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 software (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 software (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 is installed by default**.

Below, we'll use `micromamba` to demonstrate the functionality of Mamba.

The basics

How to create an environment:

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

```
$ micromamba create --name envName
# or equivalently
$ micromamba env create --name envName
```

Remove unwanted environments

Similarly, if an environment is no longer wanted or becomes corrupted, it can be deleted:

```
# Must deactivate your environment if activated:
$ micromamba env remove --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`:

```
$ micromamba activate envName
```

How to unload/deactivate an active environment:

When you are finished with your work using the environment and want to return a standard Unix/Linux command-line environment, you can deactivate your current environment:

```
$ micromamba deactivate
```

> *NOTE*: It is unnecessary to specify your environment name when deactivating, as `micromamba` knows which environment you are in and are deactivating.

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.

```
$ micromamba search pkgName
```

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

```
$ micromamba 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 `micromamba activate` your environment or specify the `--name envName` option.

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

# or, when not activated, us the `--name` option:
$ micromamba 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):

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

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

For example:

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

```
$ mm search wget
Getting repodata from channels...

pkgs/main/osx-64                                            No change
pkgs/main/noarch                                            No change
pkgs/r/osx-64                                               No change
pkgs/r/noarch                                               No change
bioconda/osx-64                                   4.3MB @   5.6MB/s  0.8s
bioconda/noarch                                   5.0MB @   5.5MB/s  1.0s
conda-forge/noarch                               22.8MB @  10.7MB/s  2.1s
conda-forge/osx-64                               41.4MB @  19.3MB/s  2.2s

     wget 1.25.0 h3a17b82_0
------------------------------------------

 Name            wget
 Version         1.25.0
 Build           h3a17b82_0
 Size            873 kB
 License         GPL-3.0-or-later
```

```
 Subdir           osx-64
 File Name        wget-1.25.0-h3a17b82_0.conda
 URL              https://repo.anaconda.com/pkgs/main/osx-64/wget-1.25.0-
h3a17b82_0.conda
 MD5              3b1ba76caf798c06766f19fb8fa3554c
 SHA256           f270e4c7665e029ab3dc23c2f8e7330392c1cb16ea49fd02268c8e99733d0e63

 Dependencies:
  - zlib >=1.2.13,<1.3.0a0
  - libiconv >=1.16,<2.0a0
  - openssl >=3.0.15,<4.0a0
  - gettext >=0.21.0,<1.0a0
  - pcre2 >=10.42,<10.43.0a0
  - libidn2 >=2,<3.0a0
  - libunistring >=0,<1.0a0

 Other Versions (8):

  Version Build
 ──────────────────────────────────────────
  1.24.5  h3a17b82_0
  1.21.4  hca547e6_0          (+ 3 builds)
  ...     (4 hidden versions)          ...
  1.19.4  h073198b_0
  1.19.1  hcb5d8a9_0
```

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

```
$ micromamba install wget=1.24.5=h3a17b82_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:
$ micromamba list

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

For example:

```
$ micromamba 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
  ...
  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:

```
$ micromamba update pkgName1 pkgName2 ...
```

To update a single package:

```
$ micromamba update wget
```

Update *all* packages in your environment:

```
$ micromamba update --all
```

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

```
$ micromamba install wget=1.21.4=hf20ceda_1
```

Remove packages from an environment:

```
$ micromamba remove pkgName1 pkgName2 ...
```

> NOTE: To remove software from a virtual environment, you must first `micromamba 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:

```
$ micromamba env list
```

Other useful commands

Viewing and 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". Which channels do you already have?:

```
$ micromamba config list channels
```

Some useful bioinformatics channels can be added to your Mamba configuration like so:

```
$ micromamba config append channels conda-forge  # for general Linux/Unix tools/libraries
$ micromamba config append channels bioconda     # for bioinformatic tools/libraries
$ micromamba config append channels anaconda     # for general data science tools/libraries
$ micromamba config append channels r            # for R software and libraries
```

> NOTE: With each of the above `micromamba` 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
$ micromamba env export >envName.yaml

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

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

```
$ micromamba 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 `micromamba run` command:

```
$ micromamba run --name envName softwareCommand
```

For example:

```
$ micromamba install --name envName samtools
$ micromamba 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:

```
$ micromamba clean --all
```

Update Mamba package manager software

```
$ micromamba self-update
```