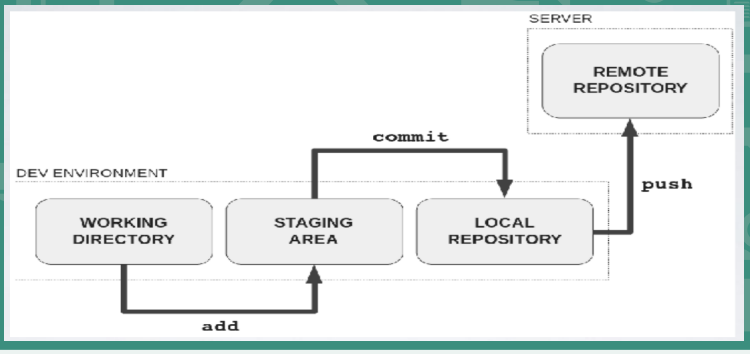
GitHub
GIT CHEAT SHEET



Git is the free and open source distributed version control system that's responsible for everything GitHub related that happens locally on your computer. This cheat sheet features the most important and commonly used Git commands for easy reference.

Host webpage from GitHub via GitHub pages:
from a repository >> ... >> settings >> GitHub pages
select source and save. Note repository must be public

INSTALLATION & GUIS

With platform specific installers for Git, GitHub also provides the ease of staying up-to-date with the latest releases of the command line tool while providing a graphical user interface for day-to-day interaction, review, and repository synchronization.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

For Linux and Solaris platforms, the latest release is available on the official Git web site.

Git for All Platforms
http://git-scm.com

SETUP

Configuring user information used across all local repositories

`git config --global user.name "[firstname lastname]"`

set a name that is identifiable for credit when review version history

`git config --global user.email "[valid-email]"`

set an email address that will be associated with each history marker

`git config --global color.ui auto`

set automatic command line coloring for Git for easy reviewing

SETUP & INIT

Configuring user information, initializing and cloning repositories

`git init`

initialize an existing directory as a Git repository

`git clone [url]`

retrieve an entire repository from a hosted location via URL

STAGE & SNAPSHOT

Working with snapshots and the Git staging area

`git status`

show modified files in working directory, staged for your next commit

`git add [file]`
use: git add . to add all files which are new or changed
add a file as it looks now to your next commit (stage)

`git reset [file]`

unstage a file while retaining the changes in working directory

`git diff`

diff of what is changed but not staged

`git diff --staged`

diff of what is staged but not yet committed

`git commit -m "[descriptive message]"`
use: --amend to modify the most recent commit and combine staged changes

commit your staged content as a new commit snapshot

BRANCH & MERGE

Isolating work in branches, changing context, and integrating changes

HEAD is the current commit checked out

`git branch`

list your branches. a * will appear next to the currently active branch

`git branch [branch-name]`
-f option before branch name forces branch to be moved, can be used with ~ (relative commits back) use to move main
-u origin/main [branch-name] set to track remote
create a new branch at the current commit
option to specify branch to move to after branch-name

`git checkout [name]`
-b [branch-name] creates and checksout
GIT version 2.23 on use: git switch
switch to another branch and check it out into your working directory

git checkout -b [branch-name] origin/main creates new branch and set to track origin/main

`git merge [branch]`

merge the specified branch's history into the current one

`git log`

show all commits in the current branch's history

git checkout main^ goes to parent of main
git checkout [name]~3 goes 3 commits up
git checkout HEAD^^ goes to grandparent of head
git checkout [name]^2 goes to parent 2

git tag [tag name] [optional commit to tag]
if no commit is given, the head is tagged
permanently tags a commit
cannot checkout a tag

git describe [commit]
if no commit given, head is used
outputs closeset anchor (tag), number of commits and hash (name) from commit
<tag>_<numCommits>_g<hash>

INSPECT & COMPARE

Examining logs, diffs and object information

`git log`

show the commit history for the currently active branch

`git log branchB..branchA`

show the commits on branchA that are not on branchB

`git log --follow [file]`

show the commits that changed file, even across renames

`git diff branchB...branchA`

show the diff of what is in branchA that is not in branchB

`git show [SHA]`

show any object in Git in human-readable format

TRACKING PATH CHANGES

Versioning file removes and path changes

`git rm [file]`

delete the file from project and stage the removal for commit

`git mv [existing-path] [new-path]`

change an existing file path and stage the move

`git log --stat -M`

show all commit logs with indication of any paths that moved

IGNORING PATTERNS

Preventing unintentional staging or commiting of files

```
logs/
*.notes
pattern*/
```

Save a file with desired patterns as .gitignore with either direct string matches or wildcard globs.

`git config --global core.excludesfile [file]`

system wide ignore pattern for all local repositories

SHARE & UPDATE

Retrieving updates from another repository and updating local repos

`git remote add [alias] [url]`

add a git URL as an alias

`git fetch [alias]`

note this does not change local files

fetch down all the branches from that Git remote

git fetch origin source:destination

used to fetch specific commits to a specific branch

if no source, creates branch

`git merge [alias]/[branch]`

merge a remote branch into your current branch to bring it up to date

`git push [alias] [branch]`

<place> can also be source:destination

git push <remote> <place> updates <remote> based on local <place>

Transmit local branch commits to the remote repository branch

delete a branch with git push origin : [branch2delete]

`git pull`

use option: --rebase to fetch and rebase instead of fetch and merge

fetch and merge any commits from the tracking remote branch

git pull origin source:destination can also be used to specify branches

git cherry-pick [commit] [commit] ...

Adds copies of commits specified to current branch

REWRITE HISTORY

Rewriting branches, updating commits and clearing history

`git rebase [branch]`

use: -i option to interactively choose commits to pick

apply any commits of current branch ahead of specified one

git rebase <to here> <move this> can move main putting main in <move this>

works well for local branches, doesn't work for remote branches

`git reset --hard [commit]`

specify branch to reset to

clear staging area, rewrite working tree from specified commit

git revert [commit]

reverses changes and shares those with others

will add commit specified ahead of old with changes (named: oldname')

TEMPORARY COMMITS

Temporarily store modified, tracked files in order to change branches

`git stash`

Save modified and staged changes

`git stash list`

list stack-order of stashed file changes

`git stash pop`

write working from top of stash stack

`git stash drop`

discard the changes from top of stash stack