Working with Python3 on the IDEA Cluster

Karan Bhanot
bhanotkaran22@gmail.com

September 24, 2020

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.

We should get a Python2 user to add comments on differences... – John

We shouldn't use Python 2 at all most of the packages required don't even support it – Andrew

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`.

Please recode shell commands using the `mytexttt` command... – John

Done – Karan

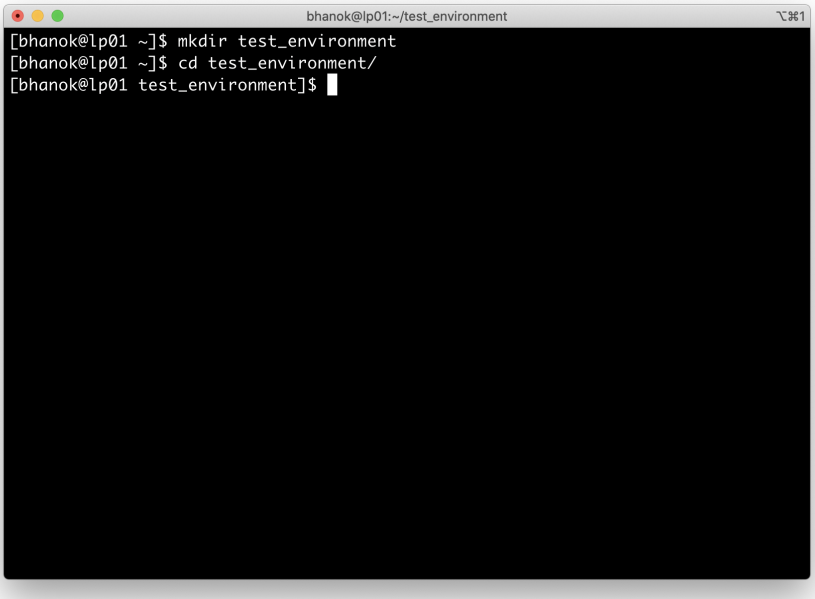

Figure 1: Creating directory and moving in it

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`

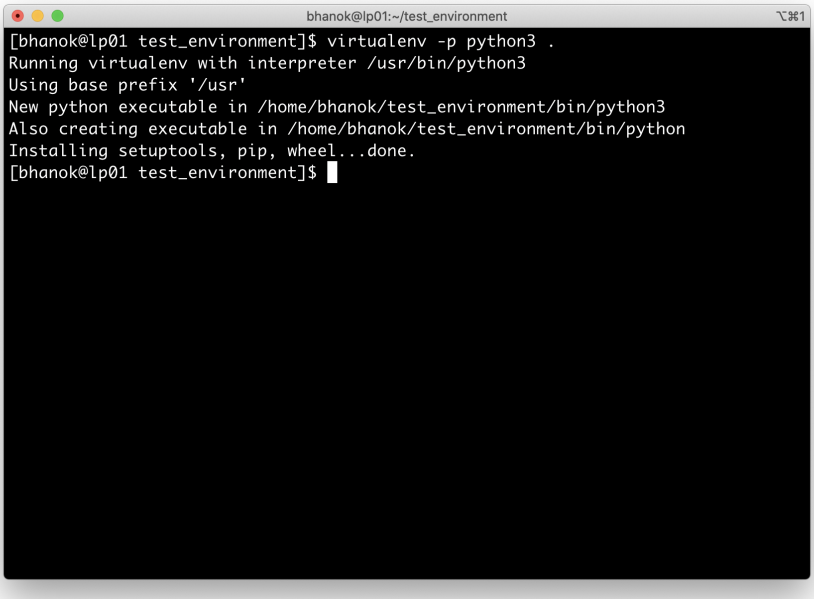

Figure 2: Creating virtual environment

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.

maybe discuss requirements files? If we are using them that is – Andrew

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.

I've created another section for requirements.txt – Karan

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.

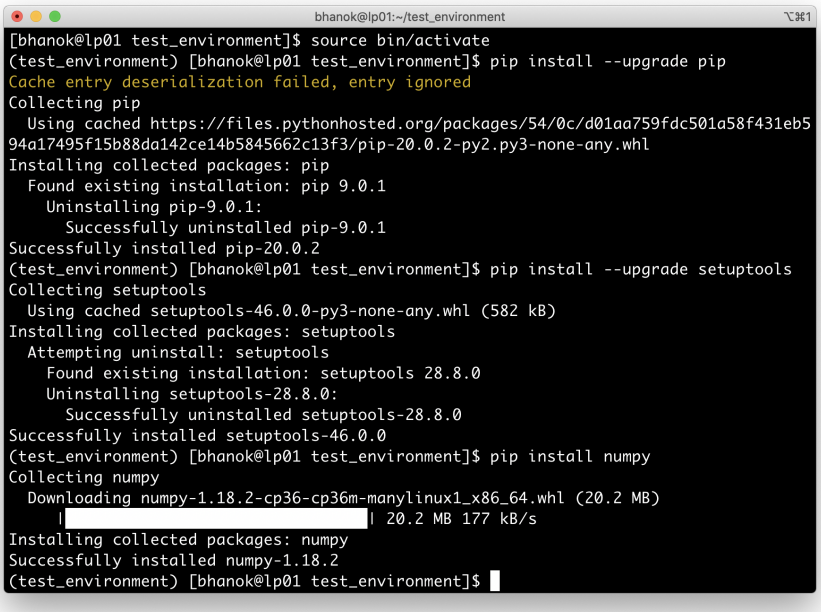

Figure 3: Activating virtual environment

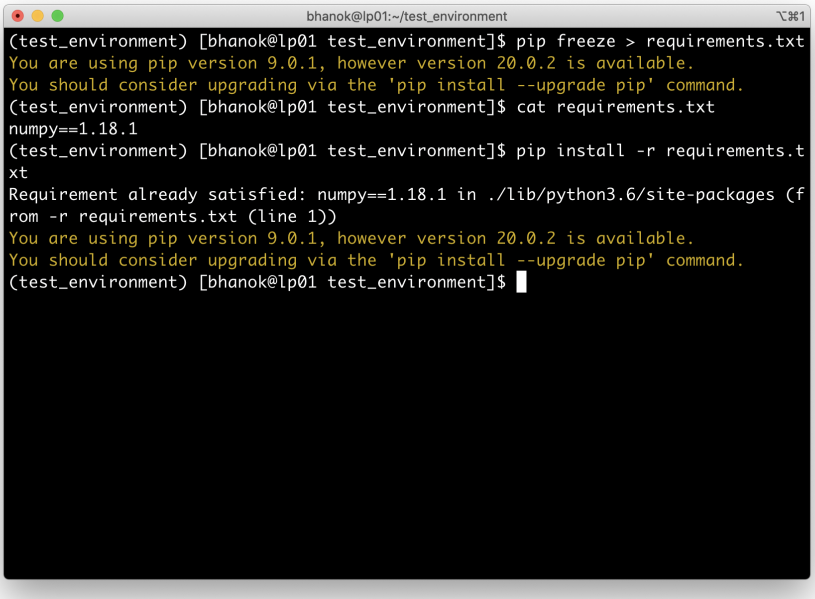

Figure 4: Working with requirements.txt

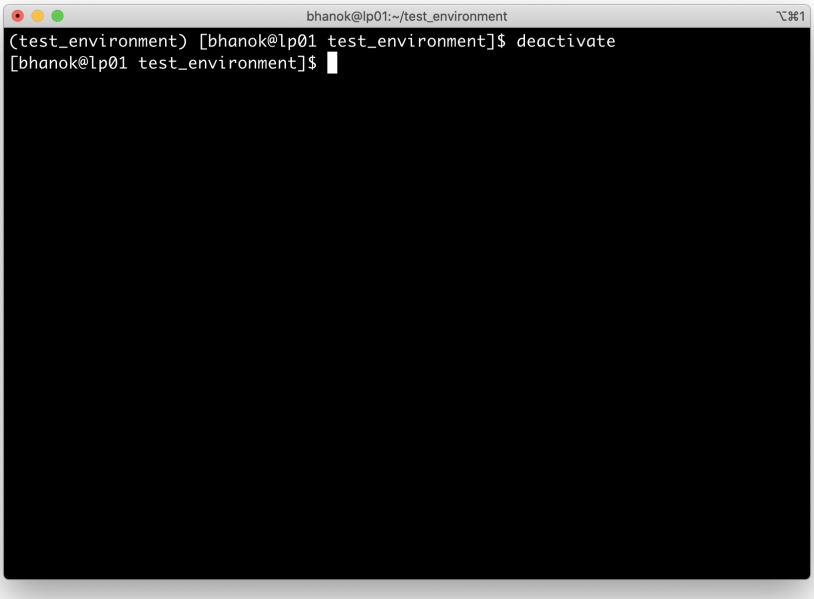

Figure 5: Deactivating virtual environment

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 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.

5 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