

Working Like A Developer

Bill Patrianakos / CTO & Software Engineer, Aplo LLC

Agenda

- Intros (we may have already done this by the time you see this slide)
- What is web development?
- Bash Commands
- Using GitHub

Introductions

What to expect

During this course we will learn (at a high level):

- Command Line / Terminal
- Git and GitHub
- Ruby programming language
- Rails web application framework
- (Basic) Database Modeling

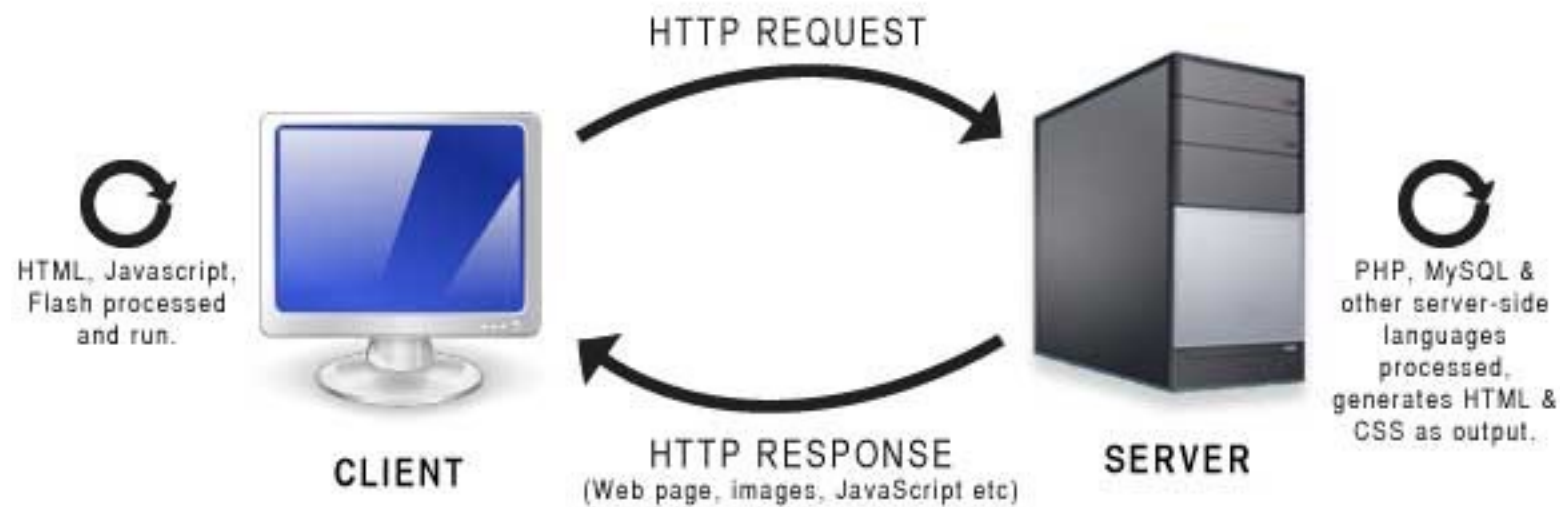
To pass the course you will need to:

- Complete at least 80% of all assigned homework
- Complete a final project

What you will get from us

- In class labs
- Homework
 - I will provide feedback on homework once per week. 99% of the time I'll send feedback every Sunday night so you can expect to hear back by Monday morning.
 - Any late homework will typically be reviewed along with the current week's homework
- Slides / student handouts available in our shared GitHub repo
- Me. I'm here to help.

The Web Application Stack



Back-end vs. Front-end Development

Let's define a few terms:

- Web Development -> apps built for the web
- Front-End Development -> client / browser code (HTML, CSS, JS)
- Back-End Development -> server-side code (Ruby, C#.NET)

Integrate into the developer community

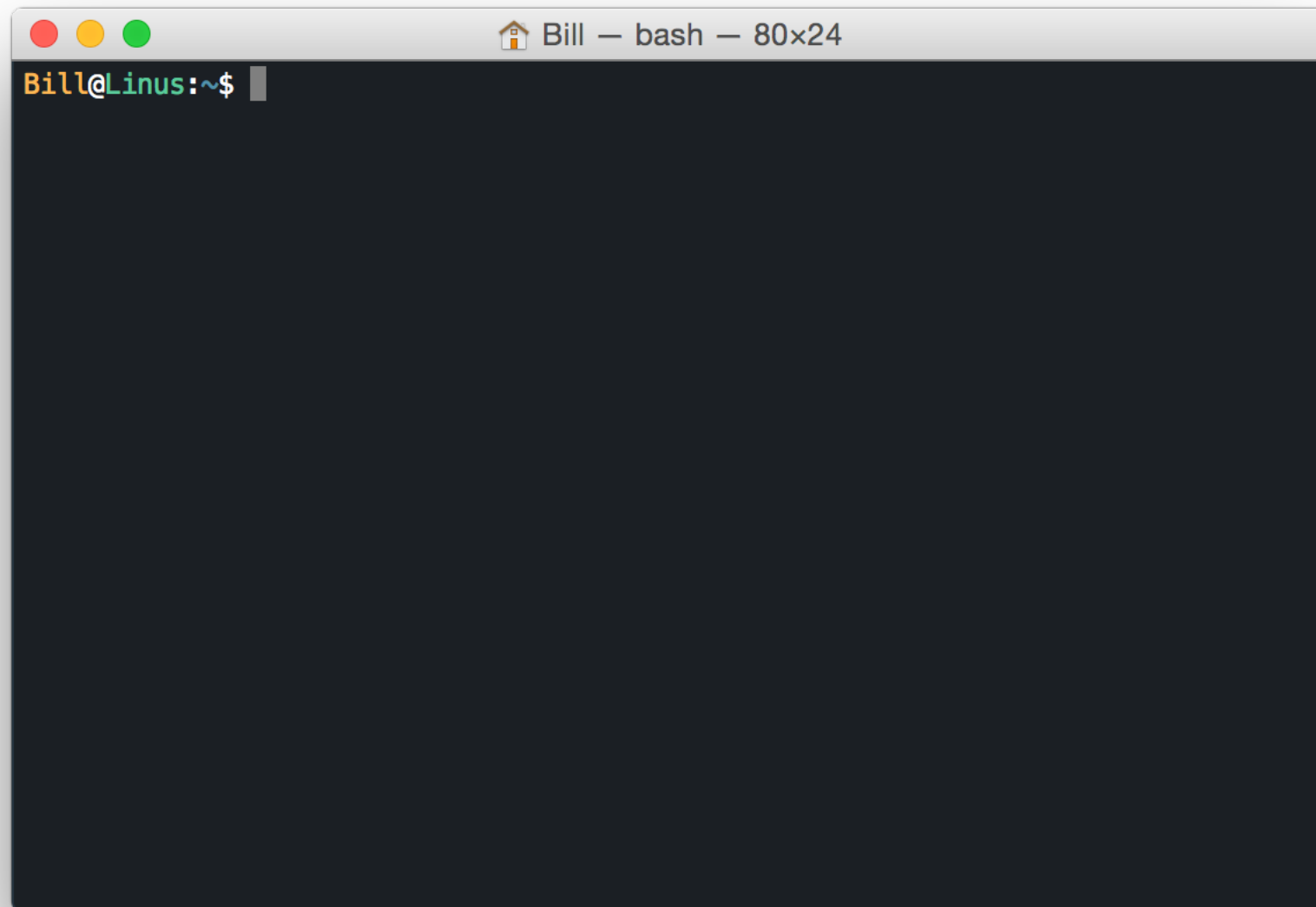
- Choose the right OS, editors, & tools for your projects.
- Leverage the online community's vast libraries and documentation.
- Spread the knowledge you gain, and give back to the community when you can.
- Take pride & joy in what you work on.

Be efficient:

- Use the keyboard as much as possible
- If you find yourself doing the same thing repeatedly, automate it



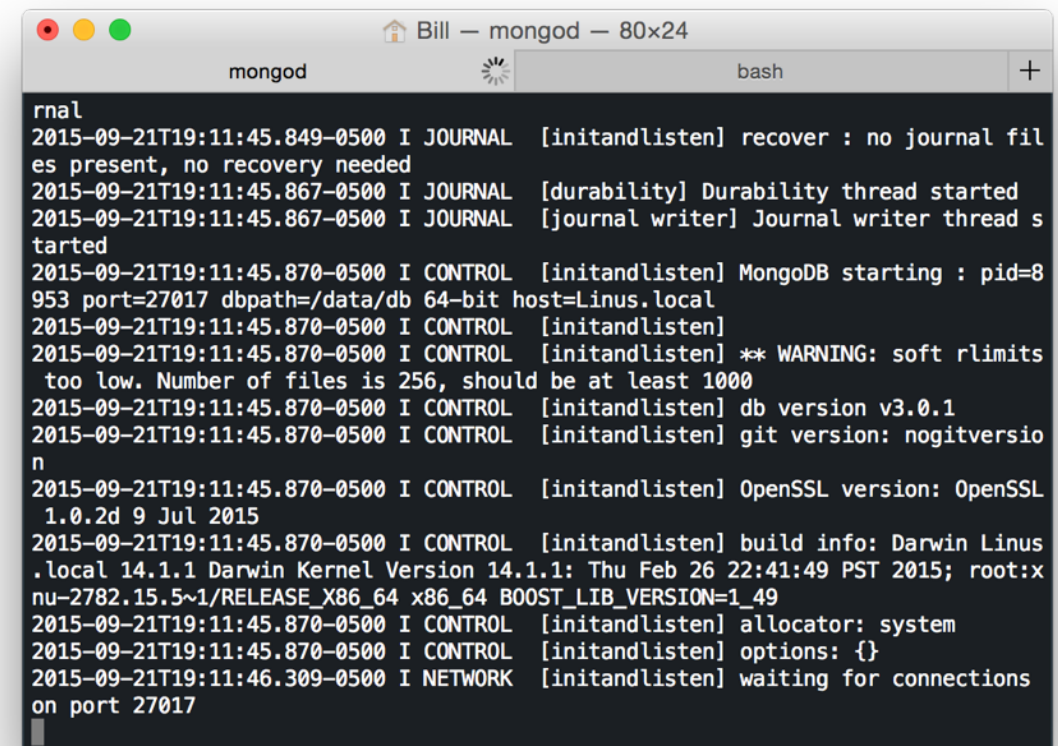
The Command Line



What is it?

The command line is a **terminal** giving you direct access to your operating system. You can enter simple commands to perform a variety of functions.

Many of the tasks we need to carry out (such as committing our code) are best performed in the command line.

A screenshot of a macOS terminal window titled "Bill - mongod - 80x24". The window has a dark background and shows the output of the MongoDB daemon (mongod) starting. The logs include timestamps, log levels (I for info), and messages from various components like [initandlisten], [durability], [journal writer], and [network]. The messages indicate that the database is starting, no journal files are present, the durability thread and journal writer thread have started, and the MongoDB version is v3.0.1. A warning is also shown about the number of files being too low. The terminal window has standard macOS window controls (red, yellow, green buttons) and a title bar.

```
rnal
2015-09-21T19:11:45.849-0500 I JOURNAL [initandlisten] recover : no journal fil
es present, no recovery needed
2015-09-21T19:11:45.867-0500 I JOURNAL [durability] Durability thread started
2015-09-21T19:11:45.867-0500 I JOURNAL [journal writer] Journal writer thread s
tarted
2015-09-21T19:11:45.870-0500 I CONTROL [initandlisten] MongoDB starting : pid=8
953 port=27017 dbpath=/data/db 64-bit host=Linux.local
2015-09-21T19:11:45.870-0500 I CONTROL [initandlisten]
2015-09-21T19:11:45.870-0500 I CONTROL [initandlisten] ** WARNING: soft rlimits
too low. Number of files is 256, should be at least 1000
2015-09-21T19:11:45.870-0500 I CONTROL [initandlisten] db version v3.0.1
2015-09-21T19:11:45.870-0500 I CONTROL [initandlisten] git version: nogitversio
n
2015-09-21T19:11:45.870-0500 I CONTROL [initandlisten] OpenSSL version: OpenSSL
1.0.2d 9 Jul 2015
2015-09-21T19:11:45.870-0500 I CONTROL [initandlisten] build info: Darwin Linux
.local 14.1.1 Darwin Kernel Version 14.1.1: Thu Feb 26 22:41:49 PST 2015; root:x
nu-2782.15.5~1/RELEASE_X86_64 x86_64 BOOST_LIB_VERSION=1_49
2015-09-21T19:11:45.870-0500 I CONTROL [initandlisten] allocator: system
2015-09-21T19:11:45.870-0500 I CONTROL [initandlisten] options: {}
2015-09-21T19:11:46.309-0500 I NETWORK [initandlisten] waiting for connections
on port 27017
```

Why are we using it?

- In order to learn Ruby without Rails, we must learn how to run Ruby programs on their own.
- To do so, we can simply create "stand-alone" Ruby applications.
- A stand-alone Ruby app consists of one or more Ruby files (files that have a .rb extension)
- Once you have written a Ruby file, you can run the file by typing: `ruby file.rb` (this would run a Ruby file named file.rb)
- This is the basis of how we will be writing and testing our Ruby applications in the initial portion of this course

Terminal Basics

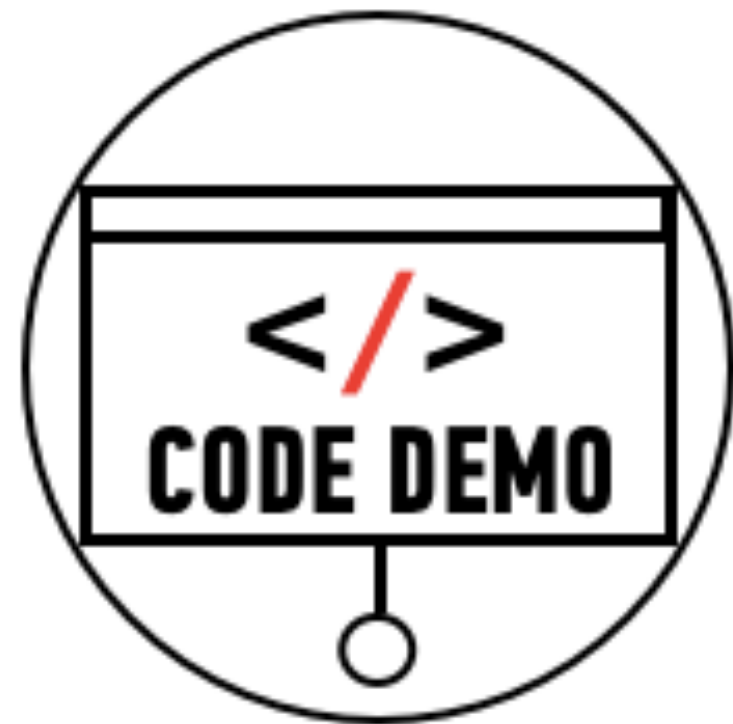
- `cd` – Change directory
 - `~` is a special alias that references your “home” folder
 - Navigate relative to `cwd` (current working directory) with variants of `./`, `../`, `../..`, etc.
- `touch` – Creates a new, empty file
- `mkdir` – Make directory
- `ls` – List files and folders in a directory. The `-l` and `-a` flags modify how the output is formatted. The flags that `ls` accepts can be combined like this `ls -la` (shows all files along with metadata including hidden files and folders)

Let's do it together now.

Your (nearly) first terminal command(s)

Task Instructions

1. Create a directory named BEWD_Class.
2. Change into this directory.
3. Create a file named "test.rb" using the touch command.
4. Demonstrate how to open "test.rb" in Sublime Text from the terminal.
5. (Now we set up Sublime)
6. Using ruby comments type the following:
`#This is how you should could take notes in class.`
7. Remove directory



Git & GitHub

They're completely different things! Remember that!

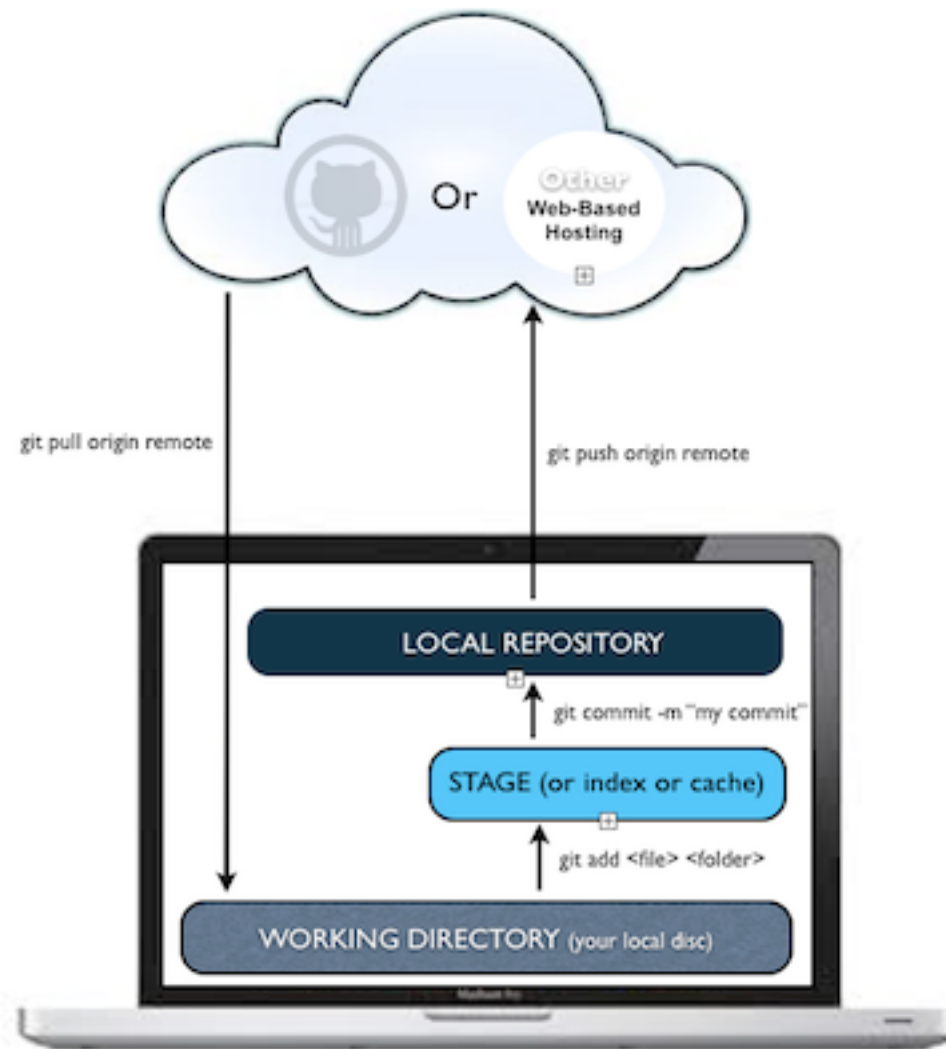
What is Git

- Git is a source control management tool.
- Git allows you to store and update your code in a structured way.
- Git includes history of changes you make, so you can create "checkpoints" and track your work better over time.
- Git is an intelligent tool, and does many things for you automatically, but can be tricky to use in some cases. It takes a bit of learning to get fully comfortable with Git.

What is GitHub

- GitHub is a service that lets you host Git repositories in the cloud.
 - In other words, they are hosted remotely by GitHub, and can be downloaded from / uploaded to over the internet. What is a Git remote?
- GitHub allows you to easily distribute code to others by sharing your repository.
- GitHub lets you view your code online easily with a web interface.
- GitHub is free to use as long as you make your code public.
 - Private repositories cost a monthly fee.

Git + GitHub Workflow



Basic Git

- **git init** - Sets up the cwd as a Git repository, tracks all subfolders and files unless specifically told not to
- **git add [files_here]** - Command that “stages” files/folders together into a group. Used just before using the commit command
- **git commit -m “My commit message”** - Saves a snapshot of the project at the moment you commit along with a message. Use good commit messages!
- **git push [remote] [branch]** - Sends or “pushes” your commits to the remote repository
- **git pull [remote] [branch]** - Pulls down changes made to the project in the remote and merges them into your local repo

Let's practice Git

Let's talk about and type out some commands together now.

Instructions

- Create a folder called Final_Project.
- Change into it
- Create 2 files using the touch command: proposal.md, model_diagram.md
- Tell git to watch this folder
- Tell git to track the 2 files
- Open proposal.md in sublime text
- Type 'For my final project, I want to...'
- Check the status of the file
- Look at the difference in the file
- Tell git it's ok to commit this file
- Commit the file
- Log into Github
- Create a new repository without a readme named BEWD_Final_Project
- Go back to your folder and add the remote branch
- PUSH your code to Github

GitHub “Forks”

- As you work on projects in this class, we want you to use GitHub.
- To gain easy access to all the class files, you will check out the class GitHub repository on your machine.
- In order to ensure that changes you make for your projects do not mix with other students work, you will create a fork of the class GitHub repository.
- You will then make changes to your fork, and occasionally pull down changes from the origin class repository.

Now it's your turn

- Fork the class repository if you haven't already. You can find it at:
https://github.com/generalassembly-studio/CHI_BEWD1
 - I will be adding class notes, slides, example code, and other resources each week before the start of each class (typically the night before class)
- Within the Homework folder, add a folder named "your_name" with a file in it named "git_together.txt" and in that file write what the most challenging part of this class so far for you was and what the easiest concept we covered today was.
- Commit your changes
- Push them to your repository
- Submit a pull request so I can access those changes ("What's a pull request", you ask? I think our friend Google might know...)
- This is how you'll submit Homework most weeks.

Help me help you

- Before you leave today, please fill out today's exit ticket which can be found here: <https://docs.google.com/forms/d/1f3uiOqDMThl77Vwm62KeNyBse41Ult93d0LjoyYt9A/viewform> (link is also in the GitHub repository README for lesson 1)



Homework

- Write about what you learned today – like a blog post
- In your fork of this class' GitHub repo, you should now have a folder named after yourself in **Week1/Homework**
- In the **Week1/Homework/your_name** folder, create a .txt file named "Homework.txt" and write your answer in there.
- When finished, add, commit, and push your work to GitHub and then create a new pull request so I can access your changes.