

Introduction to Conda

J. Seiler

BiGEst





- Notebooks let you analyze your data and describe your analyses at the same time
- Notebooks support real-time data visualization
- Interactive sections can be included within a notebook
- Notebooks support the principles of reproducibility in analysis

But

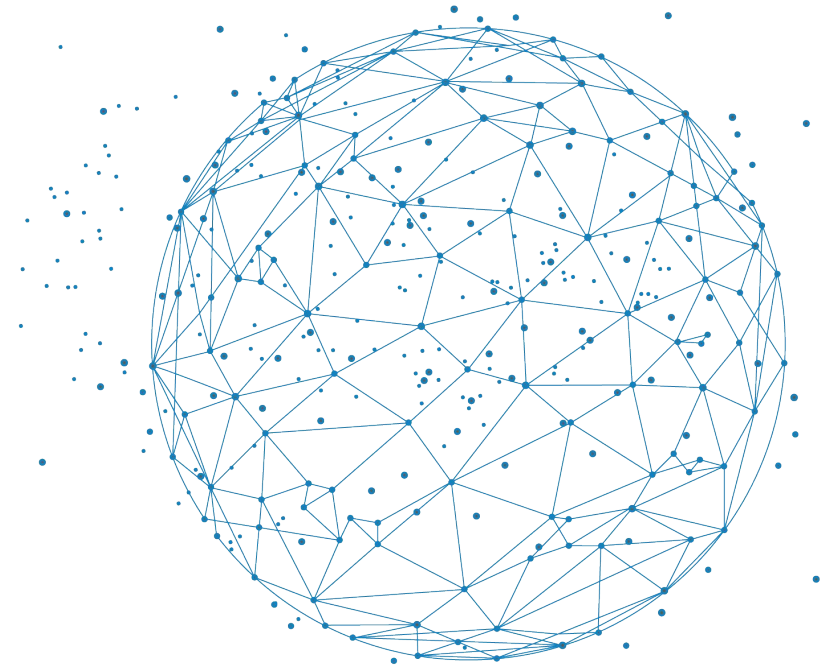
- Notebooks requires a JupyterLab server
- Notebooks don't include external dependencies



Tool	Version
fastqc	0.12.1
hisat2	2.2.1
htseq	0.13.5
samtools	1.15.1
r-base	4.2.3
deseq2	
zenodo_get	
bash_kernel	

During this course we will discover how to use **conda** to simplify the installation of these tools for any users

Installing software is hard





Installing software can be a tricky operation:

Configure

Compile

Install

README License

Building Samtools

See [INSTALL](#) for complete details. [Release tarballs](#) contain generated files that have not been committed to this repository, so building the code from a Git repository requires extra steps:

```
autoheader          # Build config.h.in (this may generate a warning about
                    # AC_CONFIG_SUBDIRS - please ignore it).
autoconf -Wno-syntax # Generate the configure script
./configure          # Needed for choosing optional functionality
make
make install
```

By default, this will build against an HTSlib source tree in `../htslib`. You can alter this to a source tree elsewhere or to a previously-installed HTSlib by configuring with `--with-htslib=DIR`.

<https://github.com/samtools/samtools>

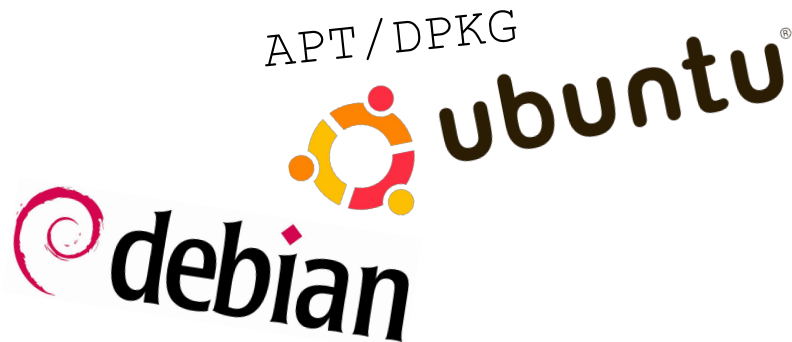
To facilitate software deployment, a number of package systems are available:

- offers pre-compiled software versions
- automatically manages dependencies
- offers updates

Most Linux distributions offer a package system for easy installation of :

However, these package systems have a number of **drawbacks**:

- require administrative rights
- often only one version of a tool can be installed (sometimes without choice)
- poor support of scientific software
- hard to contribute



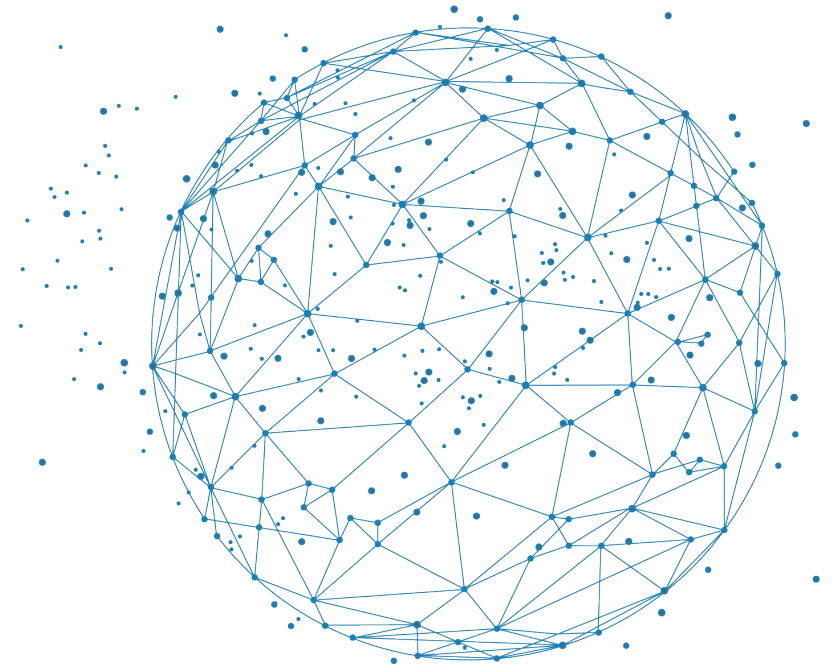
YUM/RPM
Red Hat



Rocky Linux



Introducing Conda





CONDA is a multi-platform package management system designed for users



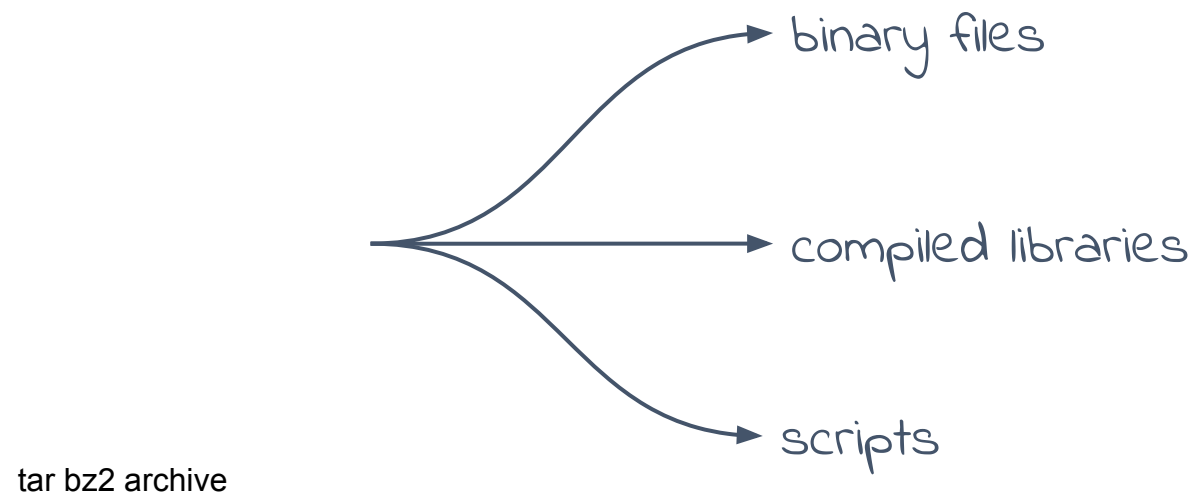
Conda's strengths:

- Does **not** require administrator rights
- Allows you to **choose the version** of the tool you want to install
- Allows **multiple versions of the same tool** to coexist (through environments)
- Many tools in the **scientific field** (bioconda)
- A very **open contribution** model
- Compatible with any **Linux** distribution
- Also compatible with **MacOS** and **Windows** (in theory)



How does it work ?

- Each software and library is available in the form of a package (a tar bz2 archive).
- A package contains the compiled version of the software and the list of packages on which it depends.



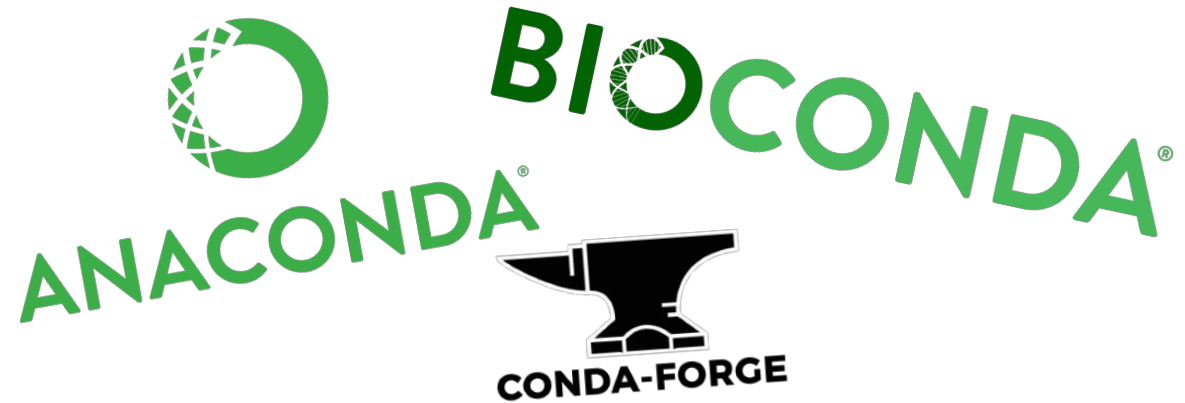


How does it work ?

- The packages are hosted on a central server: **anaconda.org**



Free




Organised by channel



By the way, what is a channel ?

A channel is a set of packages available on the conda repositories and managed by the same organisation.

The creators of conda offer a set of packages in the  **ANACONDA** channel.

For several years, two main channels have grouped together most of the packages:

- **CONDA-FORGE** : for all user tools, languages and libraries
- **BIOCONDA** : for bioinformatics tools and libraries.

There are also numerous channels proposed by conda users. On these channels you can sometimes find tools in more recent versions, **but reliability is not guaranteed.**



How does it work ?

- For a given software version, there are several archives corresponding to the different OS and CPU architectures for which the software is available.

bioconda / packages / samtools ☆ 25

Tools for dealing with SAM, BAM and CRAM files

Conda **Files** Labels Badges

Filters

Type: All ▾ Version: 1.19.1 ▾ Label: All ▾

<input type="checkbox"/>	Type	Size	Name	Uploaded	Downloads	Labels
<input type="checkbox"/>	conda	465.5 kB	osx-64/samtools-1.19.1-hd510865_0.tar.bz2	📅 1 month and 22 days ago	164	main
<input type="checkbox"/>	conda	462.2 kB	linux-64/samtools-1.19.1-h50ea8bc_0.tar.bz2	📅 1 month and 22 days ago	1018	main



How does it work ?

- The **conda** tool allows you to install packages from the **command line**.
- For each package, it determines the list of dependencies (other packages) required, downloads the packages and installs them.

```
(base) $ conda install samtools=1.19.1
```



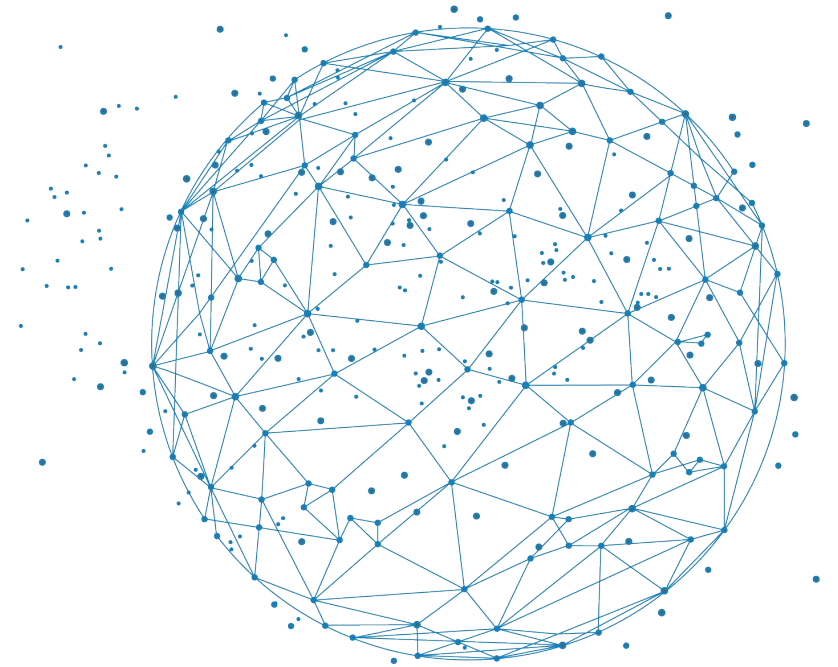
How does it work ?

- The anaconda.org website has a search engine that lets you look for packages across all the channels.

The screenshot shows the Anaconda.org search interface. At the top, there is a search bar containing 'samtools' and a green search button. Below the search bar, there are filter options: 'Type: All', 'Access: All', and 'Platform: All'. The main content is a table of search results. The table has columns for 'Favorites', 'Downloads', 'Artifact (owner / artifact)', and 'Platforms'. The results are as follows:

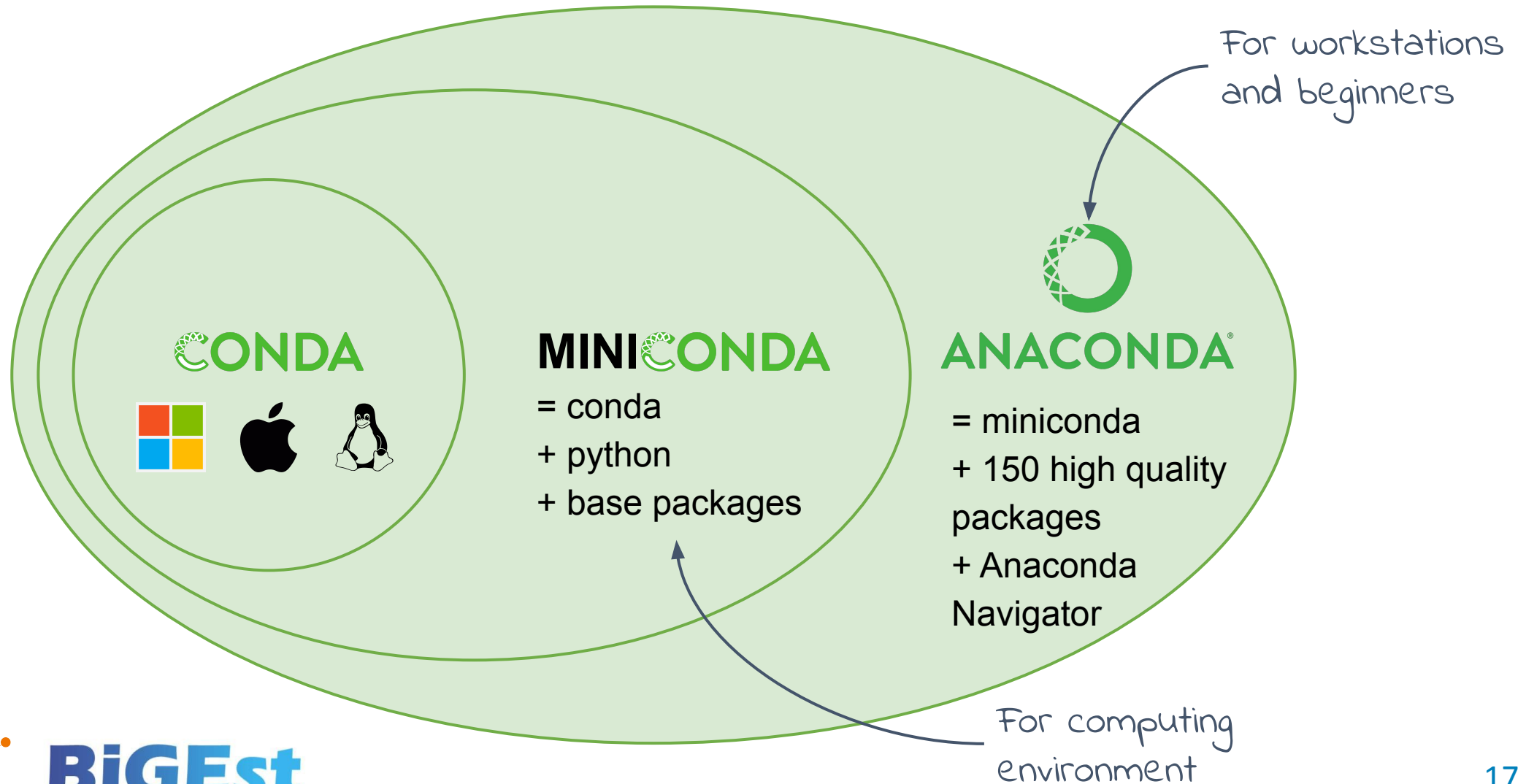
Favorites	Downloads	Artifact (owner / artifact)	Platforms
25	4977181	bioconda / samtools 1.19.2 Tools for dealing with SAM, BAM and CRAM files	linux-64 linux-aarch64 osx-64
2	878734	bioconda / bioconductor-rsamtools 2.18.0 Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import	linux-64 osx-64
3	215435	soil / samtools 1.6 Tools for dealing with SAM, BAM and CRAM files	linux-64 osx-64
0	193215	bioconda / perl-bio-samtools 1.43 Read SAM/BAM files	linux-64 osx-64
0	8097	bioconda / msamtools 1.1.3 microbiome-related extension to samtools	linux-64 osx-64
0	5555	BioBuilds / samtools 1.6.0	linux-64 linux-

Installing Conda





Conda is available in three different flavors





Conda is already available on the IFB Core Cluster so you don't need to install it.

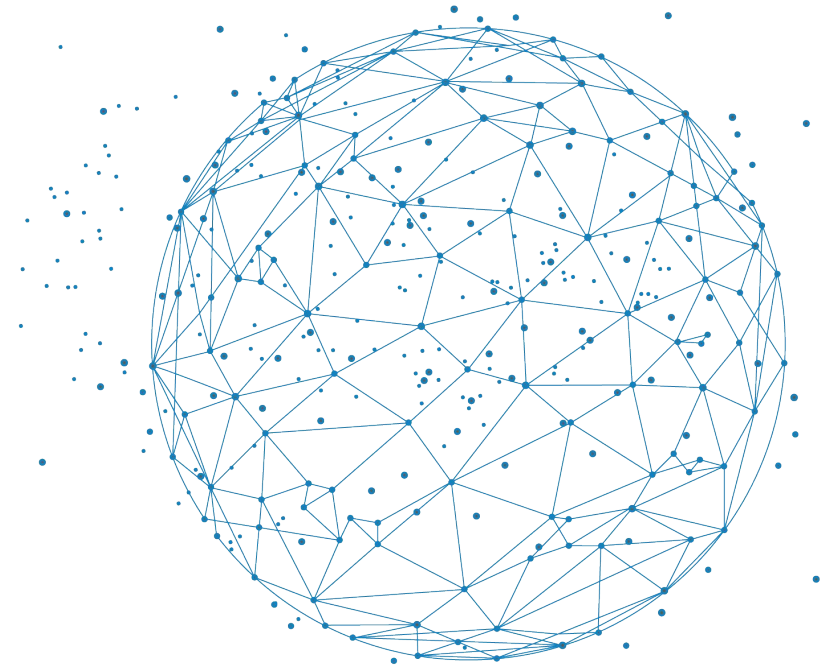
```
$ module load conda  
$ conda --version  
conda 22.9.0
```

Conda can run natively on many operating systems without the need to install special system packages. **Conda can therefore be used as a non-administrator user.**

Similarly, the same installation of Conda can run on several OS. This is the case on the Core cluster where some of the compute nodes are running Centos and others Ubuntu.

Conda also supports OS updates very well.

First steps with Conda





Let's load conda in our shell session :

```
$ module load conda  
$ conda --version  
conda 22.9.0
```



Conda needs to be initialized in your shell if you want to use its full power.

Try the following command to check if conda is already initialized :

```
$ conda activate --help
CommandNotFoundError: Your shell has not been properly configured to use 'conda activate'.
To initialize your shell, run
```

```
$ conda init <SHELL_NAME>
```

Currently supported shells are:

- bash
- fish
- tcsh
- xonsh
- zsh
- powershell

See 'conda init --help' for more information and options.

IMPORTANT: You may need to close and restart your shell after running 'conda init'.

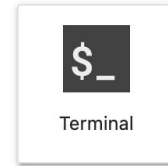


If you get the `CommandNotFoundError` message it means that conda is not initialized for your shell yet

Run :

```
$ conda init bash
```

Now close and relaunch your terminal



You will now see the indication (base) in your prompt :

```
(base) ~$
```

This means that conda has loaded the **base** conda environment by default.



Configuration

Before installing a package, you need to determine which package sources or channels you want to use and their order of priority.

```
(base) $ conda config --add channels defaults
(base) $ conda config --add channels bioconda
(base) $ conda config --add channels conda-forge
(base) $ conda config --set channel_priority strict
(base) $ conda config --set notify_outdated_conda false
```

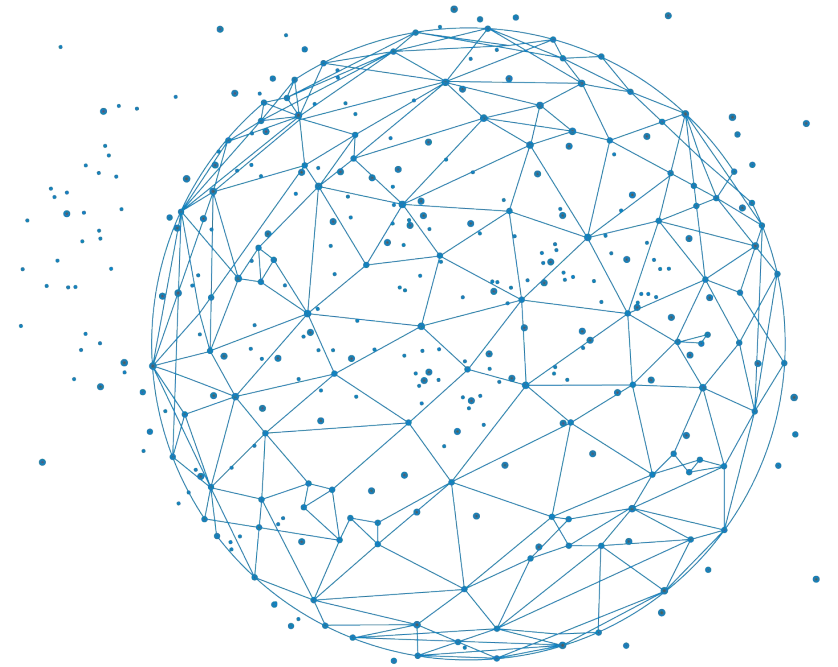
What does it mean ?

Specify the channel order in which conda will search for packages

*Conda will first search the **conda-forge** channel, then **bioconda** and finally **defaults**.*

Ask conda not to inform us if it is out of date (**notify_outdated_conda false**).

Conda environments



With conda, **each piece of software must be installed in an environment.**

An environment is a folder containing all the files needed for the software to work.

This folder looks like **a miniature operating system.**

You can install multiple software in an environment but **only one version of a software in a given environment.**

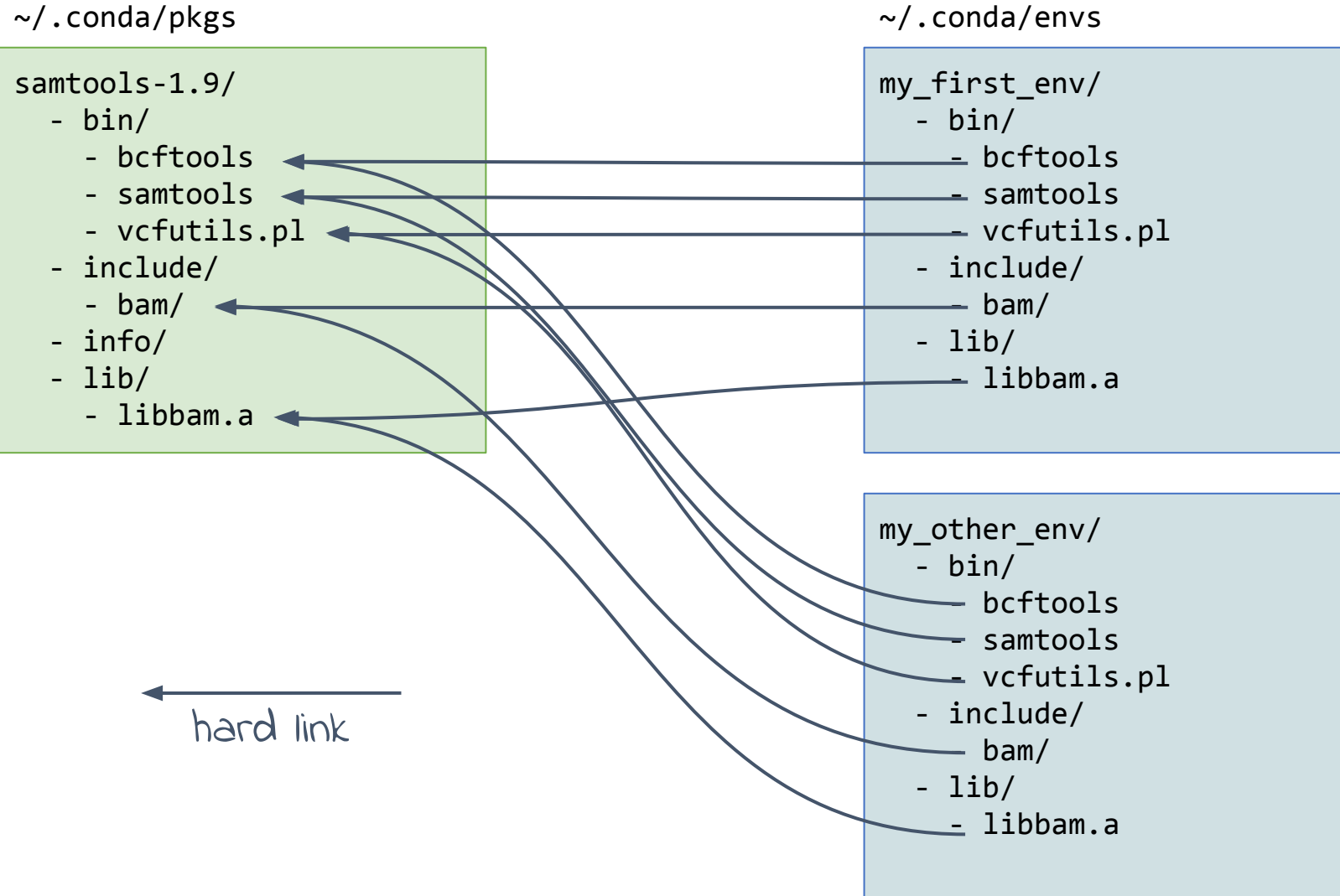
By default, environments are created in the **.conda/envs** folder **in your home directory.**

Each package provides files which are installed in an environment through hard links.

The package files are stored in a central location in order to limit the space occupied by the environments.

This means that if the same package is installed in two different environments, the space occupied by that package will only be used once.

By default, packages are unzipped into the `.conda/pkg` folder in your home directory.





Let's try installing a tool

```
(base) $ conda install fastqc
```



Let's try installing a tool...

```
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

```
## Package Plan ##
```

```
environment location: /shared/ibfstor1/software/miniconda
```

```
added / updated specs:
- fastqc
```

```
The following NEW packages will be INSTALLED:
```

```
fastqc          bioconda/noarch::fastqc-0.12.1-hdfd78af_0 None
font-ttf-dejavu-s~ conda-forge/noarch::font-ttf-dejavu-sans-mono-2.37-hab24e00_0 None
fontconfig      conda-forge/linux-64::fontconfig-2.14.2-h14ed4e7_0 None
freetype        conda-forge/linux-64::freetype-2.12.1-h267a509_2 None
libpng          conda-forge/linux-64::libpng-1.6.43-h2797004_0 None
libuuid         conda-forge/linux-64::libuuid-2.38.1-h0b41bf4_0 None
libxcrypt       conda-forge/linux-64::libxcrypt-4.4.36-hd590300_1 None
openjdk         conda-forge/linux-64::openjdk-8.0.382-hd590300_0 None
perl            conda-forge/linux-64::perl-5.32.1-7_hd590300_perl5 None
```

```
Proceed ([y]/n)?
```

```
Preparing transaction: done
Verifying transaction: failed
```

```
EnvironmentNotWritableError: The current user does not have write permissions to the target environment.
environment location: /shared/ibfstor1/software/miniconda
uid: 100002
gid: 100002
```

what's wrong ?





Let's try installing a tool...

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /shared/ibfstor1/software/miniconda

added / updated specs:
- fastqc

The following NEW packages will be INSTALLED:

fastqc                bioconda/noarch::fastqc-0.12.1-hdfd78af_0 None
font-ttf-dejavu-s~   conda-forge/noarch::font-ttf-dejavu-sans-mono-2.37-hab24e00_0 None
fontconfig           conda-forge/linux-64::fontconfig-2.14.2-h14ed4e7_0 None
freetype             conda-forge/linux-64::freetype-2.12.1-h267a509_2 None
libpng              conda-forge/linux-64::libpng-1.6.43-h2797004_0 None
libuuid            conda-forge/linux-64::libuuid-2.38.1-h0b41bf4_0 None
libxcrypt          conda-forge/linux-64::libxcrypt-4.4.36-hd590300_1 None
openjdk            conda-forge/linux-64::openjdk-8.0.382-hd590300_0 None
perl              conda-forge/linux-64::perl-5.32.1-7_hd590300_perl5 None

Proceed ([y]/n)?

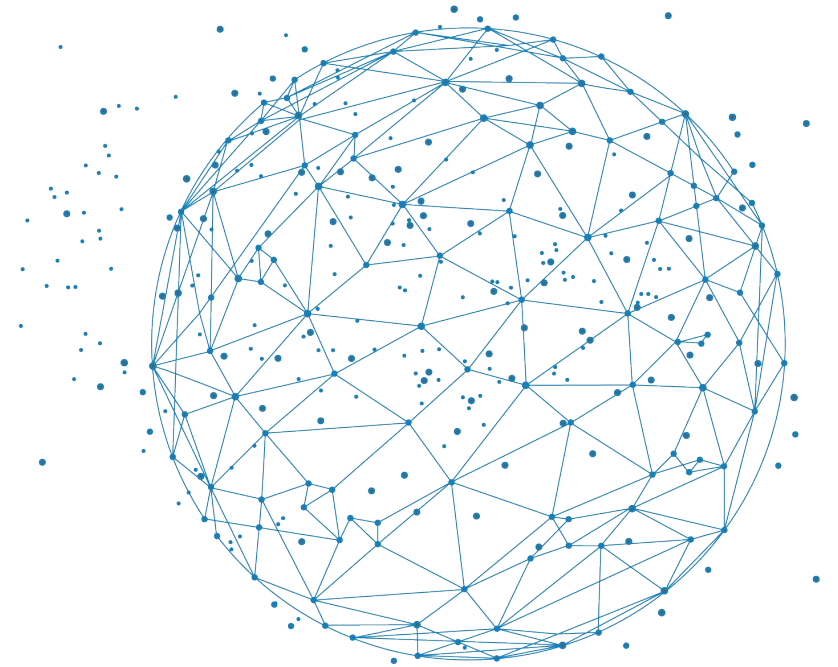
Preparing transaction: done
Verifying transaction: failed

EnvironmentNotWritableError: The current user does not have write permissions to the target environment.
environment location: /shared/ibfstor1/software/miniconda
uid: 10002
gid: 10002
```

Conda tells us that it cannot run the installation because it cannot write to the environment (`/shared/ibfstor1/software/miniconda`).

This is because the base environment has been created by the cluster administrator and cannot be modified by users.

Creating an environment for our pipeline





Let's create an environment for our pipeline:

```
(base) $ conda create -n pipeline
```

The **-n** option is used to specify the name of the environment.

The environment is created in the `~/.conda/envs/pipeline` folder

Alternatively, you can use the **-p** option to indicate a path rather than a name.



We can now activate our new environment:

```
(base) $ conda activate pipeline  
(pipeline) $
```

From now on, you will see the indication **(pipeline)** in your prompt.

Conda has set up environment variables to indicate the current environment. He has also modified your shell's **\$PATH** so that it can give priority access to tools (executables) installed in the environment.



The `conda list` command is used to list the packages installed in the environment

```
(pipeline) $ conda list
# packages in environment at /shared/home/jseiler/.conda/envs/pipeline:
#
# Name                Version                Build Channel
```

For the moment there are none.



Let's install the first tool we need

```
(pipeline) $ conda install fastqc=0.12.1
```

Creating an environment for our pipeline



```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /shared/home/jseiler/.conda/envs/pipeline
  added / updated specs:
    - fastqc=0.12.1

The following packages will be downloaded:

package | build | size | channel
-----|-----|-----|-----
graphite2-1.3.13 | h59595ed_1003 | 95 KB | conda-forge
libdeflate-1.20 | h590300_0 | 70 KB | conda-forge
libtiff-4.6.0 | h16d3f0c_3 | 276 KB | conda-forge
Total: 440 KB

The following NEW packages will be INSTALLED:

_libgcc_mutex | conda-forge/linux-64::_libgcc_mutex-0.1-conda_forge None
_openssl_mutex | conda-forge/linux-64::_openssl_mutex-1.1-0-conda_forge None
alibaba | conda-forge/linux-64::alibaba-1.2-1-h590300_1 None
bzip2 | conda-forge/linux-64::bzip2-1.0.8-h590300_5 None
ca-certificates | conda-forge/linux-64::ca-certificates-2024.2.2-hbc0c5d_0 None
cairo | conda-forge/linux-64::cairo-1.18.0-h3fae72a_0 None
expat | conda-forge/linux-64::expat-2.6.2-h59595ed_0 None

fastqc | bioconda/noarch::fastqc-0.12.1-hdfd78af_0 None
font-ttf-dejavu-sans | conda-forge/noarch::font-ttf-dejavu-sans-condensed-2.37-hub24e0_0 None
font-ttf-inconsolata | conda-forge/noarch::font-ttf-inconsolata-3.000-h77eed37_0 None
font-ttf-source-code-pro | conda-forge/noarch::font-ttf-source-code-pro-2.038-h77eed37_0 None
font-ttf-ubuntu | conda-forge/noarch::font-ttf-ubuntu-0.83-h77eed37_1 None
fontconfig | conda-forge/linux-64::fontconfig-2.14.2-h14ed4e7_0 None
fonttools | conda-forge/noarch::fonttools-4.53.0-0-conda-forge None
freefont | conda-forge/linux-64::freefont-2.12.1-h267a509_2 None
giflib | conda-forge/linux-64::giflib-5.2.1-h0b41bf4_3 None
graphite2 | conda-forge/linux-64::graphite2-1.3.13-h59595ed_1003 None
harfbuzz | conda-forge/linux-64::harfbuzz-8.3.0-h3d44e6_0 None
icu | conda-forge/linux-64::icu-73.2-h59595ed_0 None
keyutils | conda-forge/linux-64::keyutils-1.6.1-h166bdaf_0 None
lcms2 | conda-forge/linux-64::lcms2-2.16-hb7c19ff_0 None
lerc | conda-forge/linux-64::lerc-4.0.0-h27087fc_0 None
libcup | conda-forge/linux-64::libcup-2.3-h46170d4_4 None
libdeflate | conda-forge/linux-64::libdeflate-1.20-h590300_0 None
libedit | conda-forge/linux-64::libedit-3.1.20191231-he2ba2c2_2 None
libexpat | conda-forge/linux-64::libexpat-2.6.2-h59595ed_0 None
libffi | conda-forge/linux-64::libffi-3.4.2-h7f98852_5 None
libgcc-ng | conda-forge/linux-64::libgcc-ng-13.2.0-h8b7b88a_5 None
libglib | conda-forge/linux-64::libglib-2.80.0-hf2295e7_1 None
libgomp | conda-forge/linux-64::libgomp-13.2.0-h8b7b88a_5 None
libiconv | conda-forge/linux-64::libiconv-1.17-h590300_2 None
libjpeg-turbo | conda-forge/linux-64::libjpeg-turbo-3.0.0-h590300_1 None
libng | conda-forge/linux-64::libng-1.6.43-h2797064_0 None
libstdcxx-ng | conda-forge/linux-64::libstdcxx-ng-13.2.0-h7e041cc_5 None
libtiff | conda-forge/linux-64::libtiff-4.6.0-h16d3f0c_3 None
libuuid | conda-forge/linux-64::libuuid-2.38.1-h0b41bf4_0 None
libwebp-base | conda-forge/linux-64::libwebp-base-1.3.2-h590300_0 None
libxcb | conda-forge/linux-64::libxcb-1.15-h0b41bf4_0 None
libxcrypt | conda-forge/linux-64::libxcrypt-4.4.36-h590300_1 None
libxml | conda-forge/linux-64::libxml-2.13-h590300_5 None
ncurses | conda-forge/linux-64::ncurses-6.4.20240210-h59595ed_0 None
openjdk | conda-forge/linux-64::openjdk-21.0.2-haa376d0_0 None
openssl | conda-forge/linux-64::openssl-3.2.1-h590300_1 None
pcre2 | conda-forge/linux-64::pcre2-10.43-hca000b1_0 None
perl | conda-forge/linux-64::perl-5.32.1-h590300_perl5 None
pixman | conda-forge/linux-64::pixman-0.43.2-h59595ed_0 None
pthread-stubs | conda-forge/linux-64::pthread-stubs-0.4-h0c2e6d0_1001 None
xorg-fixesproto | conda-forge/linux-64::xorg-fixesproto-5.0-h7f98852_1002 None
xorg-inputproto | conda-forge/linux-64::xorg-inputproto-2.3.2-h7f98852_1002 None
xorg-rgbproto | conda-forge/linux-64::xorg-rgbproto-1.0.7-h7f98852_1002 None
xorg-libice | conda-forge/linux-64::xorg-libice-1.1.1-h590300_0 None
xorg-libsm | conda-forge/linux-64::xorg-libsm-1.2.4-h7931055_0 None
xorg-libt1 | conda-forge/linux-64::xorg-libt1-1.8.7-h8e461d_0 None
xorg-libxau | conda-forge/linux-64::xorg-libxau-1.0.11-h590300_0 None
xorg-libxdmcp | conda-forge/linux-64::xorg-libxdmcp-1.3.0-h7f98852_0 None
xorg-libxext | conda-forge/linux-64::xorg-libxext-1.3.4-h0b41bf4_2 None
xorg-libxfixes | conda-forge/linux-64::xorg-libxfixes-5.0.3-h7f98852_1004 None
xorg-libxi | conda-forge/linux-64::xorg-libxi-1.7.10-h7f98852_0 None
xorg-libxrender | conda-forge/linux-64::xorg-libxrender-0.9.11-h590300_0 None
xorg-libxt | conda-forge/linux-64::xorg-libxt-1.3.0-h590300_1 None
xorg-libxtst | conda-forge/linux-64::xorg-libxtst-1.2.3-h7f98852_1002 None
xorg-recordproto | conda-forge/linux-64::xorg-recordproto-1.14.2-h7f98852_1002 None
xorg-renderproto | conda-forge/linux-64::xorg-renderproto-0.11.1-h7f98852_1002 None
xorg-xextproto | conda-forge/linux-64::xorg-xextproto-7.3.0-h0b41bf4_1003 None
xorg-xproto | conda-forge/linux-64::xorg-xproto-7.3.31-h7f98852_1007 None
xz | conda-forge/linux-64::xz-5.2.6-h166bdaf_0 None
zlib | conda-forge/linux-64::zlib-1.2.13-h590300_5 None
zstd | conda-forge/linux-64::zstd-1.5.5-hfc522b1_0 None

Proceed [y/n]?
```

Conda proposes a large list of packages to install.

These are all dependencies required to run **fastqc**.

For each package it indicates the source channel and the version it will deploy.

We can see that it has found **fastqc** in the **bioconda** channel but that most of the dependencies come from the **conda-forge** channel.

The **bioconda** channel offers tools specifically for bioinformatics.

The **conda-forge** channel offers genetic tools and libraries.



fastqc is now available

```
(pipeline)$ fastqc --version  
FastQC v0.12.1
```



Now let's install hisat2 version 2.2.1

We can explicitly specify the channel we want to use to speed up dependency resolution

```
(pipeline)$ conda install -c bioconda hisat2=2.2.1
```

Solving dependencies takes a loooooooong time.

Conda's dependencies solver algorithm is **notoriously inefficient**.

The conda solver will make several attempts to find a solution to match the dependencies of `fastqc` and `hisat2`.



What is dependency resolution?

Each conda package is dependent on other packages. These may be libraries (openssl, jpeg, etc.), interpreters (python, java, perl, etc.) or tools (bzip2, pip, etc.).

Each of these dependencies may itself require a number of sub-dependencies.

A dependency tree can therefore be created for each package.

For example, we can consult the direct dependencies of the **fastqc** tool using the **conda search** command:

```
(pipeline)$ conda search fastqc=0.12.1 --info
```

```
(pipeline)$ conda search hisat2=2.2.1 --info
```




```
hisat2 2.2.1 hdbdd923_6
-----
file name   : hisat2-2.2.1-hdbdd923_6.tar.bz2
name        : hisat2
version     : 2.2.1
build       : hdbdd923_6
build number: 6
size        : 16.2 MB
license     : GPL-3.0
subdir      : linux-64
url         :
https://conda.anaconda.org/bioconda/linux-64/hisat2
-2.2.1-hdbdd923_6.tar.bz2
md5         : 98bb9f67fee33c105f41a9f86c3f8c23
timestamp   : 2023-05-16 08:44:56 UTC
dependencies:
  - libgcc-ng >=12
  - libstdcxx-ng >=12
  - perl
  - python >3.5
```

← other conda packages

We can see that **several revisions** have been proposed **for the same version of hisat2**.

Revision 6 has been installed in our environment.

Dependencies are not defined precisely.

A tool can therefore work with several different versions of a library or tool. This allows the solver to adjust the dependency versions in order to find a dependency tree common to several tools.

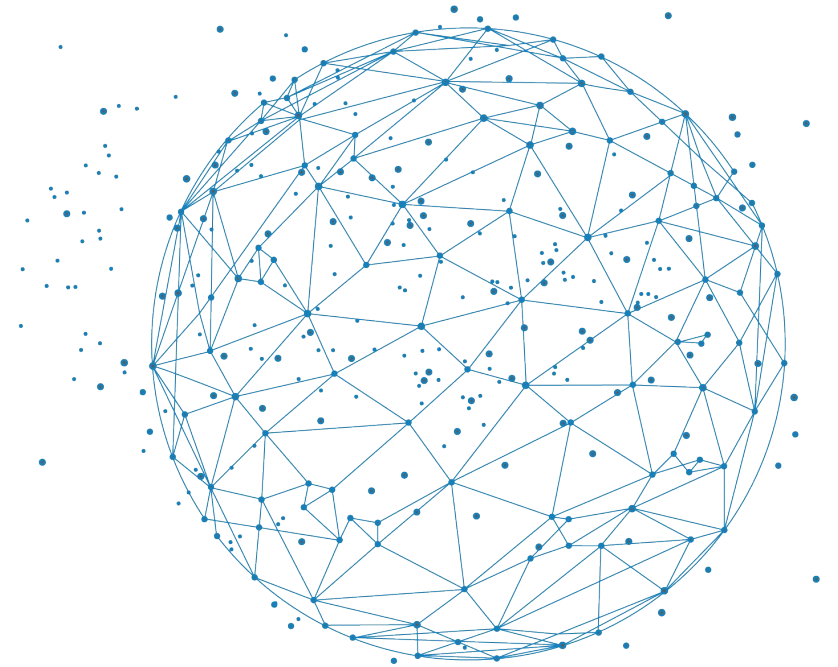


We can use the `conda remove` command to uninstall a package from our environment:

```
(pipeline)$ conda remove hisat2
```

Conda resolves dependencies to ensure that the environment is as up-to-date as possible.

mamba, the super powered solver





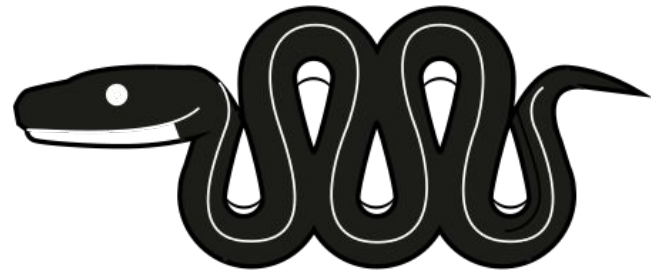
A faster solver as been developed for conda : libmamba

*This library has been developed by a french company called **QuantStack***

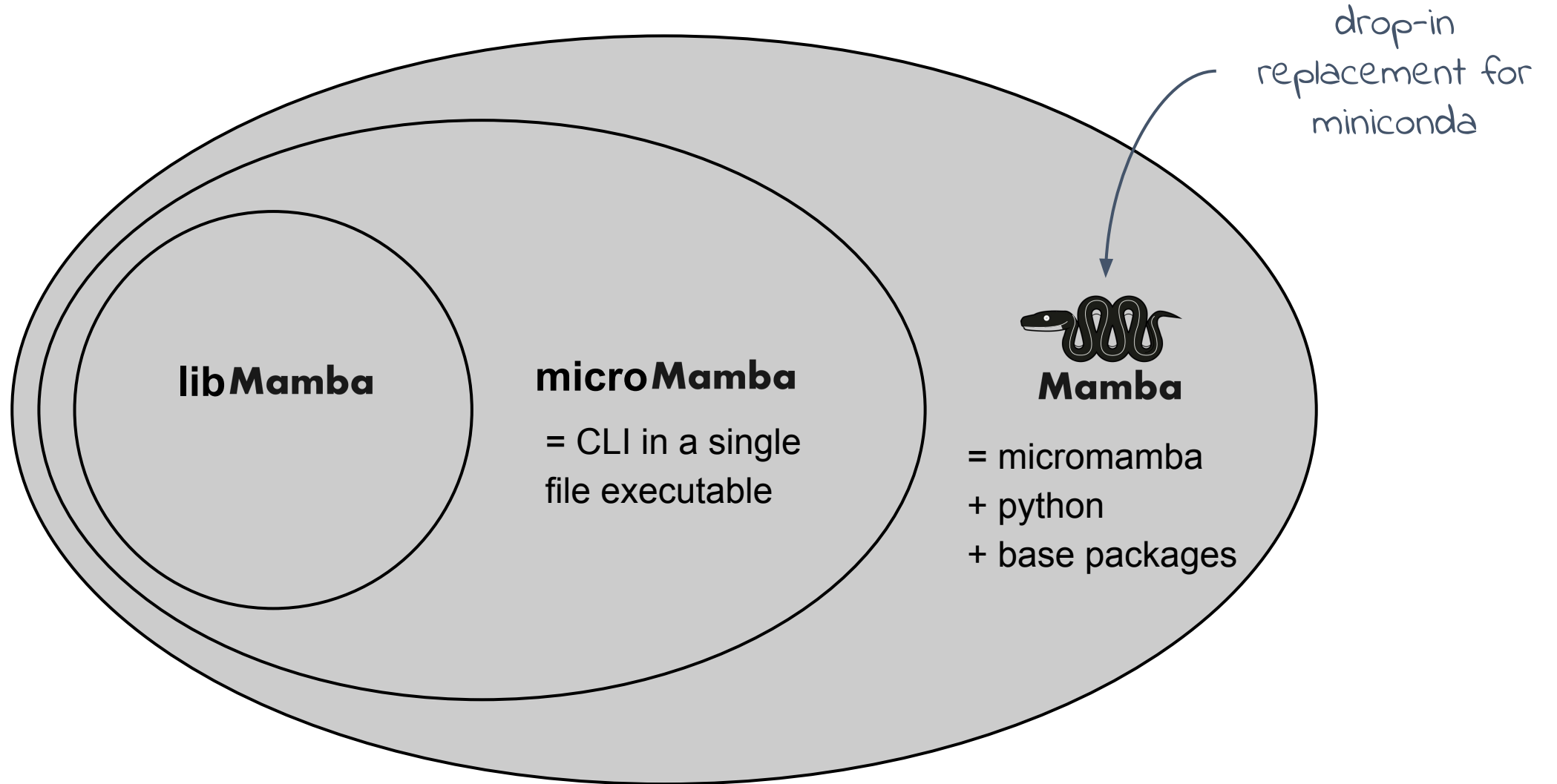


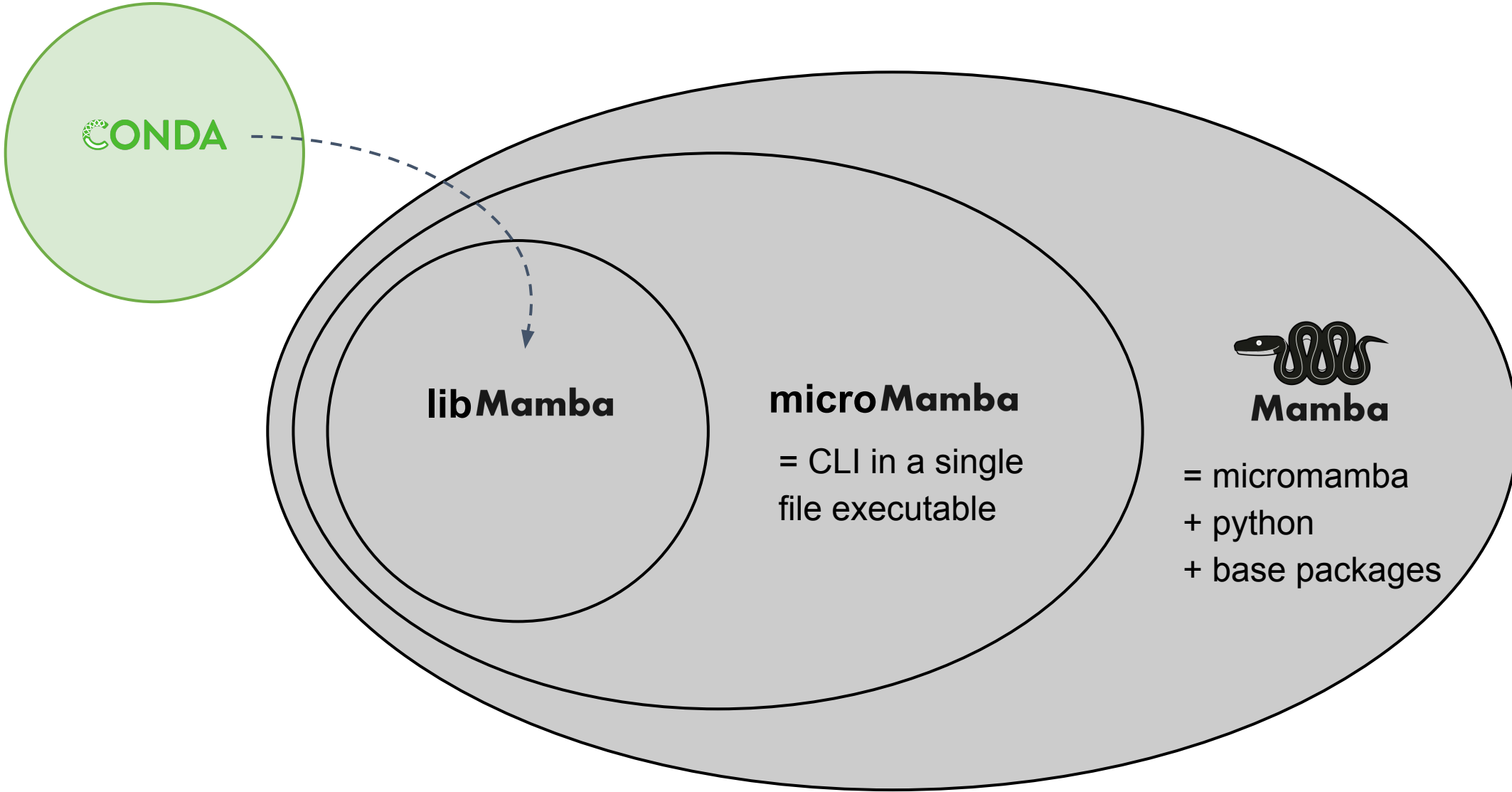
A new package manager based on libmamba is now available : mamba

It uses the same channels as conda.



Mamba







Let's install hisat2 again using the mamba solver.

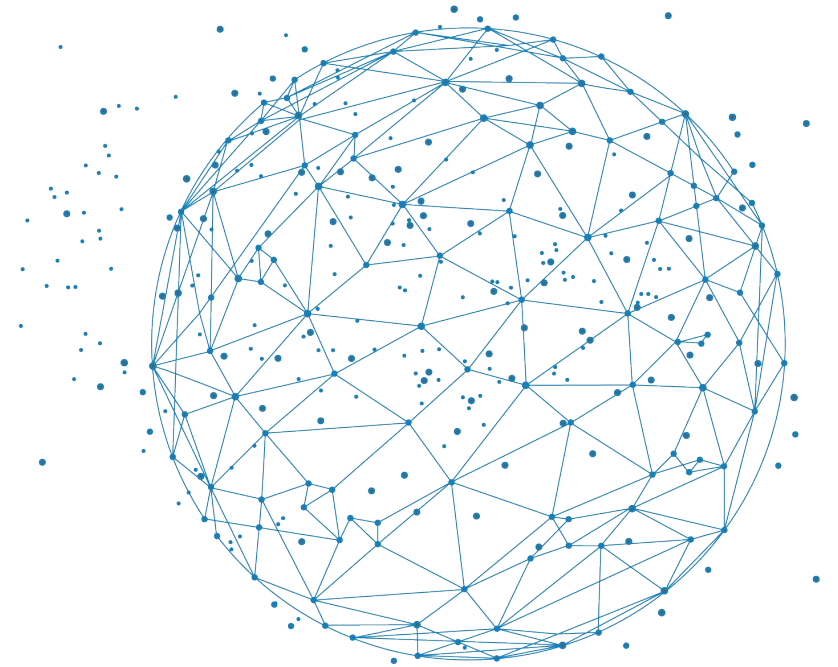
```
(pipeline)$ conda install --experimental-solver=libmamba -c bioconda hisat2=2.2.1
```

In the latest conda version, libmamba is no more considered as “experimental”:

conda install --solver=libmamba

Solving should be much faster !

Sharing your environment



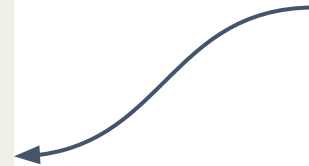


We can't share our environment directly, which is an assembly of hard links, but we can describe its dependencies in a YAML file.

Create a new file called `environment.yml` :

```
name: pipeline
channels:
  - conda-forge
  - bioconda
  - defaults
dependencies:
  - fastqc
  - hisat2
  - htseq
  - samtools
  - r-base
  - bioconductor-deseq2
  - zenodo_get
  - bash_kernel
```

This file gives the direct dependencies of our pipeline. It does not indicate the libraries and other tools required, i.e. the dependencies of dependencies.





We can easily deploy our conda environment with this file

```
(pipeline)$ conda deactivate  
(base)$ conda env remove -n pipeline  
(base)$ conda env create --experimental-solver=libmamba -f environment.yml
```



If we install this environment today, conda will install the requested tools in the following versions:

- fastqc: 0.12.1
- hisat2: 2.2.1
- htseq: 2.0.5
- samtools: 1.19.2

If we need to recreate the environment in a few months' time, it is possible that conda will propose us new versions of certain tools.

So we have an environment that is **easy to update** but **not reproducible**.



To improve reproducibility, you can specify the desired versions for each direct dependency:

```
name: pipeline
channels:
  - conda-forge
  - bioconda
  - defaults
dependencies:
  - fastqc=0.12.1
  - hisat2=2.2.1
  - htseq=2.0.5
  - samtools=1.19.2
  - r-base=4.2.3
  - bioconductor-deseq2
  - zenodo_get
  - bash_kernel
```

We have made an improvement in terms of **reproducibility**, however, when we install this environment a number of additional dependencies will be installed without us having fine control over their versions.

These dependencies can therefore change over the course of the installations and have an impact on the result of our workflow.



Conda can be used to generate a YAML file from the contents of an environment.

```
(pipeline)$ conda env export > environment.lock.yml
```

This time we have a complete and precise list of dependencies. It is **highly reproducible**.

However, we lose our ability to easily **update our direct dependencies**.

By changing the version of one of our direct dependencies, it is highly likely that conda will no longer be able to find a functional dependency tree.



In brief

Dependencies specification	Reproducibility	Upgradability
Direct	Awful	Automatic
Versionned direct	Okish	Good
Through env export	Perfect	Awful

In order to ensure perfect reproducibility of our environment, while at the same time being able to upgrade it if necessary, we need to keep two files:

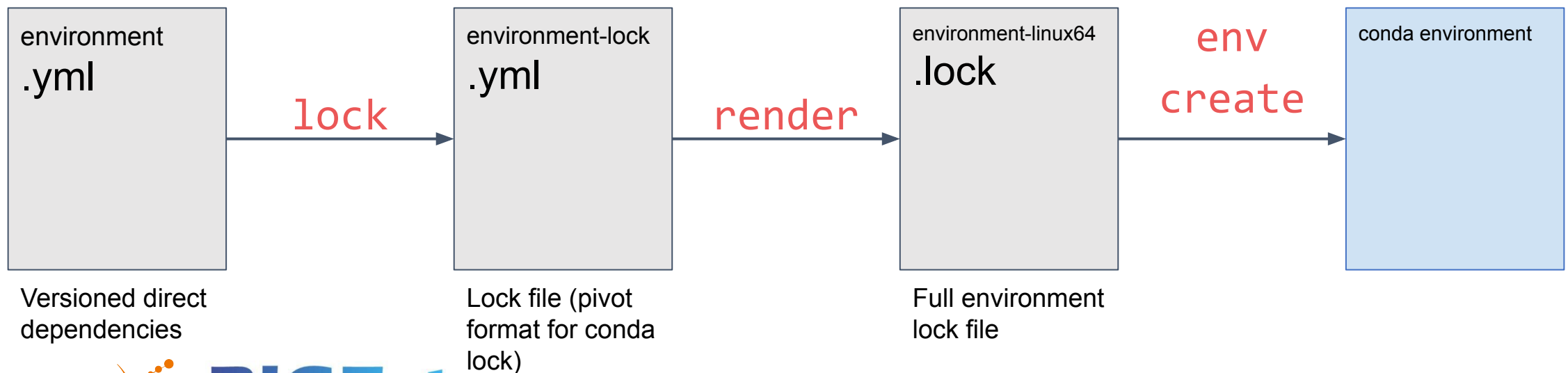
- a file with direct versioned dependencies
- an environment lock file to create the identical environment each time we want to install it



The `conda lock` tool generates an environment lock file from a conda environment file in YAML format.

Whereas `conda env export` creates a precise list of dependencies, `conda lock` creates a list of package URLs to download. This avoids the dependency resolution stage when you want to install the environment.

In addition, `conda lock` can be used to generate lock files for **any operating system**.





Create a lock file for linux:

```
(base)$ conda lock lock --mamba -p linux-64 --lockfile environment-lock.yml -f environment.yml
```

Not a mistype ;)

Render a lock environment file for linux:

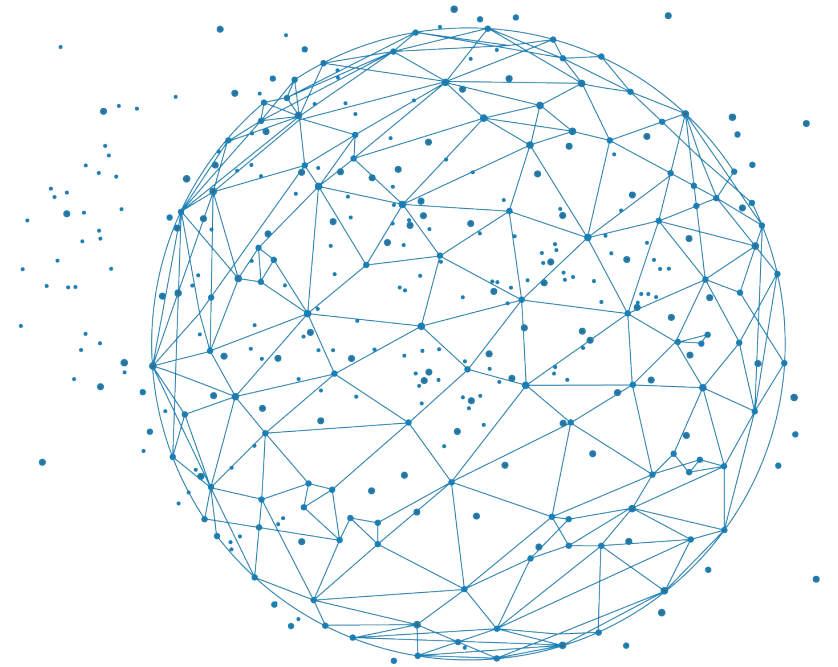
```
(base)$ conda lock render -p linux-64 --filename-template environment-{platform}.lock environment-lock.yml
```

Create the environment:

```
(base)$ conda env remove -n pipeline  
(base)$ conda create -n pipeline -f environment-linux-64.lock
```

conda doesn't need to do any solving !

About package building



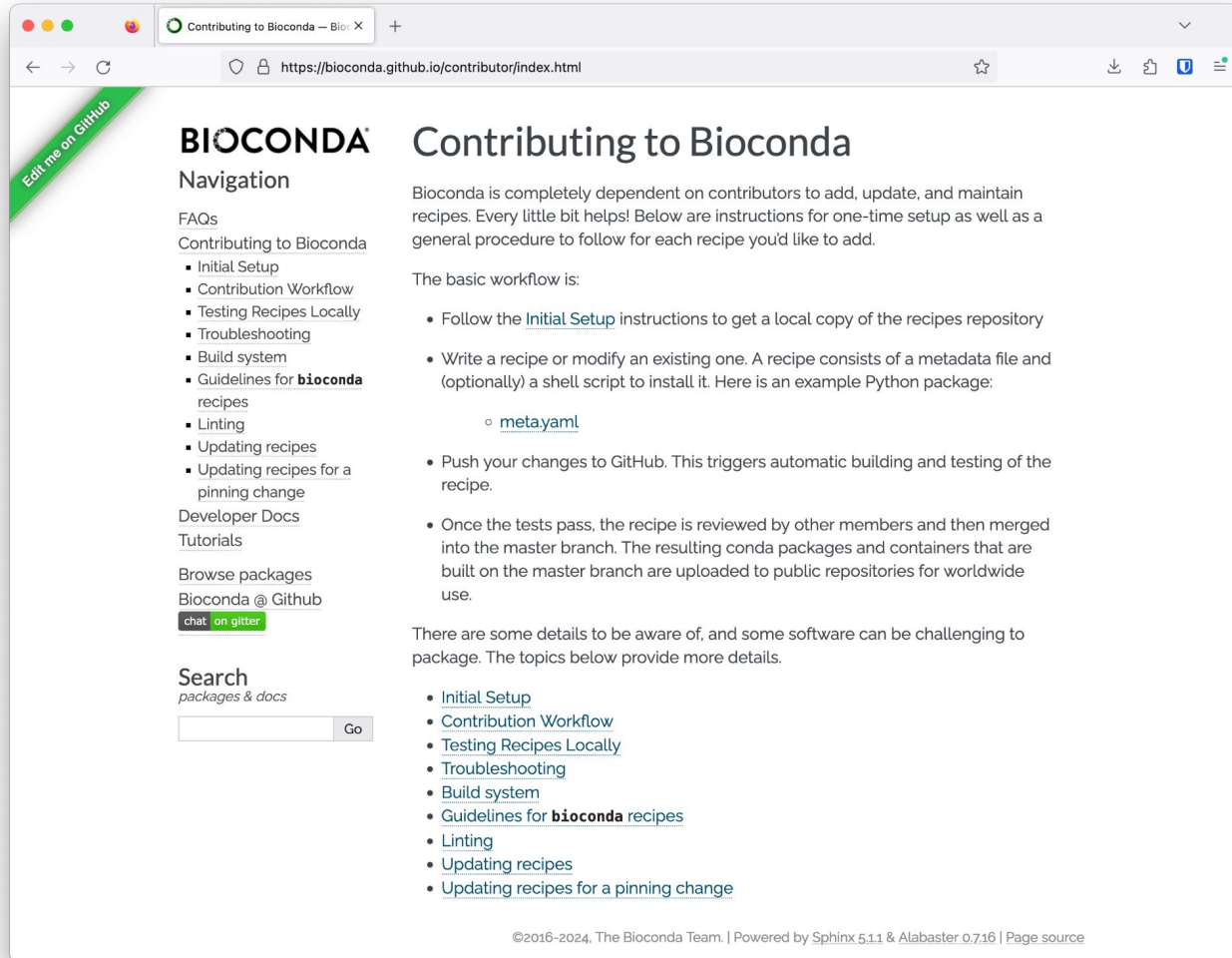


Creating a package is “easy”

1. Create a meta.yml file to describe your package and its dependencies
2. Create a build.sh script to describe the compilation workflow

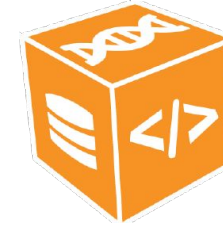
`conda build` let you build your package and publish it

Contribute to **BIOCONDA**



The screenshot shows a web browser window with the URL <https://bioconda.github.io/contributor/index.html>. The page title is "Contributing to Bioconda". It features a navigation menu on the left with links for "Navigation", "FAQs", "Contributing to Bioconda" (with sub-links for Initial Setup, Contribution Workflow, Testing Recipes Locally, Troubleshooting, Build system, and Guidelines for bioconda recipes), "Developer Docs", "Tutorials", "Browse packages", and "Bioconda @ Github". A search bar is located at the bottom left. The main content area contains the heading "Contributing to Bioconda" and a paragraph explaining that Bioconda is dependent on contributors. It lists the basic workflow: 1. Follow Initial Setup instructions. 2. Write a recipe or modify an existing one. 3. Push changes to GitHub. 4. Once tests pass, the recipe is reviewed and merged. A list of links for more details is provided at the bottom of the page.

BIOCONDA®



BIOCONTAINER



<https://bioconda.github.io/contributor/>