Git 101

Abe Cohen

This document should basically contain everything you need to know about the repository for Highpost and Hummingbird. (Side note: ideally we’d move the Hummingbird project to its own separate repository, but only Marco can do that, and so until that happens everything’s in the same repository but uses different branches. I’ll update this when that happens). I’m writing this tutorial so that by the end, you should know how to do anything you want to do from the command line.

Before you do anything, just know: the single source of truth for Highpost is the highpost-dev branch, and the single source of truth for Hummingbird is the hummingbird branch. **DO NOT MESS WITH EITHER BRANCH UNLESS YOU KNOW WHAT YOU ARE DOING**. However, by the time you finish reading this document, you should know what you’re doing. So that’s a positive, and hopefully you shouldn’t run into too much trouble. If you do, don’t hesitate to come over to me and we can resolve it together.

**Section 1**: Setup

After you’ve successfully downloaded git, if you haven’t already generated and added an SSH Key to your account, read the SSH Section (Section 5) at the bottom of this document first. Go to an empty directory (by the way, to me directory and folder mean the same thing so just roll with it) on your computer. That’s gonna be your workspace. In the command line, run:

git init

All that does is initialize an empty git repository in that directory on your computer.

Now, in order to be able to interact with the remote repository, or the origin, we’ll set up what’s called a remote, running:

git remote add origin git@github.com:Markypc3/mindweaver-highpost.git

What that just did was set up a remote so that you can push and pull from the repository. (Side note, if this is not going well for you, you might need to add an SSH key to your github account. Go to that section and do what it says there, then come back here) And that’s exactly what we’re gonna do now. Run

git pull origin master

git pull

The first command just does exactly what you think it does. It pulls the branch master from the origin. The second gets all the branches and their information. At this point you should be set to start doing work on the repository.

**Section 2**: Getting to the Right Branch in the Repository

Once you’ve completed Section 1, you should be able to move on and begin doing real work. Before we start this section, I just want to mention that you can run

git status

at any time just to see what’s going on. Git status is your friend. Trust me.

Anyway, if you’re not on the correct branch, or you’re coming from section 1 and you’re not on a branch yet, run

git checkout <branch name> example: git checkout hummingbird-intern

Now, you might have a problem doing this if you have uncommitted changes on your computer, (your local environment). If you do you have a couple options: If the local changes are important, then commit them. If not, run

git checkout .

and it will get rid of your local changes for you. If you git status and you still have local changes, run

git stash –u (this will be for if you want to get rid of a local folder that you created for some reason, but you should only have to run this pretty rarely)

Once you git status and it tells you your working directory is clean, nothing to commit, then you should be able to check out the branch that you want.

**Section 3**: Everyday Work Process

Once you’re in the correct branch, the first thing you’re gonna want to do is pull down the latest changes with

git pull

If you can’t because your working directory has some local changes in it, either get rid of the changes or commit them. Anyway, after you pull, make your changes to the code. Then you’ll stage the files you want and commit them with

git add . (this stages all local changes. If you only want to commit some files and not others, you can do git add <filename> <filename> etc. so for example: git add index.js import.js )

git commit –m “This is my commit message” example: git commit –m “My first commit”

What you’ve just done is added your changes to your local copy of the repository. At this point you’ll want to run a

git status

and it’ll either tell you to push your commits, or it’ll tell you that your local repo is a certain number of commits behind the origin. If that’s the case, run

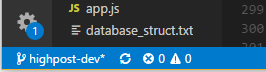
git pull

again. That will pull down the latest version of the repository and attempt to merge it with what you’ve been doing. If all goes well, you can just run

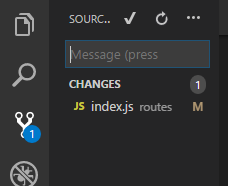
git push

and you’re done for the day. You’ve successfully pulled, edited, and pushed back the repository. If not, you’ll have what are called merge conflicts. We’ll deal with these in the Dealing with Merge Conflicts in VS Code section, because that’s how I deal with merge conflicts. It has a very simple merge conflict resolution tool.

**Section 4:** Visual Studio Code

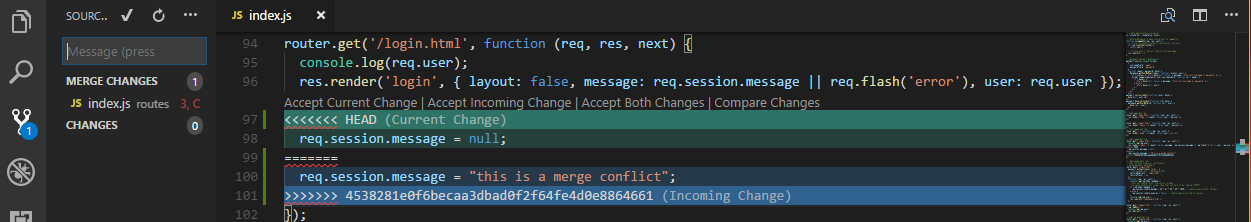
Now that you know what’s flying and can do stuff from the terminal, you can use VS Code for source control, which is really easy, and understand what you’re doing. If you have a git repository set up in a certain directory already, just open that folder with VS Code. Then it should automatically tell you which branch you’re on in the lower left corner

It should also tell you if you’re a certain number of commits ahead or behind the origin. Right now I’m up to date, but if I was ahead it would have an up arrow with a number, and if I was behind it would have a down arrow with a number. Ahead means I have commits that I haven’t pushed yet, and behind means the origin has new commits since the last time I pulled. You can also switch branches if you click on the branch name.

From the Source Control section you should be able to do all the things we were talking about in sections 2 and 3.

If you hover over a file, you can stage that to be committed, or you can hover over the “Changes” section header and stage all your changes at once from there. Also, the M next to the file means “Modified”, so I edited the file, and if you create a new file there’ll be a U there for “Untracked”. After you stage your changes, write in a commit message and hit the check to commit. In the three dots menu you can do most of the stuff we learned. You’ll mostly just push and pull from there, but there are a couple of other useful things in there that you might use every now and then.

**Section 4.1:** Dealing with Merge Conflicts in VS Code

So a merge conflict happens when two people try to edit the same line of code. In this case, while I was editing the line “req.session.message = null”, Stanley edited the same line. VS Code now allows me to either accept my change (Current Change), his change (Incoming Change) or both and just leave both lines in there. Super simple. You can see where the merge conflicts are in the Source Control section, and identify exactly where in the file they are by looking for the blue and green in the scroll bar on the right. Once you’ve decided, run the program to make sure you didn’t catastrophically mess anything up, commit your decision and push it to the repo. If you did catastrophically mess something up, 1. Obviously don’t push that to the origin 2. Get help. It’s probably not that hard to find what’s wrong, since you know exactly what lines of code could be causing the problem.

**ALWAYS GET A SECOND OPINION WHEN DEALING WITH MERGE CONFLICTS**. You don’t want to break anyone’s code, so if you know who you’re conflicting with, deal with the merge conflict together. If you don’t know who you’re conflicting with, you can find out from the Github and see who committed since you last pulled and what they committed. Go to the repository and correct branch on the github website, and click on the commits tab. Then you can find the commit that conflicts with yours and deal with that person.

**Section 5:**  Adding an SSH Key to your Github account

Follow this article. When you’re creating a new key on your computer, I’d recommend not using a password. It’s a hassle. Also, use the default key name, id\_rsa and id\_rsa.pub (.pub is the public one) if you don’t know what you’re doing (if you just leave everything blank it’ll use the defaults, which is fine)

<https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/>

By the way, I happen to think this is a good article, but if it’s confusing to you/not good, let me know and I’ll write out instructions myself.