A CLOUD GURU

Lesson Description - Project Setup

Now that we know about all of the standard building blocks that we have at our disposal, we're going to take the time in the remaining sections of the course to look at different types of projects that we could build using Python. For this section, we'll build a command-line tool to create database backups. In this lesson, we'll be setting up our project with some documentation and learn about the most important file for any Python package: `setup.py`.

Documentation for This Video

- Distributing Python Modules
- Packaging Python Projects
- Python Gitignore File

Our Project

For this project, our goal is to create a simple CLI that will allow us to do the following:

1. Back up a PostgreSQL database using `pgdump`
2. Write the backup locally *or* upload the backup to AWS S3

By the time we're finished, our tool will work like this.

- Backing up to S3:

```
  $ pgbackup postgres://bob@example.com:5432/db_one --driver s3
backups
```

- Back up locally:

```
  $ pgbackup postgres://bob@example.com:5432/db_one --driver
local /var/local/db_one/backups
```

Setting Up the Project Directory

For Python projects that we intend on distributing as a package, we only really need to have one thing: a `setup.py` file. This file is used by `pip` (specifically `setuptools` behind the scenes) to know what to do to install the package.

Let's create a project directory called `pgbackup` with some subdirectories, a `README.md` file for documentation, and a `setup.py` file.

```
$ mkdir -p ~/projects/pgbackup/src/
$ cd ~/projects/pgbackup
$ touch README.md setup.py src/.gitkeep
```

The `.gitkeep` file that we added is so that we can commit an "empty" directory with Git before we write any of the projects source code.

To go through setting up our `setup.py` file, we'll only need to make a few adjustments from what is done in the official packaging tutorial. Here's what our's looks like:

~/projects/pgbackup/setup.py

```
from setuptools import find_packages, setup

with open('README.md', 'r') as f:
    long_description = f.read()

setup(
    name='pgpackup',
    version='0.1.0',
    author='Keith Thompson',
    author_email='keith@linuxacademy.com',
    description='A utility for backing up PostgreSQL databases',
    long_description=long_description,
    long_description_content_type='text/markdown',
    url='https://github.com/your_username/pgbackup',
    packages=find_packages('src')
)
```

We want the bulk of our documentation to exist within our `README.md` file. This file will give us some documentation if we use a remote repository host like GitHub or GitLab. It also means that we won't have to modify this file often for the information about the package to change.

The two biggest things to note are that we needed to use the `setup` function from `setuptools` and that we used `find_packages` to ensure that any sub packages that we create will be picked up as part of our project without us needing to modify this file.

While we're still working with Python code, we should also initialize our project with a virtualenv using Pipenv:

```
$ pipenv --python python3.7
Creating a virtualenv for this project…
Pipfile: /home/cloud_user/projects/pgbackup/Pipfile
Using /usr/local/bin/python3.7 (3.7.2) to create virtualenv…
⠋ Creating virtual environment...Already using interpreter /usr/
local/bin/python3.7
Using base prefix '/usr/local'
New python executable in /home/cloud_user/.local/share/virtualenvs/
pgbackup-uJWPrEHw/bin/python3.7
Also creating executable in /home/cloud_user/.local/share/
virtualenvs/pgbackup-uJWPrEHw/bin/python
Installing setuptools, pip, wheel...
done.

✔ Successfully created virtual environment!
Virtualenv location: /home/cloud_user/.local/share/virtualenvs/
pgbackup-uJWPrEHw
Creating a Pipfile for this project…
$ pipenv shell
```

Writing Our README

Our `README.md` file will document a few different things:

1. How to use the tool.
2. How to get up and running when developing the project.
3. How to install the project for use from source.

~/projects/pgbackup/README.md

```
pgbackup
========

CLI for backing up remote PostgreSQL databases locally or to AWS S3.

## Usage

Pass in a full database URL, the storage driver, and destination.
```

```
S3 Example w/ bucket name:

```
$ pgbackup postgres://bob@example.com:5432/db_one --driver s3 backups
```

Local Example w/ local path:

```
$ pgbackup postgres://bob@example.com:5432/db_one --driver local /
var/local/db_one/backups
```

Installation From Source

To install the package after you've cloned the repository, you'll
want to run the following command from within the project directory:

```
$ pip install --user -e .
```

Preparing for Development

Follow these steps to start developing with this project:

1. Ensure `pip` and `pipenv` are installed
2. Clone repository: `git clone git@github.com:example/pgbackup`
3. `cd` into the repository
4. Activate virtualenv: `pipenv shell`
5. Install dependencies: `pipenv install`
```

Adding a Gitignore File

Before we commit anything, we're going to pull in a default Python .gitignore file from Github so that we don't track files in our repository that we don't need:

```
(pgbackup) $ curl -o .gitignore https://raw.githubusercontent.com/
github/gitignore/master/Python.gitignore
...
```

Our Initial Commit

Now that we've created our `setup.py`, `README.md`, and `.gitignore` files, we're in a good position to stage our changes and make our first commit:

```
$ git init
$ git add --all .
$ git commit -m 'Initial commit'
```