A CLOUD GURU

Lesson Description - Initial Project Layout

The last thing we need to do before we start implementing our `pgbackup` tool is structure the project with the required files and folders.

Documentation For This Video

- The `setuptools` package
- The `make` documentation

  Create Package Layout

  There are a few specific places that we're going to put code in this project:

  1. In a `src/pgbackup` directory. This is where our project's business logic will go.
  2. In a `tests` directory. We'll put automated tests here.

     We're not going to write the code that goes in these directories just yet, but we are going to create them and put some empty files in so that we can make a git commit that contains these directories. In our `src/pgbackup` directory, we'll use a special file called `__init__.py`, but in our `tests` directory, we'll use a generically named, hidden file.

     ```
     (pgbackup-E7nj_Bs0) $ mkdir -p src/pgbackup tests
     (pgbackup-E7nj_Bs0) $ touch src/pgbackup/__init__.py
     tests/.keep
     ```

     Writing Our `setup.py`

     One of the requirements for an installable Python package is a `setup.py` file at the root of the project. In this file, we'll utilize `setuptools` to specify how our project is to be installed and define its metadata. Let's write out this file now:

     ~/code/pgbackup/setup.py

```
from setuptools import setup, find_packages

with open('README.rst', encoding='UTF-8') as f:
    readme = f.read()

setup(
    name='pgbackup',
    version='0.1.0',
    description='Database backups locally or to AWS S3.',
    long_description=readme,
    author='Keith',
    author_email='keith@linuxacademy.com',
    packages=find_packages('src'),
    package_dir={'': 'src'},
    install_requires=[]
)
```

For the most part, this file is metadata, but the `packages`, `package_dir`, and `install_requires` parameters of the `setup` function define where setuptools will look for our source code and what other packages need to be installed for our package to work.

To make sure that we didn't mess up anything in our `setup.py`, we'll install our package as a development package using `pip`.

```
(pgbackup-E7nj_BsO) $ pip install -e .
Obtaining file:///home/user/code/pgbackup
Installing collected packages: pgbackup
  Running setup.py develop for pgbackup
Successfully installed pgbackup
```

It looks like everything worked, and we won't need to change our `setup.py` for awhile. For the time being, let's uninstall `pgbackup` since it doesn't do anything yet:

```
(pgbackup-E7nj_BsO) $ pip uninstall pgbackup
Uninstalling pgbackup-0.1.0:
  /home/user/.local/share/virtualenvs/pgbackup-E7nj_BsO/
lib/python3.6/site-packages/pgbackup.egg-link
  Proceed (y/n)? y
    Successfully uninstalled pgbackup-0.1.0
```

Makefile

In our `README.rst` file, we mentioned that to run tests we wanted to be able to simply run `make` from our terminal. To do that, we need to have a `Makefile`. We'll create a second make task that can be used to setup the virtualenv and install dependencies using `pipenv` also. Here's our `Makefile`:

~/code/pgbackup/Makefile

```
.PHONY: default install test

default: test

install:
    pipenv install --dev --skip-lock

test:
    PYTHONPATH=./src pytest
```

This is a great spot for us to make a commit:

```
(pgbackup-E7nj_BsO) $ git add --all .
(pgbackup-E7nj_BsO) $ git commit -m 'Structure project
with setup.py and Makefile'
[master 1c0ed72] Structure project with setup.py and
Makefile
 4 files changed, 26 insertions(+)
 create mode 100644 Makefile
 create mode 100644 setup.py
 create mode 100644 src/pgbackup/__init__.py
 create mode 100644 tests/.keep
```