What if you want to support tarballs instead of S3?

```
from unittest.mock import MagicMock, call, patch
from newscheck.core import find_bad_articles

class TestBadArticleFinder(unittest.TestCase):
    @patch("newscheck.core.get_file")
    @patch("newscheck.core.get_pages")
    @patch("newscheck.core.boto3.client")
    def test_find_bad_articles(self, client_mock, get_pages_mock, get_file_mock):
        page_stub = [{
                "Contents": [
                    {"Key": 'news/'},
                    {"Key": 'news/good.txt'},
                    {"Key": 'news/bad.txt'}]
        }]
        get_pages_mock.return_value = page_stub

        file_stubs = [
            "lorem ipsum dolorum\n foo baz bar biz\n hello world".split('\n'),
            "MSFT 99.32\n BAC 22.3\n F 33.2\n".split('\n'),
        ]
        get_file_mock.side_effect = file_stubs

        bad_articles = list(find_bad_articles("my_bucket"))
        self.assertEqual(["news/bad.txt"], bad_articles)

        self.assertEqual([
                call(client_mock.return_value, "my_bucket", "news/good.txt"),
                call(client_mock.return_value, "my_bucket", "news/bad.txt")
            ],
            get_file_mock.mock_calls
        )
```

What if you want different checking logic?

```
from unittest.mock import MagicMock, call, patch
from newscheck.core import find_bad_articles

class TestBadArticleFinder(unittest.TestCase):
    @patch("newscheck.core.get_file")
    @patch("newscheck.core.get_pages")
    @patch("newscheck.core.boto3.client")
    def test_find_bad_articles(self, client_mock, get_pages_mock, get_file_mock):
        page_stub = [{
                "Contents": [
                    {"Key": 'news/'},
                    {"Key": 'news/good.txt'},
                    {"Key": 'news/bad.txt'}]
        }]
        get_pages_mock.return_value = page_stub

        file_stubs = [
            "lorem ipsum dolorum\n foo baz bar biz\n hello world".split('\n'),
            "MSFT 99.32\n BAC 22.3\n F 33.2\n".split('\n'),
        ]
        get_file_mock.side_effect = file_stubs

        bad_articles = list(find_bad_articles("my_bucket"))
        self.assertEqual(["news/bad.txt"], bad_articles)

        self.assertEqual([
                call(client_mock.return_value, "my_bucket", "news/good.txt"),
                call(client_mock.return_value, "my_bucket", "news/bad.txt")
            ],
            get_file_mock.mock_calls
        )
```