GIT FOR EE's

Version control is now a fundamental tool for EE's and there is an expectation within the industry to have basic familiarity with version control tools.

Summary

My version control preference is git, although many EE's use svn. I like git over svn for a few reasons:

1. Git is lightweight. It doesn't make copies of the commits. It only saves deltas between commits.
2. 99% of software devs have adopted git so the documentation and support community is vast.
3. There are a lot of options and good support for hosting remote repos.
4. Once you learn a few key commands you can get started

Setting Up Git

- Download git
- Go to github and create an account
- open the bash shell and enter the following commands:
  - `git config --global user.email email@provider.com` Use the same email as github account
  - `git config --global user.name githubUserName` Use the same username as github account
- The command `git config user.email` should return your email
- The command `git config user.name` should return your user name

Cloning A Repository

- Find the remote repo you want to clone
- copy the URL
- open the git bash shell & navigate to the directory where you want to save the code
- type the command `git clone url` but replace url with the actual URL you have copied. You can use [shift + insert] to paste into the bash shell.

Creating A Local Repository

- open the bash shell and navigate to the directory you want to put into version control
- type the command `git init`
- add a `.gitignore` file in this directory.
  - gitignore tells git what files, file extensions, folders, etc that it should ignore
  - gitignore files can be modified and customized for each project.
  - You can usually find a gitignore file that is a good starting point. This is a good one for Microchip Studio projects
  - you can start a fresh git ignore file by typing the command `touch .gitignore` in the bash shell.
- Add a ReadMe file to the repository. I would suggest a txt or markdown file. This is for Github and will display info regarding the project when looking at the remote repository.
- Add files to the repository without tracking them (Skip Worktree)

    - Sometimes there are situations where you don't want to track changes to a file, but you want the file included in the repository.
    - gitignore will NOT work for this use-case.
    - The solution here is to use `git update-index --skip-worktree <file_name>`
    - You can add files back into the repo to be tracked with this command `git update-index --no-skip-worktree <file_name>`
    - You can see all the skipped files with this command `git ls-files -v|grep '^S'`
- Add files to the staging area
    - `git add filename` will add an individual file
    - `git add .` will add all changed files to the staging area. This is convienent as long as the ignore and skip-worktree have been properly setup
- Add files the repository
    - `git commit -m "this is a commit message"`
    - Make your commit message meaningful. Don't say "fixed stuff". Be specific.
- Checking out a branch
    - it isn't good practice to develop on the `main` or `master` branch. You should checkout a new branch to work on.
    - `git checkout -b new-feature` this command created and switched over to a branch called "new-feature".
    - Display all branches `git branch`
- Local Merge: new-feature branch into master branch
    - `git checkout master`
    - `git merge new-feature`
    - you can now delete new-feature branch if you wish `git branch -d new-feature`
- Checking status of repo `git status`
- Looking at commit history
    - `git log -n X` this will pull up the last X commits.

Adding a Local Repo to a Remote Repo

- go to github and create a new bare repository. Don't add or create any files, just the repo.
- Copy the code URL
- Open the git bash shell in the local repo directory
- `git remote add origin URL` this adds the remote repository URL to an alias called origin. This command links the local and remote repos.
- `git push origin branchName` Push a branch from the local repo up to the remote.
- Pull a branch into the local repo from the remote repo
    - `git checkout branchName` Go to the branch in the local repo
    - `git pull origin branchName` Pull the files from the remote repo into the local repo

Git Command Cheat Sheet

git command	Description
git config --global user.email email@provider.com	Sets the email address of the git user. It's a good idea to make this match the account used for remote repo hosting

git command	Description
git config --global user.name githubUserName	Sets the user name of the git user. It's a good idea to make this match the account used for remote repo hosting.
git config user.email	returns the user email
git config user.name	returns the username
git init	creates a new git repository
touch .gitignore	creates a new .gitignore file
git status	displays the repo status. What files have changed in the working directory and what files are added to the staging area.
git add .	Adds all files changed files to the staging area
git add filename	Adds a single file to the staging area
git commit -m "commit message"	Adds changes to the repository
git branch	Displays all the branches of the repo
git checkout -b new-feature	Creates a branch called new features and switches the working directory onto that branch
git clone url	clones a remote repository found at a specific URL
git update-index --skip-worktree <file_name>	Allows this file to be added to repository but the file changes aren't tracked
git update-index --no-skip-worktree <file_name>	Removes the skip flag from the file and git will now track changes made to this file
git ls-files -v|grep '-S'	Displays all the skipped files in the repo
git merge new-feature	Merge the changes on branch 'new-feature' the working directory
git log -n X	Will display the last X number of commits
git remote add origin URL	adds a remote repo at the URL location to the local repo. Alias for the remote URL is called 'origin'
git push origin new-feature	Pushes the new-feature branch to the remote repository
git pull origin new-feature	pulls the new-feature branch from the remote repository and puts in the working directory