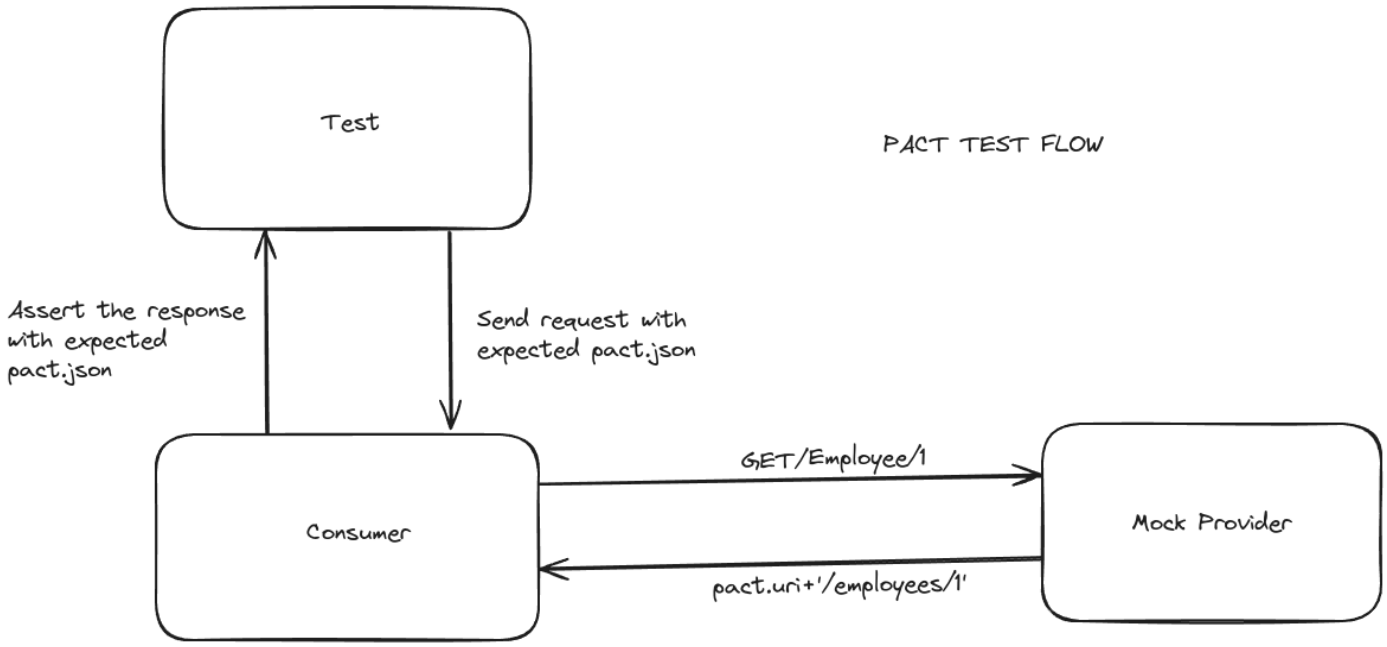
ASSIGNMENT

What is Contract Testing?

Contract testing is a testing technique used for testing endpoints, for applications which are isolated. Messages sent or received confirms that they are adhering to a written Pact or Contract. This helps the tester to confirm all calls made to the Mock Provider, returning the same response as actual service. This reference link will give more information about Contract testing.

BACKGROUND

The Python package of Pact helps in writing consumer-driven python tests. As shown in the below diagram, while running Pact Test, Consumer does not contact the actual Provider, but it will contact the Mock Provider. The mock provider will validate the defined contract and send the appropriate response. The advantage over here is you don’t have to spin up actual Provider setup.



Using pact-python

1.Define the Consumer and Provider objects that describe API endpoint and expected payload

2.Define the setup criteria for the Provider

3.Define the Consumer request using Pact

4.Define how the provider is expected to respond using Pact

5.Assert the response to validate the contract

Define the Consumer request using Pact

```
(pact
           .given('Employee creation is successful')
           .upon_receiving('a request to create an employee')
           .with_request('post', '/employees', body=expected_request)
```

Define how the Provider is expected to response using Pact

```
(pact
           .given('Employee creation is successful')
           .upon_receiving('a request to create an employee')
           .with_request('post', '/employees', body=expected_request)
           .will_respond_with(200, body=expected_response))
```

Assert the response with expected contract

```
with pact:
           result = requests.get(pact.uri+'/employees/1').json()

       # Assert the response matches the expected result
       self.assertEqual(
           result, expected, f"Actual response: {result}, Expected response:
{expected}")

       pact.verify()
```