$ git branch

$ git logs

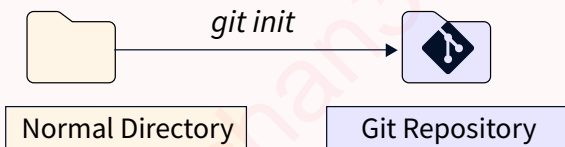# GIT CHEAT SHEET

$ git fetch

$ git status

SETUP & INIT

Configuring user information,
initializing and cloning repositories.

$ git init

Initialize an existing directory as a
Git repository.

git init

Normal Directory → Git Repository

$ git clone [url]

Retrieve an entire repository from
a hosted location via URL.

Remote
Repository at [URL]

git clone [URL]   git clone [URL]

Local Copy of the   Local Copy of the
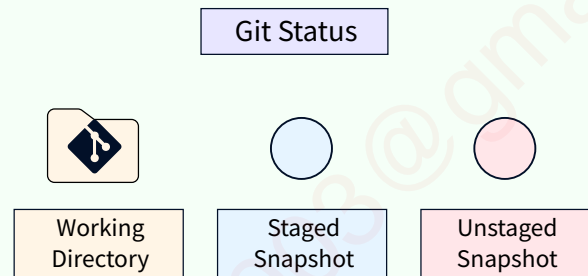Repository   Repository

STAGE & SNAPSHOT

Working with snapshots and the
Git staging area.

$ git status

Show modified files in working directory,
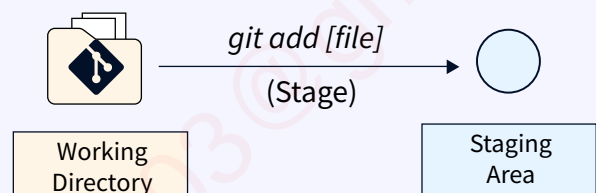staged for your next commit.

Git Status

Working   Staged   Unstaged
Directory   Snapshot   Snapshot
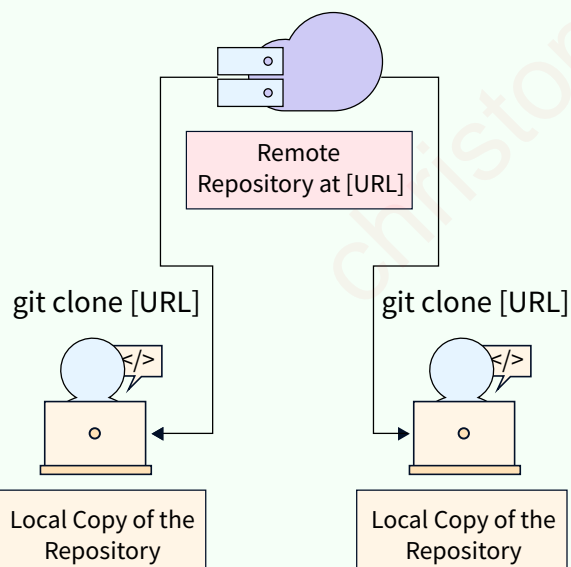
$ git add [file]

Add a file as it looks now to your
next commit (stage).

git add [file]
(Stage)

Working   Staging
Directory   Area

$ git reset [file]

Unstage a file while retaining the changes in working directory.

(Stage)

git add [file]

git reset [file]

(Unstage)

Working Directory

Staging Area

$ git diff

Diff of what is changed but not staged

≠

Working Directory

git diff

Staging Area

git diff --staged

Diff of what is staged but not yet commited.

HEAD

≠

Staging Area

git diff --staged

Commit History

$ git commit -m "[descriptive message]"

Commit your staged content as a new commit snapshot.

Working Directory

Staging Area

Git Repository (Commit History)

git add

git commit

BRANCH & MERGE

Isolating work in branches, changing context, and integrating changes.
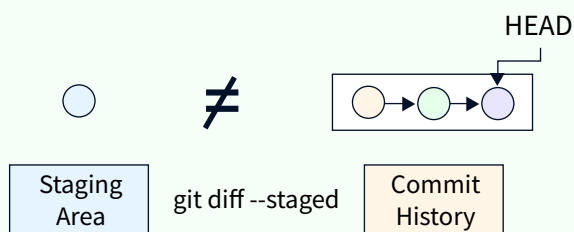
$ git branch

List your branches. A * will appear next to the currently active branch.
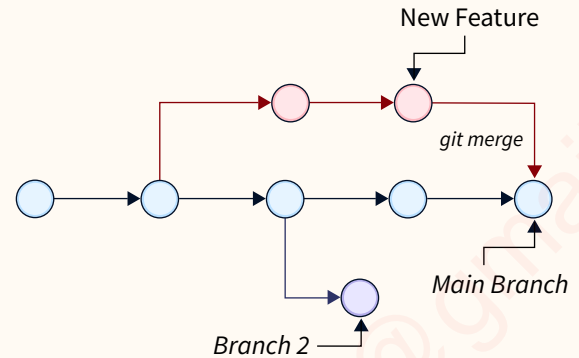
Branch 1

Main Branch
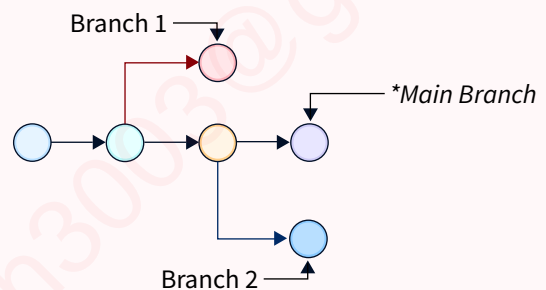
Branch 2

$ git branch [branch-name]

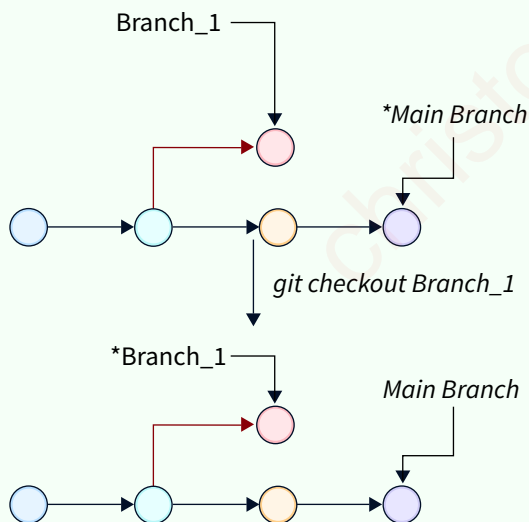Create a new branch at the current commit.
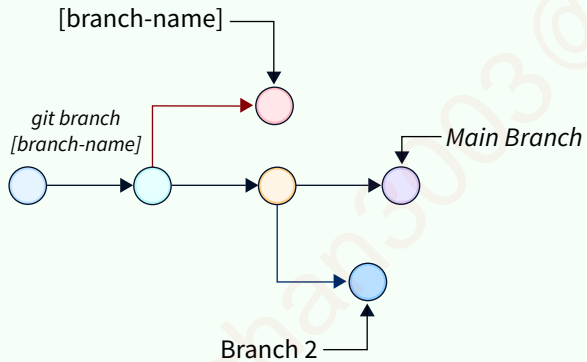
$ git merge [branch]

Merge the specified branch's history into the current one.
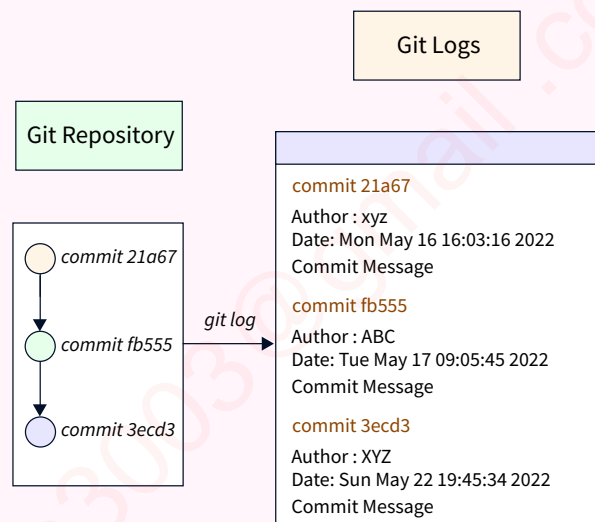

$ git checkout

Switch to another branch and check it out into your working directory.

$ git log

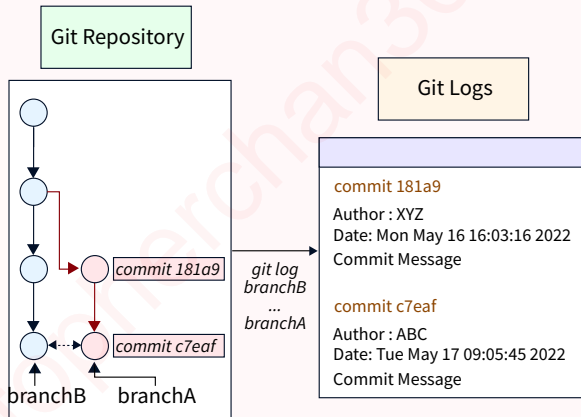Add a file as it looks now to your next commit (stage).

INSPECT & COMPARE

Configuring user information,
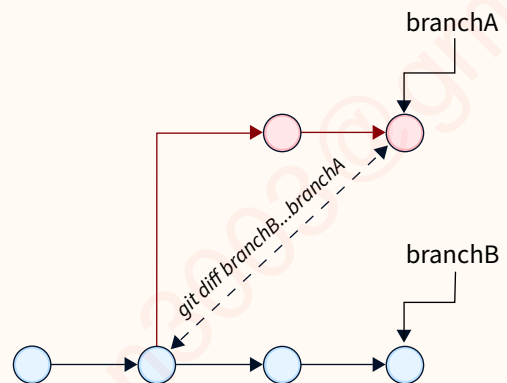initializing and cloning repositories.

$ git log branchB..branchA

Show the commits on branchA that
are not on branchB.

$ git diff branchB...branchA

Show the diff of what is in branchA
that is not in branchB.

$ git log --follow [file]

Show the commits that changed file,
even across renames.

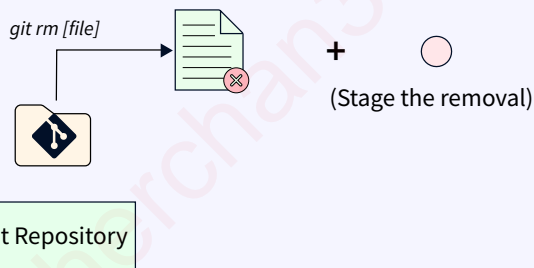## $ git show [SHA]

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 the project and
stage the removal for commit.

(Stage the removal)

Git Repository

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

Change an existing file path and
stage the move.

(Stage the removal)

Git Repository
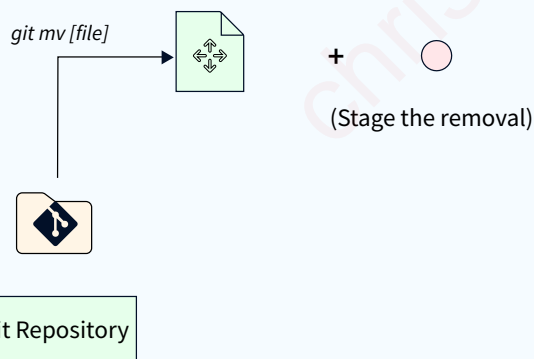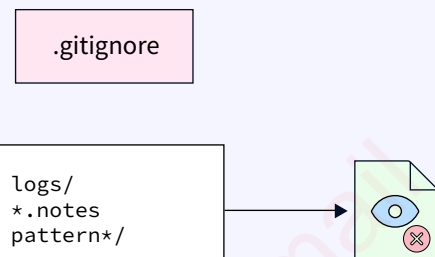
$ git log --stat -M

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.

.gitignore

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

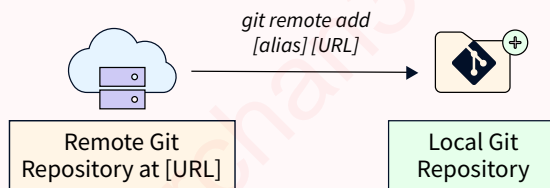### $ git config --global core.excludesfile [file]

System wide ignore pattern for all
local repositories

SHARE & UPDATE

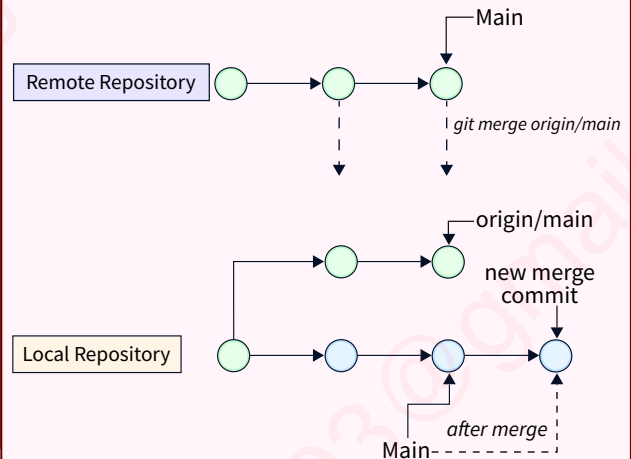Retrieving updates from another repository and updating local repos.

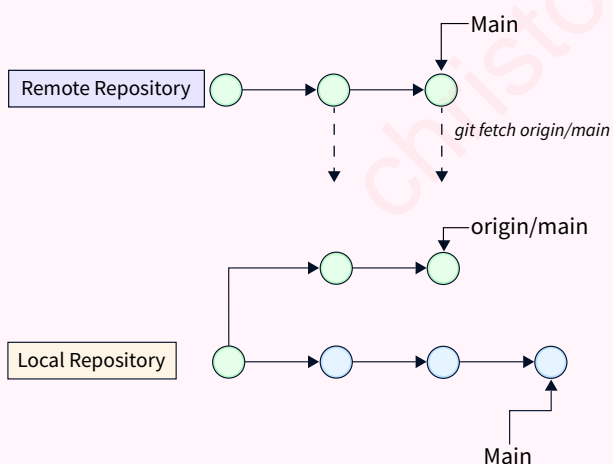## $ git remote add [alias] [url]

Add a git URL as an alias.

Remote Git Repository at [URL]

git remote add [alias] [URL]

Local Git Repository

$ git merge [alias]/[branch]

Merge a remote branch into your current branch to bring it up to date.

Remote Repository — Main

git merge origin/main

Local Repository — origin/main

new merge commit

Main — after merge

$ git fetch [alias]

Fetch down all the branches from that Git remote.

Remote Repository — Main

git fetch origin/main

origin/main

Local Repository

Main

$ git push [alias] [branch]

Transmit local branch commits to the remote repository branch.

new merge commit

remote/main — after merge

Remote Repository

merge

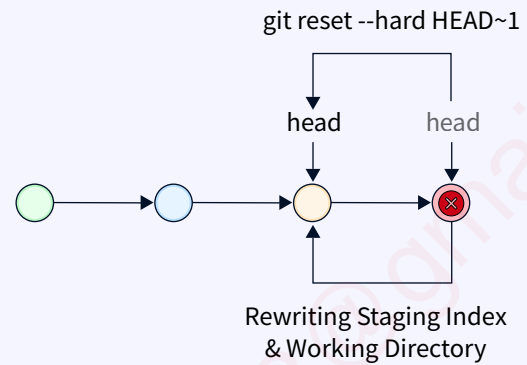git push

Local Repository

local/main
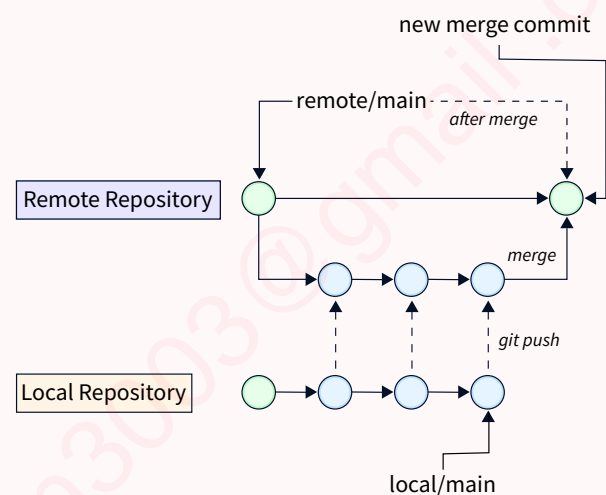
$ git pull

Fetch and merge any commits from the tracking remote branch.
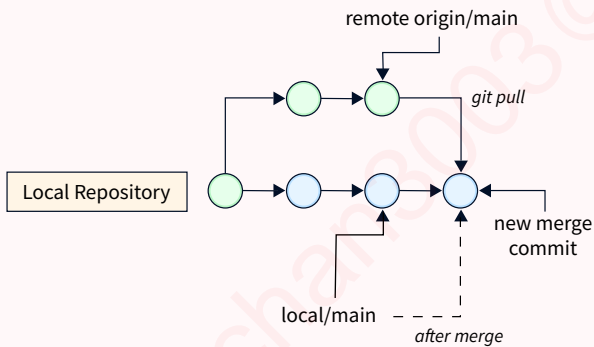
$ git reset --hard [commit]

Clear staging area, rewrite working tree from specified commit.
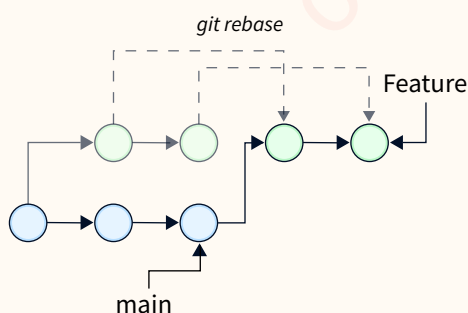

REWRITE HISTORY

Rewriting branches, updating commits and clearing history.

TEMPORARY COMMITS

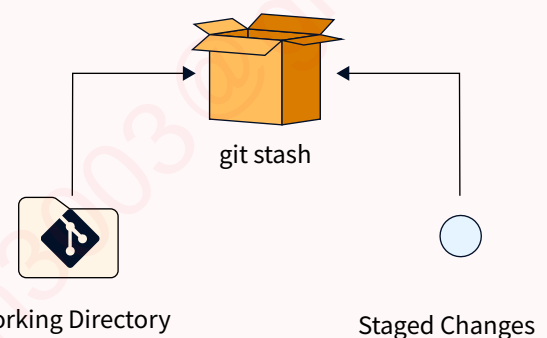Temporarily store modified, tracked files in order to change branches.

$ git rebase [branch]

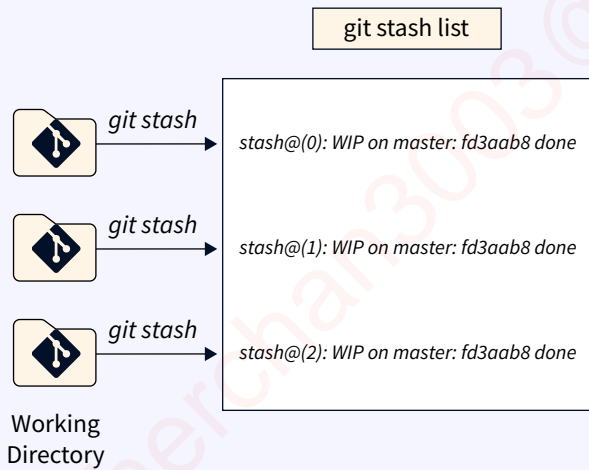Apply any commits of the current branch ahead of specified one.

$ git stash
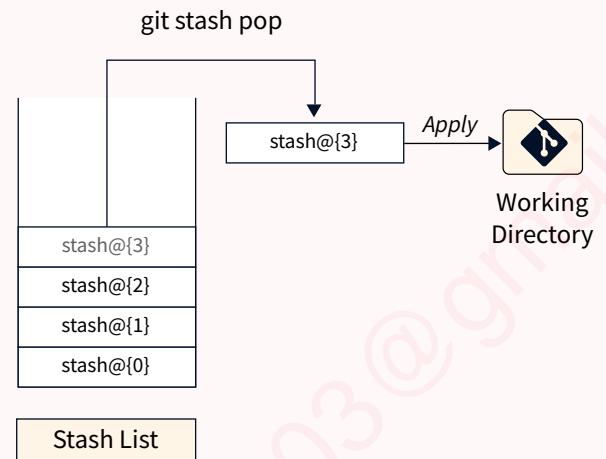
Save modified and staged changes.

$ git stash list

List stack-order of stashed file changes.

git stash list

git stash → stash@(0): WIP on master: fd3aab8 done

git stash → stash@(1): WIP on master: fd3aab8 done

git stash → stash@(2): WIP on master: fd3aab8 done

Working Directory

$ git stash pop

Write working from the top of the stash stack.

git stash pop

stash@{3} → *Apply* → Working Directory

stash@{3}
stash@{2}
stash@{1}
stash@{0}

Stash List

$ git stash drop

Discard the changes from the top of the stash stack.

git stash drop

stash@{3} ⊗
(Discard)

stash@{3}
stash@{2}
stash@{1}
stash@{0}

Stash List