# asgn08: MySQL on your Web Host

## Objectives

- Create a database on your web host
- Write additional queries
- Create folder named `database` to store database files
- Create a separate login for your host to connect to the database.
- Continue using git
- Create a GitHub repo for this assignment.

## Assignment

Watch the screencasts. I'm going to set things up a little differently than the screencast (described below). Both methods will work.

Read the section in chapter 14 on using an `includes/config.php` file as well as viewing the screencasts. So we can scale up our project if needed, I have created a new folder named `database` where I store all of my included files that are database specific. If the project were to get a lot larger, I would store different, non-database files in the `includes` folder that is described in the text. There are a lot of options here. You may choose another naming convention if you desire.

To sum it up, you can store your files in the `asgn08/includes` folder or name it `asgn08/database`.

### Setting up your files

- Create a new folder `web182/asgn08`.
- Copy the files from chapter 14 in your text (found in Moodle) to this folder.
- Create a new folder called `asgn08/database`.
- Store your database configuration information in a file called `connection.php`. It should look like

```
<?php
$server = "localhost";
$user = "wbip";
$pw = "wbip123";
$db = "test";
```

## Git

Now that you have some files set up, it is time to put version control in place.

With your terminal open

Navigate to your `asgn08` folder. In my case I would use

```
cd /Applications/MAMP/htdocs/web182/asgn08
```

Yours may be different. Use the Moodle forum or contact your instructor if you are having difficulty navigating the terminal.

Use the following commands to set up git. The commands below perform the following functions

- Initialize git for the working folder
- Add all untracked files to the staging area
- Commit the tracked files and provide a short message describing what you did.

```
git init
git add .
git commit -m "Init commit"
```

## GitHub

Now lets put our files up on GitHub.

- Go to your GitHub page.
- Click on the green **New** button to create a new repo.
- Name the repo `asgn08`
    - Side note: I'm working on a better way to organize files in GitHub but I'm not there yet, so this method of putting the course name first will have to do this term.

- For the description use

  ```
  School work for WEB 182 at A-B Tech CC using PHP and MySQL
  ```

- Click on the **Create Repository** button

Since you have already used git locally you can scroll down to section called **"...or push an existing repository from the command line"**. Copy the two lines into your terminal. In my case it looks like

```
git remote add origin https://github.com/charliekwallin/asgn08.git
git push -u origin master
```

You should see something like

```
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 226 bytes | 226.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/charliekwallin/asgn08.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Click on the link in you GitHub page to make sure the page contains your files.

## Storing private data

In the section of the chapter titled *Storing MySQL Connection Data in an Include File*, Mr. O'Kane shows you how to make your database a little more secure. It will also stop you from writing the same code more than one time. Let's take it one step further by

- Creating a `web182/asgn08/database` folder.
- Create a file inside that folder named `connection.php`. **NOTE:** This is a little different than the screencast.

Use the following login information from the book.

```
<?php

$server = "localhost";
$user = "wbip";
$pw = "wbip123";
$db = "test";
```

**Note:** if your code does not have any HTML, you do not need the closing php tag! This is considered a best practice.

## Using php constants instead

Since the database connection information is not going to change, a better choice is to use a PHP Constant. Constants are similar to variables except that you **cannot** change their value, and they syntax is different. Here is the same code from above but in this case we are using constants. You can see how this works in the third screencast 3. Modify your `connection.php` code so it works locally and on your web host.

```
<?php

define('SERVER' = 'localhost');
define('USER' = 'wbip');
define('PW' = 'wbip123');
define('DB' = 'test');
```

Referencing the `database/connection.php` file

In your php files that store the queries, you will need to reference the `connection.php` file. Here is an example

```
<?php

include_once('../database/connection.php');
```

Becuase you changed to a PHP constant, you will also need to change how you reference the constant.

Here is an example using `mysqli`

```
$connect=mysqli_connect(SERVER, USER, PW, DB);
```

Note: there are **no** dollar signs and the convention is to use all uppercase letters.

### Writing one file to connect to either database

In the third screencast, I show how you can write a file that uses PHP global server variables to detect which server your file is on (local or webhost). Someone commented on it and I like his comment so I'm including it here.

> However, you could have checked if the server variables has .com . net etc using strpos() in if statement. This way you could avoid mentioning domain name specifically. This will work for any server (with same db details).

## .gitignore

Now that you have created a file with sensitive information, it is time to hide it from the public on GitHub. Technically, you don't hide it, instead the file doesn't get pushed to your account.

The way to do this is with a file named `.gitignore`. This is a text file that contains a list of files you do not want pushed to your online repo.

Add the line `database/connection.php` to the `.gitignore` file.

While you are in your `asgn08` folder, user your text editor to create a new file named `.gitignore`.

### push your new file

Test to see if your `.gitignore` file worked correctly by pushing your code to your repo.

First check to see the state of your files

```
git status
```

`status` will show you what has changed and you can copy it for your commit.

Next

```
git add .
git commit -m "Added database/connection.php file"
git push
```

Check your work, you should see the `.gitignore` file. Click on the link and see that it just shows the name of the file you omitted.

Ask your instructor if you run into trouble.

### Writing Additional Queries

- Create a folder named `asgn08` . Put all of your files for this assignment in the folder. These are the same files from chapter 14 in the text
- Complete the coding exercises 6 – 10.
- Use your `database/connection.php` file to connect to your database. If the file path is giving you trouble, try removing the `connection.php` file from the `database` folder and put it in the same folder as your queries to see if you can get it working. Once working, you can move it back in the `database` folder. Remember that you will need to change the file path to reference it.
- Modify the `connection.php` file as described in the screencast so you can use it locally or on your web host.
- Create an `index.php` file for all of the files in your `asgn08` folder and update your `web182/index.php` file.
- Upload your `asgn08` folder to your web host.
- Test your work on your web host

## Submit

- Push your files to your GitHub account (this is where I will read your code).
- All four SQL coding exercises should be uploaded to your server so I can run them on your web host.
- Make sure you update your `web182/index.html` file so `asng08` is displayed.
- Create a file `asgn08/index.html` so I can see a list of the files on your web host.