Enable Continuous Integration with Azure Pipelines

Student lab manual

Lab requirements

- This lab requires **Microsoft Edge** or an Azure DevOps supported browser.
- **Set up an Azure DevOps organization:** If you don't already have an Azure DevOps organization that you can use for this lab, create one by following the instructions available at Create an organization or project collection.

Lab overview

In this lab, you will learn how to define build pipelines in Azure DevOps using YAML. The pipelines will be used in two scenarios:

- As part of Pull Request validation process.
- As part of the Continuous Integration implementation.

Objectives

After you complete this lab, you will be able to:

- Include build validation as part of a Pull Request.
- Configure CI pipeline as code with YAML.

Estimated timing: 45 minutes

Instructions

Exercise 0: Configure the lab prerequisites

In this exercise, you will set up the prerequisites for the lab, which consist of a new Azure DevOps project with a repository based on the eShopOnWeb.

Task 1: (skip if done) Create and configure the team project

In this task, you will create an **eShopOnWeb** Azure DevOps project to be used by several labs.

1. On your lab computer, in a browser window open your Azure DevOps organization. Click on **New Project**. Give your project the name **eShopOnWeb** and leave the other fields with defaults. Click on **Create**.

Task 2: (skip if done) Import eShopOnWeb Git Repository

In this task you will import the eShopOnWeb Git repository that will be used by several labs.

1. On your lab computer, in a browser window open your Azure DevOps organization and the previously created **eShopOnWeb** project. Click on **Repos>Files** , **Import a Repository**. Select **Import**. On the **Import a Git Repository** window, paste the following URL https://github.com/MicrosoftLearning/eShopOnWeb.git and click **Import**:

2. The repository is organized the following way:

- **.ado** folder contains Azure DevOps YAML pipelines.
- **.devcontainer** folder container setup to develop using containers (either locally in VS Code or GitHub Codespaces).
- **infra** folder contains Bicep & ARM infrastructure as code templates used in some lab scenarios.
- **.github** folder contains YAML GitHub workflow definitions.
- **src** folder contains the .NET website used in the lab scenarios.

Task 3: (skip if done) Set main branch as default branch

1. Go to **Repos>Branches**.
2. Hover on the **main** branch then click the ellipsis on the right of the column.
3. Click on **Set as default branch**.

Exercise 1: Include build validation as part of a Pull Request

In this exercise, you will include build validation to validate a Pull Request.

Task 1: Import the YAML build definition

In this task, you will import the YAML build definition that will be used as a Branch Policy to validate the pull requests.

Let's start by importing the build pipeline named eshoponweb-ci-pr.yml.

1. Go to **Pipelines>Pipelines**
2. Click on **Create Pipeline** or **New Pipeline** button
3. Select **Azure Repos Git (YAML)**
4. Select the **eShopOnWeb** repository
5. Select **Existing Azure Pipelines YAML File**
6. Select the **main** branch and the **/.ado/eshoponweb-ci-pr.yml** file, then click on **Continue**

   The build definition consists of the following tasks:

   - **DotNet Restore**: With NuGet Package Restore you can install all your project's dependency without having to store them in source control.
   - **DotNet Build**: Builds a project and all of its dependencies.
   - **DotNet Test**: .Net test driver used to execute unit tests.
   - **DotNet Publish**: Publishes the application and its dependencies to a folder for deployment to a hosting system. In this case, it's **Build.ArtifactStagingDirectory**.
7. Click the **Save** button to save the pipeline definition
8. Your pipeline will take a name based on the project name. Let's **rename** it for identifying the pipeline better. Go to **Pipelines>Pipelines** and click on the recently created pipeline. Click on the ellipsis and **Rename/Move** option. Name it **eshoponweb-ci-pr** and click on **Save**.

Task 2: Branch Policies

In this task, you will add policies to the main branch and only allow changes using Pull Requests that comply with the defined policies. You want to ensure that changes in a branch are reviewed before they are merged.

1. Go to **Repos>Branches** section.
2. On the **Mine** tab of the **Branches** pane, hover the mouse pointer over the **main** branch entry to reveal the ellipsis symbol on the right side.
3. Click the ellipsis and, in the pop-up menu, select **Branch Policies**.
4. On the **main** tab of the repository settings, enable the option for **Require minimum number of reviewers**. Add **1** reviewer and check the box **Allow requestors to approve their own changes**(as you are the only user in your project for the lab)
5. On the **main** tab of the repository settings, in the **Build Validation** section, click **+** (Add a new build policy) and in the **Build pipeline** list, select **eshoponweb-ci-pr** then click **Save**.

Task 3: Working with Pull Requests

In this task, you will use the Azure DevOps portal to create a Pull Request, using a new branch to merge a change into the protected **main** branch.

1. Navigate to the **Repos** section in the eShopOnWeb navigation and click **Branches**.
2. Create a new branch named **Feature01** based on the **main** branch.
3. Click **Feature01** and navigate to the **/eShopOnWeb/src/Web/Program.cs** file as part of the **Feature01** branch
4. Click the **Edit** button in the top-right
5. Make the following change on the first line:

```
// Testing my PR
```

6. Click on **Commit > Commit** (leave default commit message).
7. A message will pop-up, proposing to create a Pull Request (as your **Feature01** branch is now ahead in changes, compared to **main**). Click on **Create a Pull Request**.
8. In the **New pull request** tab, leave defaults and click on **Create**.
9. The Pull Request will show some pending requirements, based on the policies applied to the target **main** branch.
    - At least 1 user should review and approve the changes.
    - Build validation, you will see that the build **eshoponweb-ci-pr** was triggered automatically
10. After all validations are successful, on the top-right click on **Approve**. Now from the **Set auto-complete** dropdown you can click on **Complete**.
11. On the **Complete Pull Request** tab, click on **Complete Merge**

Exercise 2: Configure CI Pipeline as Code with YAML

In this exercise, you will configure CI Pipeline as code with YAML.

Task 1: Import the YAML build definition for CI

In this task, you will add the YAML build definition that will be used to implement the Continuous Integration.

Let's start by importing the CI pipeline named eshoponweb-ci.yml.

1. Go to **Pipelines>Pipelines**.
2. Click on **New Pipeline** button.
3. Select **Azure Repos Git (YAML)**.
4. Select the **eShopOnWeb** repository.
5. Select **Existing Azure Pipelines YAML File**.
6. Select the **main** branch and the **/.ado/eshoponweb-ci.yml** file, then click on **Continue**.

   The CI definition consists of the following tasks:

    - **DotNet Restore**: With NuGet Package Restore you can install all your project's dependency without having to store them in source control.
    - **DotNet Build**: Builds a project and all of its dependencies.
    - **DotNet Test**: .Net test driver used to execute unit tests.
    - **DotNet Publish**: Publishes the application and its dependencies to a folder for deployment to a hosting system. In this case, it's **Build.ArtifactStagingDirectory**.
    - **Publish Artifact - Website**: Publish the app artifact (created in the previous step) and make it available as a pipeline artifact.
    - **Publish Artifact - Bicep**: Publish the infrastructure artifact (Bicep file) and make it available as a pipeline artifact.

Task 2: Enable Continuous Integration

The default build pipeline definition doesn't enable Continuous Integration.

1. Click the **Edit** button in the top-right
2. Now, you need to replace the **# trigger:** and **# - main** lines with the following code:

```
trigger:
  branches:
    include:
    - main
  paths:
    include:
    - src/web/*
```

   This will automatically trigger the build pipeline if any change is made to the main branch and the web application code (the src/web folder).

   Since you enabled Branch Policies, you need to pass by a Pull Request in order to update your code.
3. Click the **Save** button (not **Save and run**) to save the pipeline definition.
4. Select **Create a new branch for this commit**.
5. Keep the default branch name and **Start a pull request** checked.
6. Click on **Save**.
7. Your pipeline will take a name based on the project name. Let's **rename** it for identifying the pipeline better. Go to **Pipelines>Pipelines** and click on the recently created pipeline. Click on the ellipsis and **Rename/Move** option. Name it **eshoponweb-ci** and click on **Save**.
8. Go to **Repos > Pull Requests**.
9. Click on the **"Update eshoponweb-ci.yml for Azure Pipelines"** pull request.
10. After all validations are successful, on the top-right click on **Approve**. Now you can click on **Complete**.
11. On the **Complete Pull Request** tab, Click on **Complete Merge**

Task 3: Test the CI pipeline

In this task, you will create a Pull Request, using a new branch to merge a change into the protected **main** branch and automatically trigger the CI pipeline.

1. Navigate to the **Repos** section.
2. Create a new branch named **Feature02** based on the **main** branch.
3. Click the new **Feature02** branch.
4. Navigate to the **/eShopOnWeb/src/Web/Program.cs** file and click **Edit** in the top-right.
5. Remove the first line:

```
// Testing my PR
```

6. Click on **Commit > Commit** (leave default commit message).
7. A message will pop-up, proposing to create a Pull Request (as your **Feature02** branch is now ahead in changes, compared to **main**).
8. Click on **Create a Pull Request**.
9. In the **New pull request** tab, leave defaults and click on **Create**.
10. The Pull Request will show some pending requirements, based on the policies applied to the target **main** branch.
11. After all validations are successful, on the top-right click on **Approve**. Now from the **Set auto-complete** dropdown you can click on **Complete**.
12. On the **Complete Pull Request** tab, Click on **Complete Merge**
13. Go back to **Pipelines>Pipelines**, you will notice that the build **eshoponweb-ci** was triggered automatically after the code was merged.
14. Click on the **eshoponweb-ci** build then select the last run.
15. After its successful execution, click on **Related > Published** to check the published artifacts:

- Bicep: the infrastructure artifact.
- Website: the app artifact.

Review

In this lab, you enabled pull request validation using a build definition and configured CI pipeline as code with YAML in Azure DevOps.