**Exercise - Set up your Azure DevOps environment**

In this unit, you ensure that your Microsoft Azure DevOps organization is set up to complete the rest of this module.

To do this, you:

* Set up an Azure DevOps project for this module.
* Move the work item for this module on Azure Boards to the **Doing** column.
* Make sure your project is set up locally so that you can push changes to the pipeline.

**Get the Azure DevOps project**

Here, you make sure that your Azure DevOps organization is set up to complete the rest of this module. You do this by running a template that creates a project for you in Azure DevOps.

The modules in this learning path form a progression, where you follow the Tailspin web team through their DevOps journey. For learning purposes, each module has an associated Azure DevOps project.

**Run the template**

Run a template that sets up everything for you in your Azure DevOps organization.

Run the template from the below URL:

https://azuredevopsdemogenerator.azurewebsites.net/?name=run-quality-tests

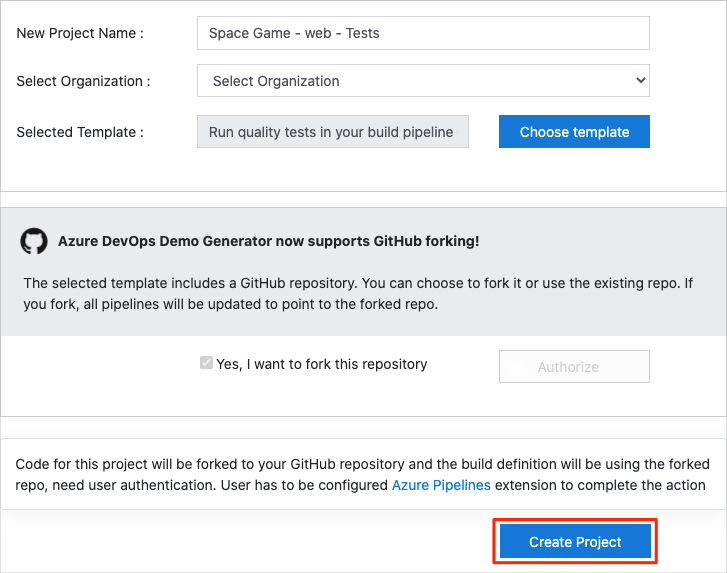
To run the template, on the Azure DevOps Demo Generator site, do the following:

1. Select **Sign In**, and accept the usage terms.
2. On the **Create New Project** page, select your Azure DevOps organization, and then enter a project name, such as *Space Game - web - Tests*.
3. Select **Fork repository on GitHub**, and then select **Authorize**. If a window appears, authorize access to your GitHub account.

**Important**

You need to select this option for the template to connect to your GitHub repository. Select it even if you've already forked the *Space Game* website project. The template uses your existing fork.

1. Select **Create Project**.



It takes a few moments for the template to run.

1. Select **Navigate to project** to go to your project in Azure DevOps.

**Important**

The **Clean up your Azure DevOps environment** page in this module contains important cleanup steps. Cleaning up helps ensure that you don't run out of free build minutes. Be sure to perform the cleanup steps even if you don't complete this module.

**Set your project's visibility**

Your fork of the *Space Game* repository on GitHub is initially public. The Azure DevOps template creates a project that's initially private.

A public GitHub repository is accessible to everyone, whereas a private repository is accessible to you and the people you share it with. In both cases, only collaborators can commit changes to a GitHub repository.

A project on Azure DevOps works the same way. Users who aren't signed in to the service have read-only access to public projects. Private projects require users to be granted access to the project and signed in to access the services.

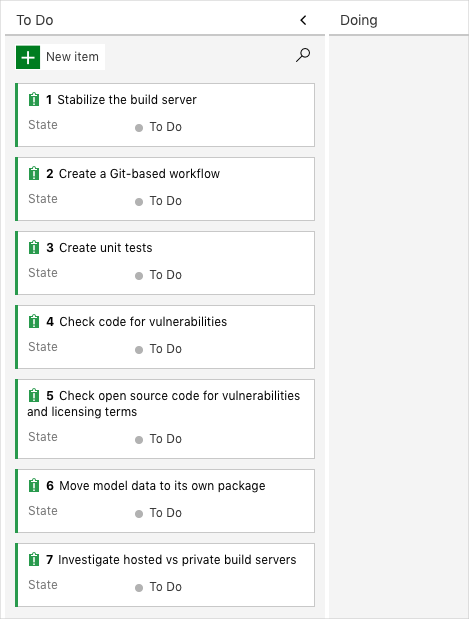
For learning purposes, you don't need to change any of these settings right now. But for your own projects, you need to decide what visibility and access you want to provide to others. For example, if your project is open source, you might make both your GitHub repository and your Azure DevOps project public. If your project is closed source, you would likely make both your GitHub repository and your Azure DevOps project private.

**Move the work item to Doing**

In this section, you assign yourself a work item that relates to this module on Azure Boards. You also move the work item to the **Doing** state. In practice, you and your team would create work items at the start of each sprint, or work iteration.

Assigning work in this way gives you a checklist to work from. It gives others on your team visibility into what you're working on and how much work is left. It also helps the team enforce work in process (WIP) limits so that the team doesn't take on too much work at one time.

Recall that the team settled on these seven top issues:

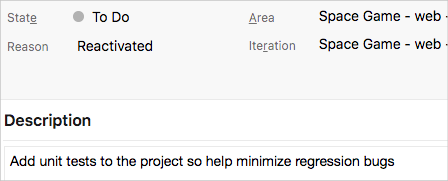


**Note**

Within an Azure DevOps organization, work items are numbered sequentially. In your project, the number that's assigned to each work item might not match what you see here.

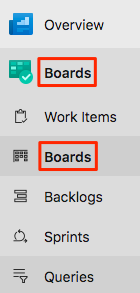
Here you move the third item, **Create unit tests**, to the **Doing** column and assign yourself to the work item.

Recall that **Create unit tests** relates to running unit tests during the build so that Amita doesn't have to deal with so many regression bugs when she tests out new builds.



To set up the work item:

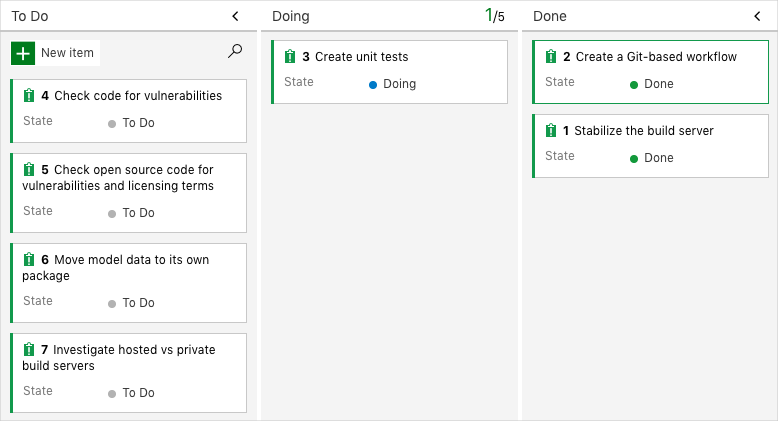
1. In Azure DevOps, go to **Boards** and then, in the left pane, select **Boards**.



1. In the **Create unit tests** work item, select the **To Do** down arrow, and then assign the work item to yourself.

Screenshot of Azure DevOps State field with the down chevron button highlighted.

1. Drag the work item from the **To Do** column to the **Doing** column.



At the end of this module, after you've completed the task, you'll move the card to the **Done** column.

**Set up the project locally**

Here, you load the *Space Game* project in Visual Studio Code, configure Git, clone your repository locally, and set the upstream remote so that you can download starter code.

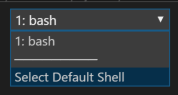
**Note**

If you're already set up with the **mslearn-tailspin-spacegame-web** project locally, you can move to the next section.

**Open the integrated terminal**

Visual Studio Code comes with an integrated terminal, so you can edit files and work from the command line all from one place.

1. Start Visual Studio Code.
2. On the **View** menu, select **Terminal**.
3. In the drop-down list, select **bash**:



The terminal window lets you choose any shell that's installed on your system, like Bash, Zsh, and PowerShell.

Here you'll use Bash. Git for Windows provides Git Bash, which makes it easy to run Git commands.

**Note**

On Windows, if you don't see **bash** listed as an option, make sure you've installed **Git** and then restart Visual Studio Code.

If you still don't see the **bash** option, see **Integrated Terminal** to manually configure your terminal settings.

1. Run the cd command to navigate to the directory you want to work from, like your home directory (~). You can choose a different directory if you want.

**Bash**

cd ~

**Configure Git**

If you're new to Git and GitHub, you first need to run a few commands to associate your identity with Git and authenticate with GitHub.

Set up Git  explains the process in greater detail.

At a minimum, you'll need to complete the following steps. Run these commands from the integrated terminal:

1. Set your username .
2. Set your commit email address .
3. Cache your GitHub password .

**Note**

If you're already using two-factor authentication with GitHub, **create a personal access token** and use your token in place of your password when prompted later.

Treat your access token like you would a password. Keep it in a safe place.

**Set up your project in Visual Studio Code**

In this part, you clone your fork locally so that you can make changes and build out your pipeline configuration.

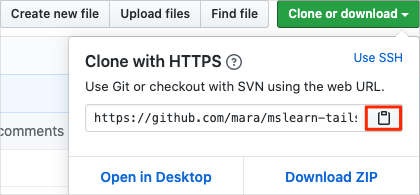
**Clone your fork locally**

You now have a copy of the *Space Game* web project in your GitHub account. Now you'll download, or *clone*, a copy to your computer so you can work with it.

A clone, just like a fork, is a copy of a repository. When you clone a repository, you can make changes, verify they work as you expect, and then upload those changes back to GitHub. You can also synchronize your local copy with changes other authenticated users have made to GitHub's copy of your repository.

To clone the *Space Game* web project to your computer:

1. Go to your fork of the *Space Game* web project (**mslearn-tailspin-spacegame-web**) on GitHub .
2. Select **Clone or download**. Then select the button next to the URL that's shown to copy the URL to your clipboard:



1. In Visual Studio Code, go to the terminal window.
2. In the terminal, move to the directory you want to work from, like your home directory (~). You can choose a different directory if you want.

**Bash**

cd ~

1. Run the git clone command. Replace the URL that's shown here with the contents of your clipboard:

**Bash**

git clone https://github.com/your-name/mslearn-tailspin-spacegame-web.git

1. Move to the mslearn-tailspin-spacegame-web directory. This is the root directory of your repository.

**Bash**

cd mslearn-tailspin-spacegame-web

**Set the upstream remote**

A *remote* is a Git repository where team members collaborate (like a repository on GitHub).

Run this git remote command to list your remotes:

**Bash**

git remote -v

You see that you have both fetch (download) and push (upload) access to your repository:

OutputCopy

origin https://github.com/username/mslearn-tailspin-spacegame-web.git (fetch)

origin https://github.com/username/mslearn-tailspin-spacegame-web.git (push)

*Origin* specifies your repository on GitHub. When you fork code from another repository, it's common to name the original remote (the one you forked from) as *upstream*.

Run this git remote add command to create a remote named *upstream* that points to the Microsoft repository:

Bash

git remote add upstream https://github.com/MicrosoftDocs/mslearn-tailspin-spacegame-web.git

Run git remote a second time to see the changes:

Bash

git remote -v

You see that you still have both fetch (download) and push (upload) access to your repository. You also now have fetch access from the Microsoft repository:

OutputCopy

origin https://github.com/username/mslearn-tailspin-spacegame-web.git (fetch)

origin https://github.com/username/mslearn-tailspin-spacegame-web.git (push)

upstream https://github.com/MicrosoftDocs/mslearn-tailspin-spacegame-web.git (fetch)

**Open the project in the file explorer**

In Visual Studio Code, your terminal window points to the root directory of the *Space Game* web project. You'll now open the project from the file explorer so you can view its structure and work with files.

1. On the **File** menu, select **Open**.
2. Navigate to the root directory of the *Space Game* web project.

(You can run the pwd command in the terminal window to see the full path if you need a reminder.)

You see the directory and file tree in the file explorer.

**Note**

You might need to open the integrated terminal a second time after you open the folder.

You're now set up to work with the *Space Game* source code and your Azure Pipelines configuration from your local development environment.