# Exercise - Set up your Azure DevOps environment

In this section, you make sure that your Azure DevOps organization is set up to complete the rest of this module.

To set up your Azure DevOps organization, you:

* Add a user to ensure that Azure DevOps can connect to your Azure subscription.
* Set up an Azure DevOps project for this module.
* In Azure Boards, move the work item for this module to the **Doing** column.
* Make sure that your project is set up locally so that you can push changes to the pipeline.

## Add a user to Azure DevOps

To complete this module, you need your own Azure subscription . You can get started with Azure for free.

You don't need an Azure subscription to work with Azure DevOps. But here you'll use Azure DevOps to deploy to resources that exist in your Azure subscription. To simplify the process, use the same Microsoft account to sign in to both your Azure subscription and your Azure DevOps organization.

If you use different Microsoft accounts to sign in to Azure and Azure DevOps, add a user to your DevOps organization under the Microsoft account that you use to sign in to Azure. For more information, see Add users to your organization or project . When you add the user, choose the **Basic** access level.

Then sign out of Azure DevOps. Sign in again under the Microsoft account that you use to sign in to your Azure subscription.

## 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 run a template that creates a project in Azure DevOps.

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

### **Run the template**

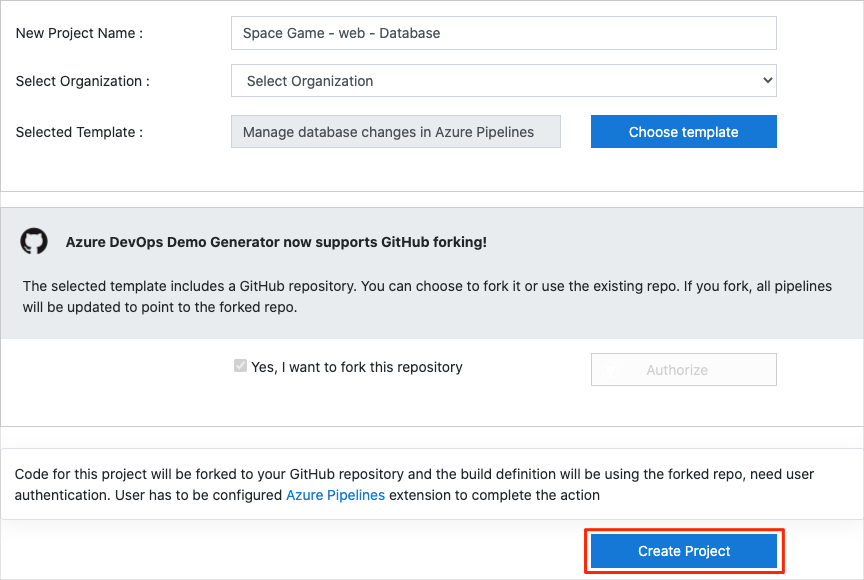
Run a template that sets up your Azure DevOps organization.

Run the template by clicking on the link below

https://azuredevopsdemogenerator.azurewebsites.net/?name=manage-database-changes

On the Azure DevOps Demo Generator site, follow these steps to run the template.

1. Select **Sign In** and accept the usage terms.
2. From the **Create New Project** page, select your Azure DevOps organization. Enter a project name, such as Space Game - web - Database.



1. Select **Yes, I want to fork this repository**, and then select **Authorize**.
2. Select **Create Project**.

The template takes a few moments to run.

1. To go to your project in Azure DevOps, select **Navigate to project**.

**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. Follow 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.

Later, you can use these resources to help you decide which option is best for your project:

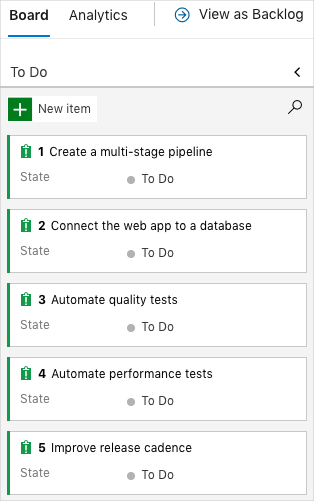
* What is a public project?
* Quickstart: Change the project visibility, public or private
* Setting repository visibility

## Move the work item to Doing

In this part, in Azure Boards you assign yourself a work item that relates to this module. You also move the work item to the **Doing** state. In practice, 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. The work item lets your teammates see what you're working on and how much work is left. It also helps the team enforce work-in-progress (WIP) limits to avoid taking on too much work at one time.

Recall that the team settled on these top issues for the current sprint:



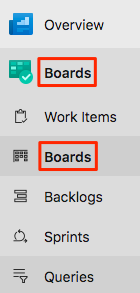
**Note**

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

Here you move the second item, **Connect the web app to a database**, to the **Doing** column. You assign yourself to the work item. **Connect the web app to a database** relates to connecting the Space Game web application to a database and managing changes to the database schema in Azure Pipelines.

To set up the work item:

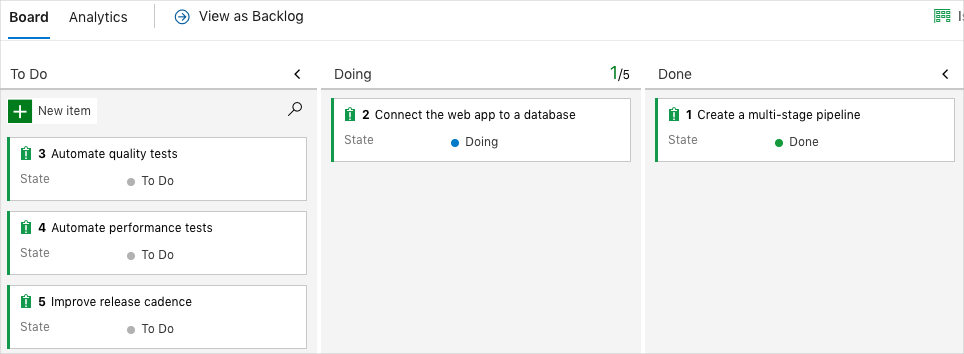
1. In Azure DevOps, navigate to **Boards**. In the menu, select **Boards**.



1. In the **Connect the web app to a database** work item, select the down arrow at the bottom of the card. Assign the work item to yourself.

Azure Boards showing assigning the work item to yourself.

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



At the end of this module, you move the card to the **Done** column after you complete the task.

## 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-deploy** project locally, you can move to the next section.

### Open the integrated terminal

Visual Studio Code comes with an integrated terminal. Here you can both edit files and work from the command line.

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

In the terminal window, you can choose any shell that's installed on your system. For example, you can choose Bash, Zsh, or 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 where you want to work. Choose your home directory (~) or a different directory if you want.

**Bash**

**cd ~**

### Configure Git

If you're new to Git and GitHub, first run a few commands to associate your identity with Git and authenticate with GitHub. For more information, see Set up Git .

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

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

**Note**

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

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

### Set up your project in Visual Studio Code

In the Build applications with Azure DevOps  learning path, you forked and then cloned a Git repository. The repository contains the source code for the Space Game website. Your fork was connected to your projects in Azure DevOps so that the build runs when you push changes to GitHub.

**Important**

In this learning path, we switch to a different Git repository, **mslearn-tailspin-spacegame-web-deploy**. When you ran the template to set up your Azure DevOps project, the process forked the repository automatically for you.

In this part, you clone your fork locally so that you can change 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 that they work as you expect, and then upload those changes to GitHub. You can also synchronize your local copy with changes that other authenticated users have made to the GitHub 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-deploy**) on GitHub .
2. Select **Clone or download**. Then select the button next to the URL to copy the URL to your clipboard:
3. In Visual Studio Code, go to the terminal window.
4. In the terminal, move to the directory where you want to work. Choose your home directory (~) or 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-deploy.git**

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

**Bash**

**cd mslearn-tailspin-spacegame-web-deploy**

### Set the upstream remote

A remote is a Git repository where team members collaborate. It's 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:

**Output**

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

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

Origin specifies your repository on GitHub. When you fork code from another repository, the original remote (the one you forked from) is often named 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-deploy.git**

Run git remote again to see the changes:

**Bash**

**git remote -v**

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

**Output**

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

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

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

### Open the project

In Visual Studio Code, your terminal window points to the root directory of the Space Game web project. You'll now open the project to 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. If you need a reminder of the full path, you can run the pwd command in the terminal window to see the path.

You see the directory and file tree.

**Note**

You might need to open the integrated terminal again 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.