

7. Code coverage

Code coverage helps us understanding how much main code is being executed during the execution of test cases. In bigger project it is common to miss some code which is not covered by test cases and by running few tools we can find areas in code which are not covered by test cases.

For checking code coverage [Coverage.py](#) is popular tool but we are going to use [pytest-cov](#) which is addon specifically for pytest providing more functionality.

Add pytest-cov to *requirements.txt*. We are not adding specific versions for dependencies as we want to have latest versions by default.

```
pytest
moto
boto3
pytest-cov
```

Next add *.coveragerc* file to root of the project and add following contents. This helps pytest-cov to generate coverage reports. In this config file we are excluding code from 'tests' to be measured under coverage. There is also [branch coverage](#) enabled to make sure all possible code paths are covered.

```
[run]
omit = src/tests/*
branch = True
```

Install all dependencies including *pytest-cov*

```
pip install -r requirements.txt
```

Execute the test cases using following command to get the detailed coverage results.

```
python -m pytest "src/tests" -p no:warnings --cov="src"
```

```
(devenv) PS C:\Ashpak\Practice\CoderClub\FirstApp> python -m pytest "src/tests" -p no:warnings --cov="src"
===== test session starts =====
platform win32 -- Python 3.9.0, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Ashpak\Practice\CoderClub\FirstApp\src\tests, configfile: pytest.ini
plugins: cov-3.0.0
collected 19 items

src\tests\test_api_lambda.py ..... [100%]

----- coverage: platform win32, python 3.9.0-final-0 -----
Name                               Stmts  Miss Branch BrPart  Cover
-----
src\__init__.py                      0      0      0      0  100%
src\api_lambda.py                   59      0      8      0  100%
src\dynamodb_lambda.py              10      0      0      0  100%
-----
TOTAL                               69      0      8      0  100%

===== 19 passed in 3.68s =====
```

```
(devenv) PS C:\Ashpak\Practice\CoderClub\FirstApp> python -m pytest "src/tests" -p no:warnings --cov="src"
===== test session starts =====
platform win32 -- Python 3.9.0, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Ashpak\Practice\CoderClub\FirstApp\src\tests, configfile: pytest.ini
plugins: cov-3.0.0
collected 19 items
```

```
src\tests\test_api_lambda.py ..... [100%]

----- coverage: platform win32, python 3.9.0-final-0 -----
Name Stmts Miss Branch BrPart Cover
src\__init__.py 0 0 0 0 100%
src\api_lambda.py 59 0 8 0 100%
src\dynamodb_lambda.py 0 0 0 0 100%

TOTAL 59 0 8 0 100%

===== 19 passed in 3.68s
=====
```

HTML version of this report can also be generated using following command

```
python -m pytest "src/tests" -p no:warnings --cov="src" --cov-report html
```

this command will generate html format report under `htmlcov` folder in root with `index.html` as main page. We can click on individual code file link to get more details around code coverage inside.

Coverage report: 100%



coverage.py v6.4.1, created at 2022-06-08 23:48 +0530

Module	statements	missing	excluded	branches	partial	coverage
src__init__.py	0	0	0	0	0	100%
src\api_lambda.py	59	0	0	8	0	100%
src\dynamodb_lambda.py	0	0	0	0	0	100%
Total	59	0	0	8	0	100%

coverage.py v6.4.1, created at 2022-06-08 23:48 +0530

Next we can update `.gitignore` file for `.coverage` and `htmlcov` location

```
devenv
__pycache__
.coverage
htmlcov
```

  Having 100% code coverage means test cases are covering each line of code but it doesn't mean we are covering every scenario in test cases. It is good to have 100% test coverage but it can't assure code is being tested for every possible scenario. Apart from having good coverage we also should be focusing on having good coverage for maximum possible scenarios.