Mamba Package Manager

Resources:

- Conda Cheat Sheet

What is Mamba and why should you use it?

The Mamba (also the legacy Conda) package manager is a cross-platform toolkit for creating and managing virtual environments --- semi-self-contained Unix command-line configurations --- that allow flexible installation of, and access to, the executables and libraries needed to perform different analyses. These virtual environments are particularly useful when programs or pipelines require conflicting dependencies (e.g., require the same executable or library to be installed, but require different versions). Mamba even allows the user to export detailed descriptions of an environment to a file that can be shared with others, allowing them to reproducibly replicate the original user's environment. This promotes reproducibility and transparency of data analyses, making Mamba a valuable component of the scientific software stack!

The `mamba` toolkit can be used to install Python programs and modules as well as other third-party interpreted or compiled softwares (like Perl, R, C, and C++ programs and their dependent libraries). `mamba` enables the user to search for available software in established repositories ('channels' in Mamba parlance), installs the desired softwares (pre-compiled), and can be used to make updates as needed. `mamba` tracks all installed software versions and build information, and even allows the user to build their own packages to be shared with others.

There are two main distributions of Mamba:

1. **Mamba** : Contains the Python-based `mamba` command-line package manager and a minimal Python interpreter and set of libraries/modules installed.
2. **Micromamba** : Contains the C++-based `micromamba` command-line package manager and a minimal set of libraries installed. It's meant to be a streamlined, super-lite version. No Python interpreter installed by default.

The basics

How to create an environment:

The following will create a minimal (i.e., "empty") virtual environment called `envName`:

```
$ mamba create --name envName
```

How to load/activate an environment:

To run executables or import libraries installed in a virtual environment, you first need to activate it. When you activate a Mamba virtual environment, the new environment inherits your current Unix environment (so you'll still be able to use `ls`, etc., for example), but now gives you access to tools that are installed in `envName`:

```
$ mamba activate envName
```

How to search for available software:

Mamba can search for software of interest from the command line. The command below will search for the `pkgName` software package and write out the versions and builds available for installation.

```
$ mamba search pkgName
```

For example, to search for the `wget` Unix command-line tool:

```
$ mamba search wget
```

How to install software:

You can install one or more software packages easily with Mamba. It will first examine the software versions already installed in your loaded environment (if applicable), determine whether the package being installed has any unsatisfied dependencies, and create a list of dependencies (if any) that also need to be installed. This is what Mamba calls "Solving your environment".

By default, Mamba then installs the software package requested. If no package version was specified, Mamba will choose a compatible version for you. If the software being installed was written in a compiled language --- such as C, C++, Java, etc. --- Mamba will choose a pre-compiled build appropriate for your system. This saves you time and avoids many headaches caused by the often-tedious compilation process.

> NOTE: To install software into a virtual environment, you must first `mamba activate` your environment or specify the `--name envName` option.

```
# assuming that envName is already activated
$ mamba install pkgName1 pkgName2 ...

# or, when not activated, us the `--name` option:
$ mamba install --name envName pkgName
```

If you want to specify a particular version (and build) of a tool you want to install, include the version, and optionally the build identifier, after the package name separated by equal (`=`) signs (the square brackets `[]` below denote optional components of the command):

```
$ mamba install pkgName[=Version[=Build]]
```

> NOTE: To install software into a virtual environment, you must first `mamba activate` your environment or specify the `--name envName` option.

For example:

1. Search for the software you want to install with Mamba:

```
$ mamba search wget

     wget 1.24.5 h3a17b82_0
----------------------------------------
 Name            wget
 Version         1.24.5
 Build           h3a17b82_0
 Size            374 kB
 License         GPL-3.0-or-later
 Subdir          osx-64
 File Name       wget-1.24.5-h3a17b82_0.conda
 URL             https://repo.anaconda.com/pkgs/main/osx-64/wget-1.24.5-
h3a17b82_0.conda
 MD5             a2b857f21972e726022923f22ec6f0e4
 SHA256          fb74164357f217b11bce50ff308e1b68eddc4b72202eebeb4a1fb5e8c4db65f6

 Dependencies:
  - libidn2 >=2,<3.0a0
  - libunistring >=0,<1.0a0
  - openssl >=3.0.14,<4.0a0
  - zlib >=1.2.13,<1.3.0a0

 Other Versions (7):

  Version Build
 ----------------------------------------
  1.21.4  hca547e6_0          (+ 3 builds)
  1.21.3  h6dfd666_0
  ...     (3 hidden versions)          ...
  1.19.4  h073198b_0
  1.19.1  hcb5d8a9_0
```

2. Then choose the version (and build) you want to install:

```
$ mamba install wget=1.19.1=hcb5d8a9_0
```

How to check packages already installed:

Mamba provides an utility to interrogate which packages (with versions and builds) are installed in your environment. This can be useful when writing up your Methods sections!

```
# Assuming your environment is activated
$ mamba list

# If the `envName` environment was not already activated
$ mamba list --name envName
```

For example:

```
$ mamba list

    List of packages in environment: "/Users/username/.micromamba/envs/envName"

      Name                 Version    Build       Channel
    ------------------------------------------------------------
      ca-certificates      2024.8.30  h8857fd0_0  conda-forge
      gettext              0.22.5     hdfe23c8_3  conda-forge
      gettext-tools        0.22.5     hdfe23c8_3  conda-forge
      libasprintf          0.22.5     hdfe23c8_3  conda-forge
      libasprintf-devel    0.22.5     hdfe23c8_3  conda-forge
      libcxx               19.1.2     hf95d169_0  conda-forge
      libgettextpo         0.22.5     hdfe23c8_3  conda-forge
      libgettextpo-devel   0.22.5     hdfe23c8_3  conda-forge
      libiconv             1.17       hd75f5a5_2  conda-forge
      libidn2              2.3.7      h10d778d_0  conda-forge
      libintl              0.22.5     hdfe23c8_3  conda-forge
      libintl-devel        0.22.5     hdfe23c8_3  conda-forge
      libunistring         0.9.10     h0d85af4_0  conda-forge
      libzlib              1.3.1      hd23fc13_2  conda-forge
      openssl              3.3.2      hd23fc13_0  conda-forge
      wget                 1.21.4     hca547e6_0  conda-forge
      zlib                 1.3.1      hd23fc13_2  conda-forge
```

How to update/upgrade Mamba packages:

One can also update older software versions:

```
$ mamba update pkgName1 pkgName2 ...
```

To update a single package:

```
$ mamba update wget
```

Update *all* packages in your environment:

```
$ mamba update --all
```

To update/roll-back to a package of a specific version, use `mamba install` instead:

```
$ mamba install wget=1.21.4=hf20ceda_1
```

Remove packages from an environment:

```
$ mamba remove pkgName1 pkgName2 ...
```

> NOTE: To remove software from a virtual environment, you must first `mamba activate` your environment or specify the `--name envName` option.

Which Mamba virtual environments do I have?

It's convenient to organize tools into environments by analysis type (i.e., one for genome assembly tools, another for variant calling tools, and another for RNA-seq analysis, etc.). This, however, can result in many Mamba environments. We can see which environments we have by running:

```
$ mamba env list
```

Other useful commands

Adding channels

There are many sources of software packages that you can install from, which are stored on servers on the web. In Mamba parlance, these source servers are referred to as "channels". Some useful bioinformatics channels can be added to your Mamba configuration like so:

```
$ mamba config append channels conda-forge
$ mamba config append channels bioconda
$ mamba config append channels anaconda
```

> NOTE: With each of the above `mamba` commands, you can specify particular channels without adding them to your Mamba configuration permanently by including the `--channel channelName` option. This option can be specified more than once on the command line.

Reproducible environments

Mamba provides a convenient utility allowing you to export the list of software (and their version and build information) installed in an environment, allowing you to share that environment with others via a compact text file. This is useful when writing your Methods sections, allowing reviewers to run your analyses themselves, increasing reproducibility.

The command below will write a YAML-formatted file called `envName.yaml` containing the information required to reproduce an environment.

```
# Assuming your environment is activated
$ mamba env export >envName.yaml

# If the `envName` environment was not already activated
$ mamba env export --name envName >envName.yaml
```

One can then re-create that environment from the YAML file:

```
$ mamba env create --file envName.yaml --name envName
```

Running a single executable command

To run a single executable installed in the environment without activating the environment, one can use the `mamba run` command:

```
$ mamba run --name envName softwareCommand
```

For example:

```
$ mamba install --name envName samtools
$ mamba run --name envName samtools depth my.bam >my.depth
```

Remove cached temporary files

When Mamba installs software in environments, it downloads and caches TAR archive files containing the software (for each version) installed. After a while, these TAR files can accumulate and occupy many gigabytes of disk space. You can remove these cached TAR files with the `clean` command:

```
$ mamba clean --all
```

For example:

```
$ mamba clean --all

    Collect information..
```

```
Cleaning index cache..
Cleaning lock files..
Package file                                          Size
-------------------------------------------------------------
/Users/username/.micromamba/pkgs
-------------------------------------------------------------

asttokens-2.4.1-pyhd8ed1ab_0.conda                    29kB
bzip2-1.0.8-hfdf4475_7.conda                         134kB
...
tzdata-2024a-h0c530f3_0                              661kB
wheel-0.43.0-pyhd8ed1ab_1                            230kB

/Users/username/.mamba/pkgs
-------------------------------------------------------------

Total size:                                           80MB
Cleaning tarballs..

Remove tarballs: [Y/n]
Package folder                                 Size
-------------------------------------------------------
bzip2-1.0.8-h10d778d_5                        784kB
c-ares-1.28.1-h10d778d_0                      577kB
...
tzdata-2024a-h0c530f3_0                       661kB
wheel-0.43.0-pyhd8ed1ab_1                     230kB

/Users/username/.mamba/pkgs
-------------------------------------------------------
Total size:                                   335MB
```