Write a Unit Test

**SPL-BE-200-DVWAUT-1 - Version 1.0.1**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab overview**

This lab picks up where the *Commit a Bugfix* lab left off. Git has been configured for a generic Student and the PresidentsApp repository in AWS CodeCommit has already been cloned to the AWS Cloud9 instance. The main.html file has been corrected so that the banner image displays correctly.

This lab demonstrates how to use AWS CodePipeline to perform unit testing for the Presidents application. Specifically, it tests the logic used to determine their age at death. You received a report that the age for President John Adams is inaccurate, which indicates that there’s a bug in the logic. You start by testing the current unit test to ensure it passes. However, you find that it’s not accurate because it calculates the Presidents’ ages based on subtracting the year they died from the year they were born. To correct this inaccuracy, you perform an update to the application logic. You use the relativedelta type in the application code logic to ensure that the application calculates the time between the President’s date of birth until the specific day they died.

After you update the application, you commit the changes and push the updates to the PresidentsApp CodeCommit repository. The Presidents-Pipeline is invoked again. After the unit test completes successfully, the application update is deployed using CodeDeploy. Finally, you verify that the application calculates the ages correctly.

OBJECTIVES

By the end of this lab, you will be able to do the following:

* Verify that the unit test passes.
* Update the Presidents application to calculate the ages of the presidents by using the relativedelta type.
* Push changes to the PresidentsApp CodeCommit repository.
* Verify that teh application calculates the ages correctly after the pipeline finishes the new unit testing and deployment stages.

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab:

* Familiarity with the basic navigation of the AWS Management Console.
* Versed in editing and running scripts by using an AWS Cloud9 code editor and terminal.
* A basic understanding of and familiarity with Git, AWS CodePipeline, and AWS CodeDeploy.
* Prior experience with AWS services and serverless computing is helpful, but isn’t required.

DURATION

This lab requires *60* minutes to complete.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Command:** A command that you must run.
* **Expected output:** A sample output that you can use to verify the output of a command or edited file.
* **Note:** A hint, tip, or important guidance.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING:** **Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

**Task 1: Verify if there is a bug in the application**

In this task, you open the Presidents application and compare whether the age for President John Adams matches the age that’s listed on Wikipedia.

1. From the navigation pane at the left of the screen, copy the **WebsiteURL** value and open it in a new browser tab.
2. Verify the **age** for **President John Adams** that’s listed on the application. The application calculates his age to be **91** years old at the time of his passing.
3. From the navigation pane at the left of the screen, copy the **JohnAdamsWikipediaURL** link and open it in a new browser tab.
4. In the **Personal Details** section, look at the column with John Adams’ portrait. The column shows that he **Died** on *July 4, 1826 (aged 90)*.

This information confirms that the application has a bug with calculating accurate ages. You address this bug in the next set of tasks.

 Congratulations! You have reviewed the Presidents application and verified that the logic used to determine the ages of the presidents doesn’t work properly. The application logic needs to be updated.

**Task 2: Update the application logic and unit tests**

In this task, you connect to the AWS Cloud9 environment and install the dependencies for the application and the unit test. After you install the dependencies, you start unit testing locally in the AWS Cloud9 environment. You identify the flaw in the logic that’s used to test the age calculation. Then, you update the logic to calculate the age based on a time interval according to a specific date. You also update the unit test accordingly.

TASK 2.1: INSTALL THE DEPENDENCIES FOR THE APPLICATION AND THE UNIT TEST

1. From the navigation pane at the left of the screen, copy the **Cloud9Environment** URL value and in a new browser tab, open the URL.
2. Close the **Welcome** tab and to have more viewing area, expand the **terminal session** pane.
3. **Command:** To change directories to the **~/environment/PresidentsApp** directory and install the application dependencies (based on the **app/requirements.txt** file), run the following grouped commands:

cd ~/environment/PresidentsApp; pip install -r app/requirements.txt

**Expected output:** Output has been truncated.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Installing collected packages: zipp, urllib3, six, pyyaml, MarkupSafe, jmespath, itsdangerous, click, Werkzeug, python-dateutil, Jinja2, importlib-metadata, flask, botocore, s3transfer, boto3

Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.2 Werkzeug-2.2.3 boto3-1.26.107 botocore-1.29.107 click-8.1.3 flask-2.2.3 importlib-metadata-6.1.0 itsdangerous-2.1.2 jmespath-1.0.1 python-dateutil-2.8.2 pyyaml-6.0 s3transfer-0.6.0 six-1.16.0 urllib3-1.26.15 zipp-3.15.0

WARNING: You are using pip version 22.0.4; however, version 23.0.1 is available.

You should consider upgrading via the '/home/ec2-user/.pyenv/versions/3.9.16/bin/python3.9 -m pip install --upgrade pip' command.

1. **Command:** Install the unit testing requirements (based on the **tests/requirements.txt** file) by running the following command:

pip install -r tests/requirements.txt

**Expected output:** Output has been truncated.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Successfully installed astroid-2.15.2 attrs-22.2.0 coverage-7.2.3 dill-0.3.6 exceptiongroup-1.1.1 iniconfig-2.0.0 isort-5.12.0 lazy-object-proxy-1.9.0 mccabe-0.7.0 packaging-23.0 platformdirs-3.2.0 pluggy-1.0.0 pylint-2.17.2 pytest-7.2.2 pytest-cov-4.0.0 pytest-mock-3.10.0 tomli-2.0.1 tomlkit-0.11.7 typing-extensions-4.5.0 wrapt-1.15.0

You have now installed all the requirements for the application and for unit testing.

TASK 2.2: START UNIT TESTING FROM THE AWS CLOUD9 ENVIRONMENT

Now that the requirements for the application and unit tests are installed, you can run a unit test locally.

Before you run the unit test, review the code for the unit test to understand the logic that’s being used.

1. Open the **~/environment/PresidentsApp/tests/test\_handler.py** file.

When you look at the unit test, you see that the response for *President X* is data that’s mocked from the data that’s in the Amazon DynamoDB table. The mock data matches the values for President John Adams and it verifies that the age equals *91*.

1. **Command:** Run the following command to run a unit test:

./run\_tests.sh

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

==================== test session starts ====================

platform linux -- Python 3.9.16, pytest-7.2.2, pluggy-1.0.0

rootdir: /home/ec2-user/environment/PresidentsApp

plugins: mock-3.10.0, cov-4.0.0

collected 1 item

tests/test\_handler.py . [100%]

---------- coverage: platform linux, python 3.9.16-final-0 -----------

Coverage HTML written to dir htmlcov

==================== 1 passed in 0.44s ====================

Though the unit test passes, you know that the correct age isn’t *91* and that this value needs to be updated.

TASK 2.3: UPDATE THE APPLICATION LOGIC

The unit test passes with the wrong logic. It’s time to update the unit test and the application code that calculates the ages of the presidents.

1. Modify the **test\_handler.py** file by updating the **age** value from **91** to **90**.
2. Save your changes to the file.

If you ran the unit test again, it would fail. Next, you need to update the logic for the Python script that calculates the age.

1. Open the **~/environment/PresidentsApp/app/presidents.py** file.

If you look at the end of the script, you see a comment reading *Initial logic to calculate age*. The logic that calculates the age is on the following line. It should match the following code snippet:

president["Aged"] = president["Died"].year - president["Born"].year

The logic calculates the age by subtracting the birth year from the death year. This logic is flawed because it doesn’t consider the age of the person if they haven’t reached their birthday for that year.

To calculate the exact time between birth and death, you need logic that calculates the actual time instead of only years. The solution is to use the *relativedelta* type. In short, *relativedelta* is a Python tool you can use to calculate different amounts of time for a range of given dates or times. To learn more about *relativedelta*, see the [Additional resources](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2FSPL-BE-200-DVWAUT-1%3A1.0.1-5adf2161/en-US#additional_resources) section.

1. In the **presidents.py** file (just after the commented line that reads **#Initial logic to calculate age**), either comment out or delete the following line of code:

president["Aged"] = president["Died"].year - president["Born"].year

1. Uncomment the line of code that follows the **#relativedelta solution** comment:

# relativedelta solution

president["Aged"] = relativedelta(president["Died"], president["Born"]).years

1. Save the changes to the file.

**Note:** A solution file named *presidents\_solution.py* is available if you need to refer to it.

With the logic in calculating the age updated and the unit test value for the age being updated, run the unit test again.

1. **Command:** Run the unit test with the following command:

./run\_tests.sh

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

==================== test session starts ====================

platform linux -- Python 3.9.16, pytest-7.2.2, pluggy-1.0.0

rootdir: /home/ec2-user/environment/PresidentsApp

plugins: mock-3.10.0, cov-4.0.0

collected 1 item

tests/test\_handler.py . [100%]

---------- coverage: platform linux, python 3.9.16-final-0 -----------

Coverage HTML written to dir htmlcov

==================== 1 passed in 0.44s ====================

The unit test passed.

**Note:** If your tests aren not passing, verify you have updated the unit test to expect the corrected *aged 90* response, you have updated the *presidents.py* calculation to use the *relativedelta* type, and that you have saved both files.

 Congratulations! You have updated the unit test to expect a value of *90* instead of *91* for the data that’s mocked from the DynamoDB table. You also updated the application logic to calculate the ages by using the Python *relativedelta* type.

**Challenge A: Push the updates and test the application**

Welcome to this lab’s challenge. At this point, you have updated both the application logic that calculates ages and the unit test. It’s now time to push the updates to the PresidentsApp CodeCommit repository.

Now that you’re familiar with using Git, your challenge is to push the changes in the application to the PresidentsApp repo.

If you get stuck on the commands to use and the order to run them in, refer to the following hints.

**First Hint**

**Second Hint**

**Third Hint**

**Last Hint**

After the code is pushed to the CodeCommit repo, this action invokes the Presidents pipeline. The unit test runs and if it is successful, CodePipeline deploys the updated application code to the Amazon Elastic Compute Cloud (Amazon EC2) instance.

To follow the progress of the pipeline, you can open the CodePipeline console.

1. At the top of the AWS Management Console, in the search bar, search for and choose

CodePipeline

.

1. Choose the link for the **Presidents-Pipeline**.
2. Observe the pipeline as it progresses through each stage.
3. After the pipeline finishes successfully, return to the browser tab with the Presidents App frontend.
4. Refresh the page and observe the change to the age for President John Adams.

The Presidents application now shows an age of **90**.

**Note:** If you look closely, you see that the ages of a couple other presidents also changed.

 Congratulations! You have successfully completed the challenge and pushed the updates to the PresidentsApp repo which then invoked the pipeline that ran a unit test and then deployed the application to the web server. After refreshing the Presidents App frontend, you were able to see that the ages were being calculated correctly.

**Conclusion**

 Congratulations! You now have successfully:

* Verified that the unit test passed.
* Updated the Presidents application to calculate the ages of the presidents by using the relativedelta type.
* Pushed changes to the PresidentsApp CodeCommit repository.
* Verified that the application is calculating the ages correctly after the pipeline finished the new unit testing and deployment stages.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional resources**

* To learn more, see [relativedelta](https://dateutil.readthedocs.io/en/stable/relativedelta.html).

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).