😊 Adding Unit Test

What is a Unit Test?

A **unit test** is a type of automated test that focuses on verifying the correctness of a small, isolated part of an application—typically a single function or method. It ensures that the logic of individual components works as expected.

In the example above, a simple unit test was written to check whether an integer variable (i) holds the correct value and whether a Boolean variable (result) is correctly set based on a condition. The test uses an **Assert** statement to validate the expected outcome.

Use Case of Unit Tests

Unit tests are primarily used in software development to:

- Validate that individual functions, methods, or classes work correctly.
- Catch errors early in development before they become bigger issues.
- Ensure that changes or updates to the code do not break existing functionality.

In the **build pipeline**, unit tests help maintain code quality by running automatically whenever changes are pushed to the repository. This ensures that faulty code does not proceed further in the development or deployment process.

Benefits of Unit Testing

1. **Early Bug Detection** – Identifies issues at an early stage, reducing debugging effort later.
2. **Code Reliability** – Ensures that each component of the application behaves as expected.
3. **Faster Development** – Developers can quickly test and verify code changes without manual testing.
4. **Easier Refactoring** – Helps maintain existing functionality when modifying or refactoring code.
5. **Automated Testing in CI/CD Pipelines** – Enables seamless integration of tests in the **Azure DevOps pipeline**, preventing bad code from being deployed.

By incorporating unit tests into the build pipeline, developers can improve **code stability, reduce deployment risks, and enhance overall software quality**.

😄 To begin with the Lab

This Lab aims to demonstrate how to incorporate **unit tests** into a build pipeline.

The process begins by ensuring that unit tests are included in the project. This is done by adding a separate **unit test project** within the **Visual Studio Solution** alongside the main web application.

The example walks through:

a. **Cloning the repository** from Azure DevOps into a local machine.

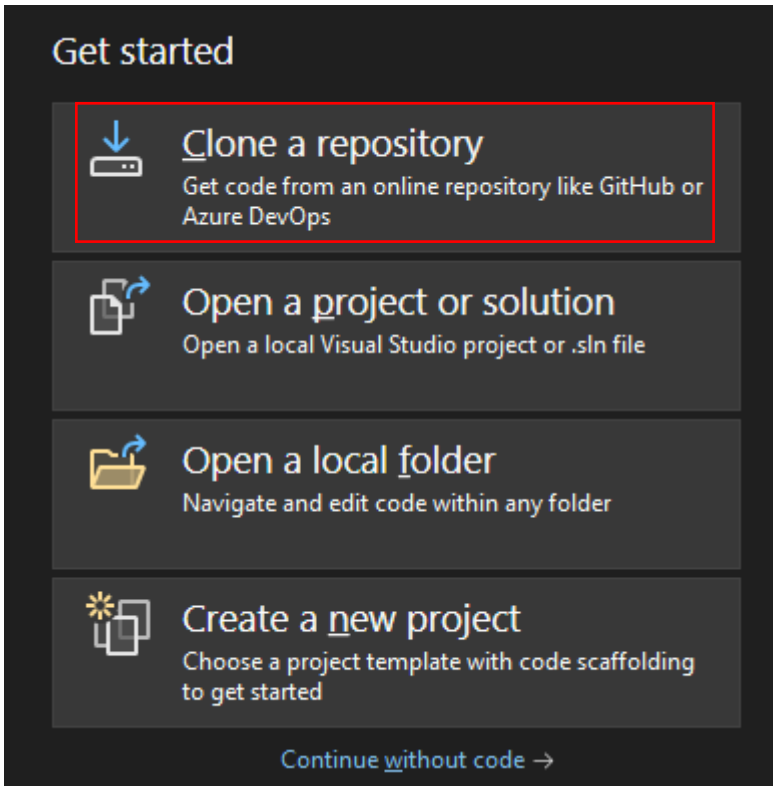
b. **Adding a test project** (using **xUnit**) to the solution.

c. **Writing a basic unit test**, where an Assert statement checks an integer variable for a specific value.

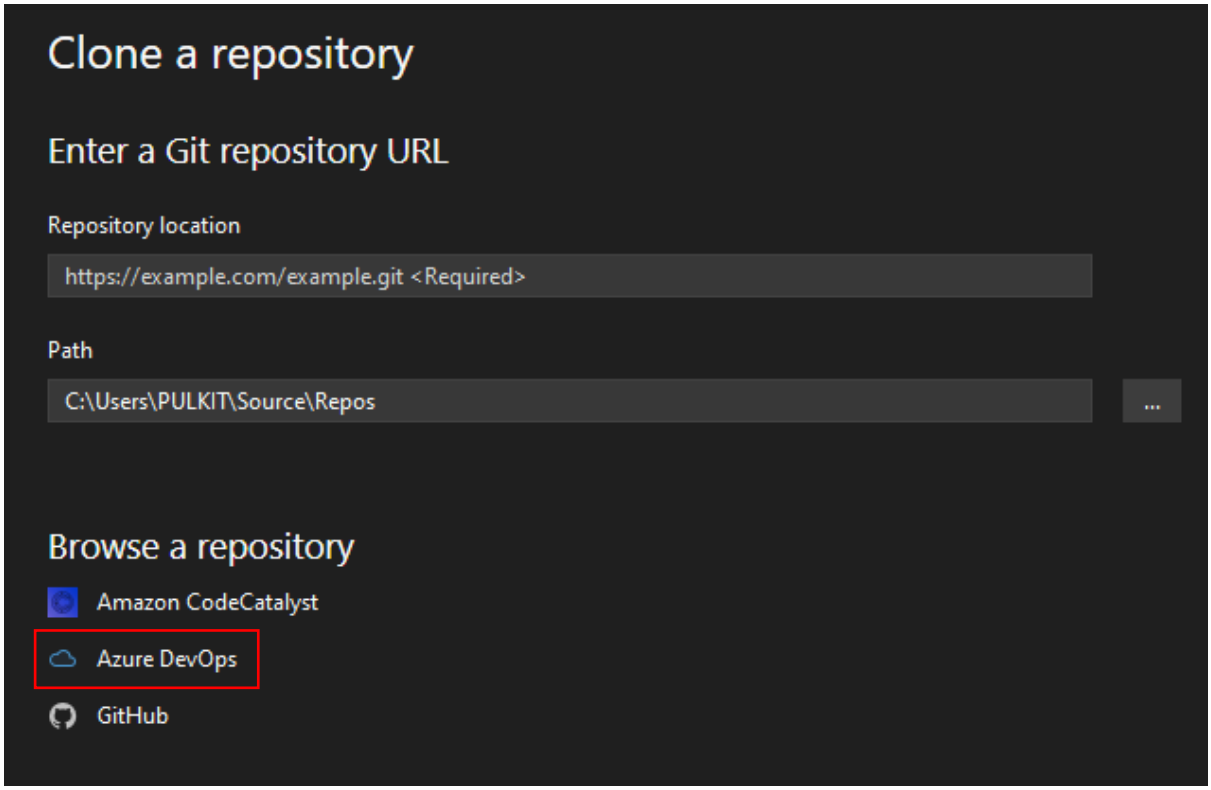
d. **Running the test in Visual Studio**, ensuring it passes successfully.

The key takeaway is that unit tests should be part of the local project before integrating them into the build pipeline.

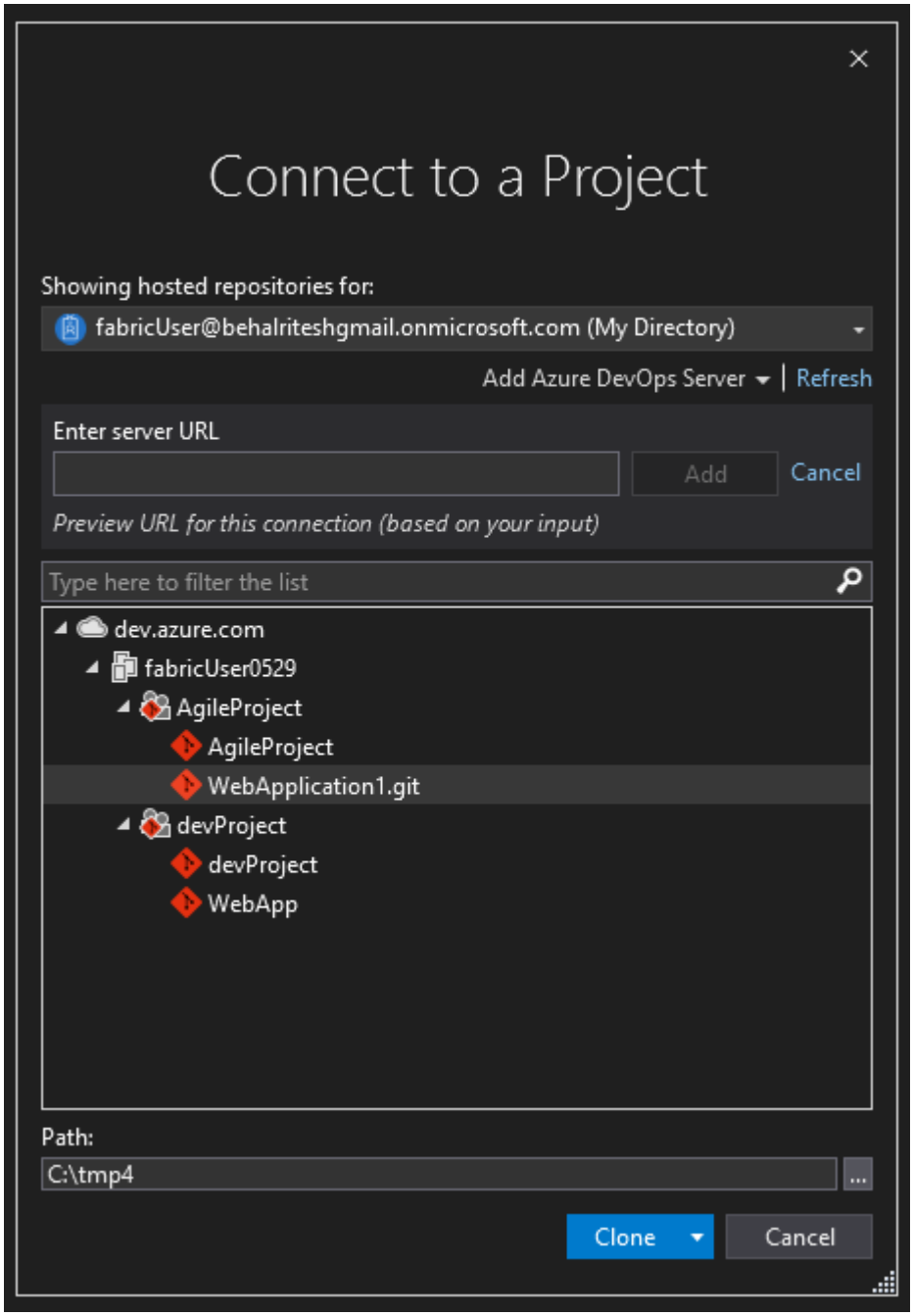
1. Open the Visual Studio and click on Clone a Repository.



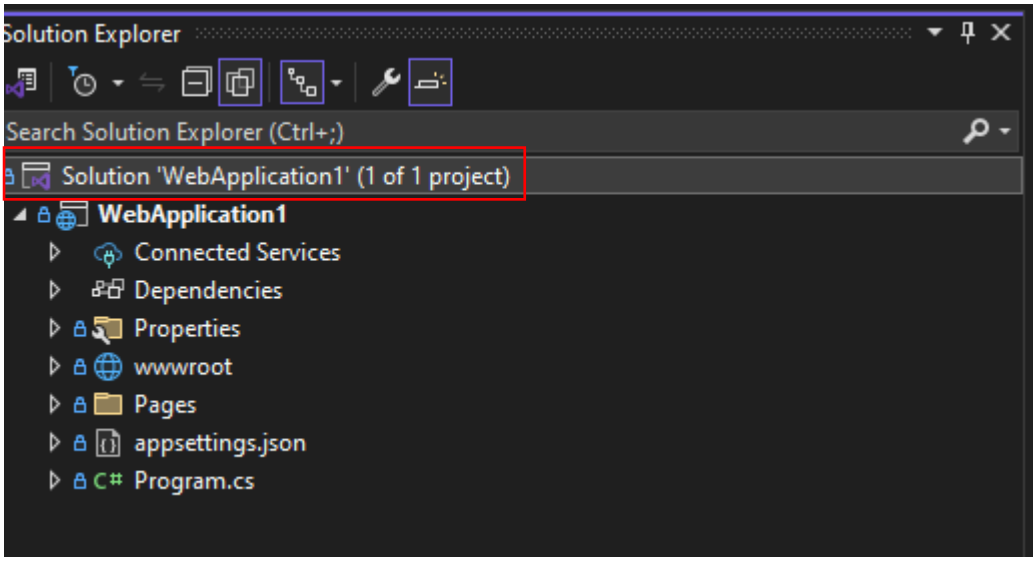
2. Then you need to clone your repository using Azure DevOps.

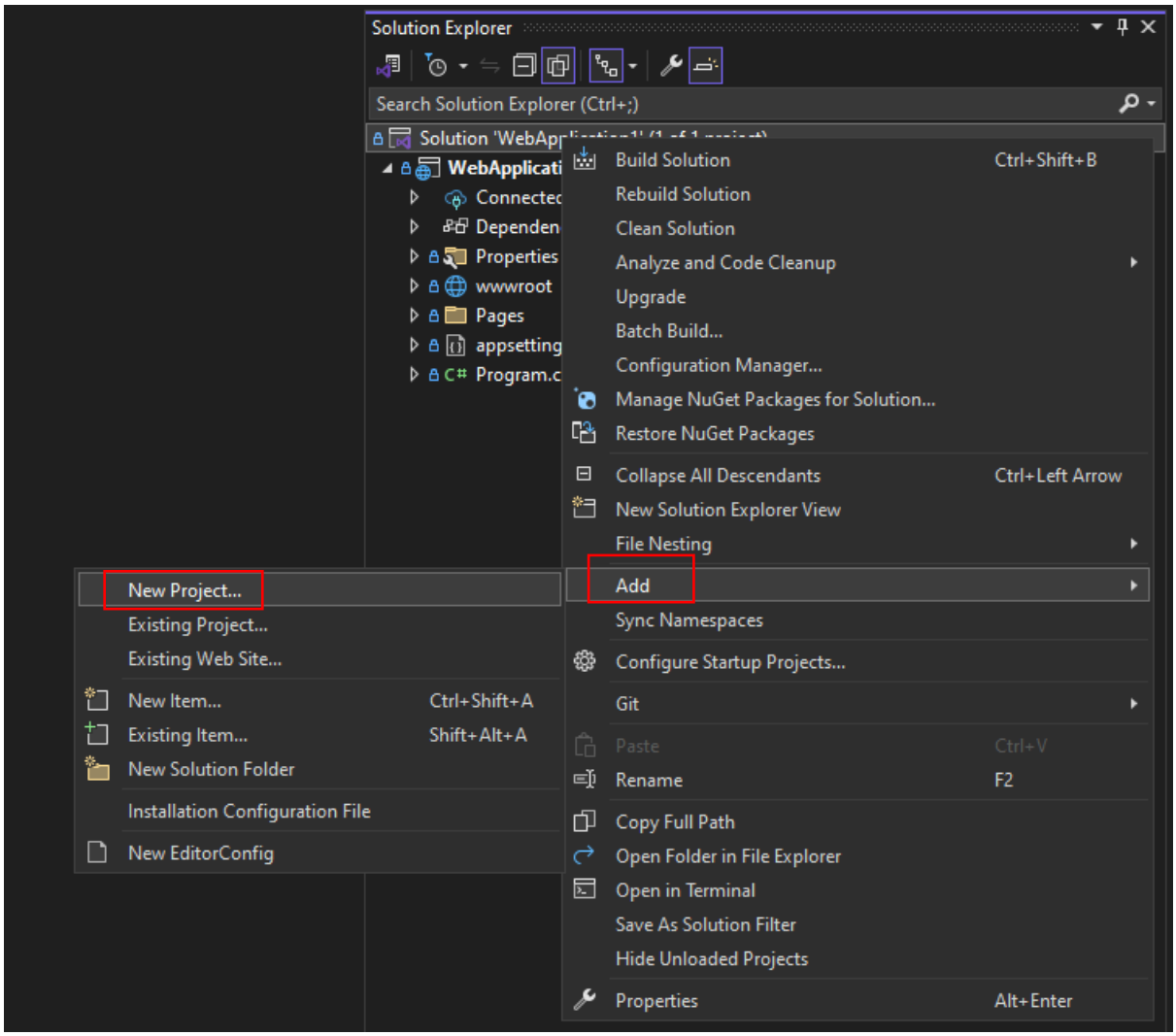


3. After that choose your repository and the path where you want to store the project then click on clone.

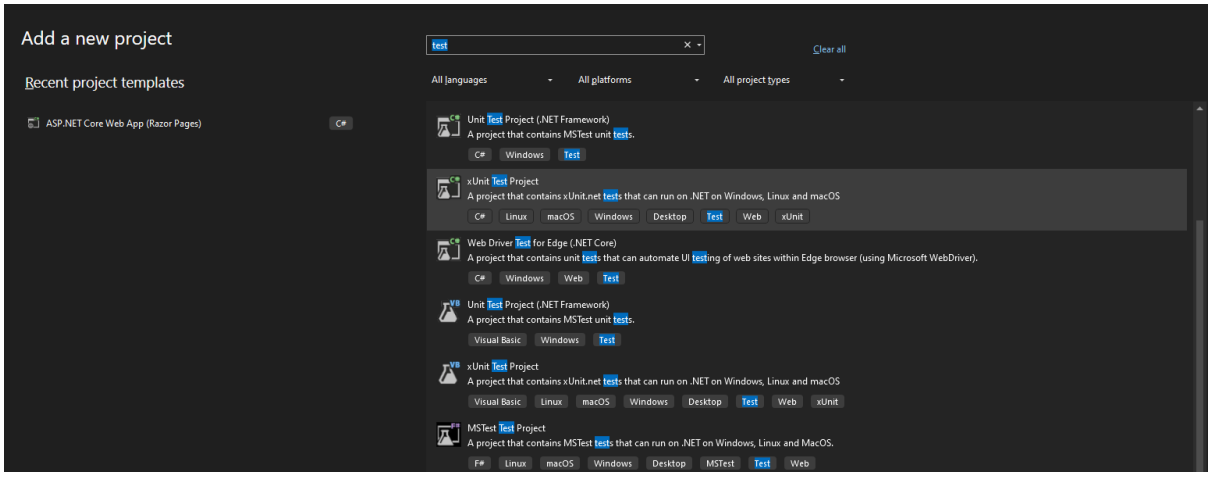


4. Once your project is opened in Visual Studio open the solution file then right-click on it to add a new project.

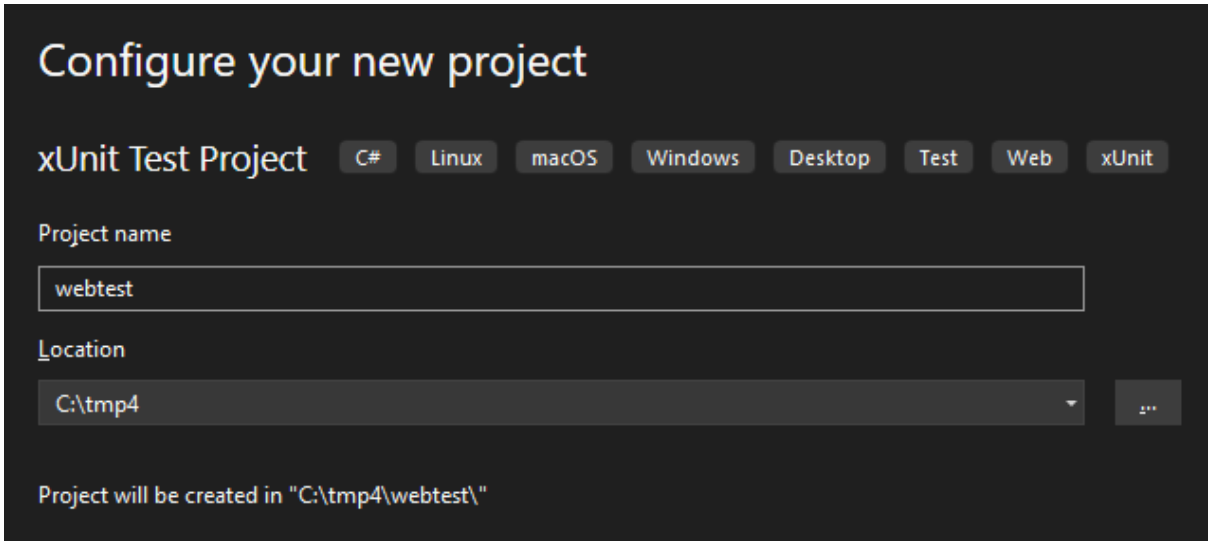


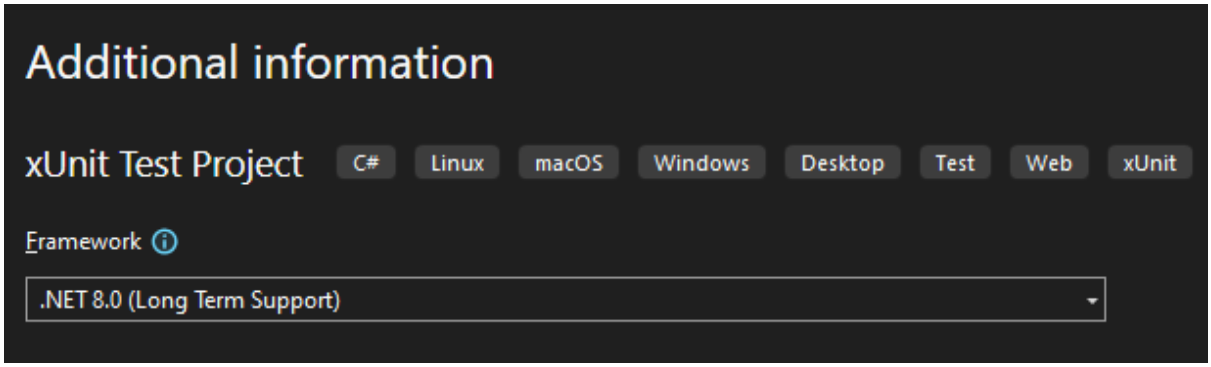


5. Search for **test** and choose the **xUnit test project** based on C#. Click on Next.

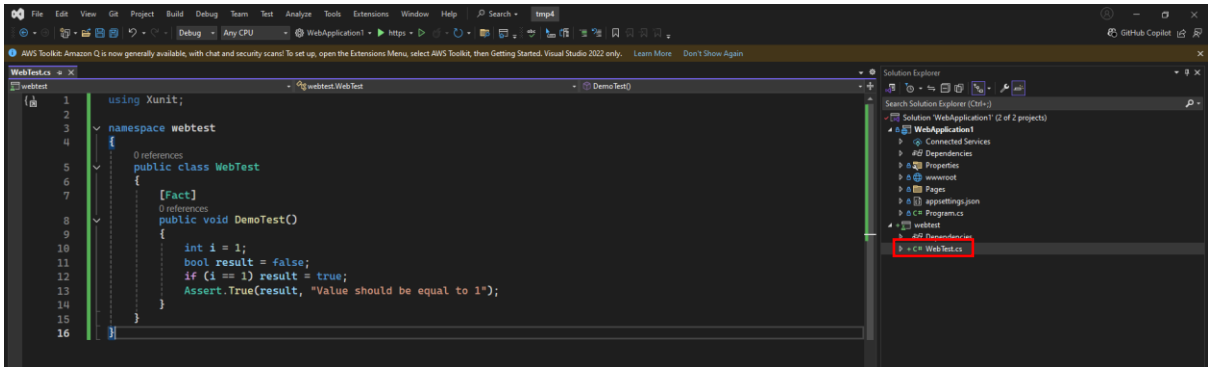


6. Then name your project and click on next and choose the desired framework. Just click on Create.

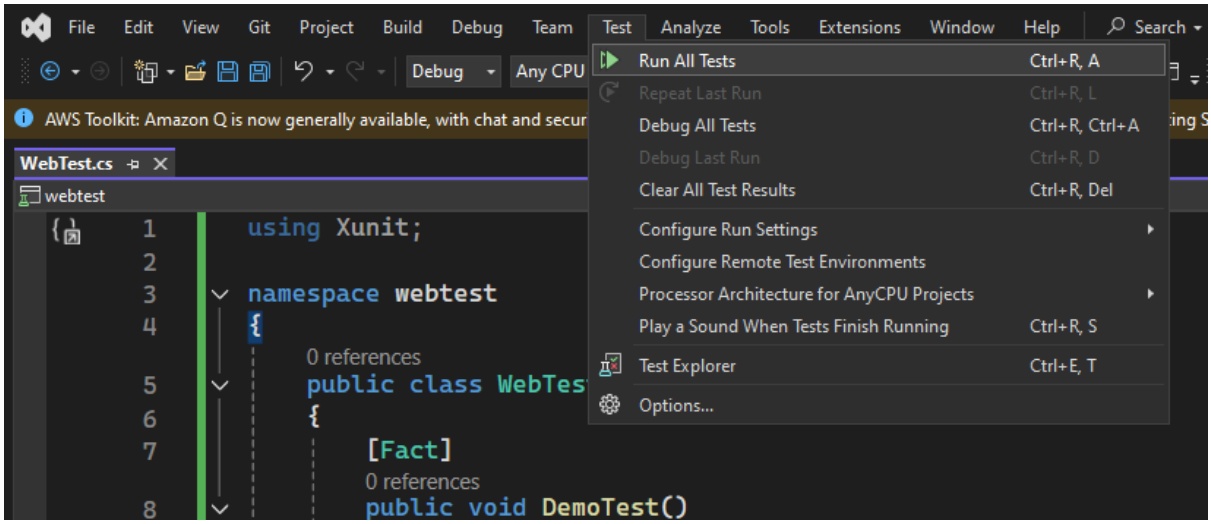




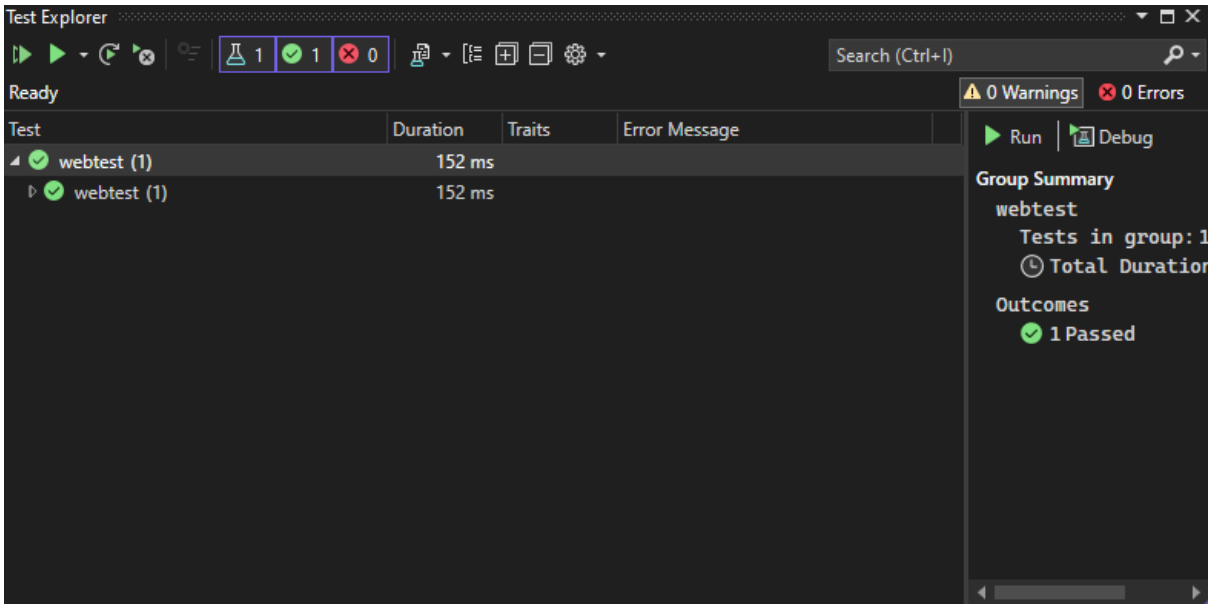
7. The first thing we need to do is rename the file from unit test to web test as you can see below in the snapshot. Then you need to add the code in this file and save it.



8. After saving everything you need to click on test and choose to run all the tests.

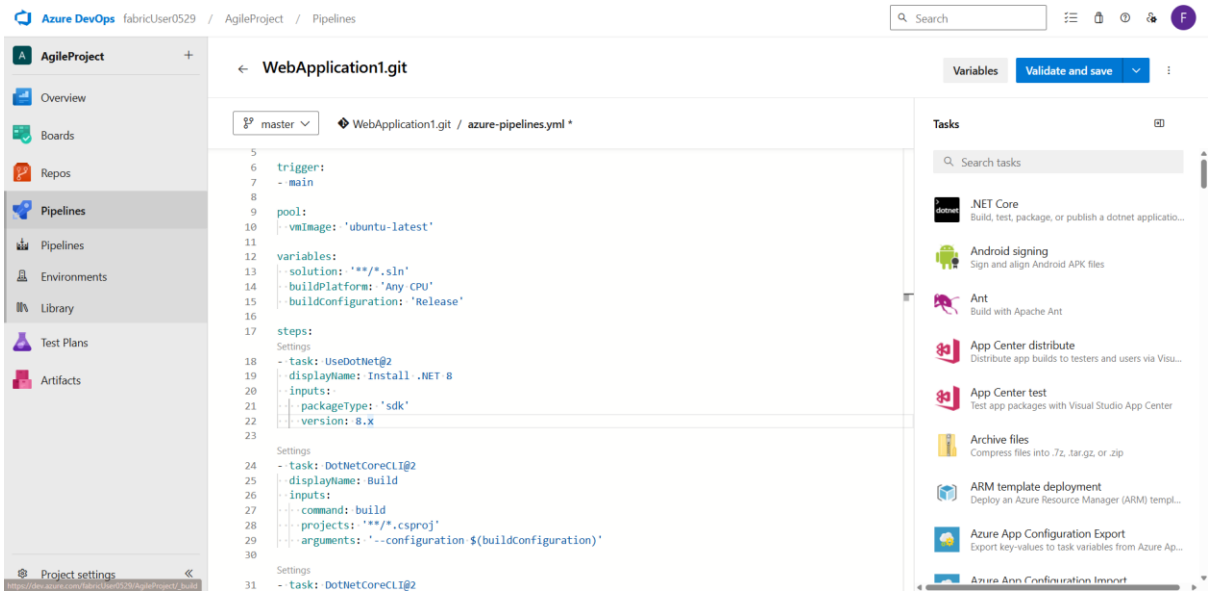


9. Here you can see that our test has been passed because it worked. It is passing because of the assert statement.

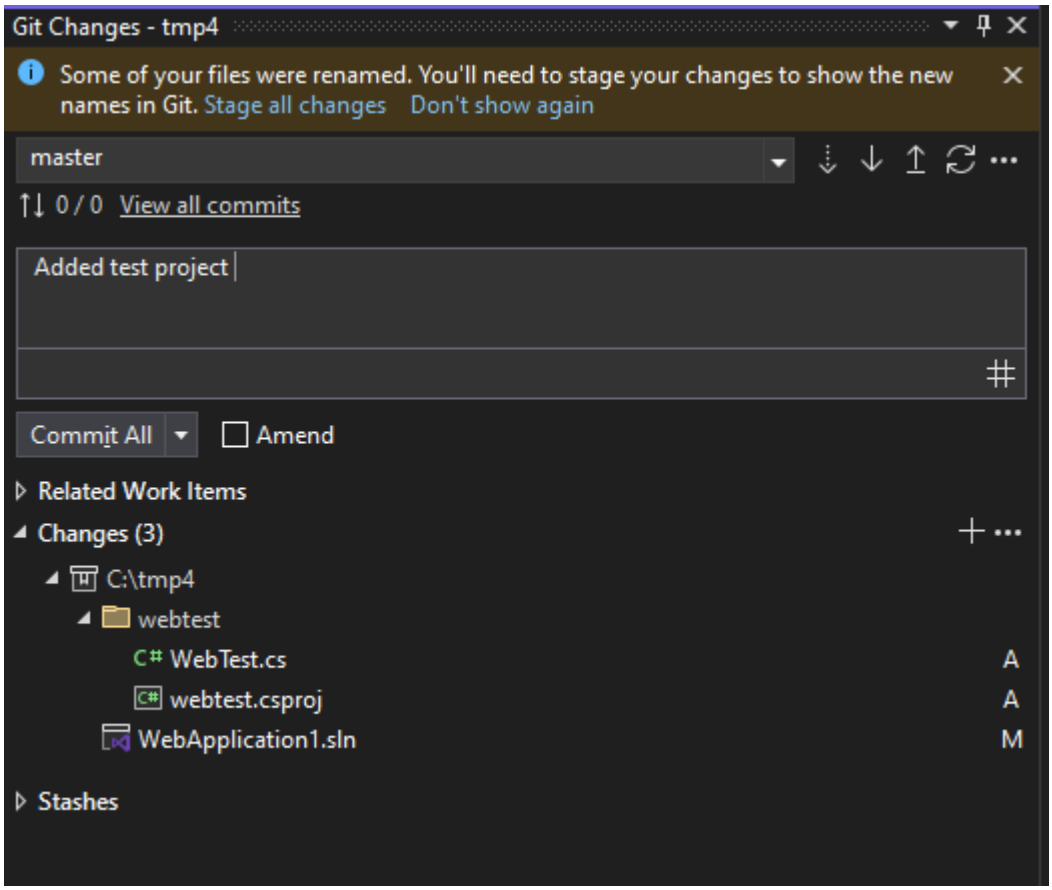


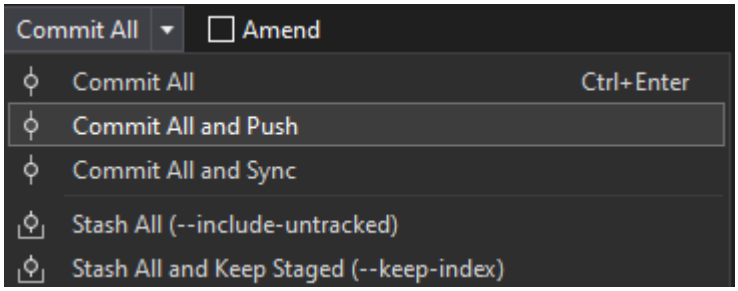
10. Now we want to push this web test project onto our repository on Azure Repos but if you remember from our previous labs, we have already run the pipeline.

11. So, go to Azure DevOps and open the pipeline click on edit then use the code you get from GitHub with this lab and paste it into the pipeline.
12. After that just validate and save the changes for the pipeline then run your pipeline.

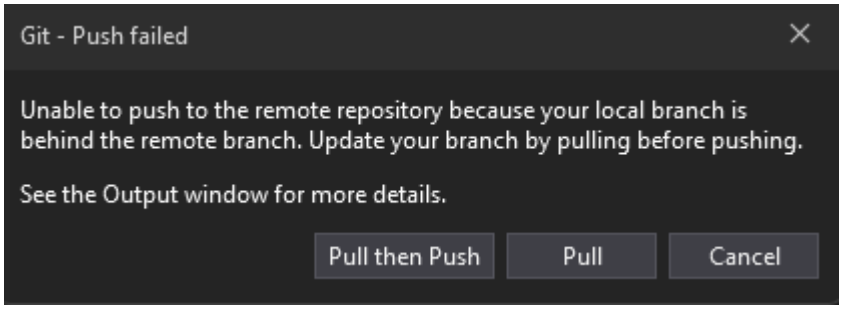


13. Now you need to come back to the Visual Studio and from the solution explorer you need to open the Git changes and give a message then from the commit all drop-down menu choose commit all and push.

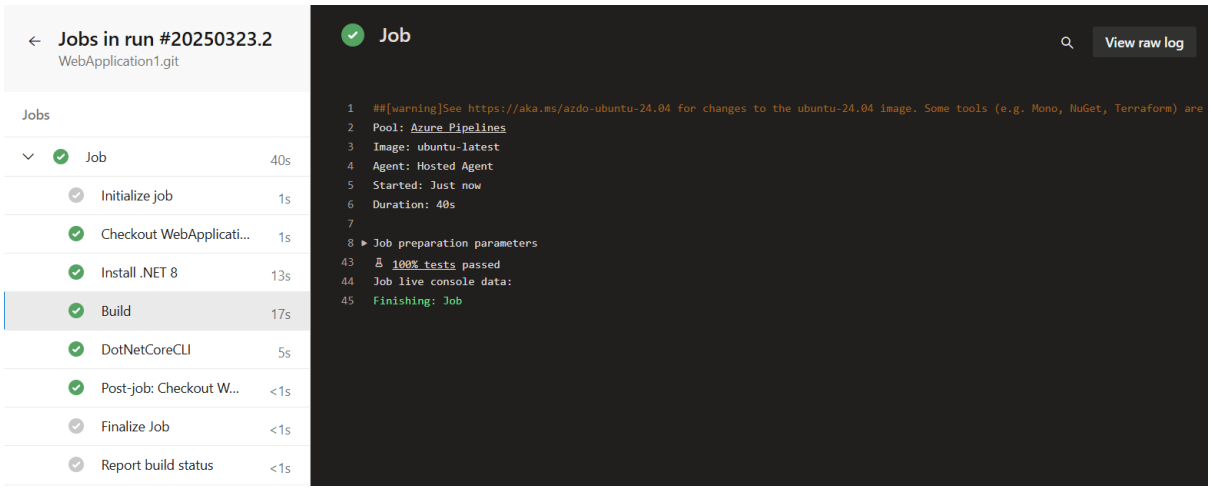




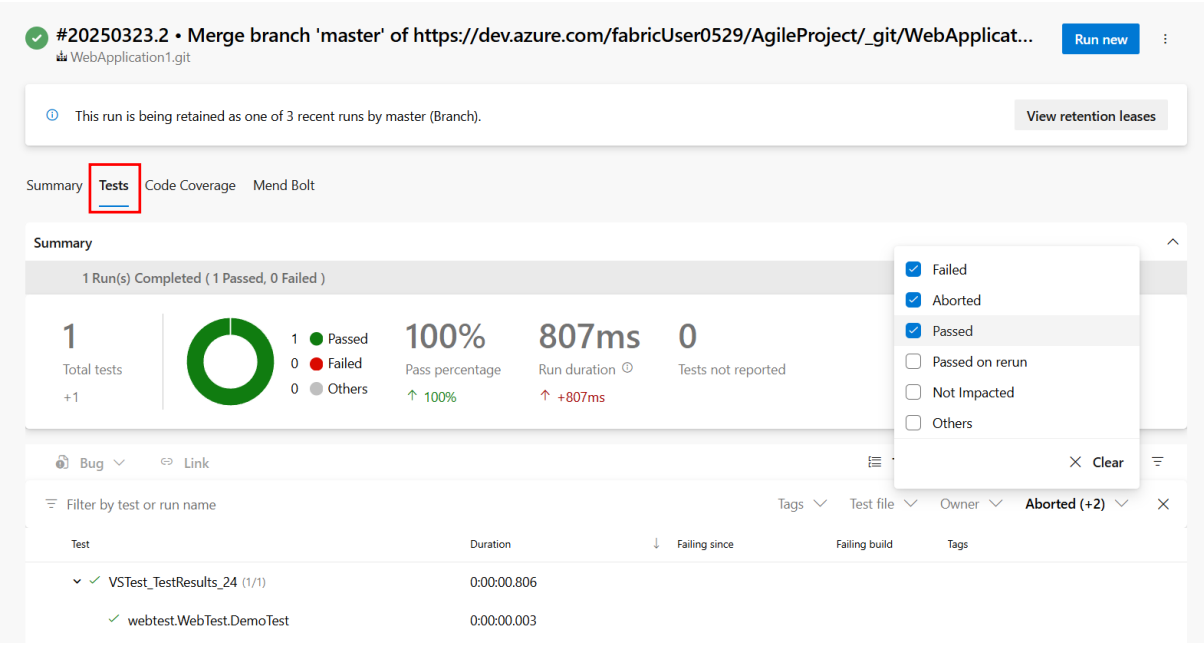
14. Then it will ask us to Pull the changes first then push the changes. Click on Pull then Push and you will see that your pipeline has started running.



15. Here you can see that the pipeline run has been completed.



16. If you go back and see your pipeline you will see a new tab for test, click on it and you can see that what has been passed.


#20250323.2 • Merge branch 'master' of https://dev.azure.com/fabricUser0529/AgileProject/_git/WebApplicat...
WebApplication1.git
Run new
This run is being retained as one of 3 recent runs by master (Branch).
View retention leases
Summary
Tests
Code Coverage
Mend Bolt
Summary
1 Run(s) Completed (1 Passed, 0 Failed)
1
Total tests
+1
1 Passed
0 Failed
0 Others
100%
Pass percentage
100%
807ms
Run duration
+807ms
0
Tests not reported
Failed
Aborted
Passed
Passed on rerun
Not Impacted
Others
Clear
Bug
Link
Filter by test or run name
Tags
Test file
Owner
Aborted (+2)
Test
Duration
Failing since
Failing build
Tags
VSTest_TestResults_24 (1/1)
0:00:00.806
webtest.WebTest.DemoTest
0:00:00.003