### Push Changes Back to the Repository

To push the changes back to the Azure DevOps repository, we need to commit the changes and push them to the remote branch. To do this, open the Team Explorer tab in Visual Studio, select the Changes option, and stage and commit the changes. Then, push the changes to the remote branch.

### Set up the Build Pipeline

Now that the changes are pushed to the remote branch, we need to set up the build pipeline to deploy the workflow in the Databricks test environment. To do this, navigate to the Pipelines section in the Azure DevOps project and click the New pipeline button.

#### **Select the Repository**

In the first step of the pipeline creation wizard, select the Azure DevOps repository that was just created.

#### **Select the Build Template**

In the next step, select the template for building a Python project. This will set up the necessary build steps for our project.

#### **Set up the Build Steps**

In the Build pipeline editor, we need to set up the build steps to deploy the workflow in the Databricks test environment. To do this, we need to add a task to deploy the workflow in the Databricks test environment and run the test case notebook.

##### Deploy Workflow in Databricks Test Environment

To deploy the workflow in the Databricks test environment, we need to add a task to deploy the workflow using the Databricks CLI. To do this, add a Command Line task to the pipeline and set the command to deploy the workflow using the Databricks CLI. Here is an example command:

bashCopy code

databricks workspace import\_dir /path/to/workflow /path/to/workflow/notebook

##### Run Test Case Notebook

Once the workflow is deployed, we need to run the test case notebook to capture the results. To do this, add another Command Line task to the pipeline and set the command to run the test case notebook using the Databricks CLI. Here is an example command:

bashCopy code

databricks jobs run-now --notebook-path /path/to/test/notebook --output /path/to/output.xml

### Capture Results in XML and Send Notifications

Finally, we need to capture the results of the test case notebook and send notifications via Outlook and MS Teams.

#### **Capture Results in XML**

To capture the results of the test case notebook, we need to redirect the output of the Databricks CLI command to an XML file. This can be done by adding the **--output** flag to the command, as shown in the previous section.

#### **Send Notifications via Outlook and MS Teams**

To send notifications via Outlook and MS Teams, we need to set up the notification settings in Azure DevOps. To do this, navigate to the Project settings and click the Notifications option. From here, we can set up notifications for various events, such as when a build fails or succeeds. We can also choose which channels to send notifications to, such as email or Microsoft Teams.

## Conclusion

In this technical page, we have walked through the process of setting up an automated testing workflow using Azure DevOps. We started by setting up the development environment, including installing Python, Pytest, Visual Studio, Azure CLI, and Databricks CLI. Then, we set up the Azure DevOps workflow, including creating a new project and repository, connecting the repository to Visual Studio, creating a new branch, writing test cases, pushing changes back to the repository, and setting up the build pipeline to deploy the workflow in the Databricks test