https://guides.github.com/

Git Tutorial

Installing Git

- Linux $ sudo apt-get install git
- Mac (homebrew) $brew install git
- Windows (https://git-for-windows.github.io/)

Configuring Git

- $ git config --global user.name "My Name"
- $ git config --global user.email myEmail@example.com

Creating repository

- $ cd Desktop/git_exercise/
- $ git init
- output

  Initialized empty Git repository in /home/user/Desktop/git_exercise/.git
- Status $ git status

Add & commit

- $ git add filename.extension
- $ git add -A (should be inside the folder)
- git commit -m "Write comments"

Remote repo

- $ git remote add origin uri
- $ git push origin master (branch name)
- $ git clone uri
- $ git pull origin master (branch name)(for getting updated one)

Contd...

- $ git branch name
- $ git branch name * master (switch)
- $ git log (for log)

  Sample to add a new file to the branch
- $ git add filename.extension
- $ git commit -m "comments"

- Switch to master.

  $ git checkout master

- To merge
- $ git merge branch name
- $ git branch -d branch name

Git Cheat Sheet

Remember!
git <COMMAND> --help

Global configuration is stored in ~/.gitconfig.
git config --help

master is the default development branch.
origin is the default upstream repository.

Create

From existing data
```
cd ~/my_project_directory
git init
git add .
```

From existing repository
```
git clone ~/existing_repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://user@host.org/project.git
```

Show

Files changed in working directory
```
git status
```

Changes made to tracked files
```
git diff
```

What changed between ID1 and ID2
```
git diff <ID1> <ID2>
```

History of changes
```
git log
```

History of changes for file with diffs
```
git log -p <FILE> <DIRECTORY>
```

Who changed what and when in a file
```
git blame <FILE>
```

A commit identified by ID
```
git show <ID>
```

A specific file from a specific ID
```
git show <ID>:<FILE>
```

All local branches
```
git branch
```
star (*) marks the current branch

Revert

Return to the last commited state
```
git reset --hard
```
This cannot be undone!

Revert the last commit
```
git revert HEAD
```
Creates a new commit

Revert specific commit
```
git revert <ID>
```
Creates a new commit

Fix the last commit
```
git commit -a --amend
```
(after editing the broken files)

Checkout the ID version of a file
```
git checkout <ID> <FILE>
```

Update

Fetch latest changes from origin
```
git fetch
```
(this does not merge them)

Pull latest changes from origin
```
git pull
```
(does a fetch followed by a merge)

Apply a patch that someone sent you
```
git am -3 patch.mbox
```
In case of conflict, resolve the conflict and
```
git am --resolved
```

Publish

Commit all your local changes
```
git commit -a
```

Prepare a patch for other developers
```
git format-patch origin
```

Push changes to origin
```
git push
```

Make a version or milestone
```
git tag v1.0
```

Branch

Switch to a branch
```
git checkout <BRANCH>
```

Merge BRANCH1 into BRANCH2
```
git checkout <BRANCH2>
git merge <BRANCH1>
```

Create branch BRANCH based on HEAD
```
git branch <BRANCH>
```

Create branch BRANCH based on OTHER and switch to it
```
git checkout -b <BRANCH> <OTHER>
```

Delete branch BRANCH
```
git branche -d <BRANCH>
```

Resolve merge conflicts

View merge conflicts
```
git diff
```

View merge conflicts against base file
```
git diff --base <FILE>
```

View merge conflicts against your changes
```
git diff --ours <FILE>
```

View merge conflicts against other changes
```
git diff --theirs <FILE>
```

Discard a conflicting patch
```
git reset --hard
git rebase --skip
```

After resolving conflicts, merge with
```
git add <CONFLICTING_FILE>
git rebase --continue
```

Workflow