Published in The Startup

Swati Kanchan Follow
Dec 3, 2020 · 4 min read · Listen

Setting Up a Conda Environment in Less Than 5 Minutes

Why do you even need virtual environments?

As a beginner, it is usually tempting to use the base Python 2x or 3x program for running each and every Python code you have on your computer, but once your projects get more and more complex, versatile, or structured, Virtual Environments comes to rescue in neatly organizing the several codebases. It helps you to keep separate projects in isolated environments secluded from the other environments' dependencies

> *You must run different projects on separate environments!*

For more details please read Why you should use a virtual environment for EVERY python project!.

Accordingly, there can be a few scenarios in which virtual environments are required to be set-up. Match your scenario in which your situation fit in

Scenario 1: You want to replicate a Conda environment from a different machine

If you are currently working on some project in a different machine and want to create the same Conda environment in another machine, make one YAML file of that environment, which will contain all the packages along with the versions. To do that, follow the steps below:

Step 1: On Windows open up a Anaconda Prompt, on Linux and MacOS open up a Terminal.

Step 2: Activate the environment using `conda activate <name_of_environment>`. Replace `<name_of_environment>` with name of environment you want to replicate.

Step 3: Then export your active environment to a new file using `conda env export > environment.yml`

Now you will have one environment.yml file with all the nitty-gritty versioned packages of the environment. Copy it and get that file to your destination machine where you want to set-up the new environment.

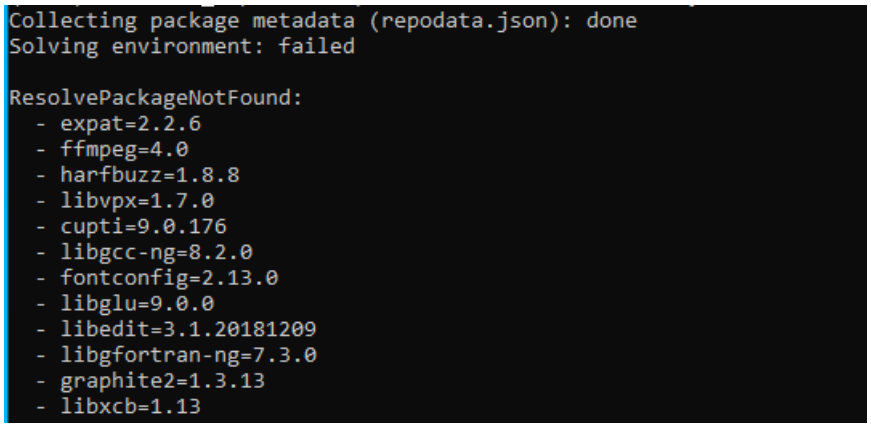
Step 4: Follow **Scenario 2**.

Step 2: Create the environment by running `conda env create -f environment.yml`.

This creates a brand new environment with the same name as before. To verify that it was created successfully, execute `conda info --envs` or `conda info -e`.

. . .

Note: You might face *ResolvePackageNotFound: failure* while creating. Following the ResolvePackageNotFound: line in the prompt, you will find some package names along with their versions. It might look something like this:



Anaconda Prompt or the terminal error

export your environment once again, this time with a `--no-builds` option (`conda env export --no-builds > environment.yml`).

If the error persists, you still have an option to install those specific “ResolvePackageNotFound” packages with `Pip`.

Open up the environment.yml file in a text-editor and go to the last line to add an extra line with `-pip:`. Then hit enter and in the next line list all those “ResolvePackageNotFound” packages along with the versions with an extra indentation. Also, remember to remove all these packages from the `-dependencies:` list in the environment.yml file. It would finally look something like this:

```
  - defaults
dependencies:
  - _tflow_select=2.1.0
  - absl-py=0.7.0
  - astor=0.7.1
  - blas=1.0
  - bzip2=1.0.6
  - c-ares=1.15.0
  - ca-certificates=2019.1.23
  - cairo=1.14.12
  - pip:
    - libglu==9.0.0
    - readline==7.0
    - jasper==2.0.14
    - graphite2==1.3.13
    - libgfortran-ng==7.3.0
    - libedit==3.1.20181209
    - libuuid==1.0.3
    - hdf5==1.9.0

```

environment.yml

Finally, try creating the new environment once again with `conda env create -f environment.yml`. If you still face any package issue (which I hope you would not), try removing that package from the yml file and create that environment again! Hopefully, the new environment will be created successfully now.

. . .

and run `pip install requirements.txt`. This `requirements.txt` contain a list of the python libraries along with their versions written in plain text format.

Step 4: Verify that the Conda environment was created successfully by executing `conda info --envs` or `conda info -e`.

Step 5: Activate your environment using `conda activate <environment_name>`.

Scenario 3: You need to set-up an environment from scratch or just from a requirement.txt

In order to create a Conda environment right from scratch, you might need to first choose a Python version.

Step 1:

If you just want to create an environment without a specific Python version, run `conda create --name <env_name>` . Replace `<env_name>` with environment name.

When conda asks you to proceed, type `y`.

Or

If you want to create an environment with a specific Python package, run `conda`

Or

If you want to create an environment with a package with a specific version, run `conda create --name <env_name> python=<version> <package>=<version>` . Replace `<package>=<version>` with package name and version. (eg. `hdf5=1.10.2`

Step 2: If you also have a requirements.txt file, you might want to ensure that those packages are installed. To install them, go to that directory in the Terminal and run `pip install requirements.txt` . This `requirements.txt` contain a list of the python libraries along with their versions written in plain text format.

Step 3: Verify that the Conda environment was created successfully by executing `conda info --envs` or `conda info -e`.

Step 4: Activate your environment using `conda activate <environment_name>`.

. . .

Cherry on the Cake:

- After you create the environment, they are stored in /envs/ directory of your Anaconda or Miniconda folder
- To rename your environment, run:

- Want to manage your Conda environments more? Head to its Docs page.