Automated testing is a critical component of software development, and Azure DevOps provides a comprehensive suite of tools to streamline the testing process. This technical page will walk through the process of setting up an automated testing workflow using Azure DevOps, from pulling the latest code from the development branch of a DevOps repository into a local branch in Visual Studio IDE, designing test cases using pytest assert function, pushing changes back to the repository, deploying a workflow in a Databricks test environment, running a test case notebook, capturing results in XML, and sending notifications via Outlook and MS Teams.

**Setting up the Development Environment**

Before setting up the automated testing workflow, we need to ensure that the development environment is properly configured. This involves installing the necessary tools and dependencies to develop and test code.

**Install Python and Pytest**

The first step is to ensure that Python and Pytest are installed on the local machine. Pytest is a popular testing framework that allows us to write and run tests for our code. To install Pytest, open a command prompt or terminal and run the following command:

bashCopy code

pip install pytest

**Install Visual Studio**

Next, we need to install Visual Studio, which will be used to pull the latest code from the DevOps repository and create a local branch. Visual Studio is a powerful integrated development environment (IDE) that provides a suite of tools to develop and test code. To install Visual Studio, follow the instructions on the Visual Studio website.

**Install Azure CLI**

To interact with Azure DevOps from the command line, we need to install Azure CLI. Azure CLI is a command-line tool that provides a streamlined interface to interact with Azure services, including Azure DevOps. To install Azure CLI, follow the instructions on the Azure CLI website.

**Install Databricks CLI**

Finally, we need to install Databricks CLI, which will be used to deploy the workflow in the Databricks test environment. Databricks CLI is a command-line tool that provides a streamlined interface to interact with Databricks services. To install Databricks CLI, follow the instructions on the Databricks CLI website.

**Setting up the Azure DevOps Workflow**

Now that the development environment is set up, we can set up the Azure DevOps workflow to automate the testing process.

**Create a New DevOps Project**

The first step is to create a new project in Azure DevOps. This can be done by logging in to Azure DevOps and creating a new project.

**Create a DevOps Repository**

Once the project is created, we need to create a repository to store our code. To create a new repository, navigate to the Repos section of the project and click the New repository button. Give the repository a name and click the Create button.

**Connect the Repository to Visual Studio**

Now that the repository is created, we need to connect it to Visual Studio. To do this, open Visual Studio and select the Team Explorer tab. Click the Connect button and select the Azure DevOps option. Enter the URL of the Azure DevOps project and select the repository that was just created.

**Create a New Branch**

Now that the repository is connected to Visual Studio, we can create a new branch to work on. Click the Branches option in the Team Explorer tab and select the New branch option. Give the branch a name, such as "test-branch", and click the Create Branch button.

**Write Test Cases**

Now that the local branch is created, we can write test cases using the Pytest framework. Pytest allows us to write tests in Python and provides a suite of assert functions to test the code. Here is an example test case:

pythonCopy code

def test\_addition(): assert 2 + 2 == 4

**Push Changes**