[version_1.0]

Note

The exercises in this course will have an associated charge in your AWS account. In this exercise, you will create the following resources:

- AWS CodePipeline pipeline

This exercise includes instructions to delete all the resources that you created in the exercises.

Familiarize yourself with **AWS CodePipeline pricing** and the **AWS Free Tier**.

Exercise: Creating a Pipeline

In this exercise, you will first start by using AWS CodePipeline to create a new pipeline. You will create a new feature branch in your repository and edit some application code. You will then commit the changes to your feature branch and merge the feature branch into your main branch.

Task 1: Creating a pipeline

In this task, you use AWS CodePipeline to create a pipeline.

1. In the console, choose **Services**, and search for and select **CodePipeline**.
2. Choose **Create pipeline**.
3. For **Pipeline name**, enter `trivia-pipeline` and choose **Next**.
4. In the **Add source stage** step, configure the following settings:
    - **Source provider**: *AWS CodeCommit*
    - **Repository name**: *trivia-app*
    - **Branch name**: *main*
5. Choose **Next**.
6. In the **Add build stage** step, configure the following settings.
    - **Build provider**: *AWS CodeBuild*
    - **Project name**: *trivia-unittests*
7. Choose **Next**.
8. In the **Add deploy stage** step, choose **Skip deploy stage** and then choose **Skip**.

9. In the **Review** step, choose **Create pipeline**. You should see a *Success* message.

Task 2: Creating a feature branch

In this task, you create a new feature branch in your AWS CodeCommit repository.

1. In a separate browser tab, switch to the **AWS Cloud9** development environment. Make sure that you are in the `trivia-app` directory.

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

   Create a feature branch by running the following command:

   ```
   git checkout -b feature-bonus-scores
   ```

   You should see that you created a new branch and switched to that branch:

   ```
   Switched to a new branch 'feature-bonus-scores'
   ```

Task 3: Editing the code

In this task, you edit the code to implement a bonus scores feature.

1. Open the `trivia-app/back-end-python/gameactions/app.py` file.
2. In the `trivia_calculate_scores` function, locate the code where `last_answer` is set.
3. On the next line, add code to set the `bonus` variable:

   ```
   last_answer = connection["lastAnswer"] if "lastAnswer" in connection else ""
   bonus = question["bonus"] if "bonus" in question else 0
   ```

4. You also need to add the `bonus` variable to the calculation logic. Update the code incrementing `score` so that it includes `bonus`.

   Code before:

   ```
   if last_question_id == question["id"] and last_answer == question["answer"]:
       score += 20
   ```

   Code after:

   ```
   if last_question_id == question["id"] and last_answer == question["answer"]:
           score += 20 + bonus
   ```

5. Save the file.

6. You also need to update the unit tests code so that it tests the new bonus score. Open the `trivia-app/back-end-python/tests/unit/test_handler.py` file.

7. Replace `SCORES_EVENT` with a new event that includes a bonus score:

```
SCORES_EVENT = {
"gameid" : "01234567012301230123012345678901",
"questions" : [
    { "id" : "q-1111", "question" : "Good question?", "answer" : "Yes", "bonus": 20},
],
"iterator" : { "questionpos" : 0 }
}
```

   Save the file.

8. Find the `test_trivia_calculate_scores_correct` section and change the assert statements to expect a score of 40.

```
app.TABLE.update_item.assert_called_with(
    Key={'gameId': '01234567012301230123012345678901', 'connectionId': 'connection-1'},
    AttributeUpdates={'score': {'Value': 40, 'Action': 'PUT'}}
)

app.MANAGEMENT.post_to_connection.assert_has_calls([
    mock.call(Data='{"action": "playerlist", "players": [{"connectionId": "connection-1'
    mock.call(Data='{"action": "gameover"}', ConnectionId='connection-1')
    ])
```

9. Save the file.

10. Verify that the code is stable by running the unit test.

```
./local_build.sh
```

    You should see that everything passed.

11. (*Optional*) You can verify that the coverage is still 100 percent by previewing the `htmlcov/index.html` file.

Task 4: Committing and merging the new code

In this final task, you commit the new code to your feature branch in AWS CodeCommit. You then merge the changes into the `main` branch.

1. In the **AWS Cloud9** terminal, view the status of your git repository by running the following command:

```
git status
```

You should see that status of the two files changed to *not staged for commit*.

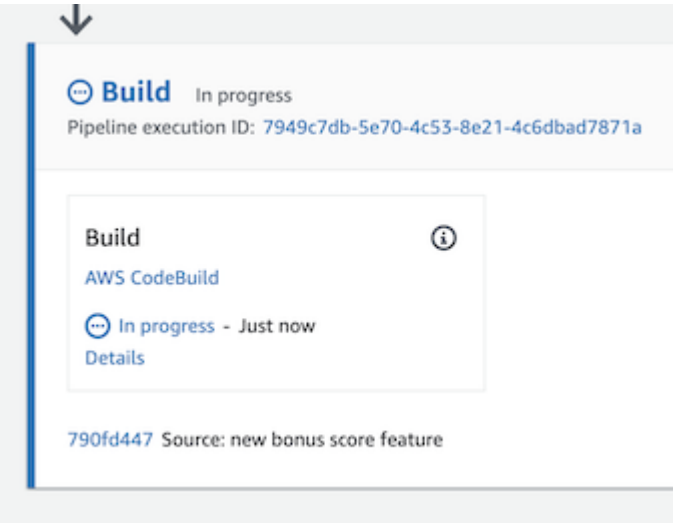
```
modified:   back-end-python/gameactions/app.py
modified:   back-end-python/tests/unit/test_handler.py
```

2. Add the files, create a commit, and push the changes to the `origin` remote.

```
git add *
git commit -m "new bonus score feature"
git push origin feature-bonus-scores
```

   As of now, you haven't made any changes to the `main` branch yet.

3. Switch back to the **CodePipeline** tab.
4. Choose **Services** and then select **CodeCommit**.
5. In CodeCommit, open the `trivia-app` repository and in the navigation pane, choose **Commits**.
6. On the right, view the bonus score commit by opening the dropdown menu with `main` and selecting `feature-bonus-scores`.
7. At the top of the window, choose the `trivia-app` breadcrumb.
8. Choose **Create pull request**.
9. For **Destination**, keep `main` selected and for **Source**, choose `feature-bonus-scores`.
10. Choose **Compare**. You can scroll down to see the code changes that you made.
11. In **Details > Title**, enter `New feature: Bonus scoring`. Choose **Create pull request**.
12. Choose **Merge**.
13. Keep both **Fast forward merge** and **Delete source branch feature-bonus-scores after merging?** selected. Choose **Merge pull request**.
14. In the navigation pane, see the new merge commit by choosing **Commits**.
15. In the navigation pane, open the **CodePipeline** console by choosing **Pipeline > Pipelines**.
16. View the pipeline details by opening **trivia-pipeline**. In the **Source** section, you should see the new commit: *Source: new bonus score feature*
17. Review the **Build** section. The recently merged commit on the `main` branch triggered a pipeline build.



pipeline_build

18. Switch back to the **AWS Cloud9** tab.
19. The new features you committed have been merged to `main`. Update the main branch locally by running the following commands:

    ```
    git checkout main
    git pull origin main
    git log
    ```

    In the Git log, you should see the *new bonus score feature* commit.

Task 5: Deleting all exercise resources

Congratulations! You have successfully completed the course project. In this task, you delete the AWS resources that you created for this project.

1. Open the **AWS CodePipeline** console.
   - Delete the **trivia-pipeline** pipeline.
   - Delete the **trivia-unittests** build project.
2. Open the **AWS CodeCommit** console.
   - Delete the **trivia-app** repository.
3. Open the **Amazon Simple Storage Service (Amazon S3)** console.
   - Empty and delete the **-trivia-app-bucket**.
   - Empty and delete the **aws-sam-cli-managed-default-samclisourcebucket-** bucket.
   - Empty and delete the **codepipeline-** bucket.
4. Open the **AWS CloudFormation** console.
   - Delete the **trivia-app** stack.
   - Delete the **aws-sam-cli-managed-default** stack.
5. Open the **AWS Identity and Access Management (IAM)** console.
   - Delete the following IAM roles.
     - **AWSCodePipelineServiceRole-<region>-trivia-pipeline**
     - **codebuild-trivia-unittests-service-role**
     - **cwe-role-region-trivia-pipeline**
6. Open the **AWS Cloud9** console.
   - Delete the **trivia-game** environment.