Section 0: Prerequisites

GitHub account
Azure for Students account
Visual Studio Code + Azure Static Web Apps extension for Visual Studio Code
GitHub Desktop

By completing this Technical Onboarding Process, you will grow your skills with Azure, you will become trained on the use of GitHub, you will complete one of the requirements to reach the Beta milestone in the program – moving you closer to accessing more exclusive Student Ambassador benefits – and you will also get a certificate that will recognize your new technical abilities!

As you are navigating the technical onboarding, please reference the Technical Onboarding channel to ask questions and/or share issues you encounter. Make sure to review other Ambassadors' questions to help or get help!

Section 1: Create a repository

This article uses a GitHub template repository to make it easy for you to get started. The template features a starter app used to deploy using Azure Static Web Apps.

1. Navigate to the following location to create a new repository:
   a. https://github.com/microsoft/SATechnicalOnboarding/generate
2. Name your repository **SATechnicalOnboarding**

Note: Azure Static Web Apps requires at least one HTML file to create a web app. The repository you create in this step includes a single *index.html* file.

Select **Create repository from template**.

Section 2: Clone the repository

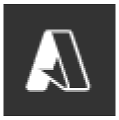
1. Open the GitHub Desktop App
2. Navigate to **File** > **Clone Repository**
3. Open the **URL** tab
4. Input https://github.com/<YOUR_GITHUB_ACCOUNT_NAME>/SATechnicalOnboarding.git into the URL input, and select **Clone**

Note: Make note of the location where the repository is saved.

Section 3: Create a static web app

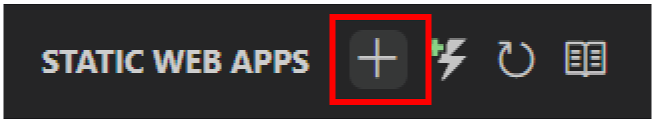
Next, open Visual Studio Code and go to File > Open Folder to open the cloned repository in the editor.

1. Inside Visual Studio Code, select the Azure logo in the Activity Bar to open the Azure extensions window.



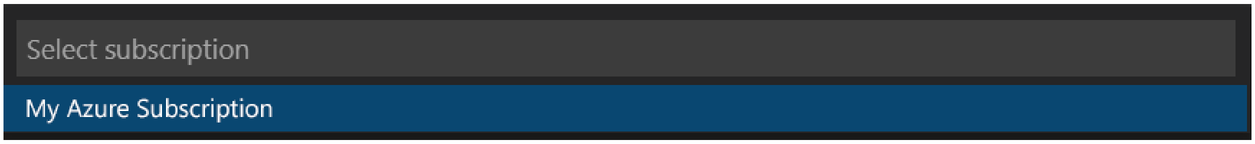
Note: You are required to sign in to Azure and GitHub in Visual Studio Code to continue. If you are not already authenticated, the extension will prompt you to sign in to both services during the creation process.

2. Under the *Static Web Apps* label, select the plus sign.



3. The command palette opens at the top of the editor and prompts you to select a subscription name.

   Select your subscription and press Enter.

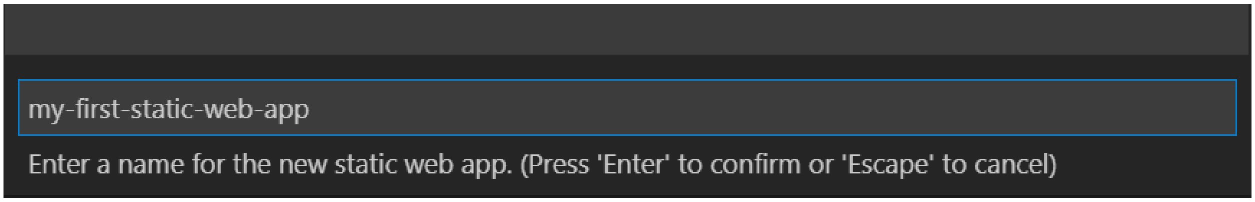


Select subscription

My Azure Subscription

4. Next, name your application.

   Type SATechnicalOnboarding and press Enter.



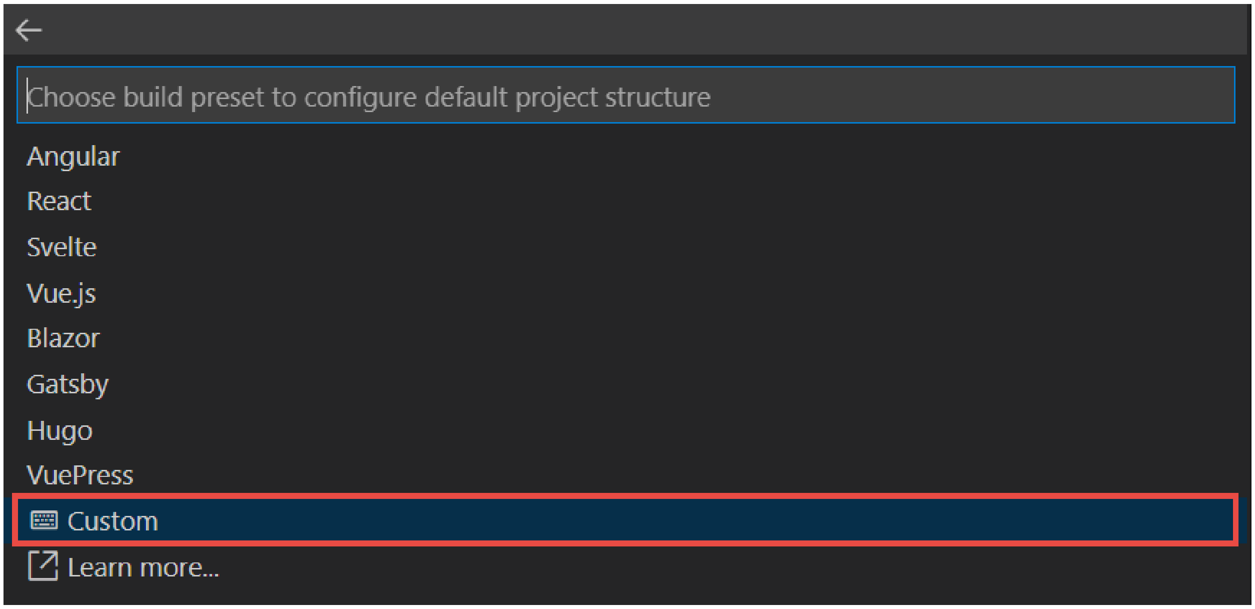
my-first-static-web-app

Enter a name for the new static web app. (Press 'Enter' to confirm or 'Escape' to cancel)

5. Select a region close to you.

Note: Azure Static Web Apps globally distributes your static assets. The region you select determines where your optional staging environments and API function app will be located.

6. Select Custom

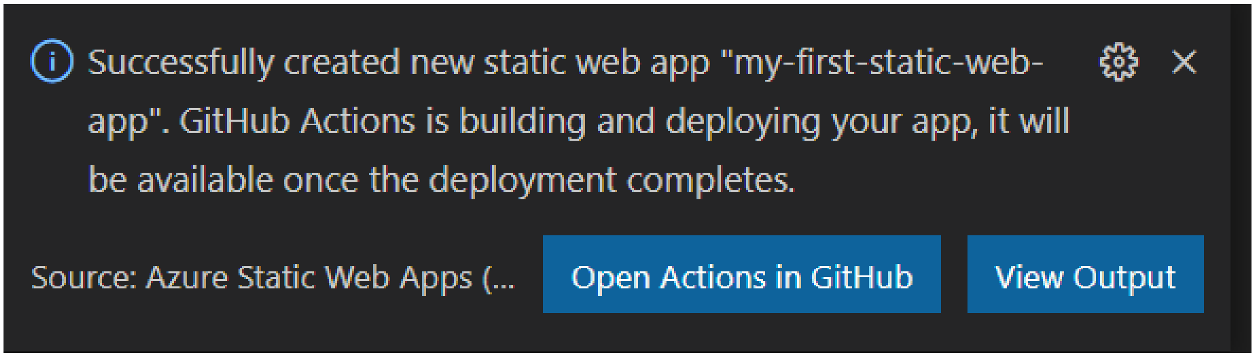


Enter /src as the location for the application files and press Enter.

This app does not produce a build output. Ensure the build output location is empty and press Enter.

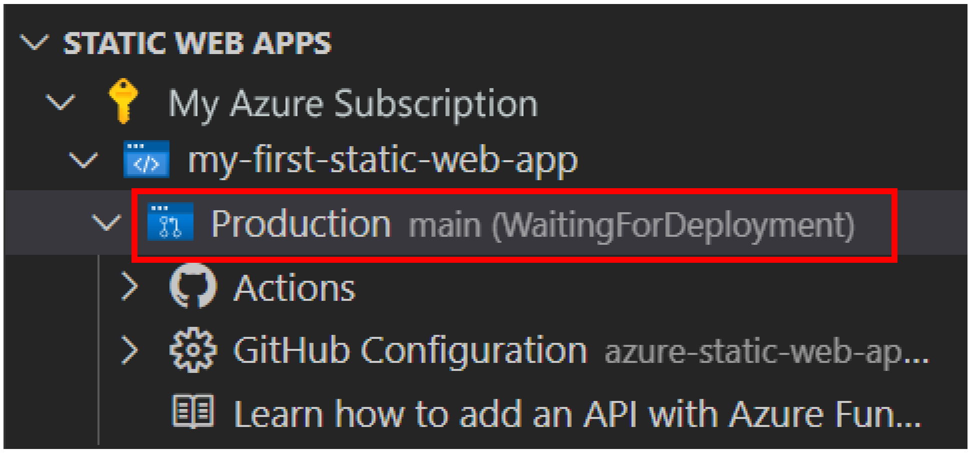
7. Select “Open Actions in GitHub.” This will launch the GitHub Actions tab in your browser. You will see the workflow run as it is creating the app. When your webapp is built and deployed, you will see a check for the workflow run.

   Once the app is created, navigate back to Visual Studio Code - a confirmation notification is shown there. The Visual Studio Code extension also reports the build status to you as the deployment is in progress – and will update you when your app is built and deployed.



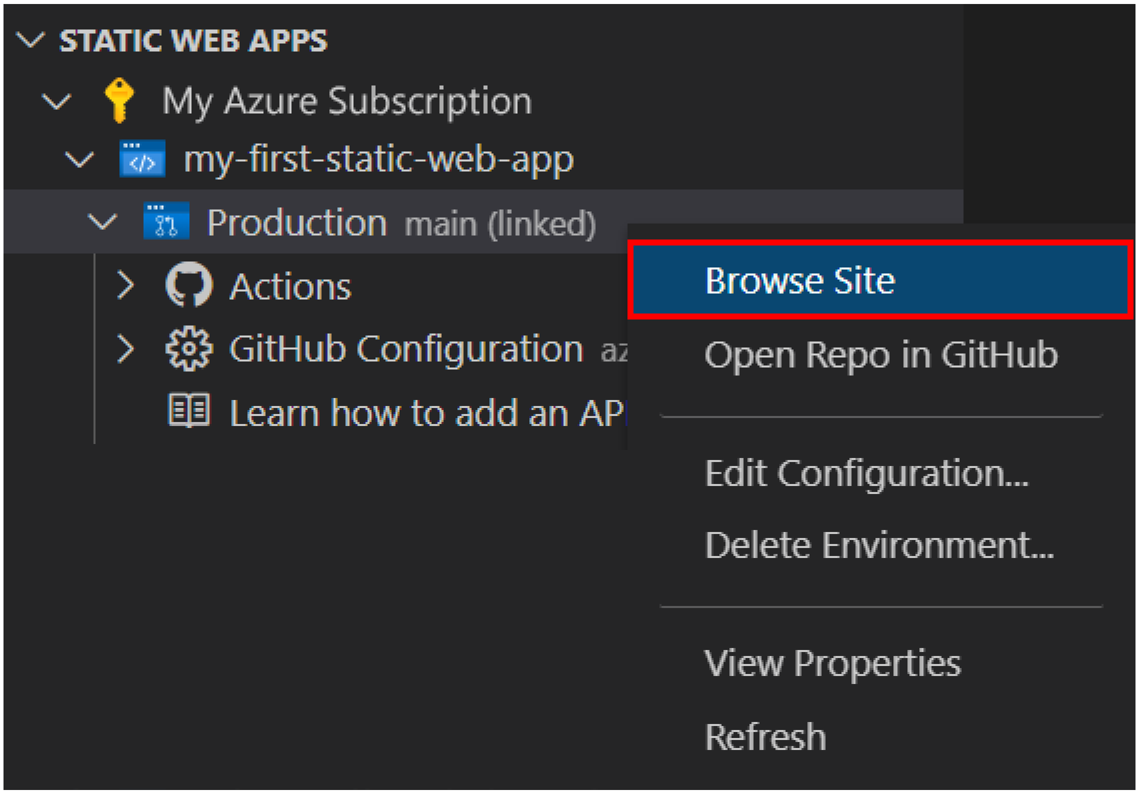
Successfully created new static web app "my-first-static-web-app". GitHub Actions is building and deploying your app, it will be available once the deployment completes.

Source: Azure Static Web Apps (...


STATIC WEB APPS
My Azure Subscription
my-first-static-web-app
Production main (WaitingForDeployment)
Actions
GitHub Configuration azure-static-web-ap...
Learn how to add an API with Azure Fun...

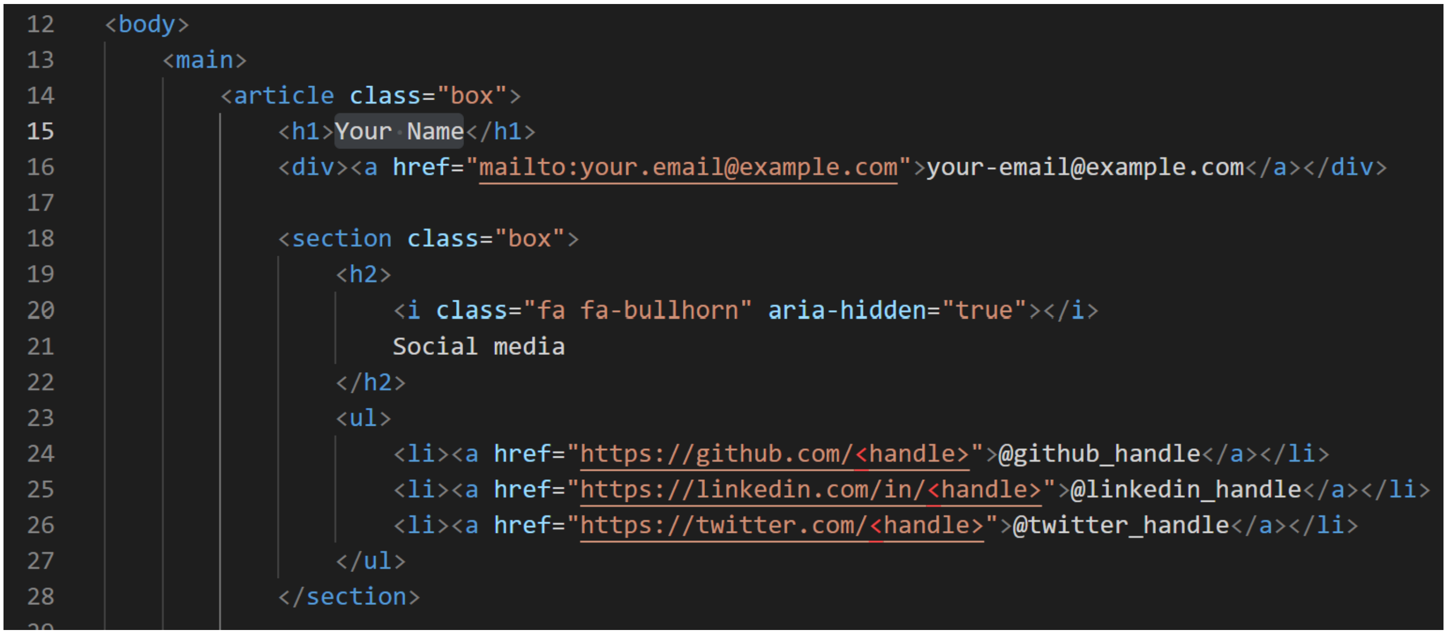
Once the deployment is complete, you can navigate directly to your website.

8. To view the website in the browser, right-click on the project in the Static Web Apps extension, and select Browse Site.



Section 4: Customize Your Static Web App

1. We will now do a small customization step customize your webapp you can edit it in VS Code and push changes using GitHub Desktop
   a. Navigate to the Explorer Tab in Visual Studio Code
   b. Navigate to the “src” folder, and within it, the “index.html” file
   c. Find the “Your Name” Heading. Replace “Your Name” with your real name.

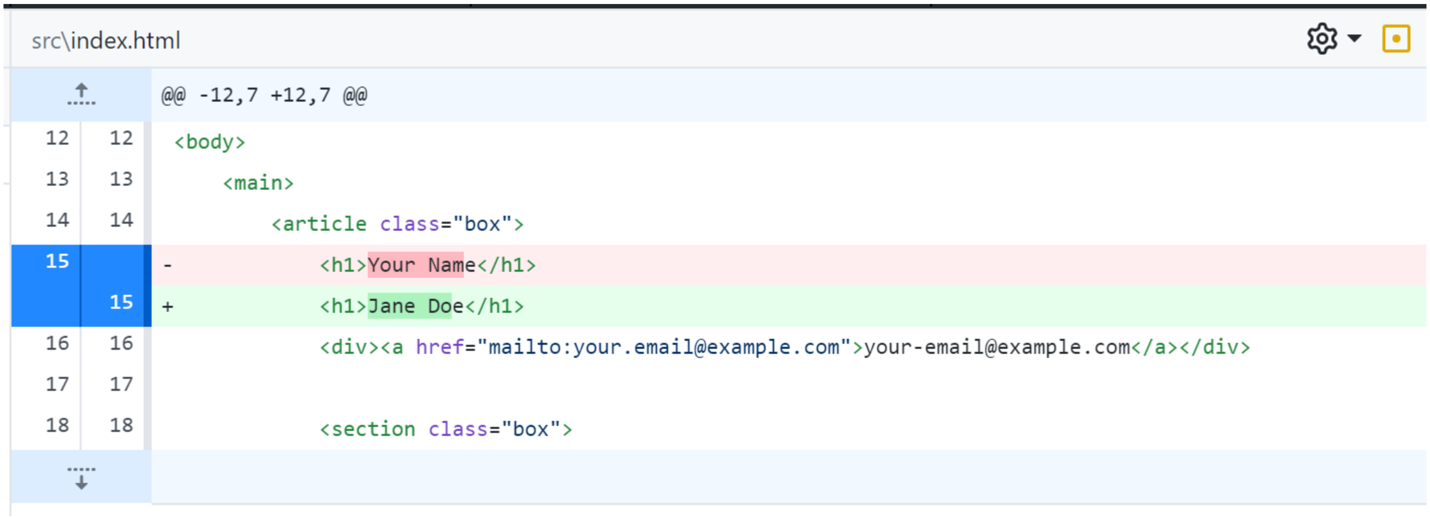


```
<body>
    <main>
        <article class="box">
            <h1>Your Name</h1>
            <div><a href="mailto:your.email@example.com">your-email@example.com</a></div>

            <section class="box">
                <h2>
                    <i class="fa fa-bullhorn" aria-hidden="true"></i>
                    Social media
                </h2>
                <ul>
                    <li><a href="https://github.com/<handle>">@github_handle</a></li>
                    <li><a href="https://linkedin.com/in/<handle>">@linkedin_handle</a></li>
                    <li><a href="https://twitter.com/<handle>">@twitter_handle</a></li>
                </ul>
            </section>
```

d. Save your changes in VS Code

e. Navigate to GitHub Desktop and ensure that your Current Repository is your Ambassador-SWA-Pilot repo (GitHub Desktop should reflect your recent changes like below)



src\index.html

```
@@ -12,7 +12,7 @@
  <body>
      <main>
          <article class="box">
-             <h1>Your Name</h1>
+             <h1>Jane Doe</h1>
              <div><a href="mailto:your.email@example.com">your-email@example.com</a></div>

              <section class="box">
```

f. In the bottom left of GitHub Desktop, give your changes a title (like “Added My Name” and add a description

g. Select “Commit to Main”

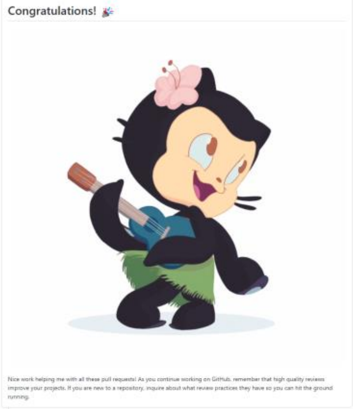
h. Now, push your changes to GitHub by selecting “Push Origin”

i. Navigate to Actions in GitHub and view your web app’s build progress

j. When the build is complete, refresh your app and the changes should be reflected

Section 5: Complete 200 Level GitHub Training on Reviewing Pull Requests

1. Navigate to the Reviewing Pull Requests Learning Lab and sign into your GitHub account to start the 200 Level GitHub training.
2. Select 'Start free course' and complete all course steps of the lab.
3. Navigate back to the Reviewing Pull Requests Lab page to confirm you have completed all course steps of the Lab. Input https://github.com/<YOUR_GITHUB_ACCOUNT_NAME>/reviewing-a-pull-request/pull/4 into the search bar of your browser to view the "change title on README #4" pull request, scroll down, and confirm that you see the following comment from the github-learning-lab bot at the end of the conversation in the pull request.



Congratulations! 🎉

Nice work helping me with all these pull requests! As you continue working on GitHub, remember that high quality reviews improve your projects. If you are new to a repository, inquire about what review practices they have so you can hit the ground running.

Section 6: Submit your onboarding details, feedback, and get your certificate of completion

1. You've made it to the end! The last step is to fill out the Onboarding Completion Form (using your @studentambassadors account) to provide a few details about you, your webapp, the link to your final pull requests from the GitHub Learning Lab, and your experience with this process. You will be sent a certificate recognizing your completion of the technical onboarding and you will be advanced to the Beta milestone (once you have also hosted an event or actively participated in a social impact league project) after a Program Team member verifies that your web app and learning lab are complete.