Working with Python3 on the IDEA Cluster

Karan Bhanot
bhanotkaran22@gmail.com

March 1, 2022

Contents

1 Python

Python is a programming language that is very popular for various applications such as web scraping, web application development, machine learning, deep learning and more. The language is backed by a very active community and houses thousands of libraries and packages.

There are two versions of Python: *Python2* and *Python3*. These two versions are still being extensively used but *official support* for Python2 was suspended on January 1, 2020 and only Python3 is now supported now. As a result, many prominent packages have added support for Python3. This document focuses on Python3.

2 Python Virtual Environments

Python Virtual Environments provide an isolated environment on a machine which has Python packages that do not interfere with the packages globally installed on the machine. A virtual environment can be based on a specific version of Python, and any number of packages can be installed inside it.

The most commonly used and easily understandable virtual environment manager is `virtualenv` but many other tools exist as well such as `pew`, `venv` etc.

3 Python3 on IDEA Server

To access and install packages for Python3 on the IDEA server, you need to work inside a virtual environment. This ensures that packages installed by a certain user do not overwrite the packages available at the global level. At the time of writing this document, the IDEA server has Python 3.6.8 but the process for set up generally does not change and should work for all future versions of Python3 as well.

3.1 Set Up Working Directory

The first step is to create a working directory for your Python environment. In the Linux shell, browse to the location where you want to create the directory and create the new directory using the command `mkdir`.

I'll create the directory `test_environment` in the main directory and then use `cd test_environment` to go inside the newly created directory as seen in Figure 1.

3.2 Create the Virtual Environment

Once you're inside the virtual environment, the command `virtualenv -p python3 .` will create the virtual environment for you using *virtualenv*. Figure 2 shows how the virtual environment is set up.

The command involves the word `virtualenv` which tells that the machine that we're using it as the virtual environment generator. `-p python3` tells the machine that we're going to use python3 installed on the machine. Finally, the command ends with a dot which means that the environment must be set up in the current directory.

3.3 Activate the Virtual Environment

The environment is now completely ready to be used. The command `source bin/activate` will activate the environment. The prompt will now be appended by the name of the environment (test _environment in our case) which shows that we are now inside the environment.

Anything we install here would not be installed or available outside this environment. Figure 3 shows how to activate the environment, update modules and install a sample package *numpy* inside the environment. We update *pip* and *setuptools* to the latest version using the command `pip install --upgrade pip`

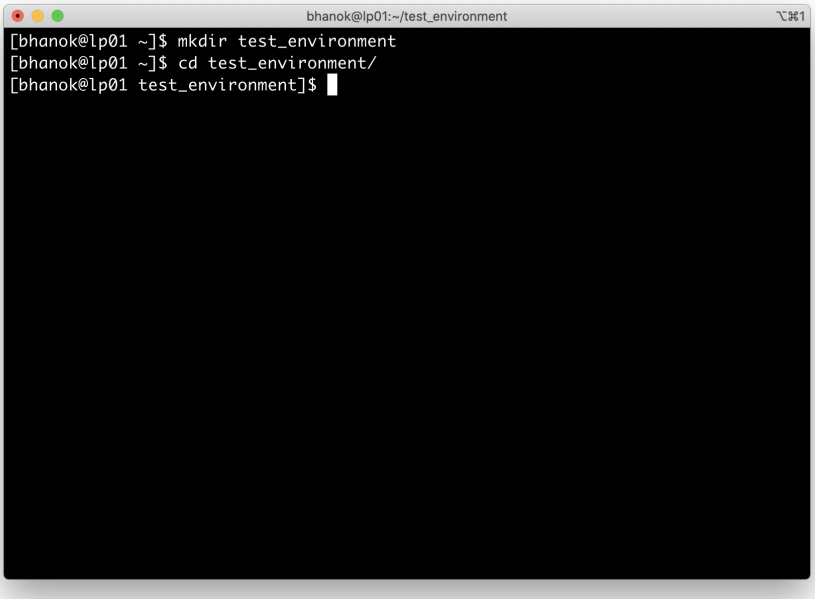

Figure 1: Creating directory and moving in it

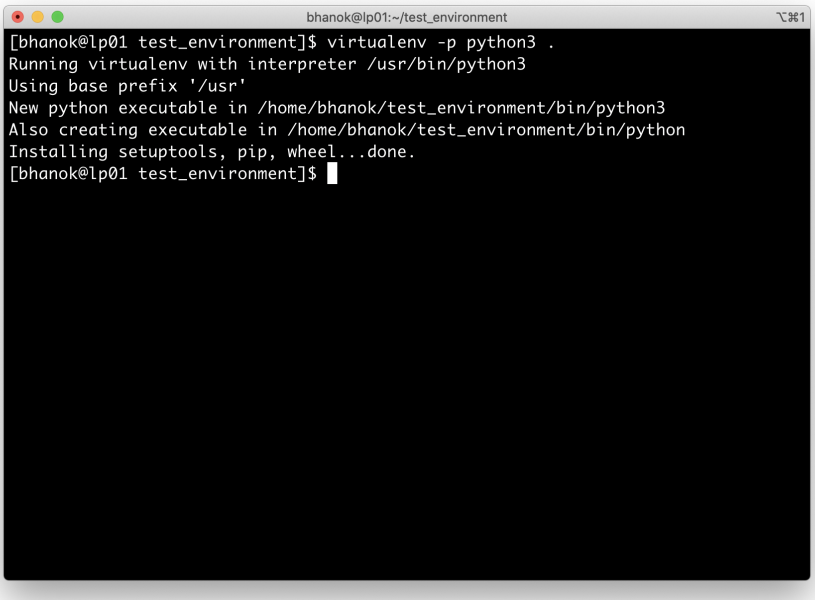

Figure 2: Creating virtual environment

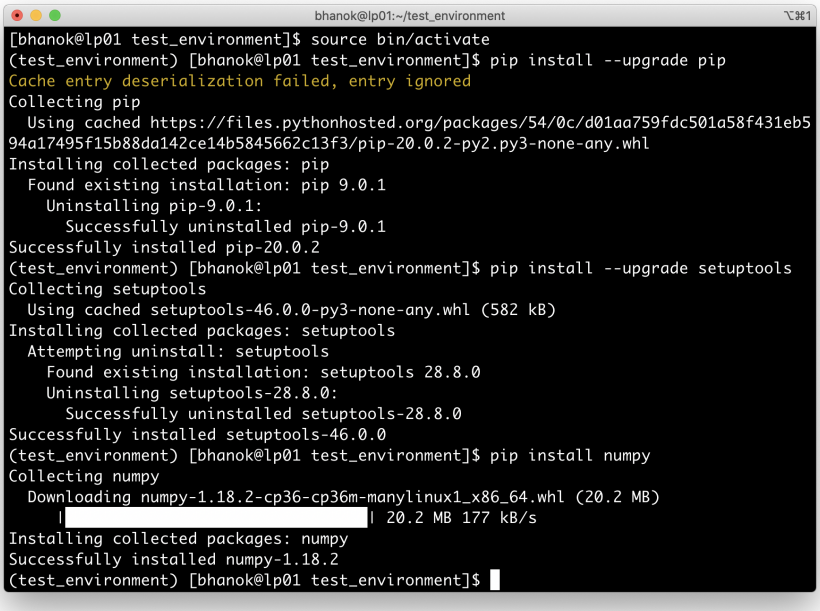

Figure 3: Activating virtual environment

followed by `pip install --upgrade setuptools` and then install our sample package numpy using `pip install numpy`.

Inside this environment, you can install your necessary packages, play with them and work on your Python scripts without interfering with anything else.

3.3.1 Using requirements.txt

Once your work is complete and you want others to use it, they would need to install the packages that you use in your projects as well as ensure compatibility amongst them. Python provides the option to work with a file called *requirements.txt* which includes the list of all packages needed by the project along with their specific versions.

Once you're inside the virtual environment just use the command `pip freeze > requirements.txt` which creates the requirements file with the list of all required packages. Any user who is using your project can simply run the command `pip install -r requirements.txt` and install the required packages for your project to be replicated easily.

In Figure 4, we can see that we save the list of installed libraries using `pip freeze > requirements.txt` which is demonstrated by showing the contents of the file *requirements.txt* using `cat requirements.txt`. To install the packages from the list, we used the command `pip install -r requirements.txt` but as in this case we already have *numpy* installed, it just echoed that the requirement is already satisfied.

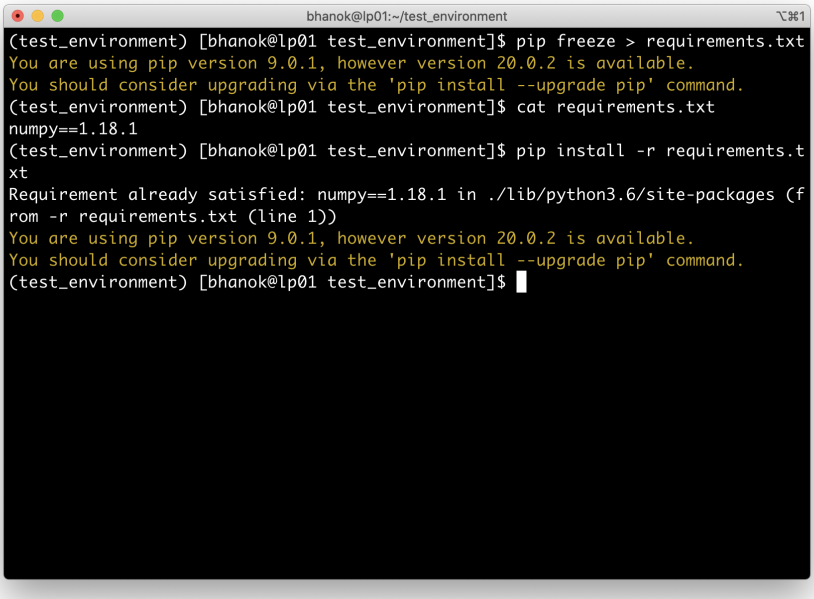

Figure 4: Working with requirements.txt

3.3.2 Using requirements.txt

Once your work is complete and you want others to use it, they would need to install the packages that you use in your projects as well as ensure compatibility amongst them. Python provides the option to work with a file called *requirements.txt* which includes the list of all packages needed by the project along with their specific versions.

3.4 Deactivate the Virtual Environment

Once you're done working for the current session, you should deactivate the environment using `deactivate`. You will see that the name of the virtual environment is now removed indicating that we're outside the environment as can be seen in Figure 5.

4 Working with Jupyter Notebooks

4.1 Accessing Jupyter Notebooks

Jupyter notebooks server is also running on the IDEA Cluster and the user needs to just directly access it by logging in to their account. Simply, go to the link https://lp01.idea.rpi.edu/jupyter/hub/login and then login with your RCS username and password and you'll get access to your account where you can create/update/delete Jupyter notebooks.

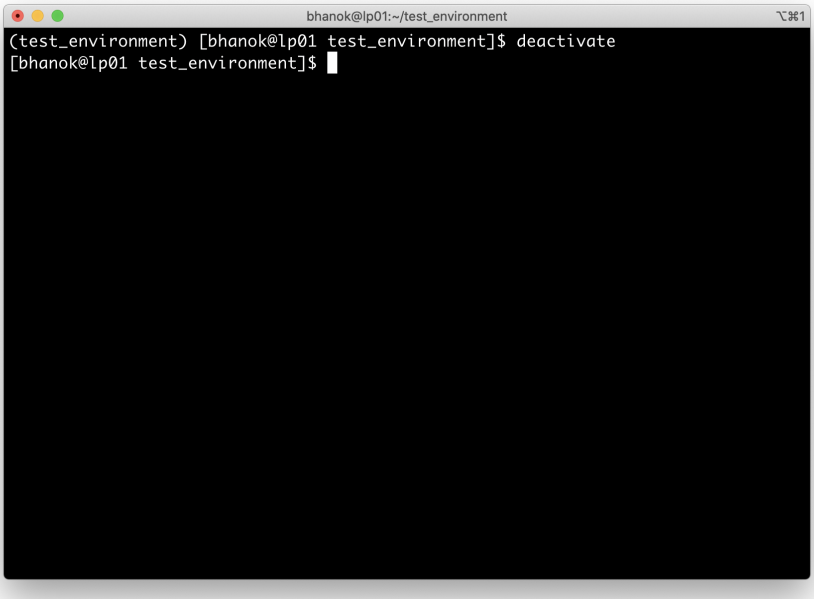

Figure 5: Deactivating virtual environment

NOTE: There are several environments under which the Jupyter notebooks can be created. Clicking on `New` dropdown, you'll be provided with a list of environments under which you wish you create the Jupyter notebook as seen in Figure 6.

4.2 Jupyter notebooks in Custom Environments

Not all packages are available under all environments and you may wish to set up your own environment, under which you can install your own set of packages and run the Jupyter notebooks. This can be done on the server for both non-GPU and GPU-enabled Jupyter notebooks.

The steps to run custom environment Jupyter notebooks are:

1. Go to the url https://lp01.idea.rpi.edu/jupyter-gpu/hub/login and login with your RCS credentials.

2. Click on the `New` dropdown on the right top corner and select the last option `Terminal`. A new tab will open with a terminal connected to the server.

3. Run the command `conda create -p /software/anaconda3/envs/ENV_NAME` where you will replace `ENV_NAME` with the name of your environment. If we consider the environment to be `custom_env`, then the command will be `conda create -p /software/anaconda3/envs/custom_env`

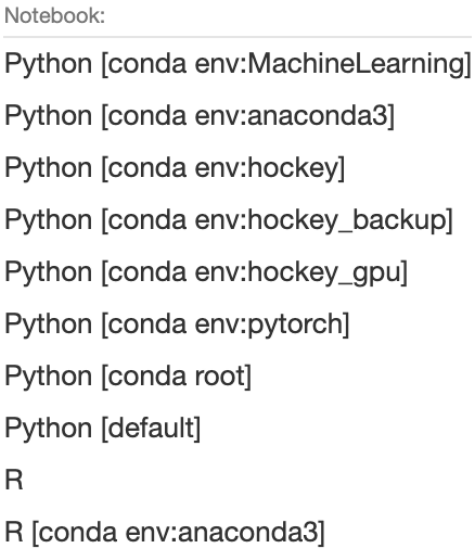

Figure 6: Several environments for Jupyter notebooks

4. The environment is now created. We activate the environment by using the command `conda activate ENV_NAME` by replacing `ENV_NAME` with the name of the environment (`custom_env` in this example).
5. We need to install `nb_conda_kernels` which enables us to show and access the environment via the Jupyter Hub. Run the command `conda install nb_conda_kernels`.
6. Install any other packages you want in the environment using either `pip` or `conda`.
7. Once the environment is ready, deactivate it using `conda deactivate`. Close this terminal window.
8. Again, go to https://lp01.idea.rpi.edu/jupyter-gpu/hub/login and login if needed (you will probably be logged in by default this time).
9. Click on the `New` dropdown on the right top corner and select the option `Python [conda env:ENV_NAME]` where the `ENV_NAME` will be replaced by the name of your environment (`custom_env` in this example).
10. A Jupyter notebook will open in a new browser tab and this will be based on the new environment you created.

The environment is set up for all users on the IDEA cluster. You only need to perform the last three steps (8-10) above every time you want to start a new notebook.

4.3 Cluster notebooks on Local Machine

If you want to run a Jupyter notebook on the cluster and access it via your local machine, it's possible by using port forwarding through SSH. We con-

sider that the user has the RCS name as `USERNAME` and we are working on the `lp01.idea.rpi.edu` node.

The steps to run Jupyter notebooks on local machine are:

1. Open the terminal on your local machine and run the command `ssh USERNAME@lp01.idea.rpi.edu` where you will replace `USERNAME` with your own RCS name. It will ask for your password. Type the password (the content is hidden and you will not see any text while typing your password) and press `ENTER` on your keyboard to login.

2. Create a virtual environment as described in the previous sections. Let us consider that the environment is called `custom_env`. Activate the environment.

3. Install all the required packages including `jupyter` inside this environment. If we just install `jupyter`, use the command `pip install jupyter`.

4. Once installed, we need to run the Jupyter server without a browser. Use the command `jupyter notebook --no-browser`. You'll notice that the server starts up. Select and copy the complete URL that starts with `http://127.0.0.1:8888/?`.

5. Open another terminal on your local machine and type the command `ssh -N -f -L localhost:8888:localhost:8888 USERNAME@lp01.idea.rpi.edu` while replacing `USERNAME` with your own RCS name.

6. Finally, go to your browser and paste in the URL you copied earlier and you'll now be accessing the Jupyter notebooks inside the environment you created on your local machine.

5 Working with Multiple Projects

If you're working with multiple Python projects, it's always a good practice to have separate virtual environments for each project as each one will have their own set of required packages and this will prevent interference. For a given project, each time you want to work on it, just go inside its directory, activate the environment, complete your task and then deactivate.

6 Future Reading

There are many virtual environment tools that exist for Python. If you want to read more about them, refer to an article published by me on Medium: Comparing Python Virtual Environments

NOTE: The document was first created on September 24, 2020 and was later updated over multiple iterations with last update on March 1, 2022.