**Exercise**

**Push a change through the pipeline**

**Introduction**

In this unit, you practice the complete code workflow by pushing a small change to the *Space Game* website to GitHub.

Mara has been given the task of changing some text on the home page of the website, *Index.cshtml*. In this unit, you'll follow along.

Let's briefly review the steps to follow to complete the task:

* Synchronize your local repository with the latest master branch on GitHub.
* Create a branch to hold your changes.
* Make the code changes you need, and verify them locally.
* Push your branch to GitHub.
* Merge any recent changes from the master branch on GitHub into your local working branch, and verify that your changes still work.
* Push up any remaining changes, watch Azure Pipelines build the application, and submit your pull request.

**Fetch the latest master branch**

In the previous unit, you created a pull request and merged your code-workflow branch into the master branch on GitHub. Now you need to pull the changes to master back to your local branch.

The git pull command fetches the latest code from the remote repository and merges it into your local repository. This way, you know you're working with the latest codebase.

1. In your terminal, run git checkout master to switch to the master branch.

**Bash**

git checkout master

1. To pull down the latest changes, run this git pull command:

**Bash**

git pull origin master

You can view the list of files that were changed. As an optional step, you can open the *azure-pipelines.yml* file to verify that it contains your complete build configuration.

Recall that a Git repository where team members collaborate (such as on GitHub) is called a *remote*. Here, *origin* specifies your repository on GitHub.

Later, you'll fetch starter code from the Microsoft GitHub repository, known as *upstream*.

**Build and run the web application**

To ensure that you have a working copy to start your changes, build and run the web application locally.

1. In Visual Studio Code, go to the terminal window and run the following dotnet build command to build the application:

**Bash**

dotnet build --configuration Release

1. Run the following dotnet run command to run the application:

**Bash**

dotnet run --configuration Release --no-build --project Tailspin.SpaceGame.Web

**Tip**

If you see an error in your browser that's related to a privacy or certificate error, select Ctrl+C from your terminal to stop the running application.

Then run dotnet dev-certs https --trust and select **Yes** when prompted. Or **see this blog post** for more information.

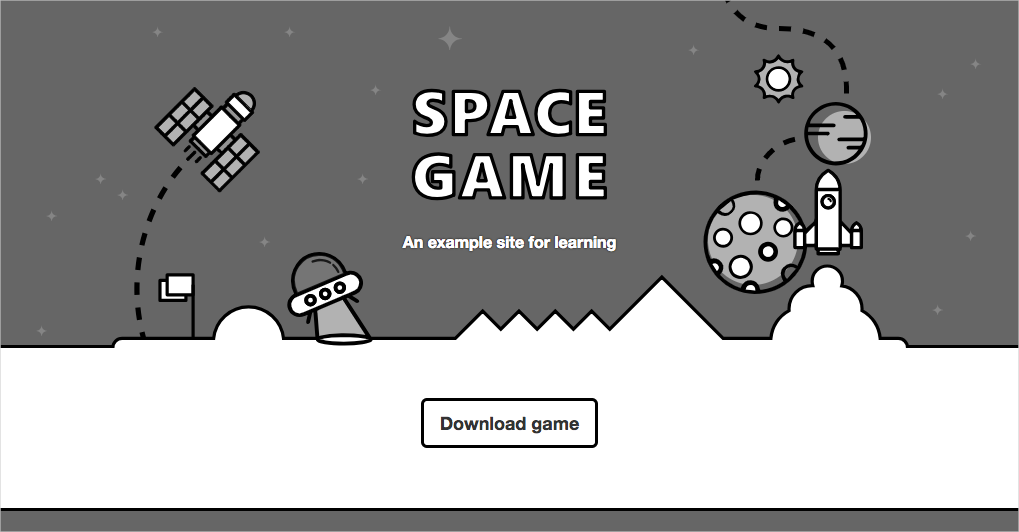
After your computer trusts your local SSL certificate, run the dotnet run command a second time and go to http://localhost:5000 from a new browser tab to see the running application.

**Verify that the application is running**

In development mode, the *Space Game* website is configured to run on port 5000.

In a new browser tab, navigate to http://localhost:5000 to see the running application.

You see this:



You can interact with the page, including the leaderboard. When you select a player's name, you see details about that player.

When you're finished, return to the terminal window and select Ctrl+C to stop the running application.

**Create a feature branch**

In this section, you create a Git branch so that you can work on files without affecting anyone else. No one will even know you're working on those files until you push them to the remote repository.

To create a branch, you use the git checkout command and give your branch a name, just like you did in the previous part.

Before you create a branch, it's a good idea to adhere to a naming convention. For example, if your branch is for working on a new feature, you might use **feature/<branch-name>.** For a bug fix, you could use **bugfix/<bug-number>**. In this example, your branch name will be feature/home-page-text.

In your terminal, run the following git checkout command:

**Bash**

git checkout -b feature/home-page-text

Like before, the feature/home-page-text is based off of master.

**Make changes and test it locally**

1. In Visual Studio Code, open *Index.cshtml* in the *Tailspin.SpaceGame.Web/Views/Home* directory.
2. Look for this text near the top of the page:

**HTML**

<p>An example site for learning</p>

**Tip**

Visual Studio Code also provides an easy way to search for text in files. To access the search pane, select the magnifying glass icon in the side pane.

1. Replace the text in the previous step with the following "mistyped" text, and then save the file:

**HTML**

<p>Welcome to the oficial Space Game site!</p>

Note that the word "oficial" is intentionally mistyped. We'll address that error later in this module.

1. In your terminal, run the following dotnet build command to build the application:

**Bash**

dotnet build --configuration Release

1. Run the following dotnet run command to run the application:

**Bash**

dotnet run --configuration Release --no-build --project Tailspin.SpaceGame.Web

1. On a new browser tab, go to http://localhost:5000 to see the running application.

You can see that the home page contains the updated text.



When you're finished, return to the terminal window, and then press Ctrl+C to stop the running application.

**Commit and push your branch**

Here you'll stage your changes to *Index.cshtml*, commit the change to your branch, and push your branch up to GitHub.

1. Run git status to check to see whether there are uncommitted changes on your branch.

**Bash**

git status

You see that *Index.cshtml* has been modified. Like before, the next step is to make sure that Git is tracking this file. This is called *staging* the file.

1. Run the following git add command to stage *Index.cshtml*.

**Bash**

git add Tailspin.SpaceGame.Web/Views/Home/Index.cshtml

1. Run the following git commit command to commit your staged file to the feature/home-page-text branch.

**Bash**

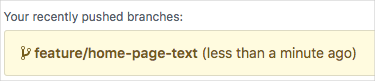
git commit -m "Improve the text at the top of the home page"

1. Run this git push command to push, or upload, the feature/home-page-text branch to your repository on GitHub.

**Bash**

git push origin feature/home-page-text

1. Go to GitHub to see that your branch has been pushed to your remote repository.



**Watch Azure Pipelines build the application**

Just as you did previously, Azure Pipelines automatically queues the build when you push changes to GitHub.

As an optional step, trace the build as it moves through the pipeline, and verify that the build succeeds.

**Synchronize any changes to the master branch**

While you were busy working on your feature, there might have been changes made to the remote master branch. Before you create a pull request, it's common practice to get the latest from the remote master branch.

To do this, you first check out, or switch to, the master branch. You then merge the remote master branch with your local master branch.

Next, you check out your feature branch. Then you merge your feature branch with the master branch.

Let's try the process now.

1. In your terminal, run this git checkout command to check out the master branch:

**Bash**

git checkout master

1. To download the latest changes to the remote master branch and merge those changes into your local master branch, run this git pull command:

**Bash**

git pull origin master

Because no one actually made any changes to your master branch, the following command tells you that everything is already up-to-date.

**Output**

From <https://github.com/username/mslearn-tailspin-spacegame-web>

\* branch master -> FETCH\_HEAD

Already up to date.

1. To check out your feature branch, run git checkout.

**Bash**

git checkout feature/home-page-text

1. Merge your feature branch with master.

**Bash**

git merge master

Again, because no one actually made any changes to your master branch, you see that everything is still up to date.

**OutputCopy**

Already up to date.

If you did incorporate any changes, you would want to test your application again to make sure that everything is still working.

**Push your local branch again**

When you incorporate changes from the remote repository into your local feature branch, you need to push your local branch back to the remote repository a second time.

Although you didn't incorporate any changes from the remote repository, let's practice the process to see what happens.

1. Run this git push command to push your changes to GitHub:

**Bash**

git push origin feature/home-page-text

Once again, the response says that you're already up-to-date since no changes were made.

**Output**

Everything up-to-date

**Submit a pull request**

In this section, you submit a pull request just as you did previously.

1. In a browser, sign in to GitHub .
2. Go to your **mslearn-tailspin-spacegame-web** repository.
3. In the drop-down list, select your feature/home-page-text branch.
4. To begin creating your pull request, select **New pull request**.
5. Ensure that the **base** drop-down list specifies your repository and not the Microsoft repository.

**Important**

Again, this step is important because you can't merge your changes into the Microsoft repository.

When you work directly with your own repository, and not a fork, your master branch is selected by default.

1. Enter a title and a description for your pull request.
   * **Title**: *Improve the text at the top of the home page*
   * **Description**: *Received the latest home page text from the product team.*
2. To complete your pull request, select **Create pull request**.

This step doesn't merge any code. It tells others that you have changes that you're proposing to merge.

The pull request window is displayed. As before, a pull request triggers Azure Pipelines to build your application by default.

1. Optionally, select the **Details** link or go to your project on Azure DevOps and watch the pipeline run.
2. When the build is finished, go back to your pull request on GitHub.
3. Select **Merge pull request**, and then select **Confirm merge**.
4. Select **Delete branch** to delete the feature/home-page-text branch from GitHub.