Conda

Important Notice: We ask you to avoid using the `Miniconda3` modules, as the `defaults` channel appears to be no longer freely available under Anaconda's updated licensing terms. Please switch to the `Miniforge3` module, which provides access to free and open-source channels like `conda-forge`. Please update your environments accordingly to ensure unrestricted access to packages. Additionally, please check your `~/.condarc` file and remove the `defaults` channel if it is still in use. You can also do this by running: `conda config --env --remove channels defaults`

Conda is a package management system which was initially created for Python, but currently also supports several other languages such as R or lua. Conda can be used to quickly find, install and update packages and their dependencies using community-maintained remote repositories (also called channels). Software packages and scientific libraries are installed in "environments" to provide the ability to maintain different, often incompatible, sets of software. Environments are also managed by Conda.

Please note: Compared to the software modules that we provide on the cluster, there are much more and newer libraries available via Conda that you can manage yourself, but they may not be as well optimized for the processor architectures on the cluster. In addition, by default, Conda installs packages in a user's home directory $HOME), which has a quota for the size and number of files. You should take care to re-set the Conda installation directory to a subdirectory of your group's $SOFTWARE directory. Otherwise, you'll probably quite quickly run into your quota in $HOME, with inconvenient consequences (like not being able to use graphical logins any more before you remove the extraneous files, possibly erroneously also deleting files that should remain).

Please note: It is not recommended to mix cluster software modules and Conda-managed software in the same work environment.

Please note: Pre-configured conda environments are available on the cluster, providing a range of packages commonly used across scientific fields. Please refer to the Pre-Configured Conda Environments section for details on accessing and using these resources.

In this section we explain how to use Conda in the cluster.

Conda usage on the cluster

On the cluster, Conda is available through the Miniforge installation, which provides default access to the free `conda-forge` channel. This installation includes only the Conda package manager, Python, and a minimal set of additional packages. The Miniforge installer also includes the *mamba* library, which offers fast package dependency resolution and speeds up package downloads by using parallel processing through multi-threading. For more details, see below.

Creating a Conda environment

In order to create and use Conda environments on the cluster, you first need to load the Miniforge3 module:

```
module load Miniforge3
```

Please note: Since loading the Miniforge module will also initialize conda for shell interaction, you do not need to additionally run the `conda init <shell>` command modifying your shell init file, e.g. `$HOME/.bashrc`. In fact, you should avoid doing so, as it may cause issues with your interactive or batch jobs. If you ran the command in the past, you may have conda entries in your shell init file. Please edit the file and remove the lines between `# >>> conda initialize >>>` and `# <<< conda initialize <<<`.

If you use Conda-installed packages, the Miniforge3 module should be the only module you load in your work environment or in your job.

The following will create a conda environment named `myenv` and install the packages `python` and `numpy` into it:

```
conda create -n myenv python numpy
```

Note that if you have loaded the `Miniforge3/23.11.0-0` module, Python 3.10.x will be installed, unless you explicitly specify the python version above. For example, to create a Python 3.8 environment, use the following command:

```
conda create -n myenvpy38 python=3.8
```

Your conda environments by default are located in your home directory under the `$HOME/.conda/envs` path. We recommend to change this location. Either use the `--prefix` flag specifying the path to the directory you want (probably `$SOFTWARE/<nameofmycondadir>`), or assign the path to the environment variable `$CONDA_ENVS_PATH`. Another way is to edit your `$HOME/.condarc` file and define your conda environment locations using the key `envs_dirs:`.

Packages that can be installed using conda are provided in channels. Popular channels are Conda Forge and Bioconda, which are set by default on the cluster. If you want to install packages from a channel which is not among the default channels, you can specity it using the `--channel <your-channel>` (or shortly `-n <your-channel>`) flag during package installation. Alternatively, you can add it permanently to your `$HOME/.condarc` file under the `channels:` key. This can also be done by running the command: `conda config --add channels <your-channel>`.

The default conda settings are defined in the cluster-wide configuration file `$EBROOTMINIFORGE3/.condarc`

```
auto_activate_base: false

envs_dirs:
  - $HOME/.conda/envs

pkgs_dirs:
```

```
  - $HOME/.conda/pkgs

channels:
  - conda-forge
  - bioconda
```

The file also defines the default location of conda's package cache directory (key `pkgs_dirs:`). The `$CONDA_PKGS_DIRS` environment variable overwrites the `pkgs_dirs:` setting. The command `conda config --show-sources` displays all identified configuration sources.

If you want to remove packages that are not used in any environment, run:

```
conda clean --all
```

Execute `conda info` to see your current conda settings, including the default channel URLs and the location of your conda environments.

The command lists all your environments:

```
conda info --envs
```

The active conda environment is marked with an asterisk (*).

To search for a package in configured channels, use the command:

```
conda search <package-name>
```

If you want to install additional packages in an existing environment, e.g. in `myenv`, it must first be activated:

```
conda activate myenv
```

Once a conda environment is activated, which may be recognized by the presence of the environment name `(myenv)$` on the command prompt, you can install additional software using either Conda or Pip. For example, the following installs `pandas`, `matplotlib` and a specific version of `scipy`:

```
(myenv)$ conda install scipy=1.6.3 pandas matplotlib
```

Note that it is strongly advised not to use conda for package installations after having installed packages via pip, as this may cause inconsistencies in the environment. For best practices, install packages using conda first, and reserve pip for any final installations.

Note that package installation may be done on both the cluster login servers as well as on the compute nodes, at least when using the default conda channels.

Should you need to see all packages installed in the environment `myenv` then run:

```
conda list -n myenv
```

Without the `-n` option, packages in the current active environment are listed.

A faster solver for Conda

The default package dependency solver of Conda is known to be slow or even fail to resolve some environments. If this is your case, you may try an alternative, faster solver for Conda using the `--solver` flag as follows:

```
conda create -n myconda --solver=libmamba package1 package2 ...
```

The option `--solver` is also available for `conda install|remove|update` commands. If you want to enable the `libmamba` solver permanently, either add `solver:libmamba` to your `$HOME/.condarc` file or run the command:

```
conda config --set solver libmamba
```

To revert back to the default `classic` solver:

```
conda config --remove-key solver
```

Using your Conda environments

To use applications from your conda environments interactively or in a job script, you first need to load the `Miniforge3` module and then activate the environment containing the applications:

```
module load Miniforge3
conda activate myenv
```

Note that we do **not** recommend putting the above lines in your shell init file, e.g. `~/.bashrc`. This may cause issues with your interactive or batch jobs.

Here is a sample job script to run an application from the conda environment:

conda-app-job.sh

```
#!/bin/bash -l
#SBATCH --job-name=my-conda-application
#SBATCH --nodes=1
#SBATCH --cpus-per-task=20
#SBATCH --mem=60G
#SBATCH --time=00:30:00
#SBATCH --mail-user=user@yourinstitute.uni-hannover.de
#SBATCH --mail-type=END

# Activate your conda environment
module load Miniforge3
conda activate <your_conda_env_name>

# Run app
```

```
<run your application>
```

As already mentioned, if you use conda managed software, you should not mix it with the cluster software modules, thus loading only the `Miniforge3` module in your interactive shell or in a job script.

Pre-Configured Conda Environments

To streamline workflows and minimize redundant installations, several pre-configured conda environments are available to all users on the cluster. These shared environments include a broad selection of packages commonly used across various scientific disciplines, such as data analysis, machine learning, bioinformatics, and more.

Using these environments helps conserve resources and ensure consistent setups, allowing you to quickly get started without needing to install the same packages individually.

Below is a list of environments and some primary packages they contain:

Path to Conda Environment	**Key Packages**
`/sw/system/home/ood/conda/envs/miniforge3-2025.10_python-3.13`	numpy, scipy, pandas, scikit-(learn|image|bio), biopython, plotly, matplotlib, seaborn, pytorch, tensorflow, keras, r-base, r-essentials
`/sw/system/home/ood/conda/envs/miniforge3-2024.08_python-3.12`	numpy, scipy, pandas, scikit-(learn|bio), biopython, sqlalchemy, pymol, matplotlib, seaborn, pytorch, tensorflow, keras
`/sw/system/home/ood/conda/envs/miniforge3-2023.08_python-3.10`	numpy, scipy, pandas, scikit-learn, matplotlib, seaborn, pytorch, tensorflow, keras, qiime2

For a complete list of packages within each environment, use the command: `conda list -p <path to conda environment>`. If you believe that a shared environment is missing packages that would be beneficial for a reasonably broad user community in the cluster, feel free to reach out to the cluster support team.

For each of the environments listed above, you can also access the corresponding GPU version by simply appending _gpu to the environment path.

To activate a shared environment, use the following command:

```
module load Miniforge3
conda activate <path to conda environment>
```

To switch to the GPU-optimized version, use:

```
module load Miniforge3
conda activate <path to conda environment>_gpu
```

In addition to the command line, these environments are also available through the JupyterLab service provided by the cluster web portal. This allows users to run Jupyter notebooks with the desired conda environments directly from the web interface. For more details on how to access and use the JupyterLab service, please refer to the cluster web portal documentation.

From:
https://docs.cluster.uni-hannover.de/ - **Cluster Docs**

Permanent link:
https://docs.cluster.uni-hannover.de/doku.php/guide/soft/miniforge3

Last update: **2025/10/03 08:02**