# DeepModeling

DeepModeling

Apr 11, 2024

# CONTENTS

CHAPTER

ONE

# CONDA/PIP ENVIRONMENTS

**Note:** The following instructions are for the Linux operating system. Most DeepModeling projects fully support the Linux OS but lack support for other OS. If you use the Windows OS, you can use Windows Subsystem for Linux (WSL).

## 1.1 How to setup a conda/pip base environment?

**Note:** The following instructions are for Miniforge but should also work for the offline packages provided by the DeepModeling projects.

An environment is a directory that contains a specific collection of packages that you have installed. Different environments are isolated from each other. A base environment is required before setting up multiple environments.

First of all, download and execute conda-forge installer:

```
wget https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Linux-
↪x86_64.sh -O miniforge.sh
bash miniforge.sh -b -p $HOME/miniforge3
```

This will install the conda base environment into `$HOME/miniforge3`. You can install the environment into a different path.

After installation, you can activate the base environment using

```
source $HOME/miniforge3/bin/activate
```

In order to initialize the environment whenever starting the bash shell, run

```
conda init --all
```

Now you can use `conda install` or `pip install` command to install conda or pip packages.

## 1.2 What is the difference between conda and pip?

The largest difference is that pip can only install Python packages, but conda can install any packages including C, C++, and Fortran packages without Python involved. We suggest reading the Anaconda blog for more information.

## 1.3 How can I check what packages have been installed?

Run `conda list` and `pip list`, which show all conda/pip packages that have been installed in the current environment.

## 1.4 How to create a new conda/pip environment?

To create a conda/pip environment named `my_env`, run

```
conda create -n my_env python=3.11
```

Now you can activate the environment using

```
conda activate my_env
```

If you want to activate the environment in a shell script, use `source` instead:

```
source activate my_env
```

## 1.5 Why do I need multiple environments instead of one?

Due to multiple reasons, dependencies of packages often conflict with each other. For example, Python 3.11 and 3.12 cannot be installed in the same environment; CUDA 11 and CUDA 12 packages cannot be installed in the same environment; Open MPI and MPICH cannot be installed in the same environment; packages built against zlib 1.2 and 1.3 cannot be installed in the same environment. While conda-forge has tried its best to reduce package conflicts, it may still happen when installing multiple packages. Installing different packages into different environments will reduce the risk of meeting the conflict packages.

## 1.6 How to use conda to install CUDA packages without CUDA installed?

Usually, `conda` detects the CUDA driver on the system to install the compatible CUDA packages. You don't need to worry about installing the CUDA Toolkit in advance. However, sometimes, you may want to install CUDA packages into a machine that doesn't install CUDA drivers (for example, the login node of an HPC cluster). In this case, you must manually set the environment variable `CONDA_OVERRIDE_CUDA` to tell conda which packages are compatible.

For example, you would like to use the CUDA packages on another machine with installed CUDA drivers. Run `nvidia-smi` on that machine,

```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 545.29.06              Driver Version: 545.29.06    CUDA Version: 12.3     |
|-----------------------------------------+----------------------+----------------------+
```

The output shows that the CUDA driver version is `12.3`. Then, on the machine for performing installation, you may set the environment variable `export CONDA_OVERRIDE_CUDA=12.3` before running `conda install`.

## 1.7 How to install conda/pip packages without the Internet access?

To install conda/pip packages without Internet access, you need to have a machine that can connect to the Internet, and use `conda` or `pip` to install all packages in an environment. Then, install conda-pack:

```
conda install conda-pack
```

Next, running `conda pack` to pack an environment, saying `my_env`:

```
conda pack -n my_env -o my_env.tar.gz
```

Transfer `my_env.tar.gz` to the machine that doesn't have the Internet access. On that machine, you can unpack the environment:

```
mkdir -p my_env
tar -xzf my_env.tar.gz -C my_env
source my_env/bin/activate
conda-unpack
```

## 1.8 How to speed up conda/pip in China?

When you are in China, we suggest adding Tsinghua University tuna mirrors to your conda/pip configs to speed up downloading conda/pip packages.

In order to set up Anaconda mirrors, append the following lines to the `$HOME/.condarc` file (if it doesn't exist, you can create one):

```
default_channels:
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r
custom_channels:
  conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
```

In order to set up PyPI mirrors, run

```
pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
```

---

- genindex
- modindex
- search