[version_1.0]

Exercise: Testing the Application

In this exercise, you will return to the AWS Cloud9 instance and run the local build for the project. The local build contains linting (or static analysis) of the code and unit tests. Next, you will run an integration test. The integration test finds the API Gateway Websocket endpoint, and it simulates a player completing a game. You will then add a simple feature to the code. Finally, you will fix the unit tests and commit the changes to your repository.

Task 1: Running unit tests

In this task, you run unit tests against the application code. You then view the reports that are outputted from these tests.

1. In your **AWS Cloud9** terminal, make sure you are still in the `trivia-app` folder. The prompt should also show the top-level Git branch.

   ```
   cd ~/environment/trivia-app
   ```

2. Install the backend code requirements and the unit test requirements by using `pip3` to run the following commands.

   ```
   pip3 install -U -r back-end-python/gameactions/requirements.txt
   pip3 install -U -r back-end-python/tests/requirements.txt
   ./local_build.sh
   ```

   You should see the code rating after `pylint` does its static analysis:

   ```
   -----------------------------------
   Your code has been rated at 10.00/10
   ```

   Then, `pytest` runs its unit tests, which are shown like the following example:

   ```
   back-end-python/tests/unit/test_handler.py .........
   ```

   It will also output a coverage report in HTML, which is in `/trivia-app/htmlcov`.

3. In the `htmlcov` folder, right-click the `index.html` file and view the coverage report by choosing **Preview**.

Task 2: Running the integration test

For this task, you run an integration test against the application code.

1. Start the integration test (which will simulate a game) by running the following command.

```
AWS_SAM_STACK_NAME=trivia-app pytest -s back-end-python/tests/integration/test_api_gatev
```

The integration test uses the AWS SAM stack to find the Websocket endpoint, so the stack is passed in the `AWS_SAM_STACK_NAME` environment variable.

At the end of the test, you should see the following message: ***1 passed***

Task 3: Updating the code and unit tests

In this task, you update the application code to change the score value from *10* to *20*. You also update the unit test code.

1. Open the `trivia-app/back-end-python/gameactions/app.py` file.
2. Scroll down to the `trivia_calculate_scores` function.
3. Update the code so the game increments the score by `20` instead of by `10`. Locate `score += 10` and change it to `score += 20`. Make sure to save the file.

   ```
   score += 20
   ```

4. Verify that you left the code in a stable state by running the `./local_build.sh` command.

   At the top of the failure report, you should see that the `test_trivia_calculate_scores_correct` test failed:

   ```
   ================================================ FAILURES ================================
   _____________________________________ test_trivia_calculate_scores_correct _____________
   ```

   You also need to update the unit test to match the new score.

5. Open the `trivia-app/back-end-python/tests/unit/test_handler.py` file.
6. Find the `test_trivia_calculate_scores_correct` function.
7. In the function's `TABLE.update_item.assert_called_with` call, find the `AttributeUpdates` field and update the `Value` from `10` to `20`.

   ```
   AttributeUpdates={'score': {'Value': 20, 'Action': 'PUT'}}
   ```

8. You must also update `app.MANAGEMENT.post_to_connection.assert_has_calls` (which tests API Gateway Websocket calls). Change the value of `score` from `10` to `20`.

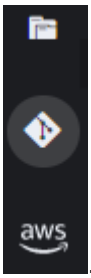
   ```
   mock.call(Data='{"action": "playerlist", "players": [{"connectionId": "connection-1", "p
   ```

9. Save the file and re-run `./local_build.sh`. The test summary should again show that everything passed.

Task 4: Committing to source control through the AWS Cloud9 GUI or the AWS CLI

In this task, you commit the changes from the previous tasks to your code repository. *You can commit the code by using either the **AWS Cloud9 GUI**, or AWS CLI commands in the **AWS Cloud9 terminal***.

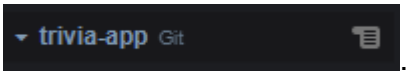
1. In the menu, at the left of **AWS Cloud9**, choose the source control icon:



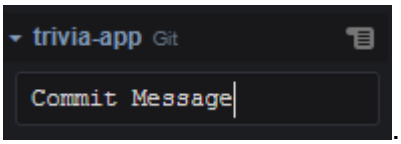
2. Under **Changes**, select either the `app.py` or `test_handler.py` file to view the diffs between the original files and your changes.

3. Under **Changes**, stage the changes in `app.py` and `test_handler.py` for a commit by placing the pointer over the file name and choosing the stage icon (**+**) for the file. The files should now appear under **Staged Changes**.

4. In the **Source Control** pane, at the right side of the **trivia-app** heading, left-click paper scroll icon and choose **Commit**. Choose **Yes**.



5. Under the **trivia-app** heading, enter the following commit message and press Enter:
`Scores now increment by 20`



6. In the **trivia-app** heading, left-click the paper scroll icon and choose **Push to…**. When prompted to select a remote, choose the remote named **origin**.

▶ Commit changes through the AWS CLI