**Integrate Github Project with Azure Pipelines**

**Overview**

With the introduction of Azure DevOps, Microsoft is offering developers a new continuous integration/continuous delivery (CI/CD) service called Azure Pipelines that enables you to continuously build, test, and deploy to any platform or cloud. It has cloud-hosted agents for Linux, macOS, and Windows; powerful workflows with native container support; and flexible deployments to Kubernetes, VMs, and serverless environments.

Azure Pipelines provides unlimited CI/CD minutes and 10 parallel jobs to every GitHub open source project for free. All open source projects run on the same infrastructure that our paying customers use. That means you’ll have the same fast performance and high quality of service. Many of the top open source projects are already using Azure Pipelines for CI/CD, such as Atom, CPython, Pipenv, Tox, Visual Studio Code, and TypeScript-and the list is growing every day.

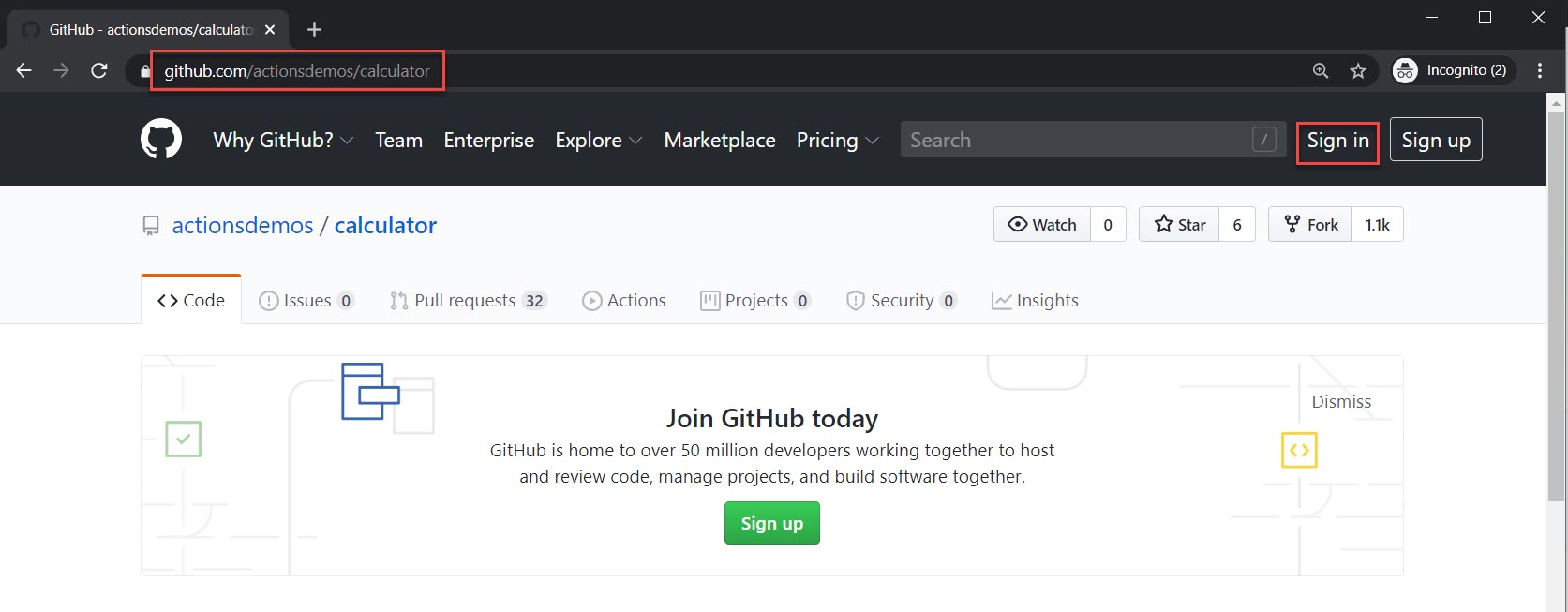
In this lab, you’ll see how easy it is to set up Azure Pipelines with your GitHub projects and how you can start seeing benefits immediately.

## Exercise 1: Getting started with Azure Pipelines

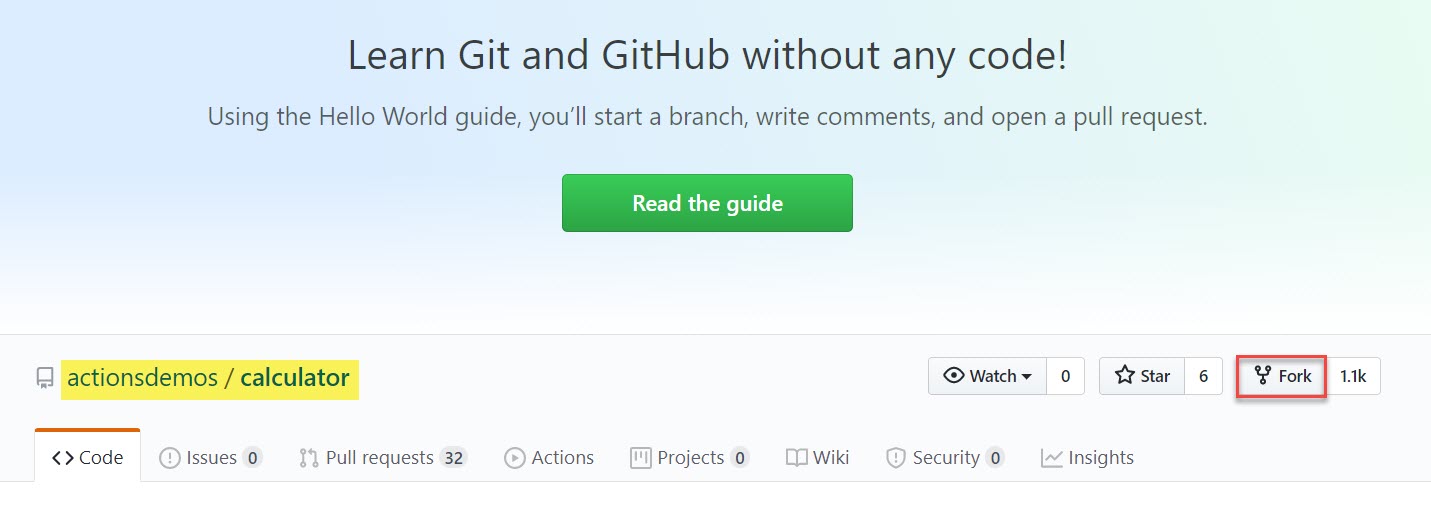
### ***Task 1: Forking a GitHub repo and installing Azure Pipelines***

1. Navigate to <https://github.com/actionsdemos/calculator>

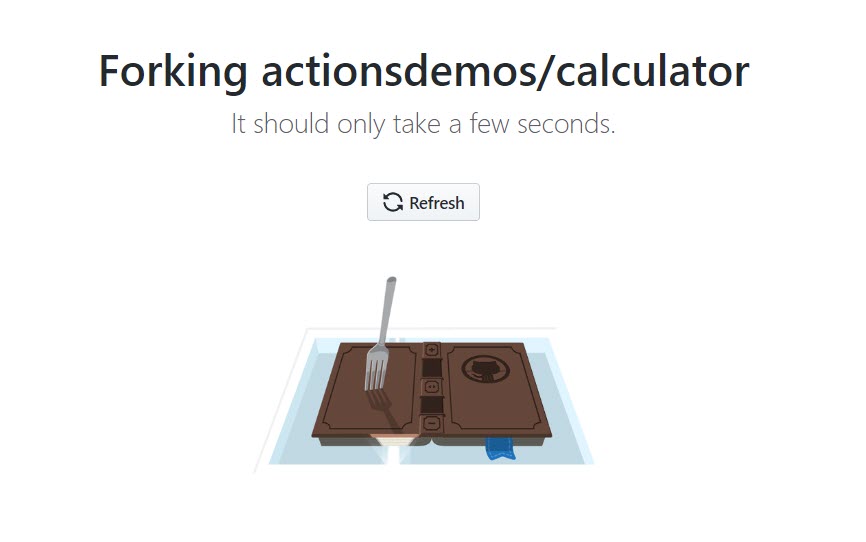
This is the baseline project we will fork and use for this lab.



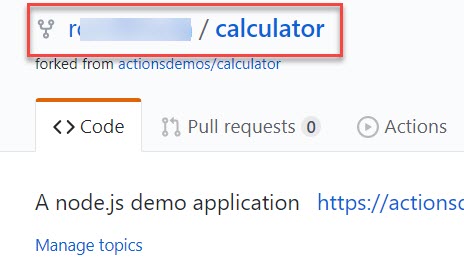
1. If you’re not already signed in to GitHub, sign in now.
2. Click **Fork** to fork the repository to your own account.



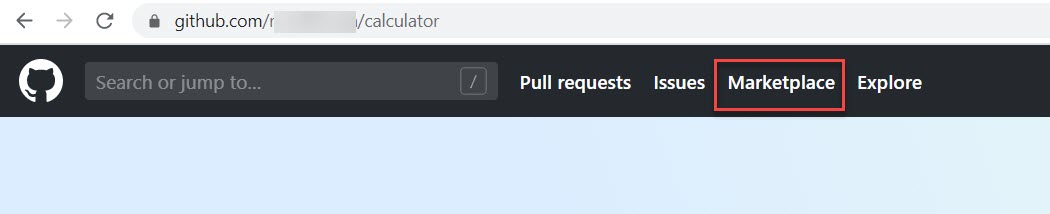
1. If prompted, select an account to fork the repository into.



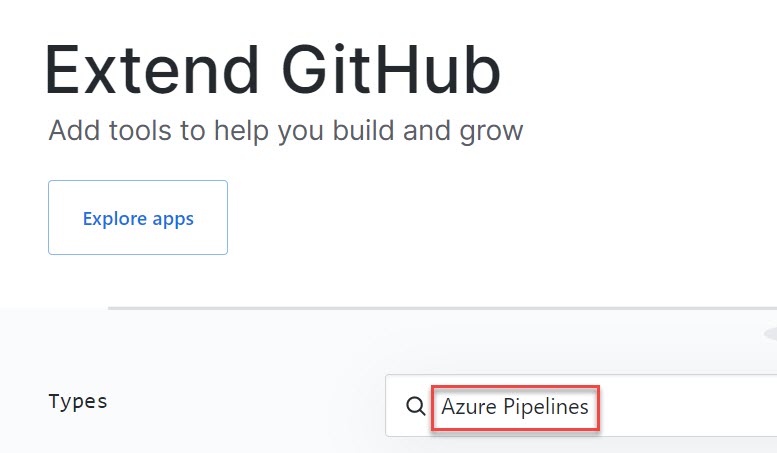
Repository will look as **name/calculator**



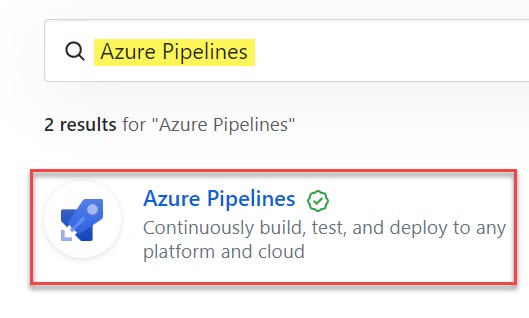
1. The **GitHub Marketplace** provides a variety of tools from Microsoft and 3rd parties that help you extend your project workflows. Click **Marketplace** from the top navigation to visit it.



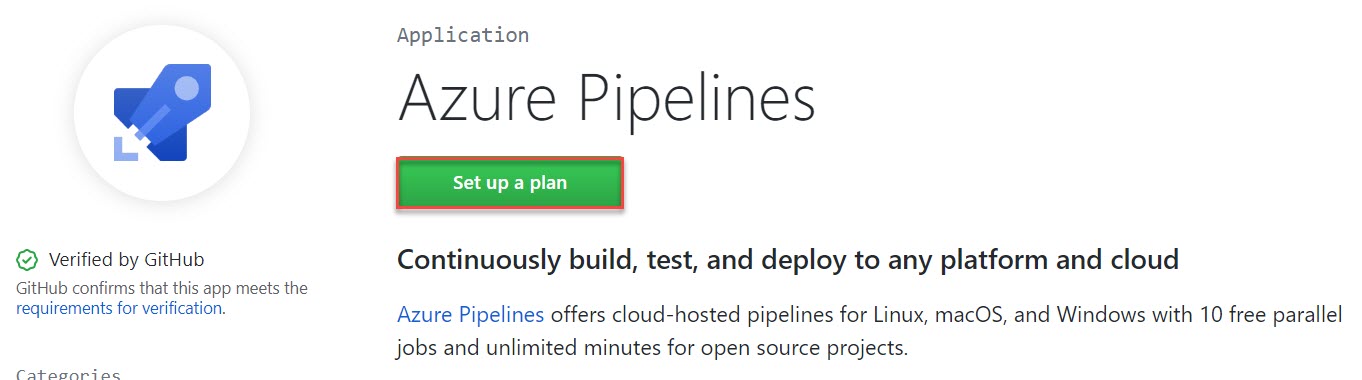
1. Search for **“Azure Pipelines”**



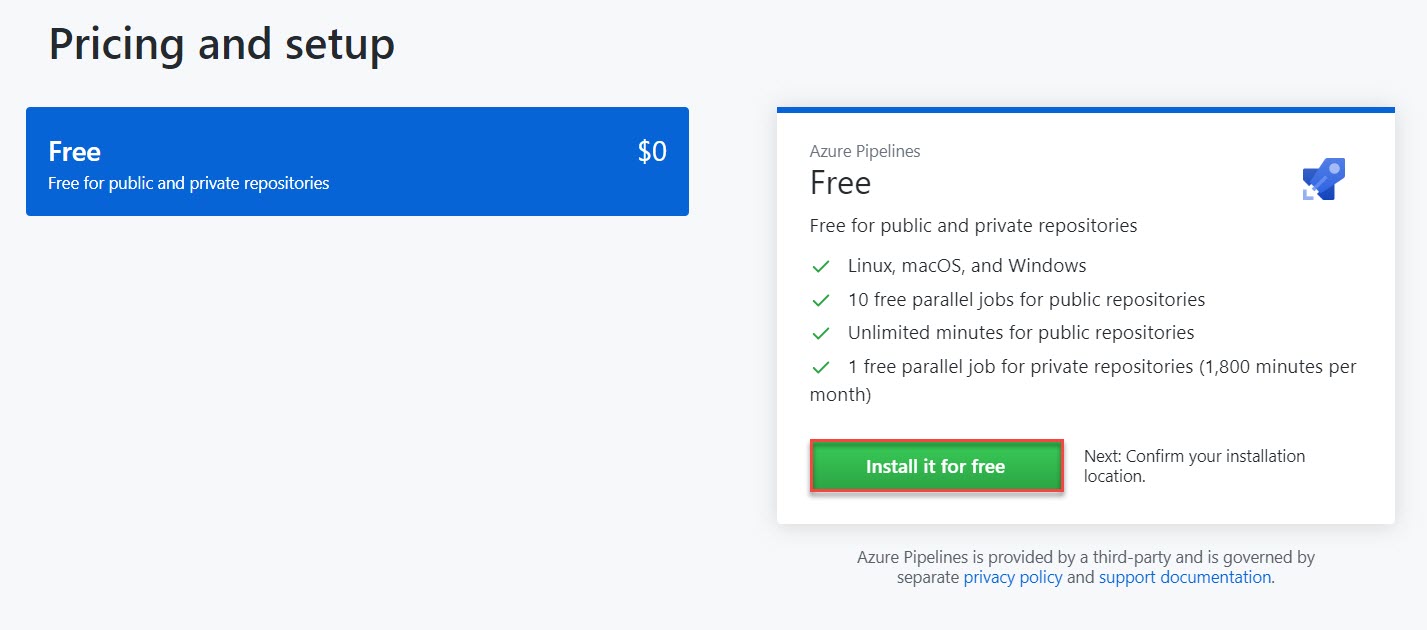
1. Click on **Azure Pipelines**



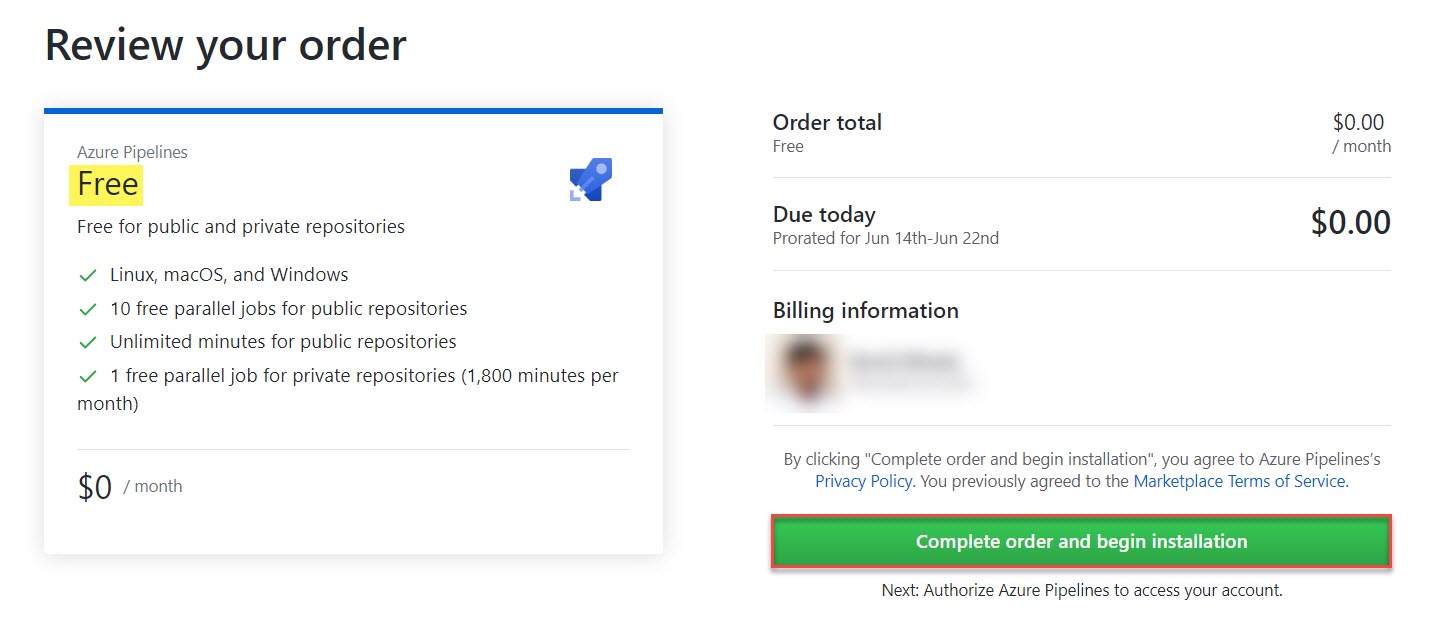
1. Click **Set up a plan**



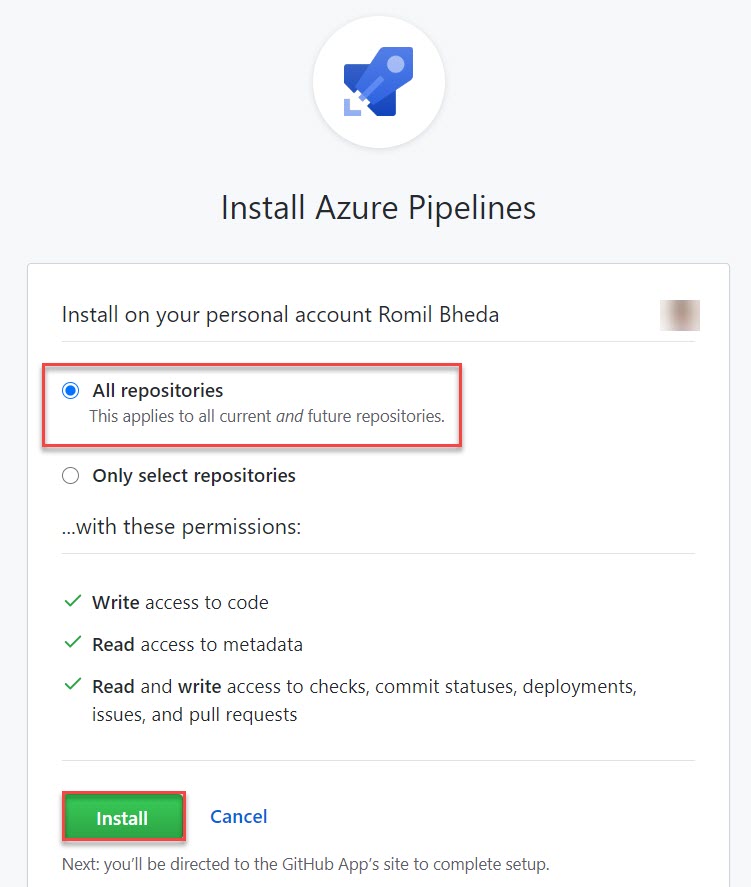
1. The Azure Pipelines offering is free for anyone to use for public repositories, and free for a single build queue if you’re using a private repository. Click **Install it for free**.



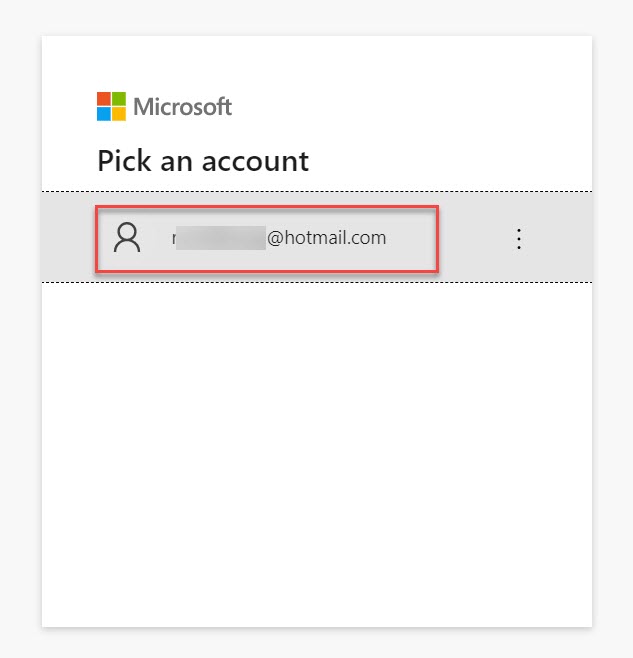
1. Click **Complete order and begin installation**.



1. You have the option to specify repositories to include, but for the purposes of this lab, just include all of them. Note that Azure DevOps requires the listed set of permissions to fulfill its services. Click **Install**.

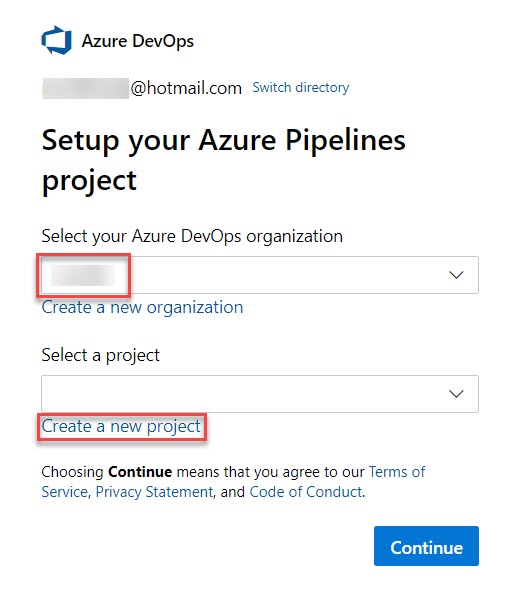


1. You may be prompted to log in to your Microsoft account. Make sure you’re logged into the one associated with your Azure DevOps account



### ***Task 2: Configuring your Azure Pipelines project***

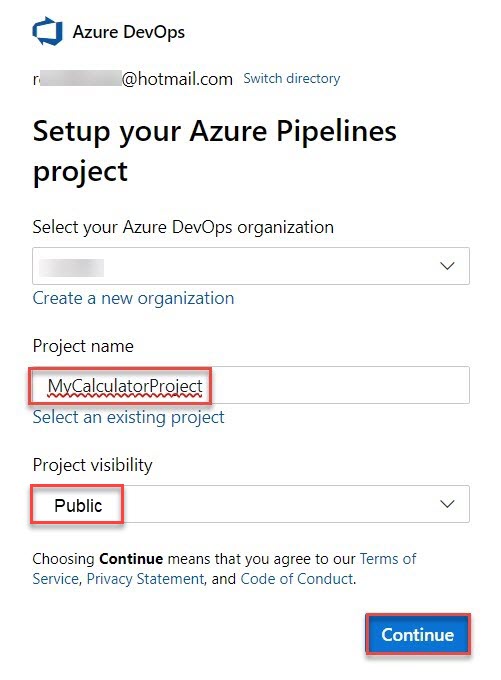
1. You are now on the Azure DevOps site and need to set up your Azure Pipelines project.



Select your Azure DevOps organization: **Choose existing**

Select a project: **Create a new project**

1. Setup Azure Pipelines project

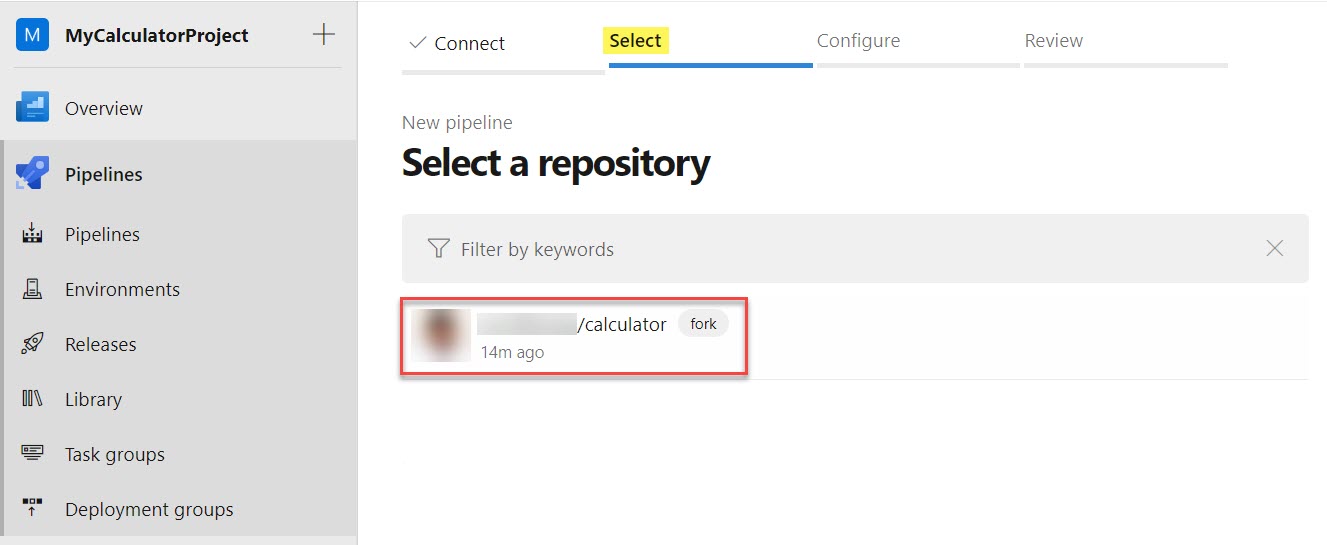


Project Name: **MyCalculatorProject**

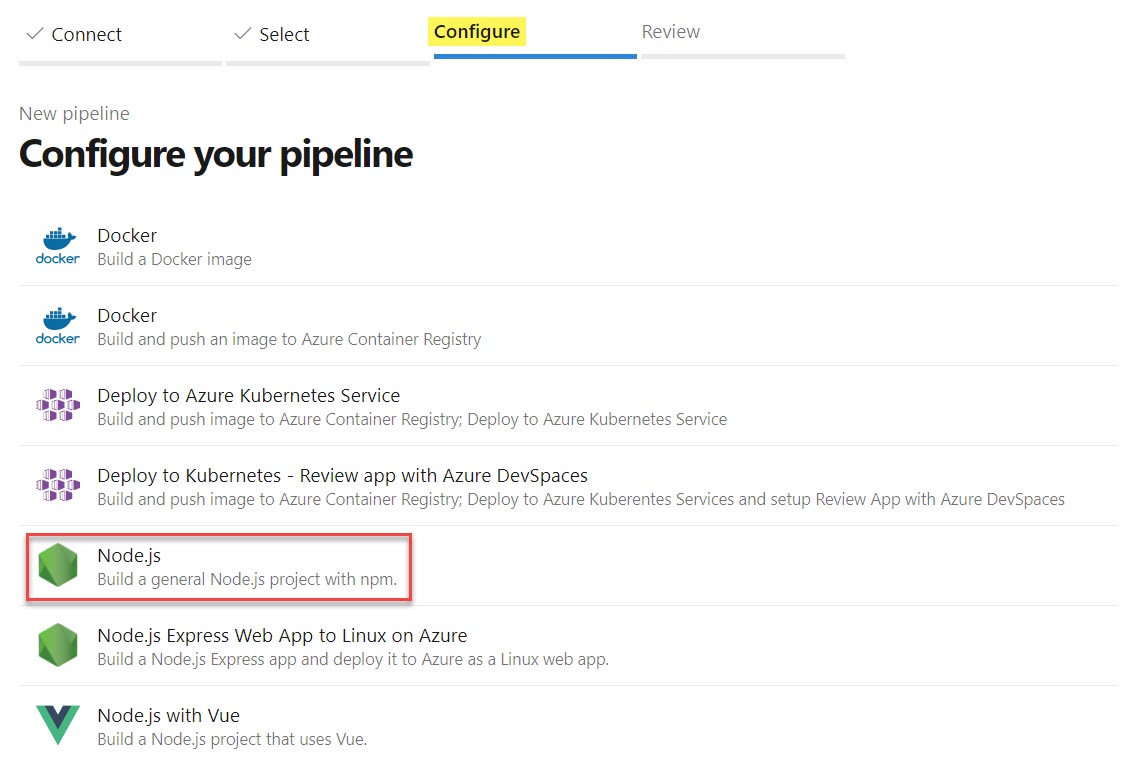
Project visibility: **Public**

click on **Continue** button

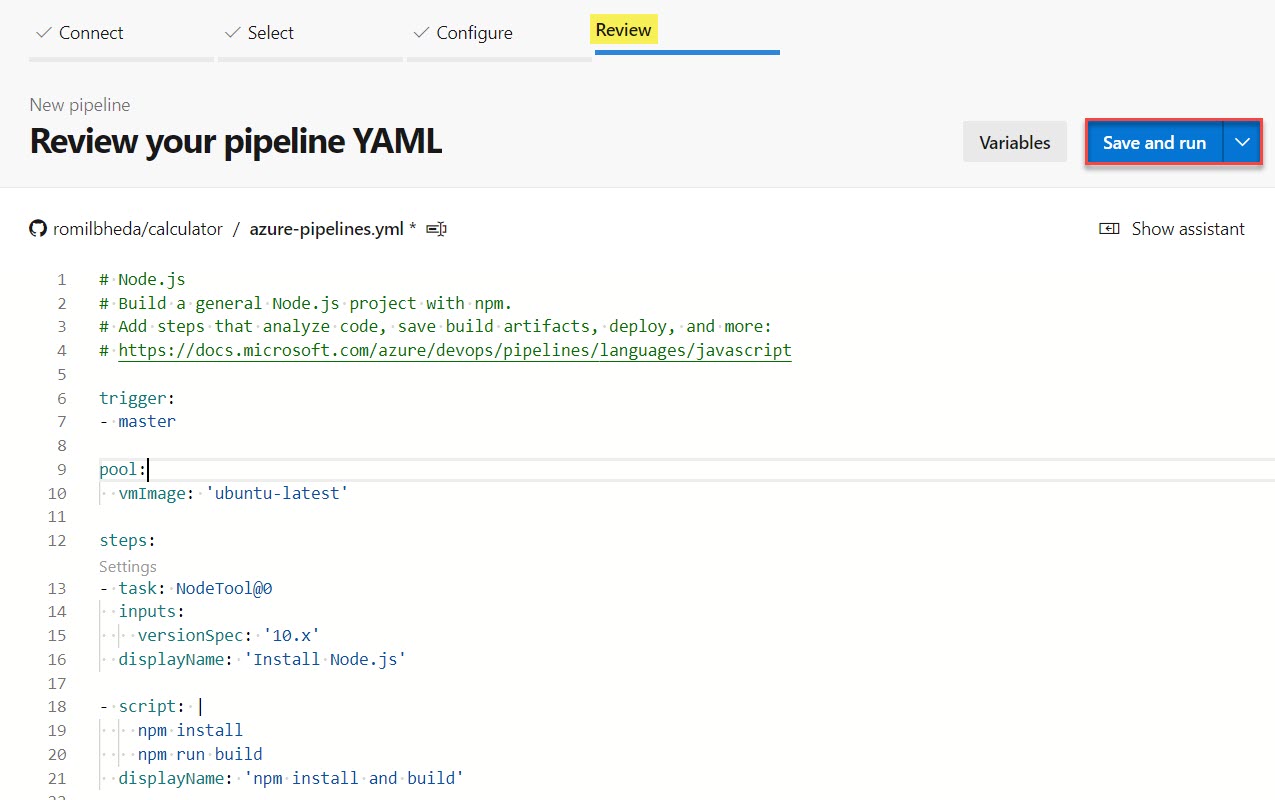
1. Select the **calculator** project from GitHub to build as part of the pipeline.



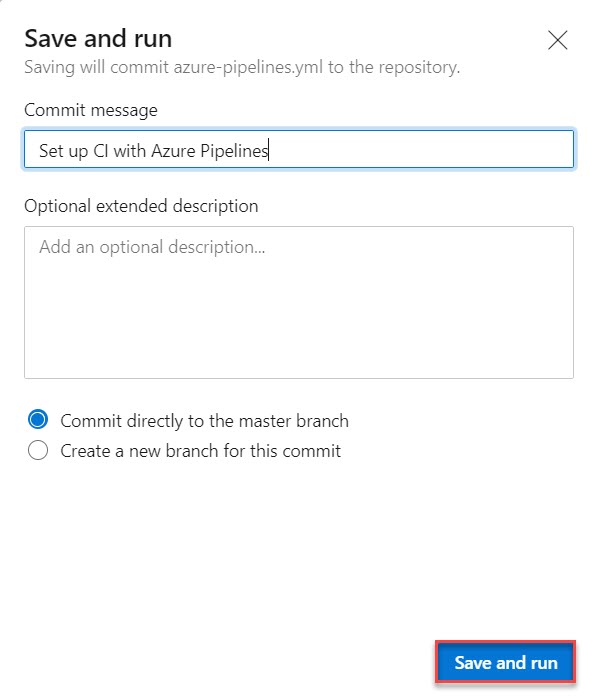
1. Azure Pipelines will analyze your project in an attempt to determine if any existing templates would be a good fit. In this case, the recommended template is for **Node.js**, which is perfect for our needs. Some alternative templates are also suggested, although the recommended one is the best for this lab. Select it to continue.



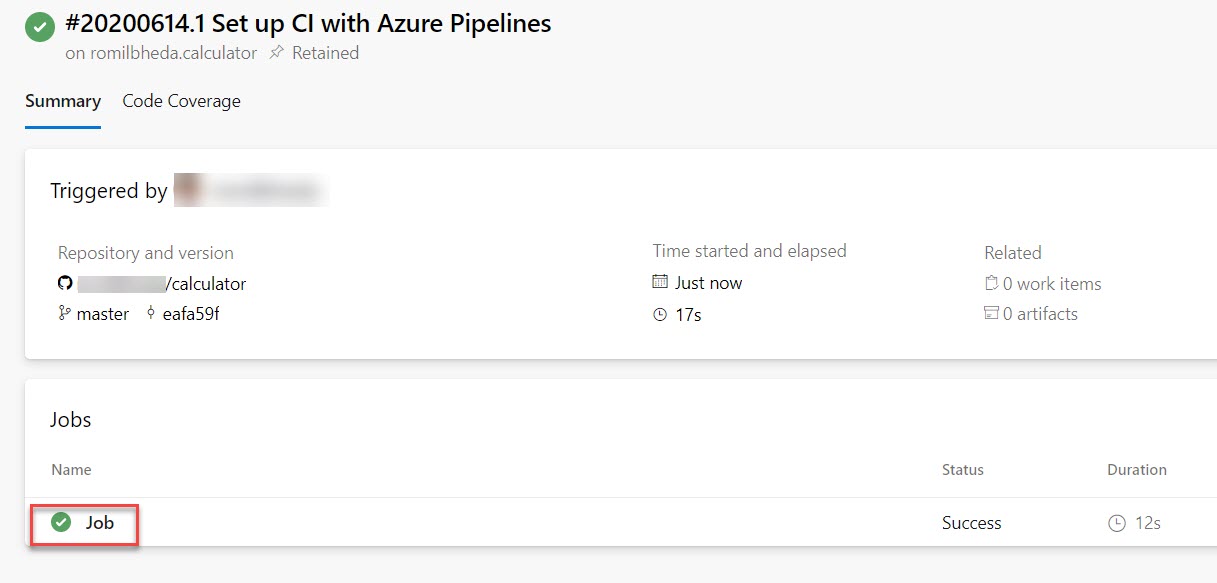
1. The build pipeline is defined as **YAML**, a markup syntax well-suited to defining processes like this because it allows you to manage the configuration of the pipeline like any other file in the repo. It’s a pretty simple template that identifies the pool to pull a VM from for building, the process to install Node.js for building, and the actual build itself. Click **Save and run** to save the pipeline and queue a new build.



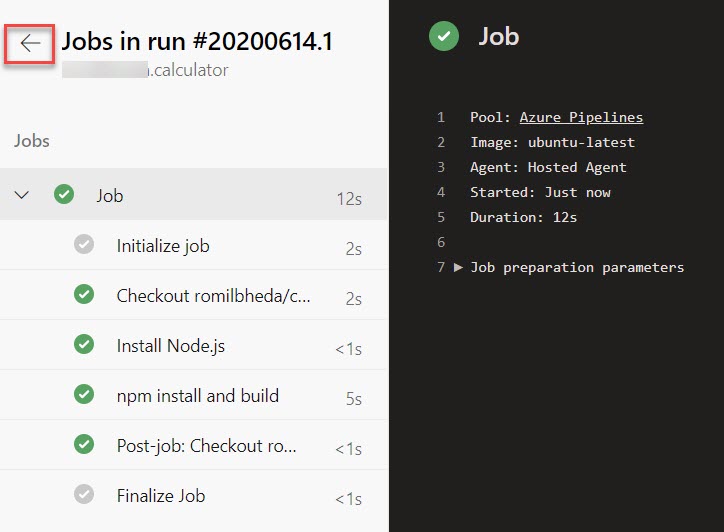
1. For the purposes of this lab, you can commit this new file directly to the master branch. Click **Save and run**.



1. It will take a moment for the pipeline to complete. During this time it will configure the build agent, pull in the source from GitHub, and build it according to the pipeline definition.

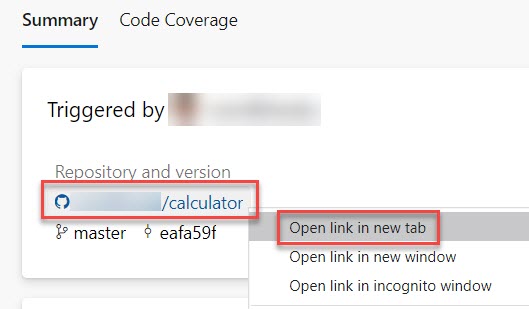


1. The build should complete successfully.

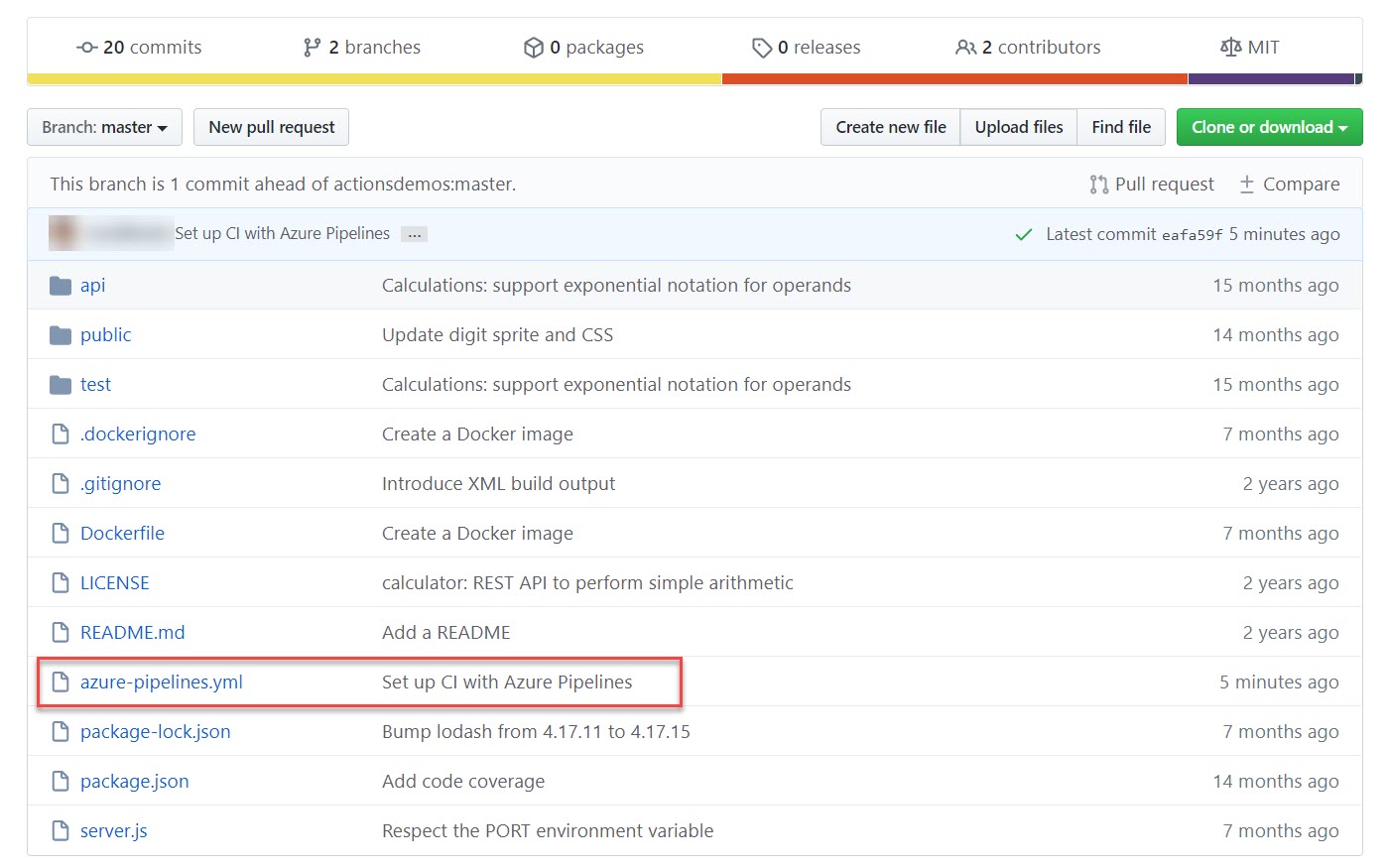


### ***Task 3: Modifying a YAML build pipeline definition***

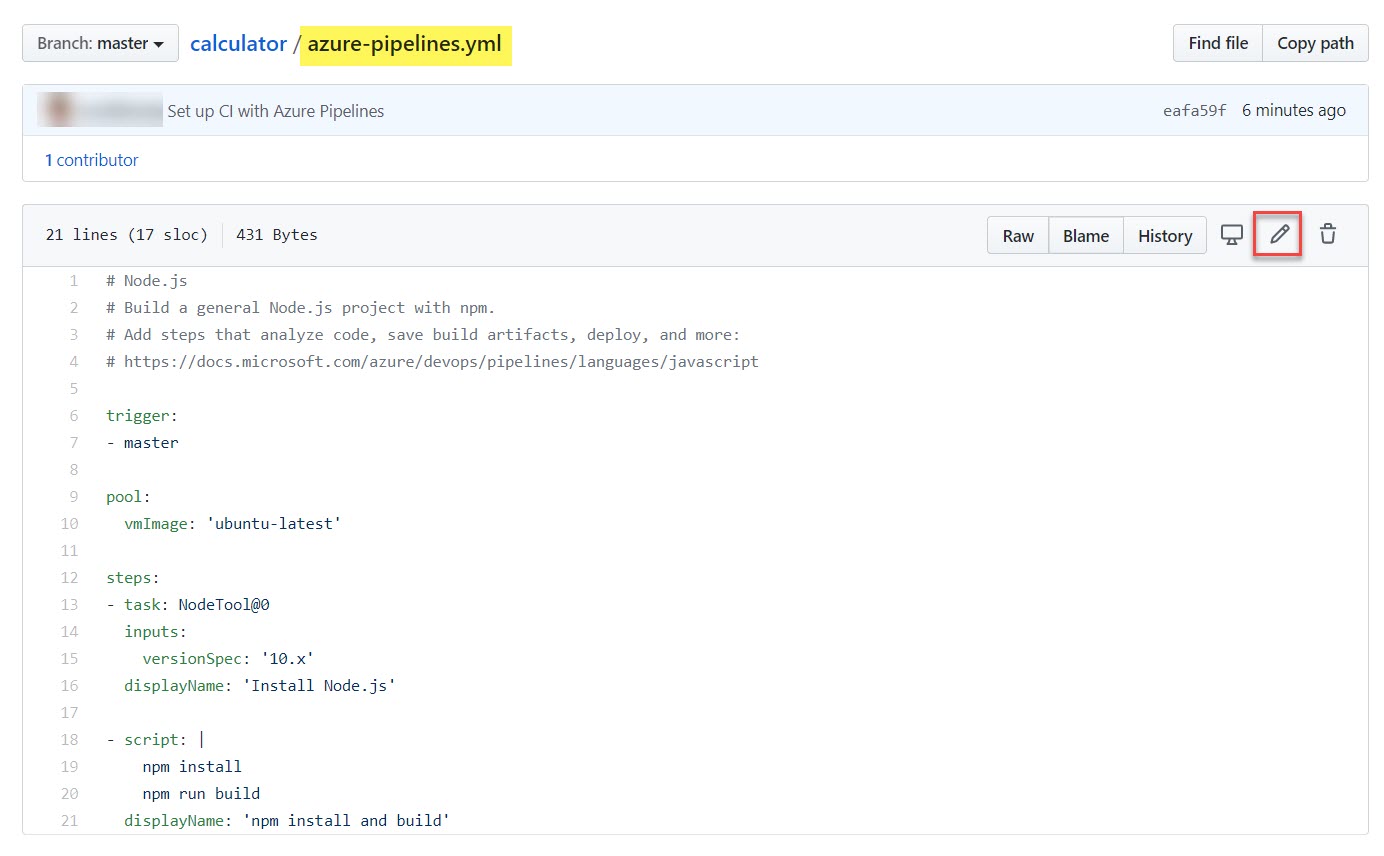
1. While the default pipeline is a great start, it doesn’t do everything we would like to have automated. For example, it would be great if it also ran our tests to confirm that the changes don’t create bugs. Let’s return to GitHub where we can edit the YAML by hand. **Right-click the GitHub project link** and select **Open link in new tab**. Since this lab will involve stepping back and forth between GitHub and Azure DevOps, it’ll be easier to keep a browser tab open to each.



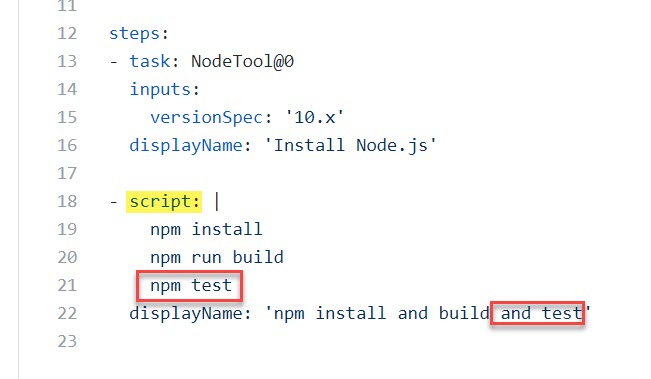
1. Click on **azure-pipelines.yml**.



1. Click the **Edit** button.



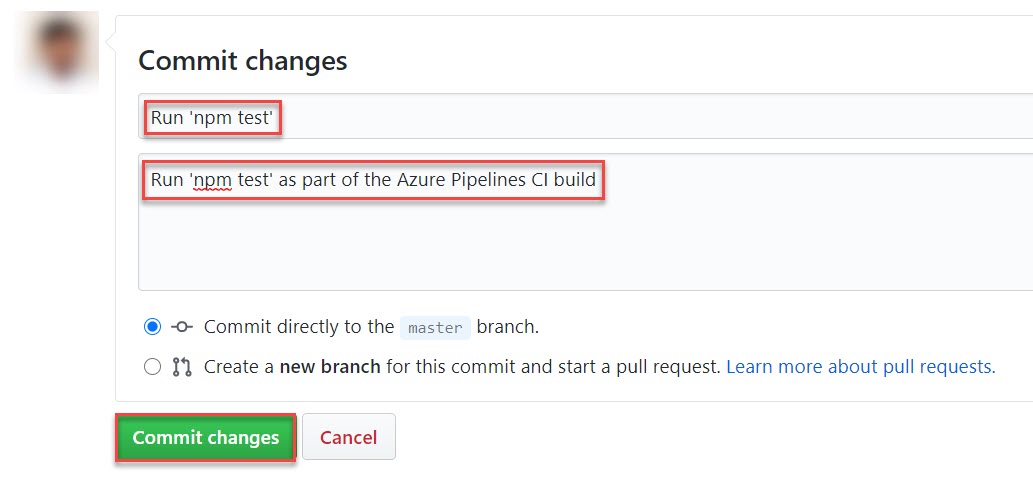
1. Our project already contains tests written using Mocha so we just need to execute them in out pipeline. To add the test run, add the **“npm test”** command at line number 21 as below. Also update the **displayName** to **‘npm install and build and test’** so that it’s easier to track what each task of the build is doing later on. These are the lines to add.



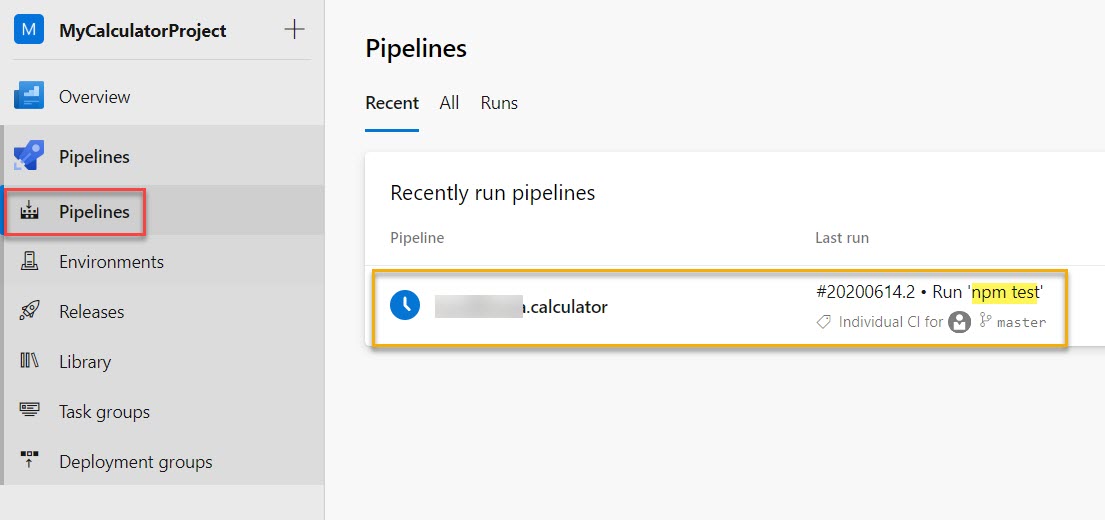
1. Scroll to the bottom of the page, provide some documentation for the change and click **Commit changes**. Again, it’s okay to commit this change directly to the master branch for the purposes of this lab.

Changes: Run ‘npm test’

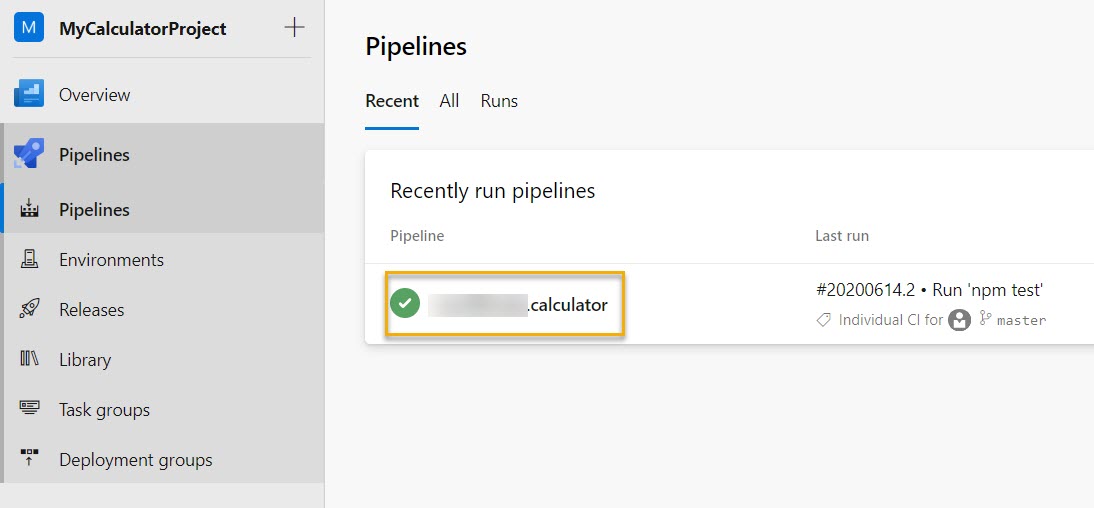
Description: Run ‘npm test’ as part of the Azure Pipelines CI build



1. back to the **Azure DevOps** browser tab. Click on **Pipelines**



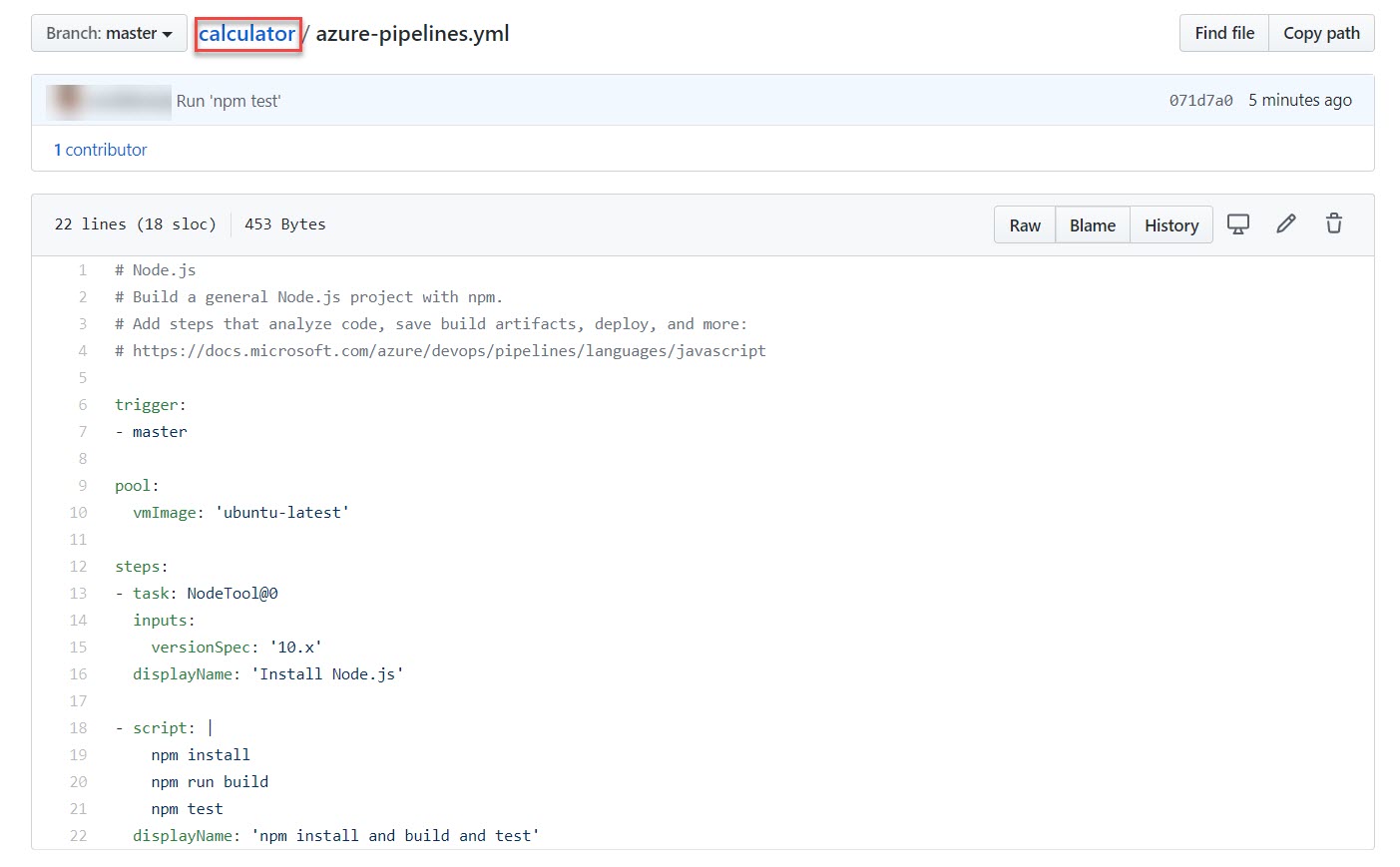
1. A new build should already be there. Click it to view progress.



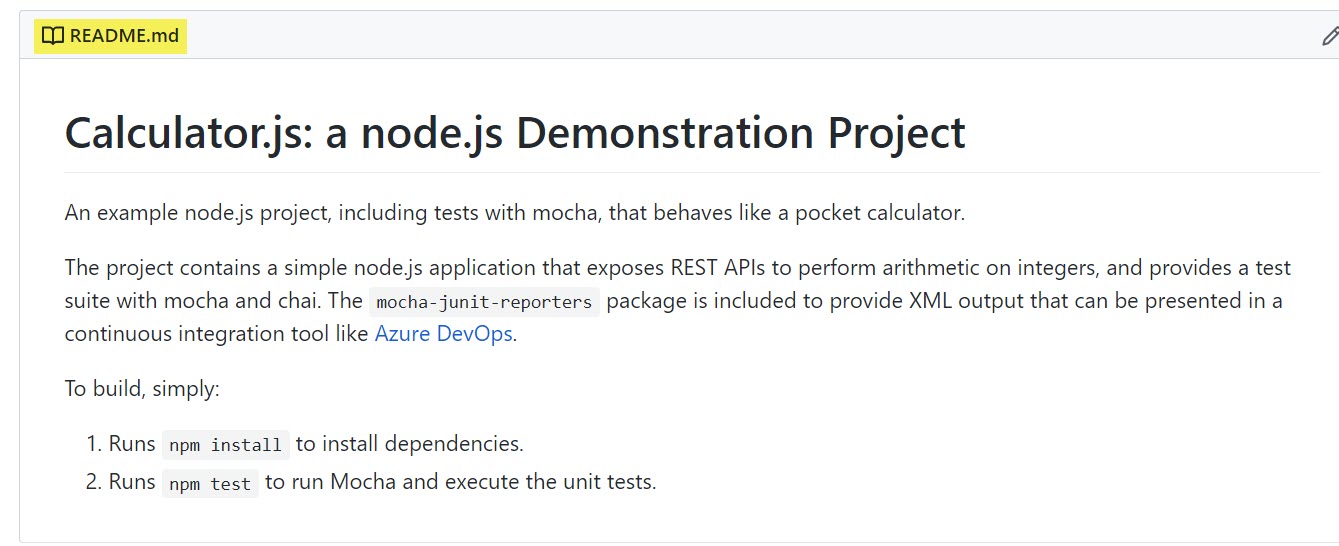
### ***Task 4: Adding a build status badge***

1. An important sign for a quality project is its build status badge. When someone finds a project that has a badge indicating that the project is currently in a successful build state, it’s a sign that the project is maintained effectively.

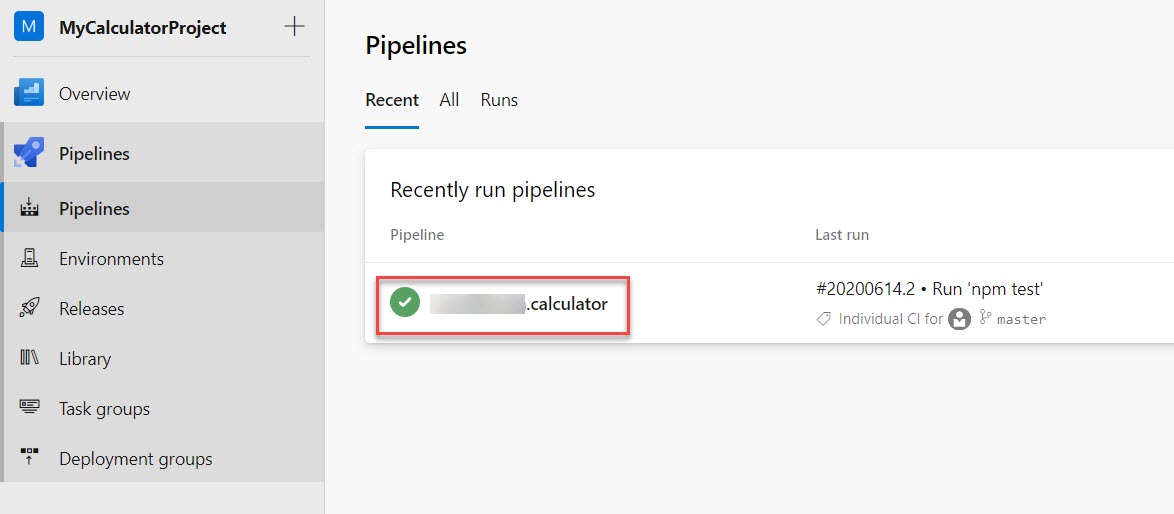
Navigate to **Github tab** in browser and click on **calculator** repository



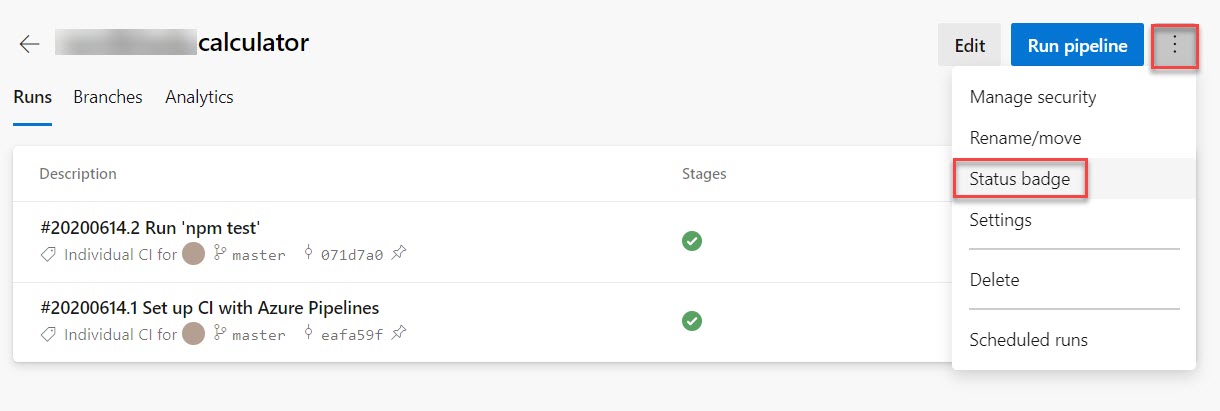
Scroll down



1. Navigate to **Azure DevOps**, Click the build pipeline to navigate to its overview page.



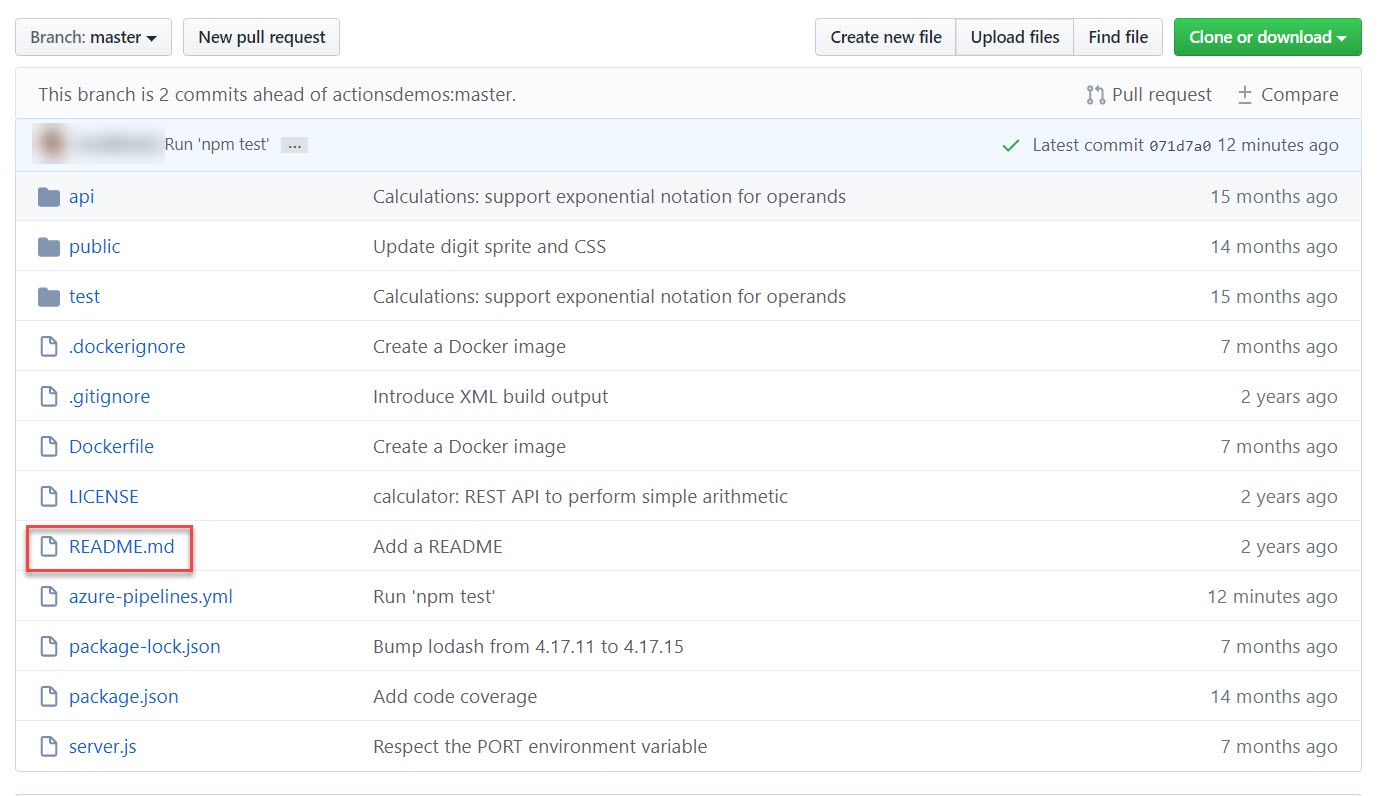
1. Click on **vertical line of** **dots** dropdown, select **Status badge**.



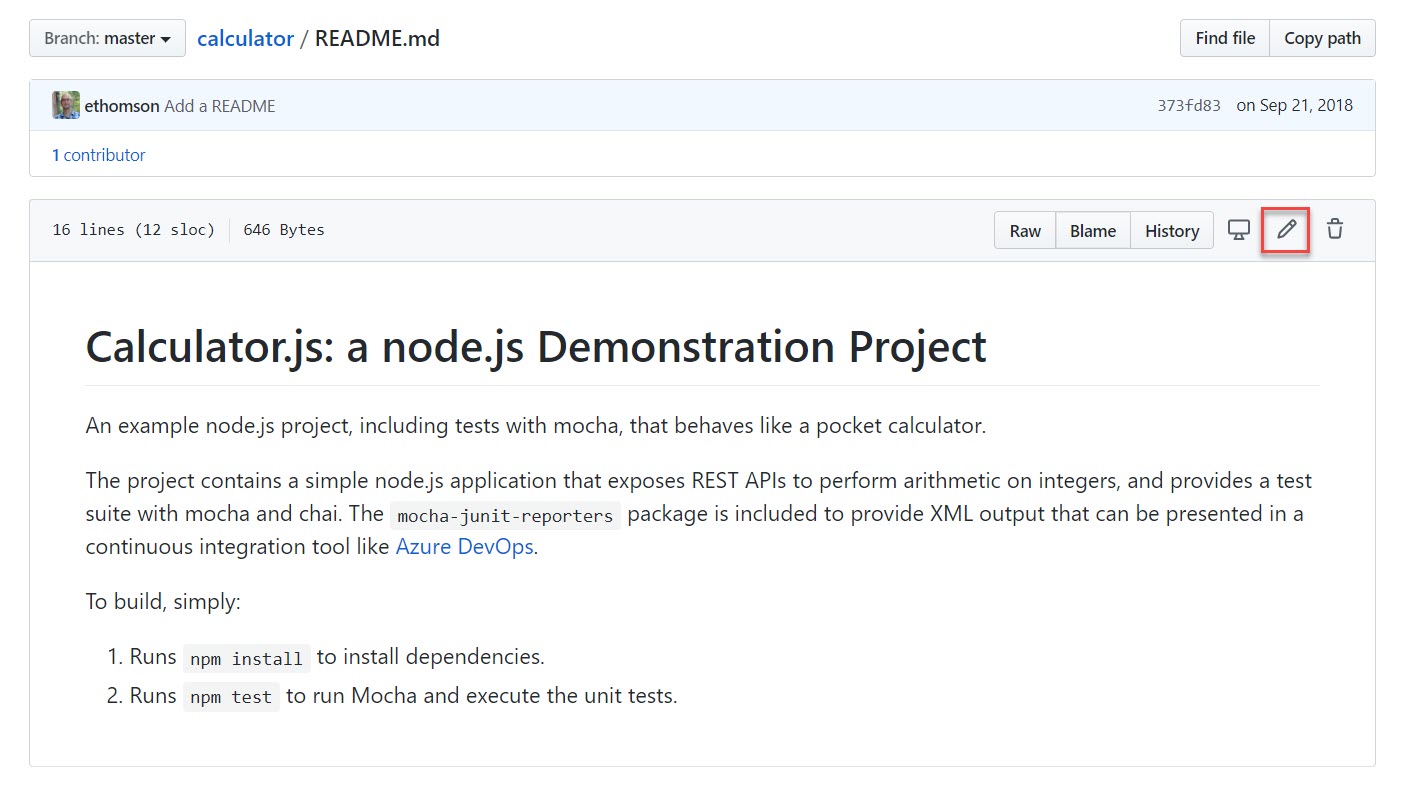
1. The **Status badge** UI provides a quick and easy way to integrate the build status wherever you want. Often, you’ll want to use the provided URLs in your own dashboards, or you can use the Markdown snippet to add the status badge to locations such as Wiki pages. Click the **Copy to clipboard** button for **Sample Markdown**.



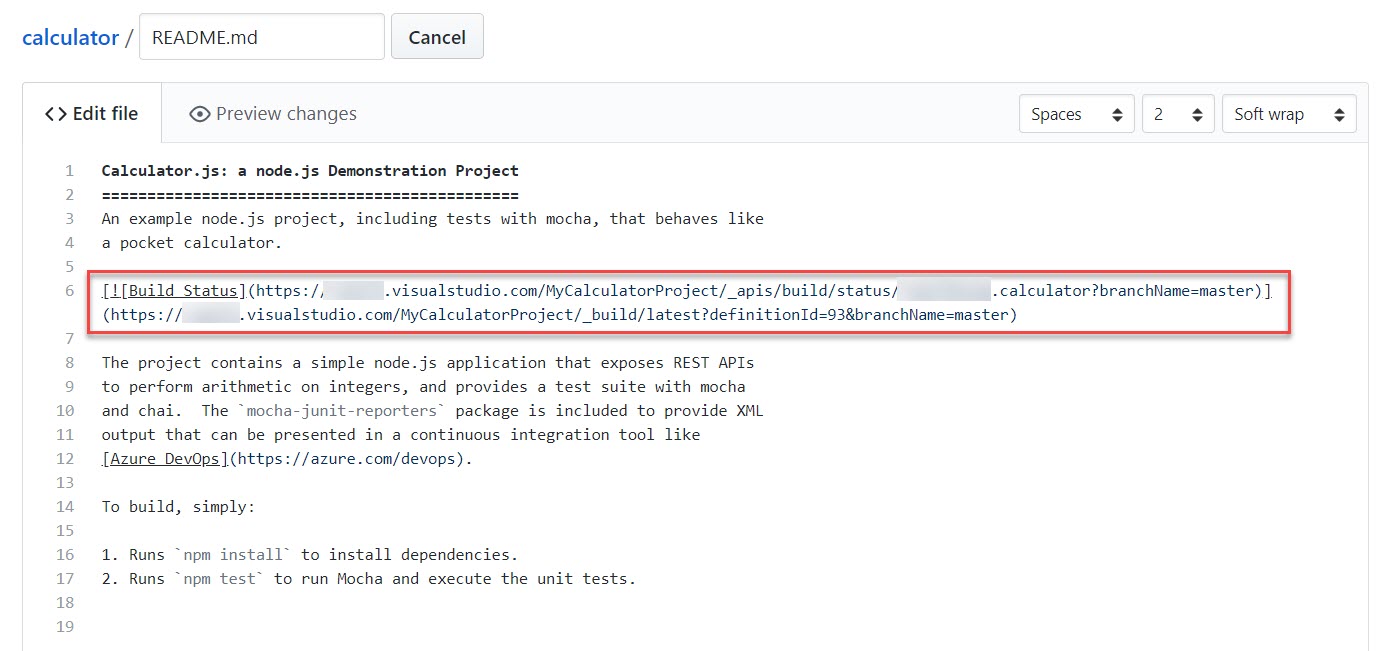
1. Return to the **GitHub** tab and click on **README.md** file



1. Click the **Edit** button.

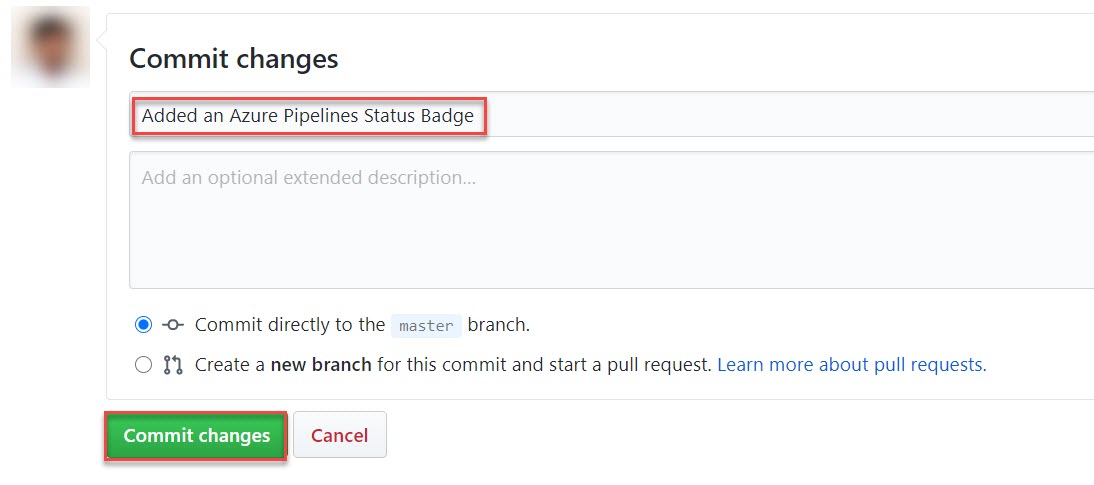


1. Paste in the clipboard contents around line **6**.

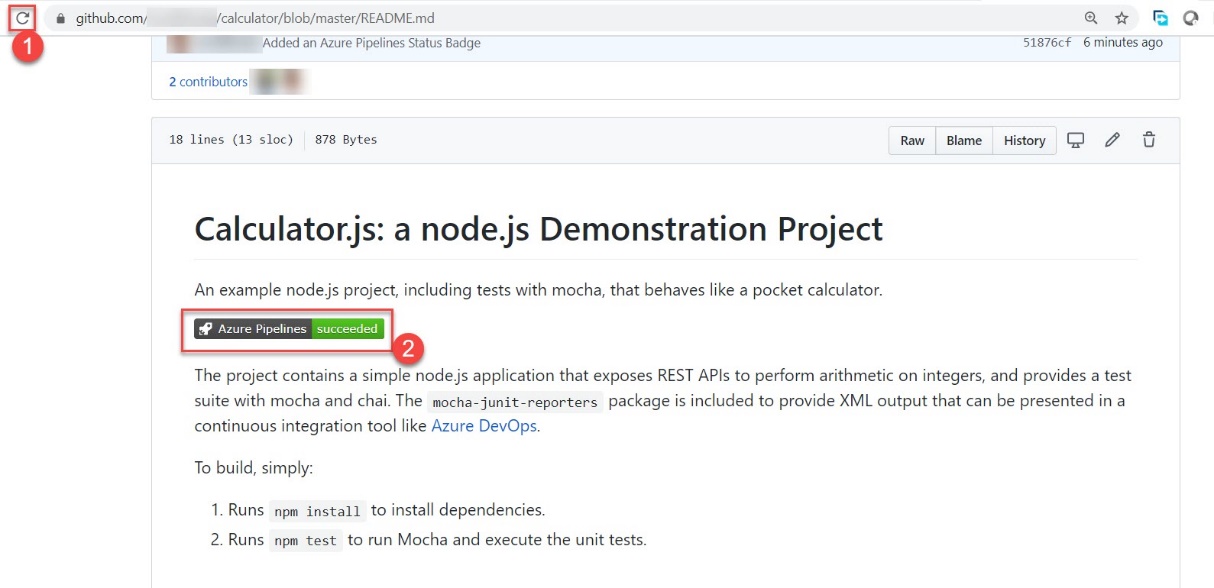


1. Scroll down and add a commit comment and click **Commit changes**.

Added an Azure Pipelines Status Badge



1. Scroll down & check the **Status**



You now have a dynamic build status badge on your project’s front page that allows everyone to know that you’re effectively managing your project.