Git Commands

- [] represent optional elements
- {} represent required elements
- Anything after # is a comment

Basics

```
git init                        # create repo
git status                      # show repo status
git add .                       # add all files to staging area
git commit -m "message here"    # commit
git push -u origin main         # push to remote repo
```

Configure settings

```
# Display user information
git config -l
git config --list

# Display username
git config [--global] user.name

# Display user email
git config [--global] user.email

# Configure username and email
git config [--global] user.name {name}
git config [--global] user.email {email}
```

Create or clone repo

```
# Create new local repo
git init

# Clone remote repo
git clone ssh://user@domain.com/repo.git
```

Manage local changes

```
# Display status of repo, changed files, etc
git status
```

```
# Display changes to tracked files, changes not yet staged
git diff

# Display changes to specific file
git diff {file name}

# Display file differences between staging and the last file version
git diff --staged

# Display differences between two branches
git diff {branch one} {branch two}

# Add all current files to stage area
git add .

# Add specific file to stage area
git add {file name}

# Commit staged changes
git commit -m {message}

# Display log of commits
git log

# Display changes over time for a specific file
git log -p {file name}
```

Working with incomplete work

```
# Temporarily store all modified tracked files
git stash

# Temporarily store all modified tracked and untracked files
git stash -u

# Restore the most recent stashed files
git stash pop

# List all stashed changesets
git stash list

# Discard the most recent stashed changeset
git stash drop
```

Working with branches

```
# Display all branches
git branch
```

```
# Create new branch
git branch {branch name}

# Switch to specified branch
git checkout {branch name}

# Create new branch and switch to it
git checkout -b {branch name}

# Delete branch
git branch -d {branch name}

# Rename current branch
git branch -M {new branch name}
```

Merging and rebasing

For these examples, we're using a `main` branch and a `dev` branch.

Merge

```
# On main branch, the following command will
# create a new commit on main branch
# with both main and dev as parents
git merge dev

# The following commands will move the
# dev branch to the new commit where the
# main branch and dev branch are parents
git checkout dev
git merge main
```

Rebase

```
# On dev branch, the following command will
# move the dev commits onto main branch
git rebase main

# The following commands will get the
# main branch up to date
git checkout main
git merge dev  # or
git rebase dev
```

Moving around the commits on a branch

```
# Checkout the commit right before main
git checkout main^

# Checkout 2 commits before main
git checkout main~2

# Move by force the main branch to 3 parents
# behind HEAD
git branch -f main HEAD~3
```

Working with remotes

```
# List remotes
git remote -v

# Set up remote repo url
git remote add origin {remote url}

# Change remote repo url
git remote set-url origin {new remote url}

# Push a branch to remote, checkout branch first
git push [{remote name} {branch name}]

# Push main branch to origin
# `-u` creates an upstream tracking connection
# and is especially useful when publishing
# a local branch on a remote for the first time
# `-u` sets up a default
# `--all` pushes all local branches
# `--delete` deletes the specified remote branch
git push -u origin main

# Get commits from remote but do not merge them
git fetch

# Get commits from remote and then merge them
git pull

# If you have local changes but local branch
# is not up to date, this will pull changes
# and rebase them
# Local changes can then be pushed
git pull --rebase
```

Reversing changes

Reset

This reverts changes by moving current branch reference backwards in time to an older commit.

It will move the branch backwards as if the commit had never been made in the first place.

This is good for local branches, but it doesn't work for remote branches that others are using.

```
git reset HEAD^
```

Revert

The following command puts down a new commit below the commit we want to reverse.

The new commit introduces changes that are the reverses of the previous commits.

```
git revert master~2
```

Note, you can checkout a previous commit and create a new branch with `git branch {new branch name}`.