Lab Homework 10: Deploying a Dynamic Website

In this lab, you'll learn how to deploy your website to a live web server.

1. Learning Objectives

- Experience deploying a dynamic website.

2. Deadline

Lab Homework	Deadline	Slip Days	Credit	Solution
All Parts	Sun 5/3 at 11:59pm	Max: 2 days	20 points (completion)	Not Provided

3. Instructions

1. **Clone** your lab repository.

   Clone the following URL: `git@github.coecis.cornell.edu:info2300-2020sp/YOUR_GITHUB_USERNAME-lab10.git`
   Replace **YOUR_GITHUB_USERNAME** in the URL with **your Cornell GitHub username**.

2. **Work together.**

   Feel free to work with your peers to complete this lab. Use your section specific chat rooms. Organize a Zoom hangout to work on the assignment together. Take this as an opportunity for some *virtual* human contact!

   Note: You are encouraged to work together so long as you do your own work and you don't give away answers.

3. **Ask questions** or **say hi** during your registered section live Zoom Q & A.

   Your section leaders will hold a Zoom Q & A during your registered section time. Feel free to pop in and **say hi** or **ask a question**! Again, use this as another opportunity to keep up with your fellow Cornell community members!

4. **Submit.**

   When you're finished, follow the instructions in **submit.md** to submit your assignment.

Part I: Hosting Your Website

In order to share your web page with the world, you'll need to *deploy* it to a server that can *host* it on the Internet. Heroku is a *service provider* or *web host* that can host your web sites written in a variety of languages, including PHP.

1. Install and Configure Heroku

Objective: Install and configure Heroku's tools on your computer.

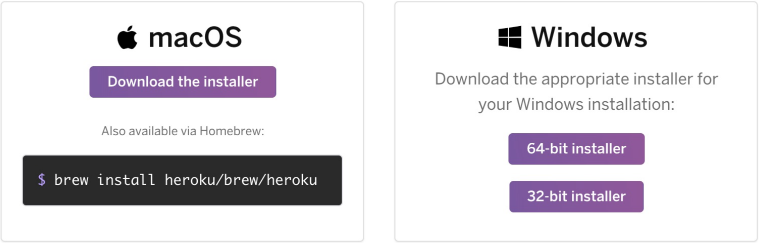
1. Sign up for a Free account at Heroku (https://signup.heroku.com/).

   Skip this step if you have already have a Heroku account.

   - Please choose PHP as your "Primary development language".
   - Verify your account using the email you registered with.
   - Set a password.

2. Download the installer for the Heroku Command Line (https://devcenter.heroku.com/articles/getting-started-with-php#set-up).

   1. If you are on a Mac, click "Download the installer" under macOS, and if you are on Windows click "64-bit installer" under Windows.

      Also available via Homebrew:

      ```
      $ brew install heroku/brew/heroku
      ```

      Download the appropriate installer for your Windows installation:

   2. Run the installer and **DO NOT** change the default settings. Just click next until it starts running.

3. Log into Heroku using the Heroku Command Line Interface (CLI) tools.

   1. Launch a terminal or command line prompt.
      - On Windows: open **Git Bash**. (**NOT the *Integrated Terminal* in VS Code and NOT "Command Prompt"**)
      - On Mac: Open a new *Integrated Terminal* in **VS Code** or open **Applications/Utilities/Terminal**.
   2. Enter this command: `heroku login`
   3. You will be prompted to write the email and password of the Heroku account you just created.
      - You have successfully logged in if you see `logged in as [YOUR HEROKU EMAIL]`.

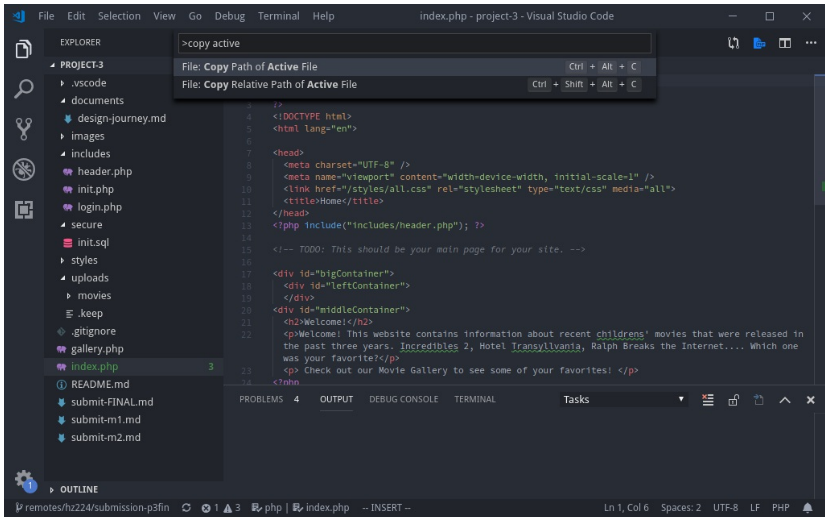
2. Connect Project Repository to Heroku

Objective: Configure Project 2 or Project 3 to be deployed to Heroku.

You'll notice this lab only has a few files. This is because for this lab you'll be deploying either your Project 2 or Project 3 to Heroku (you pick which project you want to deploy to a live web server).

Let's start off by going to your Project 2 or 3 in the command line prompt:

1. First open the project in VS Code.
2. Open **index.php** for your project.
3. From the **comand palette** select "File: Copy Path of Active File"



```
?>
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <link href="/styles/all.css" rel="stylesheet" type="text/css" media="all">
  <title>Home</title>
</head>
<?php include("includes/header.php"); ?>

<!-- TODO: This should be your main page for your site. -->

<div id="bigContainer">
  <div id="leftContainer">
  </div>
<div id="middleContainer">
  <h2>Welcome!</h2>
  <p>Welcome! This website contains information about recent childrens' movies that were released in
  the past three years. Incredibles 2, Hotel Transyllvania, Ralph Breaks the Internet.... Which one
  was your favorite?</p>
  <p> Check out our Movie Gallery to see some of your favorites! </p>
<?php
```

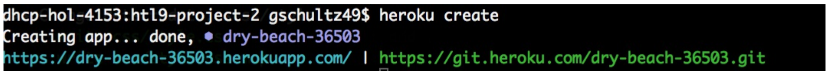
4. Then, navigate to the directory in the terminal.
   - On Mac: open a terminal and **c**hange **d**irectories to the folder by typing `cd` and pasting the full path on your clipboard (**remove index.php** from the path):

     ```
     cd PASTE_HERE
     ```

   - On PC: navigate to your project folder in the file explorer. Right click and select "Git Bash Here."
5. You should now be in your project's directory. Verify this by checking your path with the `pwd` command (**p**rint **w**orking **d**irectory).

6. Now, create your Heroku project with the command:

```
heroku create
```



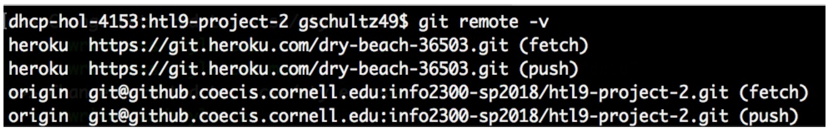
```
dhcp-hol-4153:htl9-project-2 gschultz49$ heroku create
Creating app... done, ⬢ dry-beach-36503
https://dry-beach-36503.herokuapp.com/ | https://git.heroku.com/dry-beach-36503.git
```

This will add the Heroku project as a Git *remote* named **heroku** to your local Git repository. The command generates a project and a Git remote with a random name, like **dry-beach-36503**. You will notice that this project/remote is now present in your heroku dashboard (https://dashboard.heroku.com/apps).

Note: You should only type this command **once** or else it will continue to generate random remotes. If you mess this step up and have multiple remotes, first remove the bad remote with: `git remote rm REMOTE_NAME` and then add the good remote with: `heroku git:remote -a NAME_OF_HEROKU_PROJECT`.

7. Test that you have configured the Heroku Git remote correctly:

```
git remote -v
```



```
dhcp-hol-4153:htl9-project-2 gschultz49$ git remote -v
heroku  https://git.heroku.com/dry-beach-36503.git (fetch)
heroku  https://git.heroku.com/dry-beach-36503.git (push)
origin  git@github.coecis.cornell.edu:info2300-sp2018/htl9-project-2.git (fetch)
origin  git@github.coecis.cornell.edu:info2300-sp2018/htl9-project-2.git (push)
```

From this command you should see two remotes: **heroku** and **origin**. If you see these 2 remotes, your repository is correctly hooked to your Heroku project.

3. Server Configuration Files

Your **lab** repository contains a few files that are necessary to get Heroku to properly run. You don't have to understand exactly what all of these do, but you will need to include them in any site you want hosted on Heroku.

- **resources**: A folder with files that are used to help configure the Apache web server that Heroku is using host your web site.
  - **.method**
  - **.htaccess**

- **.gitignore**: This file tells git to ignore certain files. Heroku requires that your .gitignore contains `/vendor/`.
- **composer.json**: This file declares your PHP library *dependencies* or the specific things that need to be installed on your Heroku server in order for your app to run correctly (much like a package.json in JavaScript or requirements.txt in python). In this case, we need PHP 7.1 and sqlite.
- **composer.lock**: Derived from the composer.json, this is like a snapshot of your project, showing what dependencies were installed at the most recent point in time. This file is automatically generated by the `composer` command. Note: You have not installed Composer.

Objective: Copy and paste the **resources** folder, the **composer.json** file, and the **composer.lock** file into your Project 2 or Project 3 git repository. All of these files are necessary for Heroku to setup your project.

4. Deploy to Heroku

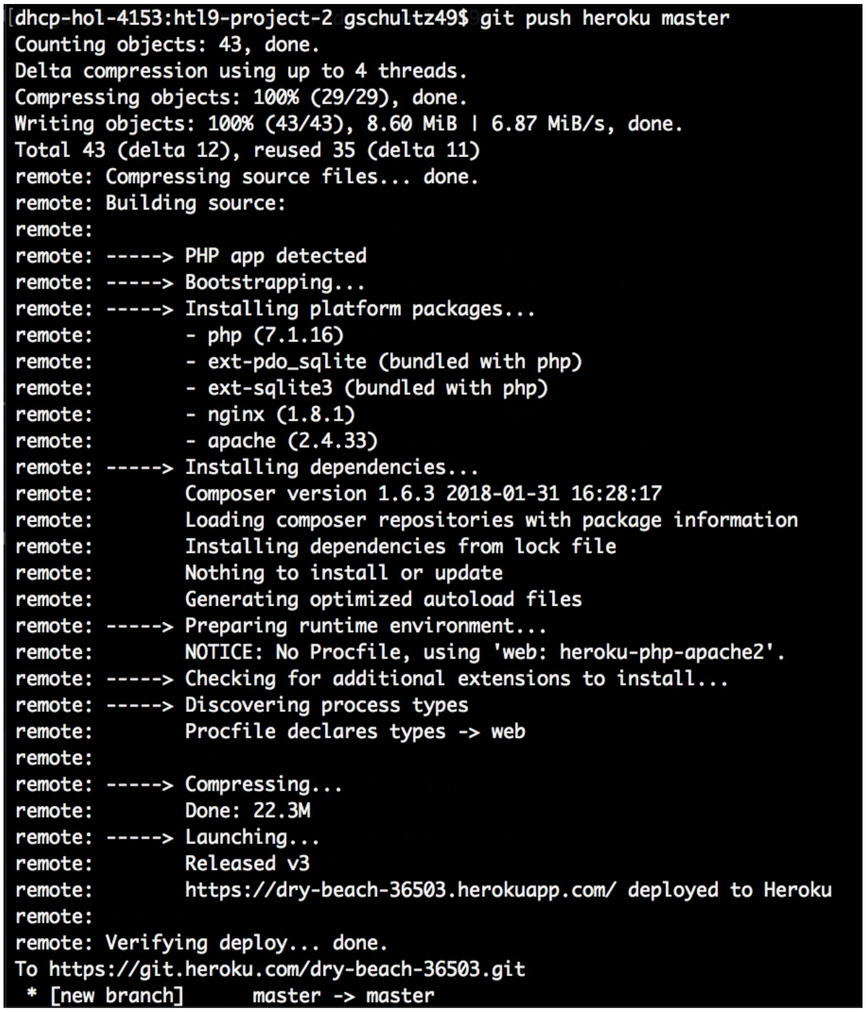
Now that your project is connected to Heroku via a Git remote and you have the necessary configuration files, let's push our work up to the Heroku server.

Objective: Before you can deploy your site to Heroku, you need to make sure you have **staged and committed the changes you want to deploy**. Otherwise, Git won't know about the changes and won't be able to send them to Heroku.

Objective: Once your website is ready and committed, you can push your work to the Heroku server using Git from the command line at our project:

```
git push heroku master
```

This pushes your local copy of the website (in "master") to Heroku's server ("heroku"), which you added in Step 2. When you run the command, you should see output that looks roughly like this:



```
dhcp-hol-4153:htl9-project-2 gschultz49$ git push heroku master
Counting objects: 43, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (29/29), done.
Writing objects: 100% (43/43), 8.60 MiB | 6.87 MiB/s, done.
Total 43 (delta 12), reused 35 (delta 11)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> PHP app detected
remote: -----> Bootstrapping...
remote: -----> Installing platform packages...
remote:        - php (7.1.16)
remote:        - ext-pdo_sqlite (bundled with php)
remote:        - ext-sqlite3 (bundled with php)
remote:        - nginx (1.8.1)
remote:        - apache (2.4.33)
remote: -----> Installing dependencies...
remote:        Composer version 1.6.3 2018-01-31 16:28:17
remote:        Loading composer repositories with package information
remote:        Installing dependencies from lock file
remote:        Nothing to install or update
remote:        Generating optimized autoload files
remote: -----> Preparing runtime environment...
remote:        NOTICE: No Procfile, using 'web: heroku-php-apache2'.
remote: -----> Checking for additional extensions to install...
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 22.3M
remote: -----> Launching...
remote:        Released v3
remote:        https://dry-beach-36503.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/dry-beach-36503.git
 * [new branch]      master -> master
```

The last line is the most unique, it is saying to take the work on your default branch (master) and to push it onto the heroku remote that we created above. This maintains a development/production relationship.

The server will start to install all dependencies and deploy your code. Once it's finished, you can go to your live url by typing in the command: `heroku open`. This will open your default browser to your Heroku URL and you should see your project!

5. Update *deploy.md*

Objective: Add the URL of your deployed Heroku website (e.g. http://dry-beach-36503.herokuapp.com) to **deploy.md** in your repo.

IMPORTANT! Failure to properly deploy your Project 2 or 3 site to Heroku and provide a valid URL in **deploy.md** will result in a 0 for this assignment.

Part II: Notes about Deploying a Site

There are a couple of gotchas that you need to be aware of when deploying your site.

1. A Note About SQLite & File Uploads on Heroku

SQLite is not a typical database system that should be used in a production environment (like Heroku). Being that the entire database operates through a single file, it is highly volatile and is particularly discouraged on Heroku. Heroku discourages this through what is known an an ephemeral filesystem. Basically, it refreshes all files (include file uploads) every 24 hours, which means that any database changes (or uploaded files) will be reset or deleted every 24 hours.

For this reason, it is recommended that you use a more robust database system for any production systems like PostgreSQL or MySQL. In particular, Heroku provides free startup plans for PostgreSQL databases: https://elements.heroku.com/addons/heroku-postgresql. However, we will not be covering how to do this in this class. If you're interested in learning more, check out the resources we've linked to here.

In order to use PostgreSQL you will also need to port your SQL queries from SQLite to PostgreSQL. This is out of scope for this class, but if you're interested, take a look at PGLoader which can port *SQLite* SQL to *PostgreSQL* SQL semi-automatically: https://pgloader.io/.

Your file uploads will also be deleted every 24 hours. To support file uploads you need to use storage service, like Amazon's Simple Storage Service (S3): https://devcenter.heroku.com/articles/s3-upload-php.

Note: The teaching staff is not qualified to help you with this. We also do not have the resources to help you with this and also grade your Project 3. If you decide to try and configure a custom domain with SSL on Heroku, you are on your own.

2. A Note About Domains and HTTPS

Heroku's free plans host your web page as a *subdomain* of the **herokuapp.com** domain. For example: **https://dry-beach-36503.herokuapp.com/**. If you wish to have your own domain name that's not *herokuapp.com*, like *mycoolsite.com*, you will need to register a domain name and configure it with Heroku. More information can be found in Heroku's Dev Center: https://devcenter.heroku.com/articles/custom-domains.

If you would like to secure your website using SSL so that you can serve your web site using **HTTPS** instead of HTTP, see the Dev Center: https://devcenter.heroku.com/articles/ssl. Also visit https://letsencrypt.org/ to learn more about SSL Certificates and Keys.

Note: The teaching staff is not qualified to help you with this. We also do not have the resources to help you with this and also grade your Project 3. If you decide to try and configure a custom domain with SSL on Heroku, you are on your own.

Submit

Follow the instructions at the beginning of this document to submit your assignment.

1. Contributors

The following individuals made contributions to this assignment:

- Greg Schultz
- Sharon Jeong
- Grant Storey
- Kyle Harms