A CLOUD GURU

Lesson Description - Introduction to TDD and First Tests

With our project structured, we're finally ready to start implementing the logic to create database backups. We're going to tackle this project using "Test Driven Development", so let's learn the basics of TDD now.

Documentation For This Video

- The `pytest` package
- The `pytest.raises` function

Installing `pytest`

For this course, we're using `pytest` as our testing framework. It's a simple tool, and although there is a unit testing framework built into Python, I think that `pytest` is a little easier to understand. Before we can use it though, we need to install it. We'll use `pipenv` and specify that this is a "dev" dependency:

```
(pgbackup-E7nj_Bs0) $ pipenv install --dev pytest
...
Adding pytest to Pipfile's [dev-packages]…
Locking [dev-packages] dependencies…
Locking [packages] dependencies…
Updated Pipfile.lock (5c8539)!
```

Now the line that we wrote in our `Makefile` that utilized the `pytest`, CLI will work.

Writing Our First Tests

The first step of TDD is writing a failing test. In our case, we're going to go ahead and write a few failing tests. Using `pytest`, our tests will be functions with names that start with `test_`. As long as we name the functions properly, the test runner should find and run them.

We're going to write three tests to start:

1. A test that shows that the CLI fails if no `driver` is specified.
2. A test that shows that the CLI fails if there is no `destination` value given.

3. A test that shows, given a driver and a destination, that the CLI's returned `Namespace` has the proper values set.

At this point, we don't even have any source code files, but that doesn't mean that we can't write code that demonstrates how we would like our modules to work. The module that we want is called `cli`, and it should have a `create_parser` function that returns an `ArgumentParser` configured for our desired use.

Let's write some tests that exercise `cli.create_parser` and ensure that our `ArgumentParser` works as expected. The name of our test file is important; make sure that the file starts with `test_`. This file will be called `test_cli.py`.

~/code/pgbackup/tests/test_cli.py

```
import pytest


from pgbackup import cli


url = "postgres://bob:password@example.com:5432/db_one"


def test_parser_without_driver():
    """
    Without a specified driver the parser will exit
    """
    with pytest.raises(SystemExit):
        parser = cli.create_parser()
        parser.parse_args([url])


def test_parser_with_driver():
    """
    The parser will exit if it receives a driver
    without a destination
    """
    parser = cli.create_parser()
    with pytest.raises(SystemExit):
        parser.parse_args([url, "--driver", "local"])


def test_parser_with_driver_and_destination():
    """
    The parser will not exit if it receives a driver
    with a destination
```

```
    """
    parser = cli.create_parser()


    args = parser.parse_args([url, "--driver","local","/some/
    path"])
    assert args.driver == "local"
    assert args.destination == "/some/path"
```

</code></pre>

Running Tests

Now that we've written a few tests, it's time to run them. We've created our Makefile already, so let's make sure our virtualenv is active and run them:

```
$ pipenv shell
(pgbackup-E7nj_BsO) $ make
PYTHONPATH=./src pytest
========================================= test session starts
=========================================
platform linux -- Python 3.6.4, pytest-3.3.2, py-1.5.2,
pluggy-0.6.0
rootdir: /home/user/code/pgbackup, inifile:
collected 0 items / 1 errors

================================================ ERRORS
================================================
_____________________________ ERROR collecting tests/
test_cli.py ______________________________________
ImportError while importing test module '/home/user/code/
pgbackup/tests/test_cli.py'.
Hint: make sure your test modules/packages have valid Python
names.
Traceback:
tests/test_cli.py:3: in
    from pgbackup import cli
    E   ImportError: cannot import name 'cli'
    !!!!!!!!!!!!!!!!!!!!!!!!!!!! Interrupted: 1 errors during
```

```
collection !!!!!!!!!!!!!!!!!!!!!!!!!!!!
    ========================================= 1 error in 0.11
seconds ==========================================
    make: *** [test] Error 2
```

We get an `ImportError` from our test file because there is no module in `pgbackup` named `cli`. This is awesome because it tells us what our next step is. We need to create that file.