

# Git Cheat Sheet

## Command

### Observe your Repository

List new or modified files not yet committed

```
git status
```

### Show the changes to files not yet staged

```
git diff
```

### Show full change history

```
git log
```

## Synchronize

Get the latest changes from origin (no merge)

```
git fetch
```

Fetch the latest changes from origin and merge

```
git pull
```

Fetch the latest changes from origin and rebase

```
git pull --rebase
```

Push local changes to the origin

```
git push
```

## Working with Branches

List all local branches

```
git branch
```

List all branches, local and remote

```
git branch -av
```

Create a new branch called my-branch

```
git branch my-branch
```

Switch to a branch and update working directory

```
git checkout branch-name
```

Delete the branch called my-branch

```
git branch -d my-branch
```

Merge branch a into branch-b

```
git checkout branch-b
```

```
git merge branch-a
```

Tag the current commit

```
git tag my-tag
```

## Make a change

Stages the file, ready for commit

```
git add [file]
```

Commit all staged files to versioned history

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

Unstages file, keeping the file changes

```
git reset [file]
```

Undoes all commits after [commit], preserving change locally

```
git reset [commit]
```

Discards all history and changes back to the specified commit

```
git reset --hard [commit]
```

## Save fragments

Lists all stashed changesets

```
git stash list
```

Temporarily stores all modified tracked files

```
git stash
```

Temporarily stores all modified tracked files with a stash name

```
git stash save "stash-name"
```

Details stash number 'n'

```
git stash show stash@{n} -p
```

Restores stash number 'n'

```
git stash apply stash@{n}
```

Discards stash number 'n'

```
git stash drop stash@{n}
```

Restores the most recently stashed files

```
git stash pop
```

## Oh Shit, Git!

I did something terribly wrong, please tell me git has a magic time machine!?!

```
git reflog
```

*# you will see a list of every thing you've done in git, across all branches!*

*# each one has an index HEAD@{index}  
# find the one before you broke everything*

```
git reset HEAD@{index}
```

*# magic time machine*

I committed and immediately realized I need to make one small change!

*# make your change*

```
git add . # or add individual files
```

```
git commit --amend
```

*# follow prompts to change or keep the commit message*

*# now your last commit contains that change!*

I need to change the message on my last commit!

```
git commit --amend
```

*# follow prompts to change or keep the commit message*

I accidentally committed something to master that should have been on a brand new branch!

*# create a new branch from the current state of master*

```
git branch branch-name
```

*# remove the commit from the master branch*

```
git reset HEAD--hard
```

```
git checkout branch-name
```

*# your commit lives in this branch now*

I accidentally committed to the wrong branch!

*# undo the last commit, but leave the changes available*

```
git reset HEAD--soft
```

```
git stash
```

*# move to the correct branch*

```
git checkout branch-name
```

```
git stash pop
```

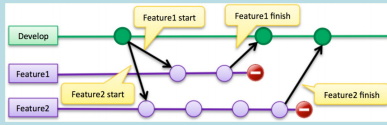
```
git add . # or add individual files
```

```
git commit -m "your message here"
```

*# now your changes are on the correct branch*

# Git Cheat Sheet

## Feature Flow



### Feature start

```
git checkout develop
git pull --rebase origin develop
git checkout -b feature/nomfeature
```

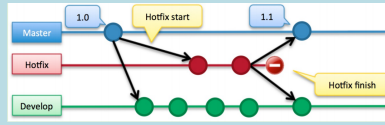
### Feature Share

```
git push origin feature/nomfeature
```

### Feature finish

```
git checkout develop
git merge --no-ff feature/nomfeature
git branch -d feature/nomfeature
git push origin develop
git push origin :feature/nomfeature (if pushed)
```

## Hotfix Flow



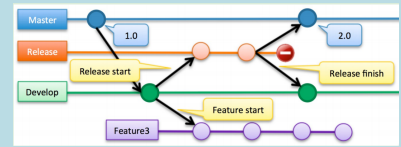
### Hotfix start

```
git checkout develop
git pull --rebase origin develop
git checkout -b hotfix/hotfix-version
```

### Hotfix finish

```
git checkout master
git merge --no-ff hotfix/hotfix-version
git tag -a hotfix-version
git checkout develop
git merge --no-ff hotfix/hotfix-version
git branch -d hotfix/hotfix-version
git push origin master
git push origin develop
git push origin --tags
git push origin :hotfix/hotfix-version (if pushed)
```

## Release Flow



### Release start

```
git checkout develop
git pull --rebase origin develop
git checkout -b release/release-version
```

### Release Share

```
git push origin feature/release-version
```

### Release finish

```
git checkout master
git merge --no-ff release/release-version
git tag -a release-version
git checkout develop
git merge --no-ff release/release-version
git branch -d release/release-version
git push origin master
git push origin develop
git push origin --tags
git push origin :release/release-version (if pushed)
```