specifies everything that has been installed and provides a way of easily recreating your environment. The `conda env export` command will also list all of the packages that were installed using pip.

```
(swe4s) $ mamba env export > environment.yml
```

`environment.yml`

```
name: swe4s
channels:
  - conda-forge
dependencies:
  - _libgcc_mutex=0.1=conda_forge
  - _openmp_mutex=4.5=2_gnu
  - bzip2=1.0.8=h7f98852_4
  - ca-certificates=2022.9.24=ha878542_0
  - ld_impl_linux-64=2.36.1=hea4e1c9_2
  - libffi=3.4.2=h7f98852_5
  - libgcc-ng=12.1.0=h8d9b700_16
  - libgomp=12.1.0=h8d9b700_16
  - libnsl=2.0.0=h7f98852_0
  - libsqlite=3.39.4=h753d276_0
  - libuuid=2.32.1=h7f98852_1000
  - libzlib=1.2.12=h166bdaf_4
  - ncurses=6.3=h27087fc_1
  - openssl=3.0.5=h166bdaf_2
  - pip=22.2.2=pyhd8ed1ab_0
  - pycodestyle=2.9.1=pyhd8ed1ab_0
  - python=3.10.6=ha86cf86_0_cpython
  - readline=8.1.2=h0f457ee_0
  - setuptools=65.4.1=pyhd8ed1ab_0
  - tk=8.6.12=h27826a3_0
  - tzdata=2022d=h191b570_0
  - wheel=0.37.1=pyhd8ed1ab_0
  - xz=5.2.6=h166bdaf_0
prefix: /home/jovyan/.conda/envs/swe4s
```

Re-creating an environment

With the environment.yml file, a nearly identical environment can be created on any system with python installed. To make software reproducibility easier, a conda environment should be curated for each of your projects and the associated environment.yml file checked into the repository. In addition to allowing others to more easily use your software, this will also more complete continuous integration testing.

```
$ mamba env create -f environment.yml
$ mamba activate swe4s
```

WAIT

Sharing your environment

While the standard `conda env export` command gives a complete listing of the packages installed in your environment and is good for recreating your environment, **IT IS NOT GOOD FOR SHARING YOUR ENVIRONMENT**. The packages selected in the dependency resolving stage depend on what operating system you are using, and what packages you already have installed. **It is not platform independent**. An environment created on a Mac is very unlikely to work in a Linux environment.

You can restrict the output to only include the packages you directly installed

```
$ mamba env export –from-history
name: swe4s
channels:
  - conda-forge
dependencies:
  - pycodestyle
prefix: /home/jovyan/.conda/envs/swe4s
```

But even this output is unlikely to work for a collaborator because the `prefix` tag depends on the existence of the `jovyan` user. As your research project grows you will likely install many different packages, but only a subset will truly be required for your software.

The **recommended approach** is to write your environment by hand, using the output from `mamba env export –from-history` as a starting point and keeping only the most essential packages (and removing `prefix`).