Git Quick Start

This is intended as a quick and handy reference for common git commands. It assumes basic familiarity with git concepts. See online documentation like git tutorial for more detailed information.

This document is partially based on *git - the simple guide* and adapted for the Nucor PSI project.

Download a client

If you need to download a client, you can get it here: https://git-scm.com/downloads

Checkout the repository

Create a local copy of a repository with `git clone`

```
# format: git clone <url>
# psi repository url is https://Nucor-NBT@dev.azure.com/Nucor-NBT/NextGen/_git/psi

git clone https://Nucor-NBT@dev.azure.com/Nucor-NBT/NextGen/_git/psi
```

Workflow

Remember that this is a local copy of a remote repository hosted on a server. The local repository consists of three "trees". - Working directory: the actual files on the file system. - Index: changes that have been staged. - HEAD: a reference to the latest commit in the active branch.

In a typical workflow you will edit files (in the working directory), stage changes (in the index), and commit them to the active branch moving the HEAD to the new commit.

These changes are still only local until you `push` them to the remote repository.

Status

Use `git status` see a summary of git status, including differences between trees.

```
git status
```

Stage

Stage a file with changes (add it to the index) with `git add`.

```
git add <filename>

# to stage all changes
git add .
```

Commit

Commit staged changes with `git commit`. A message is usually required.

```
git commit -m "Commit message"
```

A new commit is created on the active branch and HEAD now points to this new commit. *Remember: the commit only exists locally until you push it.*

Remote

By default, when you clone a repository, git will create a `remote` named `origin` using the URL of the repository you cloned from. If you initialized a new repository locally, you may need to define a remote.

```
git remote add origin <url>
```

The name origin is normally used by convention. It is possible to define multiple remotes with different names.

Push

Push changes with `git push`.

```
# if remote and upstream branch is defined
git push

# to specify the remote and branch to push to
# git push <remote> <branch>
git push origin my-feature-branch

# to push and define the upstream branch for a local branch
# git push --set-upstream <remote> <branch>
git push --set-upstream origin my-feature-branch
```

After push, your changes are in a branch on the remote server.

Pull

Update your local repository with `git pull`.

```
git pull
```

A push and pull are types of merge. You are merging changes between local and remote branches. You may need to resolve merge conflicts. Integrate with the mainline branch often to minimize the scope of potential merge conflicts.

Merge branches

Merge branches with `git merge`.

```
# to merge another branch into the active branch
# git merge <branch>
git merge my-feature-branch
```

You can preview a merge before trying.

```
git diff <source_branch> <target_branch>
```

Consider merging from the main branch on the server (origin/main) into your local development branch periodically (recommended daily) to minimize the scope of potential merge conflicts.

```
git merge origin/main
```

Branch

Create and switch to a new branch with

```
# git checkout -b <branch_name>
git checkout -b my-feature-branch
```

Switch to an existing branch with

```
# git checkout <branch_name>
git checkout main
```

Delete a branch with

```
# git branch -d <branch_name>
git branch -d my_feature_branch
```

Deleting a branch does not delete the commits. You can always create a new branch from any commit in history.

If you create a branch locally, it is not available on the server until you push it. Remember to set the upstream branch.

Tags

You can mark a commit with a helpful name using tags. This is often used for releases.

```
# to tag commit 1b2e1d63ff as v1.0.0
git tag v1.0.0 1b2e1d63ff
```

History

Use `git log` to view the commit history.

```
# To see only the commits of a certain author
git log --author=bob
```

```
# To see a very compressed log where each commit is one line
git log --pretty=oneline

# To see a ASCII art tree with all branches
git log --graph --oneline
# Try it!
```

Undo

To undo changes in a file and roll it back to match the version in HEAD.

```
git checkout -- <filename>
```

To remove all changes in the working directory and index.

```
git reset --hard
```

To store the state of the working directory and index in a stash while reverting back to HEAD.

```
git stash push -m "Stash message"

# restore the last stash from the stack
git stash pop
```

Graphical tools

There are many graphical front ends available. VSCode has a lot of good functionality baked in. VSCode extension like Git History are easy to recommend.

Helpful links

- git tutorial
- git - the simple guide
- git cheat sheet