BUS5001 – CLOUD PLATFORMS AND ANALYTICS – WORKSHOP

Use a GitHub Actions Pipeline to deploy a Static Website in Azure Storage

In last week's workshop, we hosted a static website in Azure Blob storage. If you recall, we uploaded the website files manually to the web container using the Azure portal. This process is not ideal in a real-world web development scenario and makes for more manual work. In this workshop, you will get a hands-on introduction to using a Continuous Integration (CI)/ Continuous Delivery (CD) tool, GitHub Actions to build a deployment pipeline which will automate the process of deploying a static website in Azure Blob storage.

IMPORTANT:

- The services covered in this course are only a subset of a much larger family of Azure services and GitHub. Similar outcomes can be achieved by leveraging other services and/or features not covered by this workshop. Specific business requirements may also ask for the use of different services or features not included in this workshop.
- Some concepts presented in this course can be quite complex and you may need to seek more information from different sources to complement your understanding of the Azure services and GitHub covered.

Document Structure

This document contains detailed step-by-step instructions on how to deploy a static website in Azure Blob storage using GitHub Actions workflow. It's recommended you carefully read the detailed description contained in this document to complete this workshop.

You will see the label **IMPORTANT** whenever there is a critical step. Please pay close attention to the instructions given.

Table of Contents

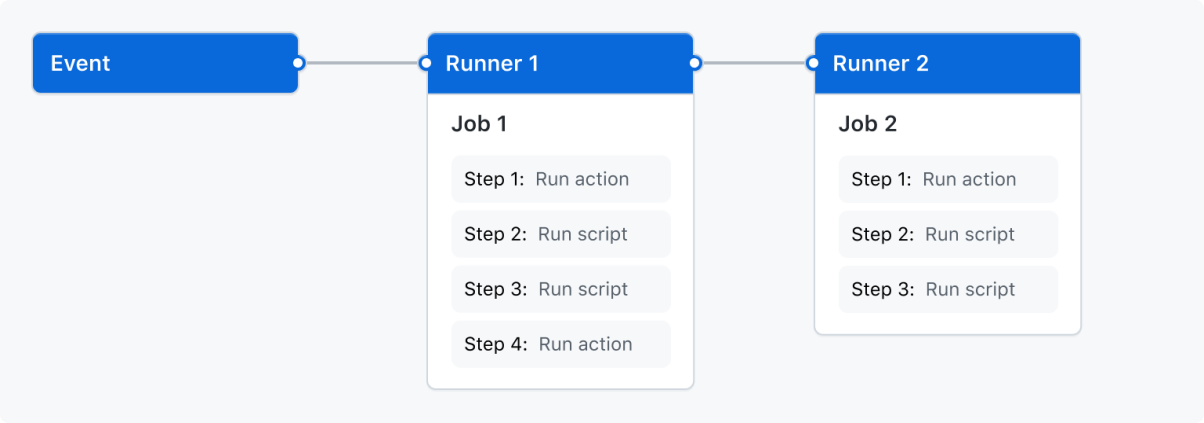
Introduction to Git and GitHub

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. GitHub is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository or deploy merged pull requests to production.

The components of GitHub Actions

You can configure a GitHub Actions workflow to be triggered when an event occurs in your repository, such as a pull request being opened, or an issue being created. Your workflow contains one or more jobs which can run in sequential order or in parallel. Each job will run inside its own virtual machine runner, or inside a container, and has one or more steps that either run a script that you define or run an action, which is a reusable extension that can simplify your workflow.



Workflows

A workflow is a configurable automated process that will run one or more jobs. Workflows are defined by a YAML file checked in to your repository and will run when triggered by an event in your repository, or they can be triggered manually, or at a defined schedule.

Workflows are defined in the .github/workflows directory in a repository, and a repository can have multiple workflows, each of which can perform a different set of tasks. For example, you can have one workflow to build and test pull requests, another workflow to deploy your application every time a release is created, and still another workflow that adds a label every time someone opens a new issue.

Events

An event is a specific activity in a repository that triggers a workflow run. For example, activity can originate from GitHub when someone creates a pull request, opens an issue, or pushes a commit to a repository. You can also trigger a workflow run on a schedule, by posting to a REST API, or manually.

Jobs

A job is a set of steps in a workflow that executes on the same runner. Each step is either a shell script that will be executed or an action that will be run. Steps are executed in order and are dependent on each other. Since each step is executed on the same runner, you can share data from one step to another. For example, you can have a step that builds your application followed by a step that tests the application that was built.

You can configure a job's dependencies with other jobs; by default, jobs have no dependencies and run in parallel with each other. When a job takes a dependency on another job, it will wait for the dependent job to complete before it can run. For example, you may have multiple build jobs for different architectures that have no dependencies, and a packaging job that is dependent on those jobs. The build jobs will run in parallel, and when they have all completed successfully, the packaging job will run.

Actions

An action is a custom application for the GitHub Actions platform that performs a complex but frequently repeated task. Use an action to help reduce the amount of repetitive code that you write in your workflow files. An action can pull your git repository from GitHub, set up the correct toolchain for your build environment, or set up the authentication to your cloud provider.

You can write your own actions, or you can find actions to use in your workflows in the GitHub Marketplace.

Runners

A runner is a server that runs your workflows when they're triggered. Each runner can run a single job at a time. GitHub provides Ubuntu Linux, Microsoft Windows, and macOS runners to run your workflows; each workflow run executes in a fresh, newly-provisioned virtual machine. If you need a different operating system or require a specific hardware configuration, you can host your own runners. For more information about self-hosted runners, see "Hosting your own runners."

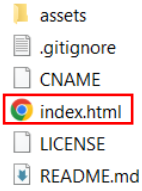
Prerequisites

- An Azure account and subscription. If you don't have a subscription, sign up for a free Azure account using your La Trobe student account.
- A GitHub account. If you do not have a GitHub account, sign up for free.

Use GitHub Actions workflow to deploy a static website in Azure Storage

Download and modify the static website content available in LMS

1. Download the resources file from LMS, unzip, go to the sample site folder and open the index.html file using notepad.

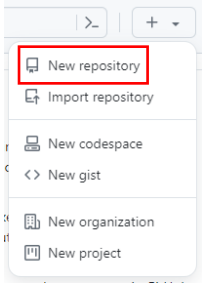


2. Replace “Your name here” with your name, save and close the file.

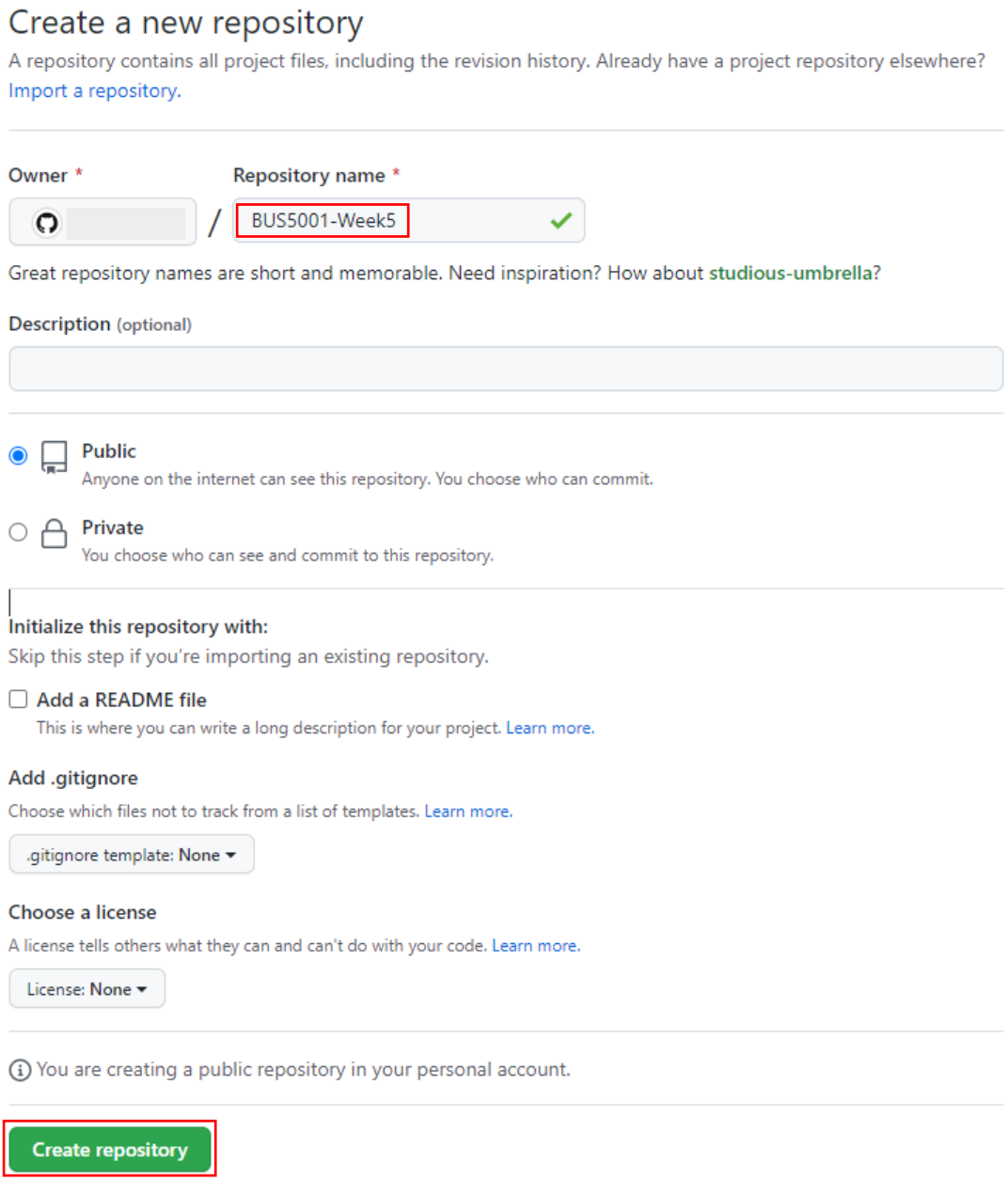
```
<!DOCTYPE HTML>
<!--
        BUS5001 - Tutorial (Week 4)
-->
<html>

<head>
        <title>Your name here</title>
        <meta charset="utf-8" />
        <meta name="description" content="">
        <meta name="keywords"
                content="student, la trobe">
        <meta name="robots" content="index, follow">
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <meta name="revisit-after" content="1 days">
        <meta name="author" content="BUS5001">
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!--[if lte IE 8]><script src="assets/js/ie/html5shiv.js"></script><![endif]-->
        <link rel="stylesheet" href="assets/css/main.css" />
        <!--[if lte IE 8]><link rel="stylesheet" href="assets/css/ie8.css" /><![endif]-->
        <!--[if lte IE 9]><link rel="stylesheet" href="assets/css/ie9.css" /><![endif]-->
        <link rel="shortcut icon" href="assets/css/images/icon.ico">
        <!--[if lte IE 9]><link rel="stylesheet" href="assets/css/images/icon.ico" /><![endif]-->
</head>

<body class="Loading">
        <div id="wrapper">
                <div id="bg"></div>
                <div id="overlay"></div>
                <div id="main">

                        <!-- Header -->
                        <header id="header">
                                <h1>Your name here</h1>
                                <p>Student  &bull;  La Trobe University </p>
                                <nav>
                                        <ul>
```

Create a new repository with the static website content

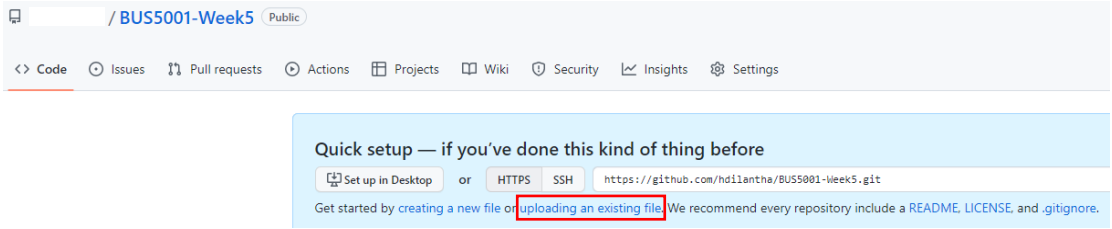
1. Sign into GitHub, In the upper-right corner of any page, use the drop-down menu, and select **New repository**.



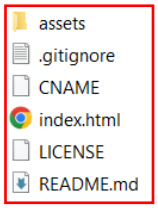
2. Give a repository name and select **Create repository**



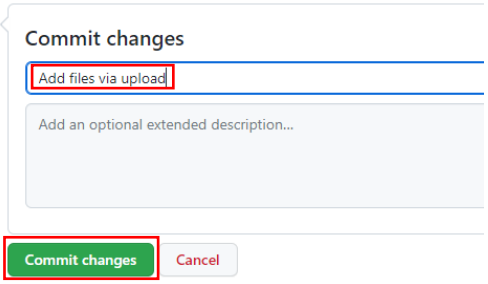
3. In the **Quick setup** tab, Select **uploading an existing file**



4. Upload all 5 files and assets folder (that you downloaded from LMS)

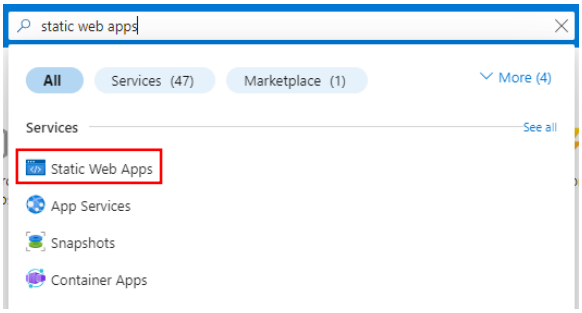


5. It will show 30 files uploaded, give a commit name, and click **Commit changes**.

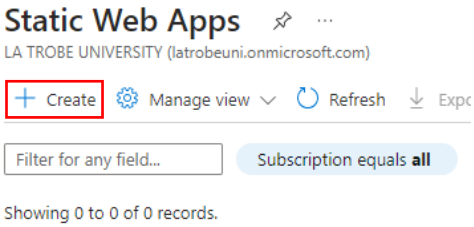


Create Azure Static Web App

1. Sign in to Azure portal.
2. Search and select **Static Web Apps**.



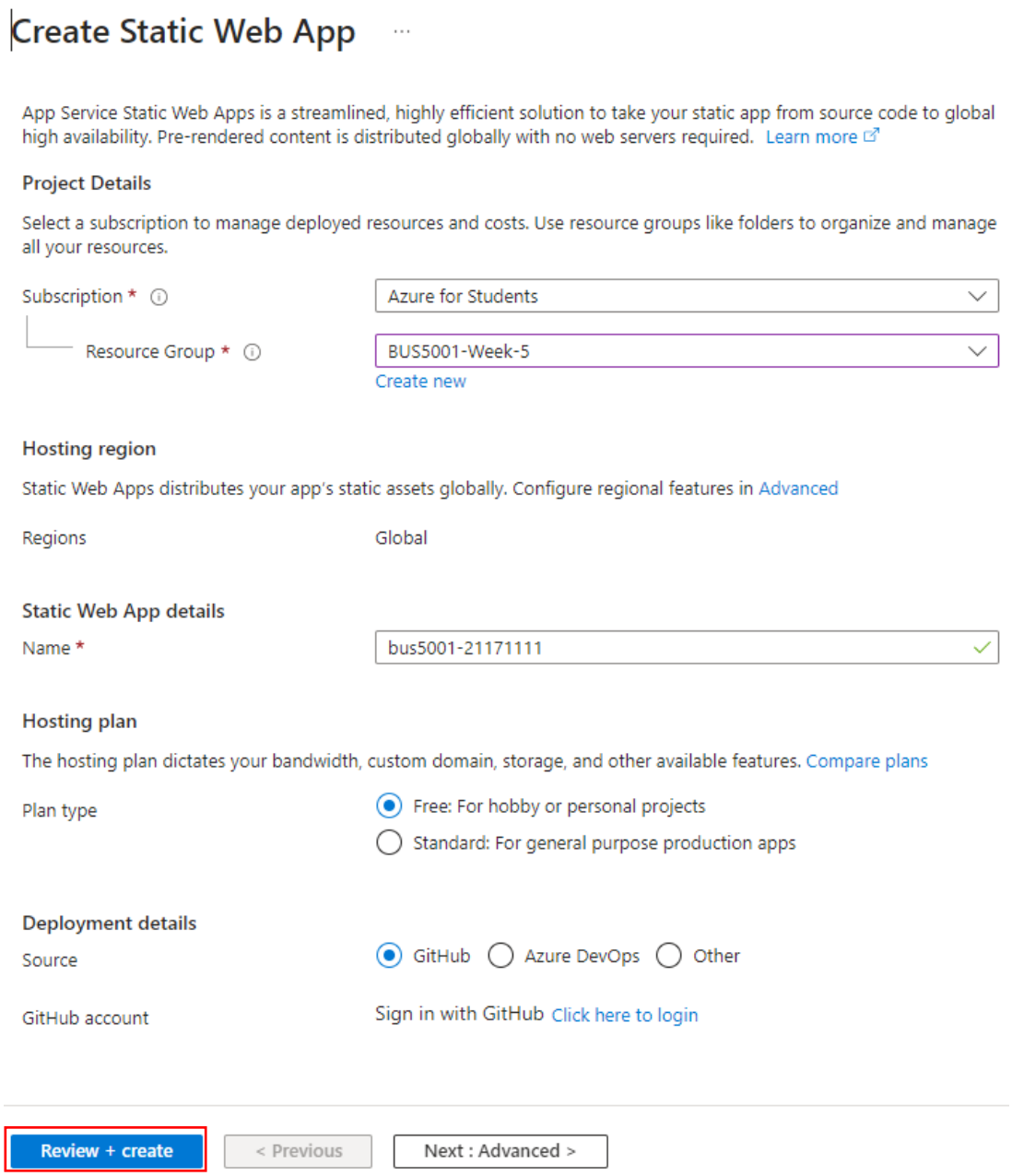
3. Select **+ Create** to create a new static web app.



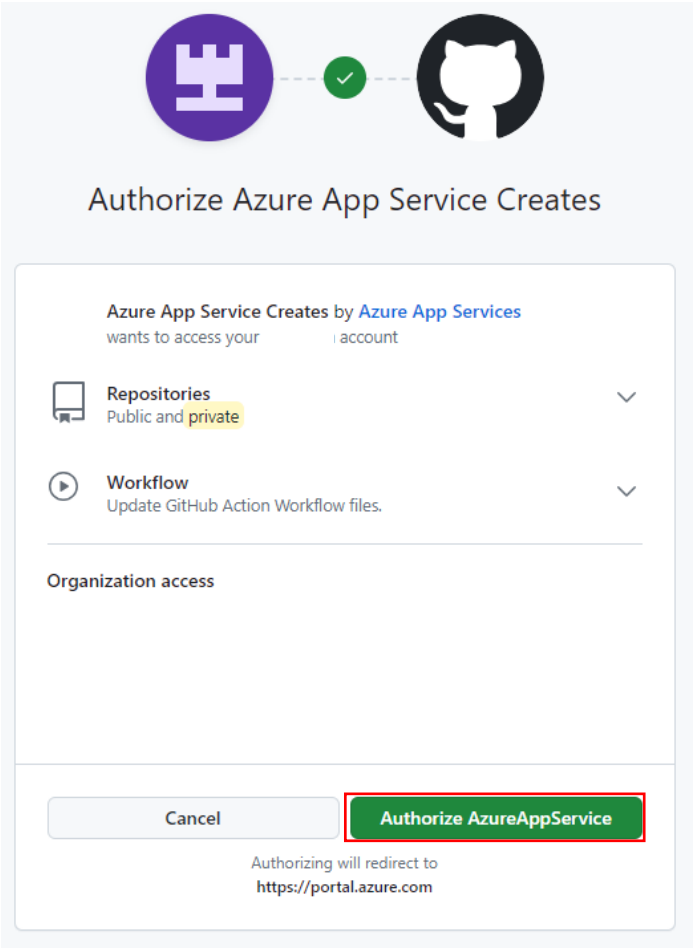
1. On the **Create Static Web App** pane, provide the following basic information about your storage account:

Property	Required	Description
Subscription	Yes	Your Azure subscription name. Select **Azure for Students**
Resource Group	Yes	The Azure resource group where you create your logic app and related resources. This name must be unique across regions and can contain only letters, numbers, hyphens (**-**), underscores (**_**), parentheses (**()**), and periods (**.**). You can reuse resource group named **BUS5001-Week-5**, created in part 1.
Static Web App Name	Yes	Your logic app name, which must be unique across regions and can contain **only letters, hyphens, and numbers**. example: bus*5001-<your_student_ID>*.
Plan type	Yes	Free: For hobby or personal projects
Source	Yes	Github

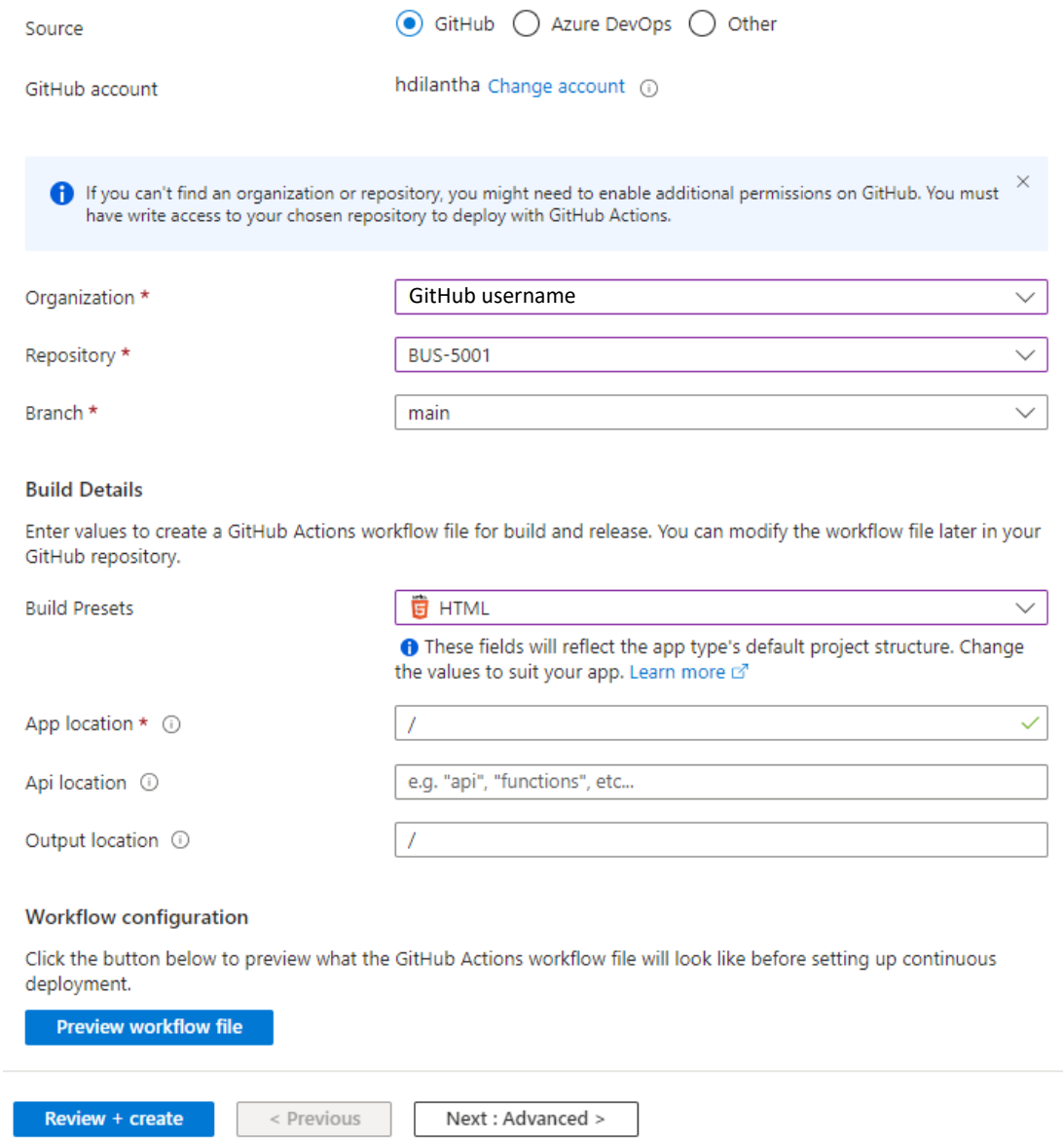
4. Select **Click here to login** to sign in with GitHub.



5. Select **Authorize AzureAppService** to continue.



6. Select the Organization (your GitHub username), Repository (Repository we created in the previous section) and Branch (main).
7. Select **HTML** as Build Presets.



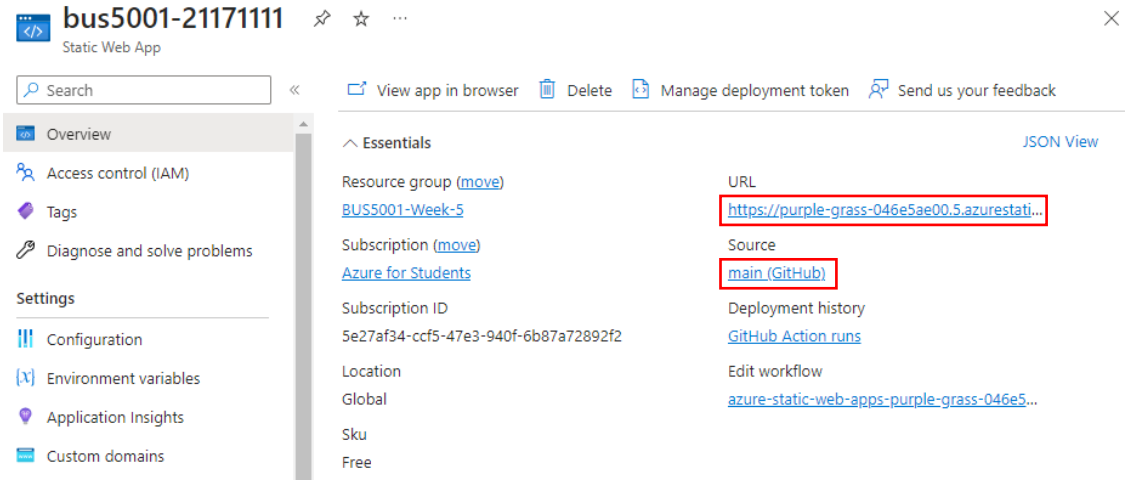
8. When you're ready, select **Review + Create**.
9. On the validation page that appears, confirm all the information that you provided, and select **Create**.

Troubleshooting Steps

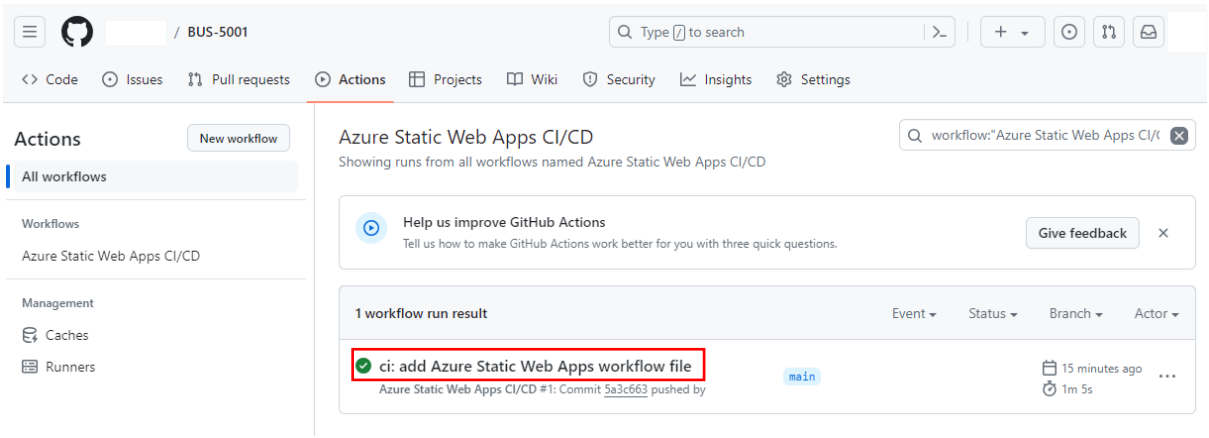
- If you don't see any repositories in step 7:
  - You may need to authorize Azure Static Web Apps in GitHub. Browse to your GitHub repository and go to Settings > Applications > Authorized OAuth Apps, select Azure Static Web Apps, and then select Grant.
  - You may need to authorize Azure Static Web Apps in your Azure DevOps organization.
- If time outs happen in step 5, try the steps again.

Review your Azure Static Web App

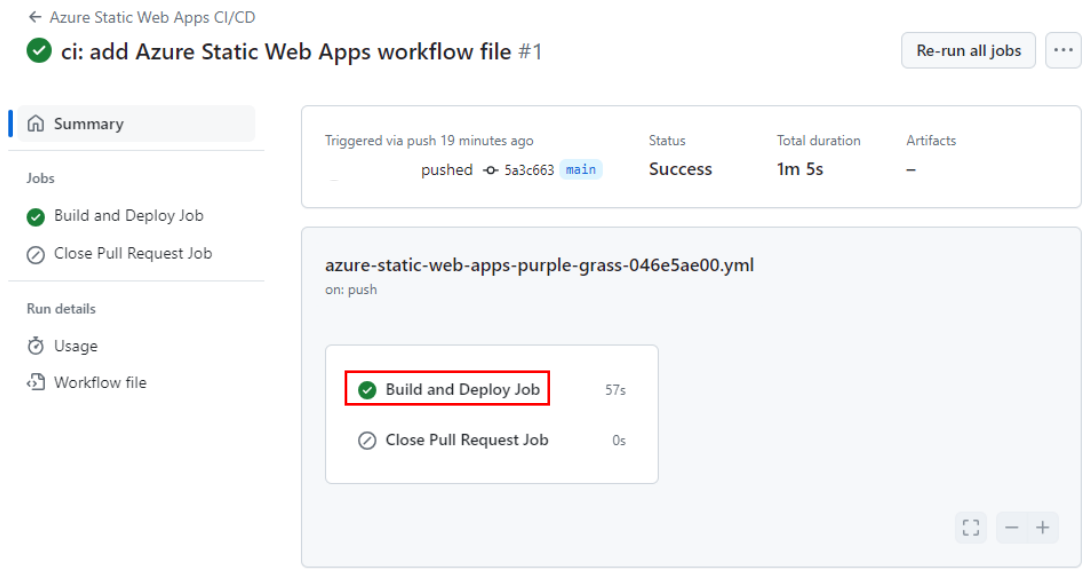
1. After the deployment is complete, navigate to the static web app that you created.
2. Select **URL** to view the static web app you hosted.



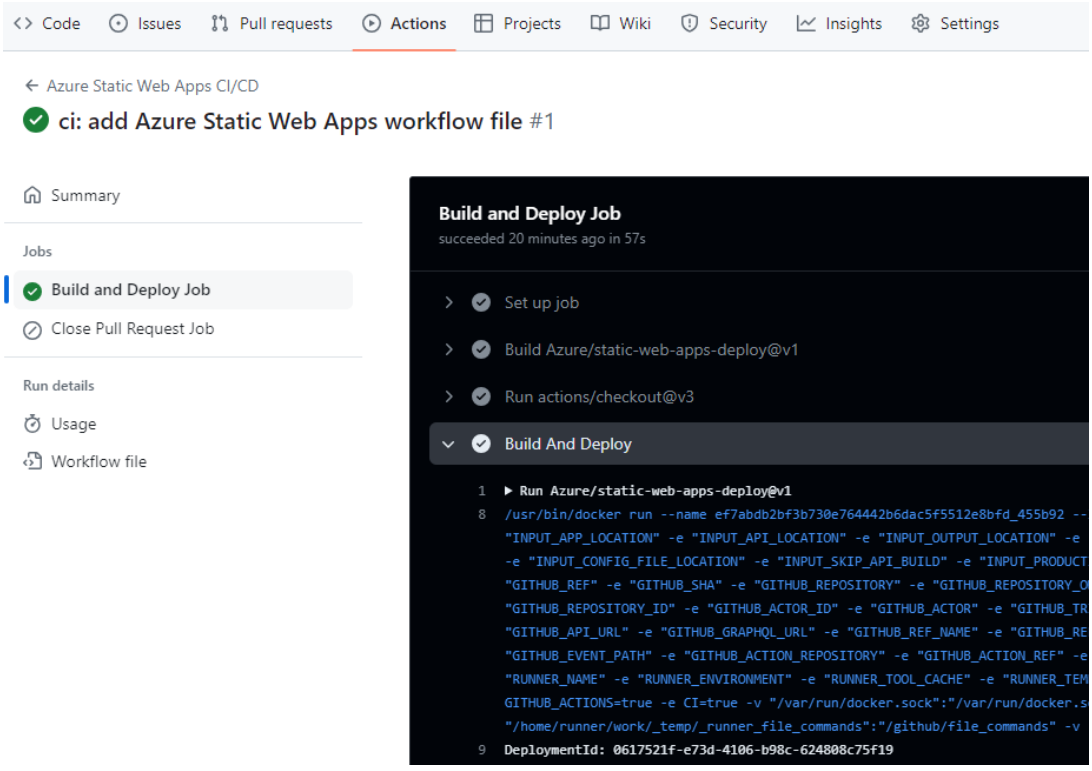
3. Click on **Deployment history** link to open your GitHub actions pipeline runs. This will take you into GitHub.
4. Select the workflow file.



5. You will see the commit details, and the Build and Deploy Job. Click on **Build and Deploy Job** to see the logs.

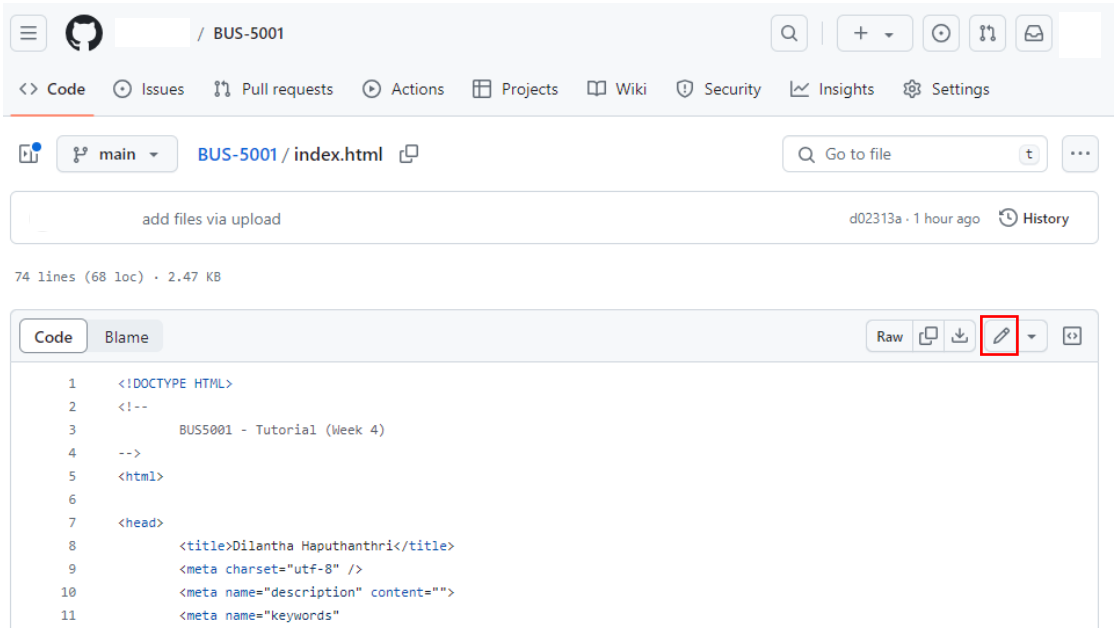


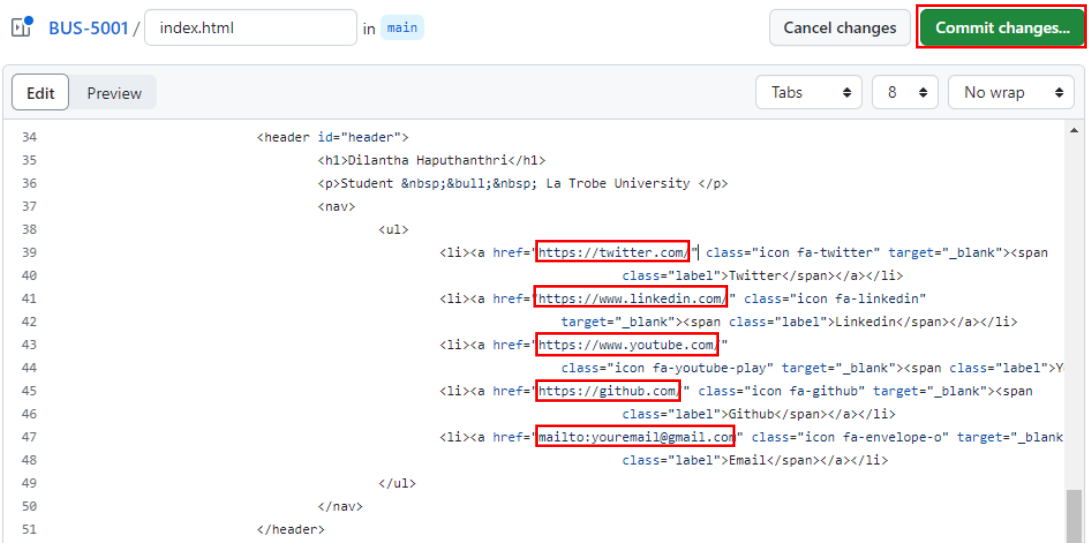
6. In the following screen, you can click on each Github action actions and see logs for each action.

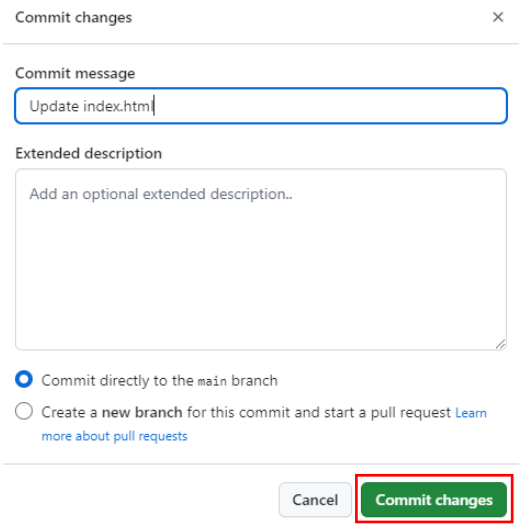


To understand how this works further,

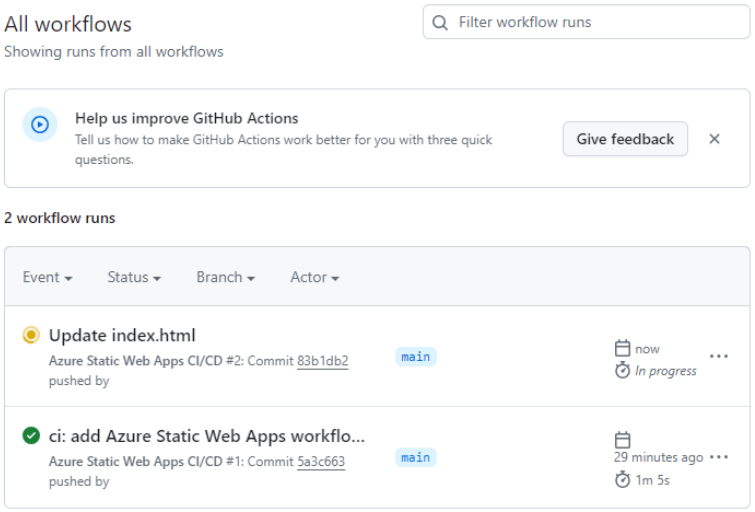
1. Navigate to the index.html file in **GitHub repository**, click edit icon, do some changes. (ex: replace social media links with your account links). Commit changes.







2. Go to **Actions** of your GitHub repository and review the workflow.



3. Upon completion of the workflow,

4. Go to the URL of the static website and check for the new changes.

IMPORTANT: CLEAN UP RESOURCES

Clean up all the Azure resources used after the completion of the workshop. Storage account, Resource group, etc.