**DEPLOY YOUR WEBSITE TO GITHUB PAGES**

**Create A GitHub Account**

There are many different ways to deploy a website to the public Internet. In this unit, we’ll use [GitHub](https://www.codecademy.com/resources/docs/general/github) Pages to deploy your website.

GitHub Pages is a service offered by GitHub. Specifically, GitHub Pages are public webpages that are hosted and published through GitHub.

Why GitHub Pages? In the last unit, we generated a site using Jekyll. GitHub Pages offers extensive integration and support for Jekyll. By using both, you’ll benefit from:

* Easy setup
* Troubleshooting your site
* Updating and maintaining your site

**Note:** Remember, it is possible to follow all of the steps outlined in this course with your *own* content — just make sure that your HTML is inside of a file called **index.html** (a GitHub Pages requirement).

**Instructions**

To succesfully deploy your site, you will need a GitHub account.

In your own browser:

1. Start a new tab
2. Navigate to <https://github.com/>
3. Create an account

If you already have GitHub account, continue to the next exercise.

After signing up, be sure to verify your e-mail address.

**Note:** The content to the right is a video. You can play the video if you’d like to view a demonstration of the instructions. You’ll come across more videos like this throughout the rest of the course.

**Initialize Your Repo**

Great! Now that you’ve created a repo with the proper naming convention, let’s upload your site to [GitHub](https://www.codecademy.com/resources/docs/general/github).

We’ll use Git to push (upload) the contents of your site’s directory to your new repo.

To do so, we’ll first initalize a Git repository in your site’s directory.

**Instructions**

Checkpoint 1 Passed

**1.**

In the terminal to the right, open a new tab.

Then, use the cd command to navigate to your site’s directory.

Hint

If you don’t see your site’s directory (personal-website) in the current directory that you made in the previous lesson, don’t worry! It may have disappeared if you reset the exercise at some point. You can recreate it with mkdir personal-website. Then, use the cd command to enter that directory and run the git init command to continue.

Checkpoint 2 Passed

**2.**

Now that you’re inside of your site’s directory, initialize a Git repository with the following command:

git init

**Add the Remote**

Next, Git needs to know what repo will store your site’s content.

In this case, the repo will be the one you created on [GitHub](https://www.codecademy.com/resources/docs/general/github) earlier.

To specify the repo using Git, we’ll have to *add* the *remote* and label it as the *origin*.

1. The *remote* is the [URL](https://www.codecademy.com/resources/docs/general/url) of the repo that will store your site’s contents.
2. The *origin* is an alias for the remote. You can think of an alias as an abbreviation or a substitute name. This means that instead of having to always type the lengthy remote URL over and over again, you can simply refer to it as origin later on.

In the terminal, you can add the remote with the following command:

git remote add origin https://github.com/your-user-name/your-user-name.github.io.git

In the example above, https://github.com/your-user-name/your-user-name.github.io.git is the remote URL that refers to the repository you created on GitHub earlier. Again, you would replace your-user-name with your actual GitHub username.

**Note:** Make sure that your remote’s URL is typed correctly. Otherwise, you risk a failed deployment.

Once the remote has been added, you can establish the primary branch of the project with the following command:

git branch -M main

By convention, we name the primary branch main.

**Instructions**

Checkpoint 1 Passed

**1.**

Make sure you’re in your **personal-website** directory. You can use pwd to find out which directory you’re in and cd to move into the **personal-website** directory if you weren’t already there.

In the terminal, add the remote that points to the repository you created earlier. Use the example above to help you.

**Important:** If you accidentally make a mistake when adding the remote URL, you can start over and remove the remote with the following command:

git remote rm origin

Hint

If you get an error that says you are not currently in a git repository, don’t worry! It may have disappeared if you reset the exercise at some point. You can recreate the repository with mkdir personal-website. Then, use the cd command to enter that directory and run the git init command to continue following along with the lesson.

Checkpoint 2 Passed

**2.**

Confirm that the remote was succesfully added, by typing the following:

git remote -v

This command lists all the Git remotes and their corresponding URLs.

Checkpoint 3 Passed

**3.**

In the same directory, run the command to create the primary branch of the repository, naming the branch main. You can use the example above to guide you.

**Commit Your Changes**

We’re almost there! Git also needs to know exactly which files should be pushed to your repo.

In this case, we want to push *all* of your site’s content to the repo. This means we will do the following two things (in order):

1. Add all of your site’s content to the Git staging area
2. Commit (save) your changes

**Instructions**

Checkpoint 1 Passed

**1.**

Add all of your site’s contents using the following Git command:

git add .

Checkpoint 2 Passed

**2.**

Save your changes using Git’s commit command and the following commit message:

git commit -m "Save my work"