Git Training

Generate SSH Key

```
Linux (terminal)
$ ssh-keygen -t rsa -b 4096 -C "your.email@wsu.com"
# Keep hitting enter
$ cat ~/.ssh/id_rsa.pub

Windows (cmd/powershell)
> ssh-keygen -t rsa -b 4096 -C "your.email@wsu.com"
# Keep hitting enter
> type $env:USERPROFILE\.ssh\id_rsa.pub

Mac (terminal)
% ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
# Keep hitting enter
% cat ~/.ssh/id_rsa.pub
```

Adding SSH key to github

- Profile pic in top right
- Settings (Not the repository settings in the middle of the page if you have one)
- SSH and GPG Keys
- New SSH Key
- Paste the key

Commands to move around in the terminal

Ubuntu Command	Windows Command	macOS Command
cd	cd	cd
ls	dir	ls
pwd	cd or echo%cd%	pwd
mkdir	mkdir or md	mkdir
rm/rm -r	del / rmdir /s	rm/rm -r
mv	move	mv
ssh	ssh	ssh

Branch Naming Convention

- `feature/` for new features.
- `bugfix/` for bug fixes.
- `hotfix/` for critical fixes that need to go into production immediately.
- `chore/` for maintenance work or minor updates.
- `release/` for preparing a release.
- `experiment/` for experimental changes or prototypes.
- `test/` for branches dedicated to testing.

Adding the Repo to your computer

```
# Starting point
# Make sure to clone the ssh link!
git clone ssh:.......
```

Creating a Branch/Moving between Branches

```
# The "-c" means create new branch
git switch -c feature/....

# If the branch alreay exists
git switch feature/....

# *Remember you can TAB to see what branches you have, if you li
# the website, do another git pull*

# *If you encounter an error, it is most likely because you have
# you haven't commited. You need to commit any changed before yo
```

Checking the files you have edited since the last commit

```
git status
```

Commiting your code

```
# Note: Commits are local to your computer, no one can see them
# You have to commit locally before you can push

# STEP 1: Add the files to you commit, you have 2 options
# Option 1: Add any and all files that have changed
git add .

# Option 2: Specify which files you want to change
git add *Hit TAB and you should see the files that have changed


# STEP 2: Add a commit message (This is what is shown next to th
git commit -m "Message here"
```

```
# STEP 3: Push to the server. Others can see this code now. You
# from a branch, never push from the main.
git push
```

Merging your code into the main

```
# STEP 1: Make sure your main branch is up to date
git switch main
git pull
git switch back_back_to_branch

# *If you encounter an error, it is most likely because you have
# you haven't commited. You need to commit any changed before yo

# STEP 2: Merge
# From the branch
git merge main

# Open vs code and go to the git option on the left. Look for an
# To open vs code where you are at, simply do "code ." (there is

# STEP 3: Now that any conflicts are resolved, push it to the se
git push

# STEP 4: Go to the github website and create a PR there
```

Temporarily Store Code Without Commiting (Useful when you need to switch to main to get the latest version but want to go right back to the branch)

```
# Temp store the changes
git stash
```

```
# Go back to the code before the most recent stash
git pop

# List all stashed
git stash list

# Pop a specific stash from the list, # is the int from git stas
git stash apply stash@{#}
```

Ignore files

```
#Make a ".gitignore" file if you don't already have one'

# Anything in your git ignore won't be added when you do a "git
```

Example .gitignore (quite long but you can choose what you need)

```
# Ignore OS-generated files
.DS_Store
Thumbs.db

# Ignore node_modules folder (Node.js projects)
node_modules/

# Ignore log files
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*

# Ignore build files
/dist/
```

```
/build/
/out/

# Ignore temporary files and directories
*.tmp
*.temp
*.swp
*.swo
*.bak
*.pid
*.seed
*.pid.lock

# Ignore compiled code
*.class
*.dll
*.exe
*.o
*.so

# Ignore dependency directories
/vendor/ # Composer vendor folder
/venv/ # Python virtual environment folder

# Ignore Python-specific files
__pycache__/
*.py[cod]
*.pyo
*.pyd

# Ignore environment files
.env
.env.local
.env.*.local

# Ignore IDE-specific files
```

```
.vscode/
.idea/
*.sublime-workspace
*.sublime-project

# Ignore macOS-specific files
._*
.Spotlight-V100
.Trashes

# Ignore temporary storage files created by editors
*.~*
*.swp
*.swo

# Ignore database files
*.sqlite
*.sql

# Ignore minified or compressed files
*.min.js
*.min.css

# Ignore compressed archive files
*.zip
*.tar.gz
*.rar
*.7z
```