Git Branching, Merging and Remote Repository

These slides will give basic discussion on each of these items

Branches

- When you create a repo, there is a single branch called **master**
- The master branch should ideally only be used to hold ‘official’ code for your software
  - Working code
  - Well tested (includes unit tests)
  - Deemed official by team members
  - NEVER PLACE THINGS ON MASTER BRANCH WITHOUT CONSULTING TEAM MEMBERS

Branches

- When you want to start working on your own items, you should create a branch for that work
- Your initial repo just has a master branch. When you create a new branch it will contain everything currently committed to the master branch
- Any work you do on your new branch will NOT affect the things in the master branch (or any other branch for that matter)

Branch Commands

- To create a branch: **git checkout -b new_branch_name**
- To move from current branch to another: **git branch branch_name_you_want_to_move_to**
- To get a list of branches in your repo: **git branch**
- To delete a branch, make sure your not in that branch, then: **git branch -d branch_name**
- Note that git will show you your current branch in parens at the end of your command prompt

Merging

- To merge the contents of one branch into another, move to the branch you would like to merge things into, then: **git merge name_of_branch_to_merge_things_from**
- If all the items in the branch you are merging from are different from your current branch, the merge will proceed without issue

Merging

- If you change a file in the branch you are merging from, git will merge that file into current branch provided there are not conflicts with that file in current branch. If there are, you must resolve conflicts (typically with a text editor)
- Adding new text/content to an existing file usually does not result in a conflict. Conflicts arise when you delete content from before or modify content from before.

Creating a GitHub repo

- Go to github.com and create yourself an account
- Once this is done you can choose New in the Repositories section
- Choose an appropriate name for the repo (for your project perhaps TeamNameDungeonAdventure)
- Allow GitHub to create a README.md file
- Add a .gitignore file based on your programming language and IDE that you intend to use
  - This file lists files and extensions that should NOT be versioned by git. It is very important to use a .gitignore to avoid tracking all the support files used by IDEs. They can cause MASSIVE merge conflicts

Working with a Remote Repository

- Once you have created a repository in a remote location (e.g. GitHub), you can copy the contents of that repository to your local machine via the clone command: **git clone address_of_remote_repo**
  - GitHub provides a means to copy/clone repo from the main page of your repo (click the Clone or Download button and copy the address you see there)
- Note that in order to access the remote repo you may be asked for your username and password for that repo/account. This is the case when working with GitHub. The good news is that once you enter those credentials for the repo git remembers them
- The name of the remote repo defaults to **origin** (since that's where it originated). You can change the name if you want, but we won't worry about that now.

Synching contents between Repos

- To grab latest content from remote repository: **git pull**
  - Note: you should do this each time you begin work on your local repo and as necessary when you are working for hours on your repo (especially if you know others may be pushing files to the remote)
- To place your latest work on the remote repo: **git push**
- Both push and pull are specific to your current branch

Synching Contents

- When you create a branch on your local repo, the first time you push it to the remote, you will get the following message when you type **git push**:
  - fatal: The current branch new_branch has no upstream branch. To push the current branch and set the remote as upstream, use

    ```
    git push --set-upstream origin new_branch
    ```

  - Simply issue the above command and your new branch will be published to the remote repo

Creating new branch on GitHub

- From your main repo on GitHub click the branch button/dropdown
- Enter the name of your new branch and place whatever files you'd like in it (note that when you create the branch it will contain the contents of the branch that was active when you created it
- To pull this branch down from your remote repo, type **git fetch origin**
- Then type **git checkout -b new_branch_on_GitHub origin/new_branch_on_GitHub** to complete the registration process

Team Based Repo Recommendations

- One team member should create a repo on GitHub
  - When creating, allow GitHub to create a README.md file so that the repo can be immediately cloned
  - You should also add a .gitignore file to the repo during creation to ensure support files that you don’t want to version are ignored. THIS IS VERY, VERY IMPORTANT. BE SURE AND DO THIS BASED ON THE LANGUAGE AND IDE YOUR TEAM IS USING
- Invite other people via Settings/Manage Access/Invite a Collaborator

Team Repo

- Once the GitHub repo is set up (including README.md and .gitignore) each person should clone the repo
- Each team member should then create a branch on their local machine with their name
- All work should be done from within that branch
  - Note you can create sub-branches within that branch as necessary
- Each team member should push their branch up to GitHub repo
- Merge items onto the master branch only after consulting one another!!
  - This can be done remotely and pushed, or from GitHub via Compare and Pull request