

# Virtual Environments and Conda

Learn how you can create you own environments with Python's `virtualenv` or with `conda`.

## Introduction

---

Many scientific software packages can be installed with `pip` (if they are Python-based) or `conda`. While we can always install these packages as modules upon request to [hpcsupport@uol.de](mailto:hpcsupport@uol.de), it might sometimes be faster or easier if you install the software yourself.

Below we describe the steps you need to take for that. Please do not hesitate to contact [hpcsupport@uol.de](mailto:hpcsupport@uol.de) anytime if you need help.


## Python `virtualenv`

---

## Conda

---

Conda is an open source package management system and environment management system for installing multiple versions of software packages and their dependencies and switching easily between them.

Below we explain the most important steps in using Conda, for more inforamtion please refer to [this documentation](#) .

## Getting started

---

In order to use Conda on the HPC cluster you need to load a module. We have currently installed different distributions of Conda ( `Anaconda3` , `Miniconda3` , and `Miniforge3` ) and they all work in the same way. We strongly recommend to use `Miniforge3` as we may remove the other distributions in the future (the are some licensing issues).

You can simply start by loading the module with

```
1 | $ module load Miniforge3 # or use e.g. Miniforge3/24.9.2-0 to load a
```

Conda organizes software in environments and you can create as many environments as you need (e.g. to have different versions of the same software). Your available environments can be seen with

```
1 | $ conda env list
```

If you are using Conda for the first time, this will likely only show a `base` environment in our central software directory.

## Create your own environment

---

If you want to create a new environment you can do so with

```
1 | $ conda create --name my_env
```

This will print some information about the environment that will be created. When you see

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

you must confirm that you want to proceed by pressing (**y** and) **enter** (if you press **n** no environment will be created). You may see a warning about updating Conda which you can (and must) ignore (if available, you can switch to a newer version by loading the corresponding module).

## Use your environments

---

You can start using your environment by activating it with

```
1 | $ conda activate my_env
2 | (my_env) $
```

You will notice that your environment was activated by the change of the prompt which is now prefixed with the name of the environment in brackets.

If the command above errors (because you are not using `Miniforge3`) and asks you to run `conda init` please do **not** do so. This will write additional things into your `$HOME/.bashrc` which will prevent you from using different versions of Conda in the future.

Because you have not installed any software in your environment yet, you cannot do much so far. If you are done using the environment you can deactivate it with

```
1 | (my_env) $ conda deactivate
2 | $
```

which makes the prompt return back to the standard one.

## Install packages in your environment

---

To install a package in your environment you can use `conda install` after activating the environment. For example, to install Python in version 3.9 you can use

```
1 | (my_env) $ conda install python=3.9
2 | ...
```

which will print a lot of information about packages that will be downloaded and installed in your environment. You may again see a warning about updating conda which you can ignore. Once you have confirmed it, the installation will proceed. Once it is completed, you can check if the program is available and which version you have with

```
1 | (my_env) $ which python
2 | ~/.conda/envs/my_env/bin/python
3 | (my_env) $ python --version
4 | Python 3.9.20
```

As you can see, within the environment you now use a `python`-executable from a directory in your `$HOME`. The version is as expected.

Conda is providing packages in so-called channels and when you install a package, Conda will tell you where it is looking for packages, which per default is

```
1 | Channels:
2 |   - defaults
3 |   - conda-forge
```

If you need to add a channel, you can do so by using the option `--channel` or `-c` with `conda install`, for example if you need to install PyTorch you could use

```
1 | my_env) $ conda install pytorch=1.9.0 -c pytorch --strict-channel-pri
```

The above has not been tested so it may or may not work. The option `--strict-channel-priority` should help to speedup the process but it could be omitted. Note that we do have a recent module for PyTorch which can be used.

You can also create a new environment and install packages in a single command. For example,

```
1 | $ conda create --name my_env python=3.9
```

would create the environment `my_env` and install Python version 3.9.

## Do not use `conda init`

---

As mentioned above we ask you not use `conda init`. After running `conda init`, your `$HOME/.bashrc` will be modified and include these lines (abbreviated)

```
1 | # >>> conda initialize >>>
2 | # !! Contents within this block are managed by 'conda init' !!
3 | __conda_setup="$('/cm/shared/uniol/sw/SYSTEM/Miniforge3/24.9.2-0/bin/"
4 | ...
5 | unset __conda_setup
6 | # <<< conda initialize <<<
```

Notice how a fixed path to a Conda installation is hard-coded here. As long as this Conda installation exists there should not be a problem. But we may in the future remove older Conda installations which could break this kind of setup. In addition, you will always activate a `base` -

environment which is also no really a problem, but maybe problematic in some situations. We therefore recommend to remove everything between `# >>> conda initialize >>>` and `# <<< conda initialize <<<` from your `.bashrc`.

The modules for `Miniforge3` automatically set all the needed environment variables for Conda and when you switch the version or unload the module, the settings are also changed or removed. This is a much cleaner approach for a system like the HPC cluster. If you are using a different Conda distribution, you can still work without `conda init` by using e.g.

```
1 | $ source $EBROOTMINICONDA3/etc/profile.d/conda.sh
```

The exact `$EBROOT`-variable depends on the module you have loaded.

Powered by [Wiki.js](#)