

Conda Environments

Conda environments can be created per user. To create a new conda environment, simply run:

```
$ conda init
$ exec bash
$ conda create --name myenv python=3.10
```

You can use whichever Python version you want (or need based on packages you intend to install) and name the environment whatever you choose.

Activating an Environment

To activate your environment and install some packages:

```
$ conda activate myenv
$ (myenv) conda install -c conda-forge pandas
```

Any packages you install using Conda while you have `myenv` activated will be limited to the `myenv` environment.

You can see all of your environments with:

```
$ conda env list
```

The currently active environment will be marked with `*`.

Deactivating an Environment

To leave the currently active environment:

```
$ (myenv) conda deactivate
```

Removing an Environment

If you no longer need an environment, you can remove it with:

```
$ conda env remove --name myenv
```

Sharing Environments

Note that you **cannot activate Conda environments created by other users**. However, it is simple for multiple users to create duplicate environments if it is necessary that all users (for example, multiple members of a lab group) have access to identical environments.

First, save an environment file from the Conda environment that you want to copy packages from:

```
$ (globalenv) conda env export > environment.yml
```

Share this `environment.yml` file with other users who want to create a duplicate environment. From their respective accounts, they can run:

```
$ conda env create -n other-env -f environment.yml
```

This creates a new Conda environment called `other-env` that installs all packages defined in `environment.yml`.

The environment name in the YAML file can be overridden by specifying `-n` when creating the environment.

Updating an Existing Environment

If an `environment.yml` file has been modified or contains packages that need to be added to an existing environment, use `conda env update`:

```
$ conda env update -n myenv -f environment.yml
```

This updates the existing `myenv` environment with the packages specified in `environment.yml`.

If you want to remove packages from the environment that are no longer specified in the YAML file, use `--prune`:

```
$ conda env update -n myenv -f environment.yml --prune
```

Use `--prune` with caution, as it can remove packages that were manually installed but are not listed in the environment file.

Exporting an Environment

To create an environment file that can be shared with other users:

```
$ (myenv) conda env export > environment.yml
```

This includes the packages and versions installed in the environment.

For a more portable environment file, you can export only the packages explicitly installed by the user:

```
$ (myenv) conda env export --from-history > environment.yml
```

The `--from-history` option is generally preferable when sharing an environment across different systems because it avoids including many automatically installed dependencies and exact platform-specific packages.

Revision #3

Created 2026-09-18 16:51:40 UTC by Tobi Azeez

Updated 2026-09-18 17:08:46 UTC by Tobi Azeez