A CLOUD GURU

Lesson Description - Implementing Local File Storage

The last few pieces of logic that we need to implement pertain to how we store the database dump. We'll have a strategy for storing locally and on AWS S3, and it makes sense to put both of these in the same module. Let's use TDD to implement the local storage strategy of our `storage` module.

Documentation For This Video

- The `tempfile` package
- The `tempfile.TemporaryFile` class
- The `tempfile.NamedTemporaryFile` class

Writing Local File Tests

Working with files is something that we already already know how to do, and local storage is no different. If we think about what our local storage driver needs to do, it really needs two things:

1. Take in one "readable" object and one, local, "writeable" object.
2. Write the contents of the "readable" object to the "writeable" object.

Notice that we didn't say files, that's because we don't need our inputs to be file objects. They need to implement some of the same methods that a file does, like `read` and `write`, but they don't have to be file objects.

For our testing purposes, we can use the `tempfile` package to create a `TemporaryFile` to act as our "readable" and another `NamedTemporaryFile` to act as our "writeable". We'll pass them both into our function, and assert after the fact that the contents of the "writeable" object match what was in the "readable" object:

_tests/test_storage.py_

```
import tempfile

from pgbackup import storage

def test_storing_file_locally():
    """
    Writes content from one file-like to another
    """
```

```
    infile = tempfile.TemporaryFile('r+b')
    infile.write(b"Testing")
    infile.seek(0)

    outfile = tempfile.NamedTemporaryFile(delete=False)
    storage.local(infile, outfile)
    with open(outfile.name, 'rb') as f:
        assert f.read() == b"Testing"
```

Implement Local Storage

The requirements we looked at before are close to what we need to do in the code. We want to call `close` on the "writeable" file to ensure that all of the content gets written (the database backup could be quite large):

src/pgbackup/storage.py

```
def local(infile, outfile):
    outfile.write(infile.read())
    outfile.close()
    infile.close()
```