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 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.

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

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.

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 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 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

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.

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.

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

1.

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

```
git add .
```

2.

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

```
git commit -m "Save my work"
```

Deploy Your Site

It's time to deploy your site!

Once again, we'll use Git to help deploy your site. This time, we'll use Git's `push` command and push the contents of your site up to your repo using the following command:

```
git push -u origin main
```

Instructions

1.

First, you'll need to obtain a Personal Access Token from GitHub. Follow GitHub's instructions on creating a personal access token (classic), ensuring that you check the box for `repo` scope. Do not close the page once the token is generated — you will not be able to view the token again once you leave this page.

In the terminal, push the contents of your site to your repo. You can use the example above to help you.

You will be prompted to enter your GitHub credentials before you can push (i.e. username and password).

When asked for a password, input your new personal access token. As you input your token, you'll notice that no text will be *visible* as you type — this is how all terminals work (as an added security measure). The terminal is still accepting your input, simply hit "Enter" on your keyboard after you are done typing/pasting your invisible password. If you typed it incorrectly, simply try again.

Wait until the terminal stops outputting information, then continue to the next exercise.

Review

Fantastic! You now have your site published on the public Internet. You can now navigate to your newly published website in your preferred browser.

The URL for your GitHub Pages site is: `your-user-name.github.io`, where `your-user-name` is your actual GitHub username.

Let's review what you accomplished in this unit:

1. Created a GitHub account
2. Created the required GitHub repo
3. Used Git to add, commit, and push your website to the repo
4. Succesfully used GitHub Pages to deploy your site to the Internet!

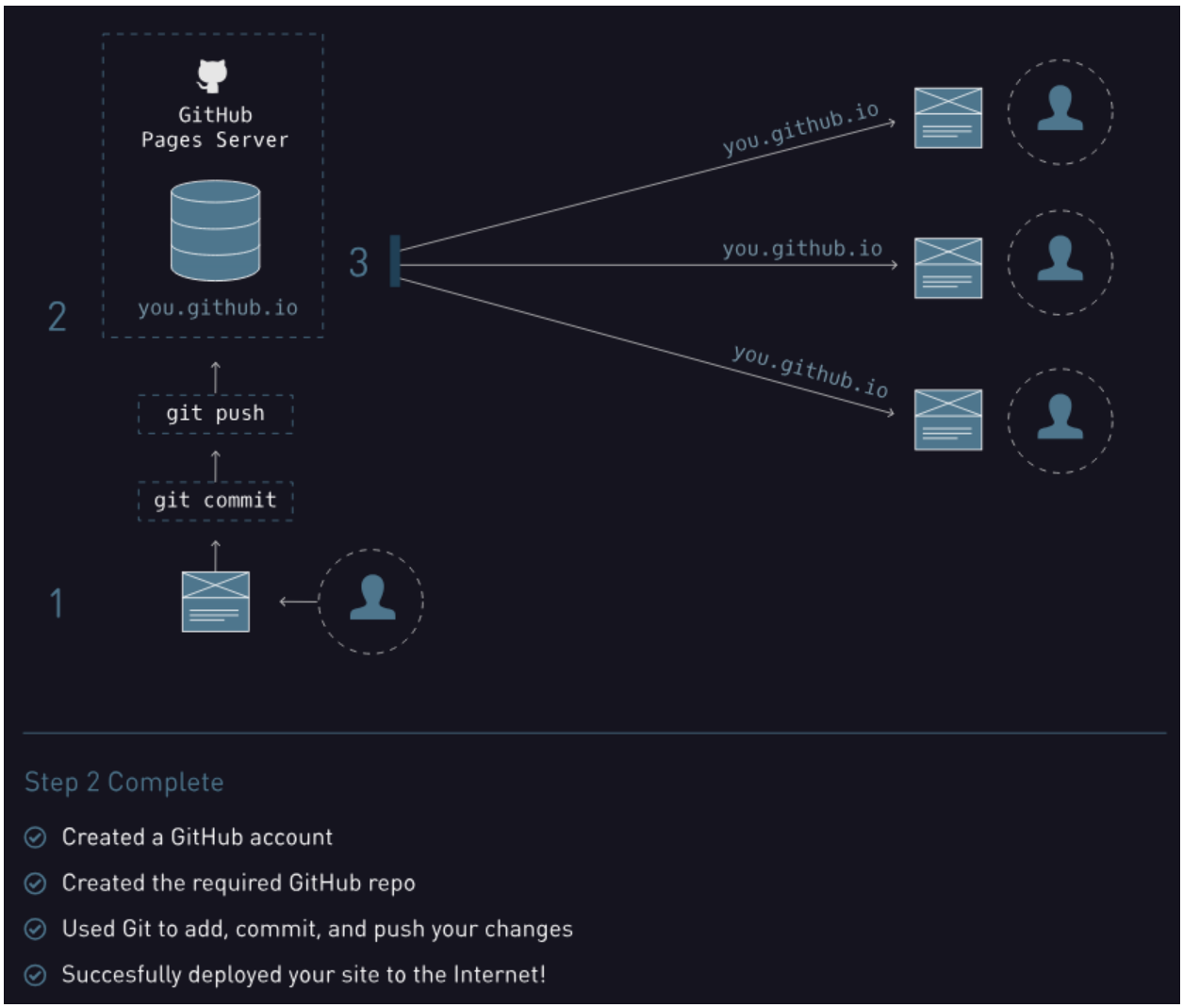
Note that your site's URL matches your repo's name — a GitHub Pages requirement. But what if you want to change the URL to a custom URL?

In the next unit, you'll learn how to keep your site hosted on GitHub, but change the URL to something besides the GitHub Pages default.

Instructions

Take a look at the diagram to the right. In this unit, you successfully accomplished the second step: deploying your site to the Internet using GitHub Pages.

Currently, the URL to your site is the GitHub Pages default URL. In the next unit, we'll move to the third step: assigning a custom URL to your site!



Step 2 Complete

- Created a GitHub account
- Created the required GitHub repo
- Used Git to add, commit, and push your changes
- Succesfully deployed your site to the Internet!