**How to Create Custom Python Module and Distribute with Git**

When we create a custom Python module, we can distribute it through a Git repository. By using a private repo, you can keep your module private and use it in your deployment pipeline by calling pip install from the private repo. Pretty cool trick.

In this example, I created a simple Python module to do S3 download and upload and pushed it to a Git repo for pip installation.

First of all, we need to have the right folder structure with [setup.py](https://the-hitchhikers-guide-to-packaging.readthedocs.io/en/latest/quickstart.html).

The folder structure will look like:

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

The key is to have the correct setup.py file. This describes the metadata about the module as well as is used for installation.

**setup.py**

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

[/cc]

The top level project folder is the name of the module. There is another folder with the same name, including \_\_init\_\_.py and actual python code that contains the class.

The init file references the class that can be imported by the module.

**\_\_init\_\_.py**

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

[/cc]

Here is the actual S3 class that does upload and download.

**s3.py**

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

[/cc]

Once everything is done, let’s push it to the repo and all done. Installation is easy. You can pass the repo url in pip install as below.

**Installation**

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

You can now test the module.

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

[/cc]

It works!

Yay!