GIT Cheatsheet

www.bongodev.com

Git Essentials

Git: Version Control System

Download: https://git-scm.com

Git Vs GitHub

Desktop	Online

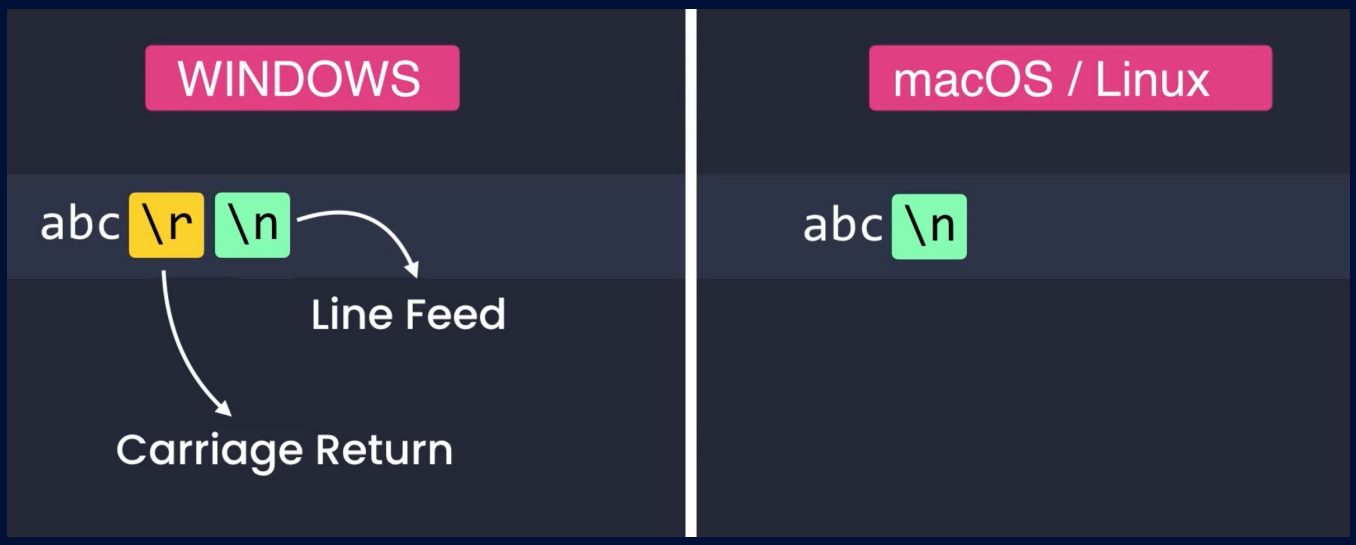
After Installing GIT

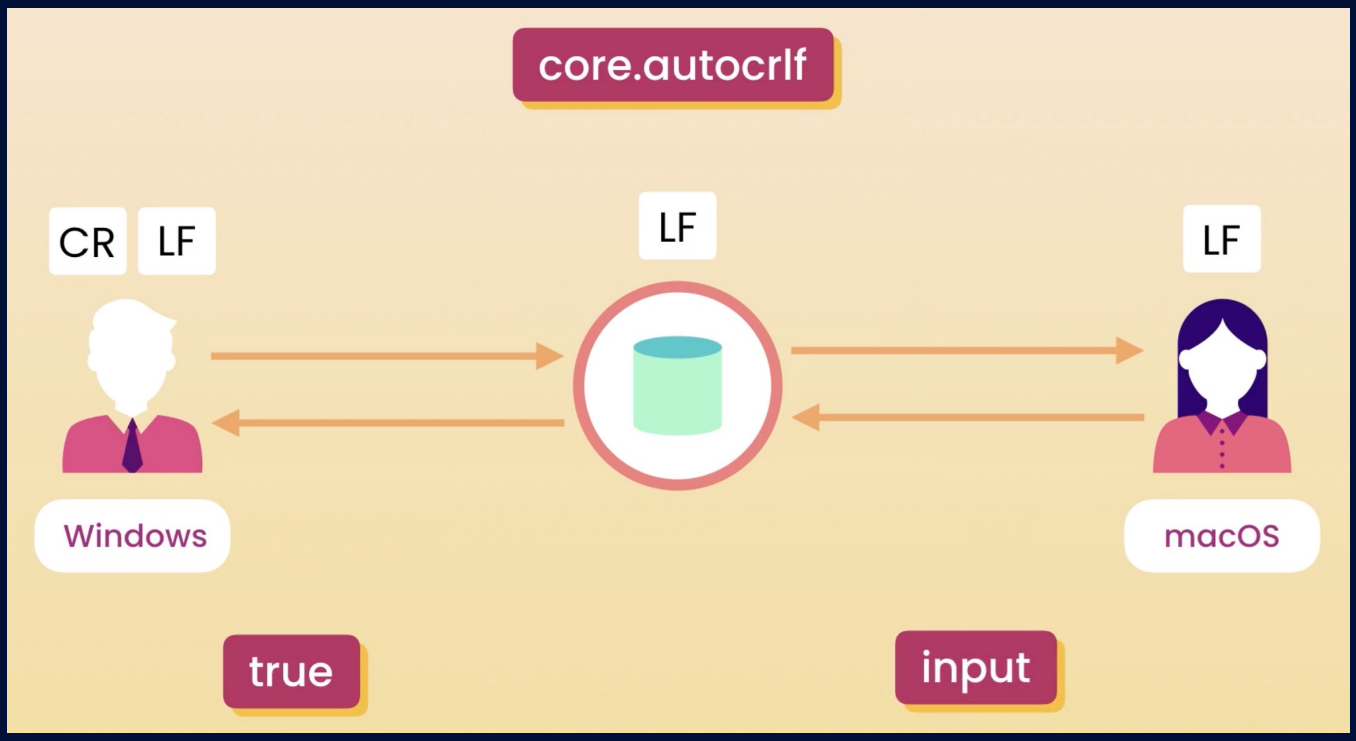
Check: `git --version | git --help`

Configure Git: (*tell git who you are*)

- `git config --global user.name "Nure Siddiq"`
  `#single name > double quote not necessary`
- `git config --global user.email nure@bongodev.com`
- `git config --global user.password ******` `#Not mandatory but having it great`
- `git config --global core.editor "code --wait"` `#Set VS code default to avoid scary VIM Editor, --wait to wait VS code until you close`
- `git config --global core.editor -e` `#Open to edit glogal config file in VS`

core.autocrlf





To fix New Line Issue, run:

```
git config --global core.autocrlf true
                    (#Mac > input, Windows > true)
```

Initializing a Repository

Create a folder: mkdir myfolder

Go to the folder: cd myfolder

Run the command: git status

```
Desktop — -zsh — 80×24
$ git status
fatal: not a git repository (or any of the parent directories): .git
$
```

Initialize Git to the folder/directory: git init

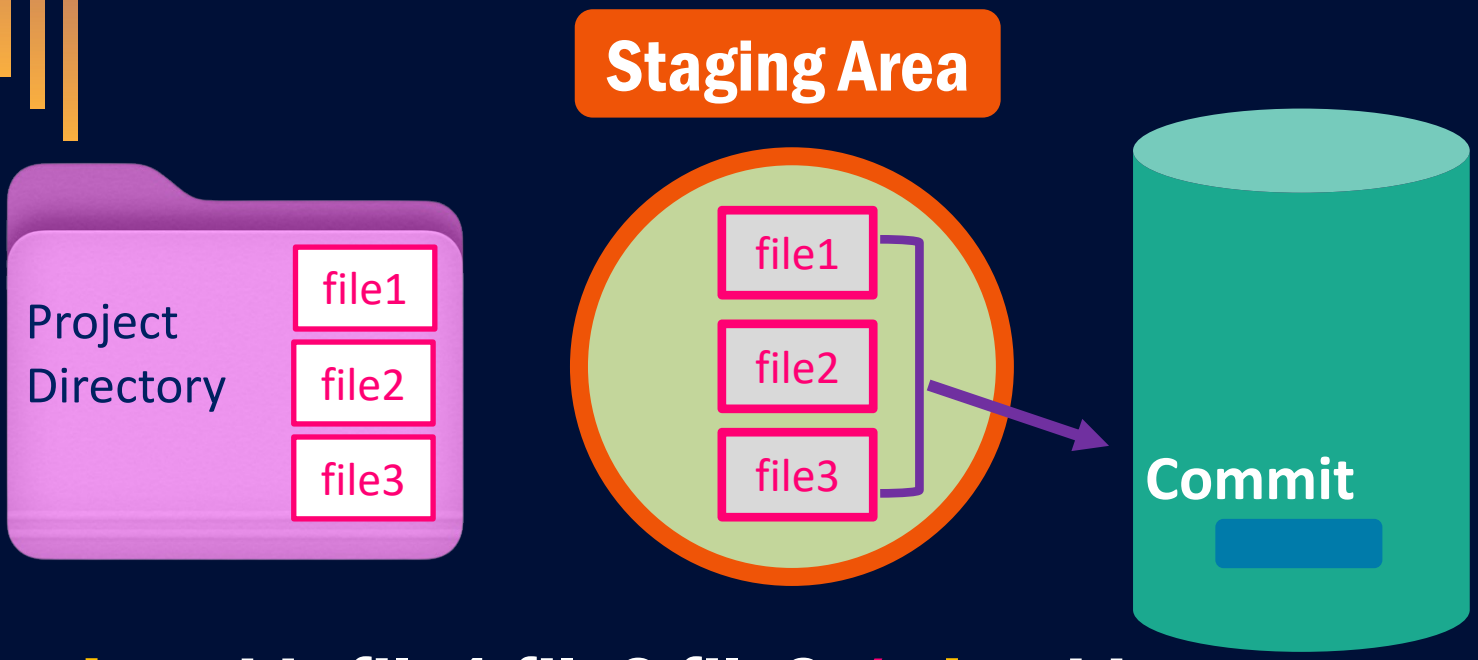
Run the command: git status

```
hello — -zsh — 80×24
$ git init
Initialized empty Git repository in /Users/welcome/Desktop/hello/.git/
$
```

Git Workflow

Hidden sub directory
in our project directory

Commit




```
git add file1 file2 file3 / git add .
git commit –m “first commit”
```

- ID
- Message
- Date/time
- Author
- Full Snapshot

Git Workflow

Without Message in command

```
git commit #will open default editor for commit message
```

View Snapshot/commit history

```
git log #shows full history (latest to first)
```

One line commit

```
git log --oneline #shows short commit history
```

Reverse commit history

```
git log --oneline --reverse #shows first to last
```

View a commit

```
git show ID or git show HEAD~commit_number
```

Unstaging file

```
git restore --staged file_name or dott( .) to unstage all files
```

Branching

```
git branch <branch_name> #create new branch

git switch <branch_name> #switch to new branch

git switch -C <branch_name> #Create & switch to new branch

git branch -D <branch_name> #Delete a branch

git branch # show all branches
```

Merging

```
git log --oneline –all --graph #log details
git switch master #go to master branch always to merge
```

Resolve Conflict During Merging

```
git merge <branch_name> #try to merge merge another branch
Resolve the conflicts first #using code editor
git add <conflict_corrected_file>
git commit –m <message> #new commit
Merge Done!
```

Shortcuts

Creating Snapshots

Initializing a repository

```
git init
```

Staging files

```
git add file1.html          # Stages a single file
git add file1.js file2.js   # Stages multiple files
git add *.html              # Stages files with a pattern
git add .                   # Stages all files
```

View current status

```
git status                  # Full status
git status -s               # Short status
```

Committing the staged files

```
git commit -m "Message"     # Commits with a one-line message
git commit                  # Opens the default editor to type
                            a long message
```

Commit skipping the staging area

```
git commit -am "Message" or
git commit -a -m "Message"
```

Removing files

```
git rm file1.js             # Removes from working directory and staging area
git rm --cached file1.js    # Removes from staging area only
```

Renaming or moving files

```
git mv file1.js file1.txt
```

View the log history

```
git log                    # Full history
git log --oneline git      # Summary
log --reverse              # Oldest to the newest commits log
```

View a commit

```
git show 921a2ff           # Shows the given commit
git show HEAD              # Shows the last commit
git show HEAD~2 git        # Two steps before the last commit
show HEAD:file.html        # Shows the version of file.js stored in
                             the last commit
```

Unstaging files (undo git add)

```
git restore --staged file.js
```

Discarding local changes

```
git restore file.css           # Copies file. from index to working directory
git restore file1.js file2.js  # Restores multiple files in working directory
git restore .                  # Discards all local changes
```

Restoring an earlier version of a file

```
git restore --source=HEAD~2 file.html
```

Checking out a commit

```
git checkout dad47ed       # Checks out the given commit
git checkout master        # Checks out the master branch
```

Undoing commits

```
git reset --soft HEAD^      # Removes the last commit, keeps changed staged
git reset --mixed HEAD^     # Unstages the changes as well
git reset --hard HEAD^      # Discards local changes
```

Reverting commits

```
git revert 72856ea          # Reverts the given commit
git revert HEAD~3           # Reverts the last three commits
```

Amending the last commit

```
git commit --amend
```

Branching & Merging

Managing branches

```
git branch hotfix           # Creates a new branch called hotfix
git checkout hotfix or      # Switches to the hotfix branch
git switch hotfix           # Same as the above
git switch -C hotfix        # Creates and switches
git branch -D hotfix        # Deletes the hotfix branch
```

Merging

```
git merge bugfix            # Merges the bugfix branch into master
git merge --abort           # Aborts the merge
```

Viewing the merged branches

```
git branch --merged         # Shows the merged branches
git branch --no-merged      # Shows the unmerged branches
```

Rebasing

```
git rebase master           # Changes the base of the current branch
```

Collaboration with GitHub & team

Cloning a GitHub repository

```
git clone url
```

Syncing with remotes

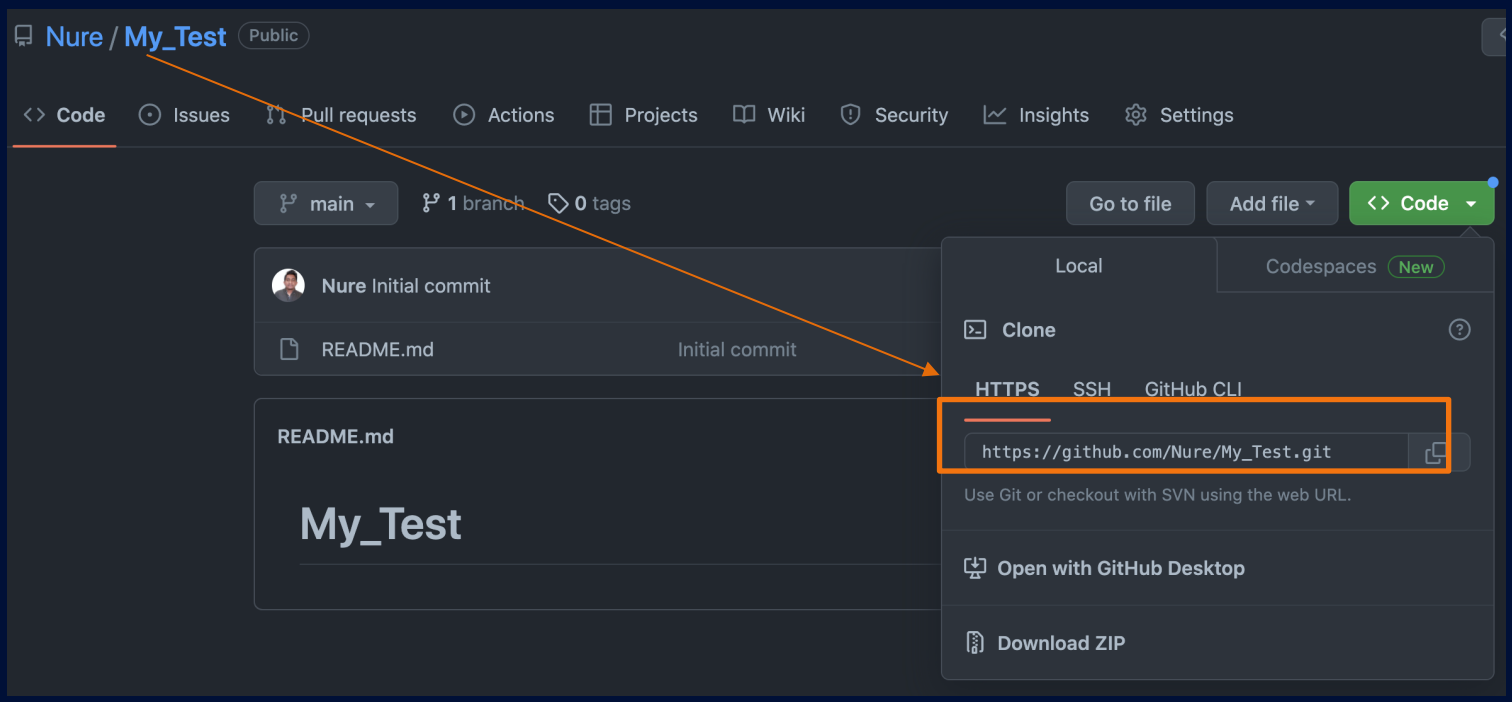
```
git fetch origin master      # Fetches master from origin
git fetch origin             # Fetches all objects from origin
git fetch                    # Shortcut for "git fetch origin"
git pull                     # Fetch + merge
git push origin master       # Pushes master to origin
git push                     # Shortcut for "git push origin master"
git push –u origin hotfix    # Pushes bugfix to origin
```

Managing remotes (GitHub)

```
git remote                       # Shows remote repos
git remote add upstream url      # Adds a new remote called upstream
git remote rm upstream           # Remotes upstream
```

আমার একটা প্রজেক্ট আমার লোকাল কম্পিউটার এ আছে। কিভাবে GitHub এ আপলোড করবো?

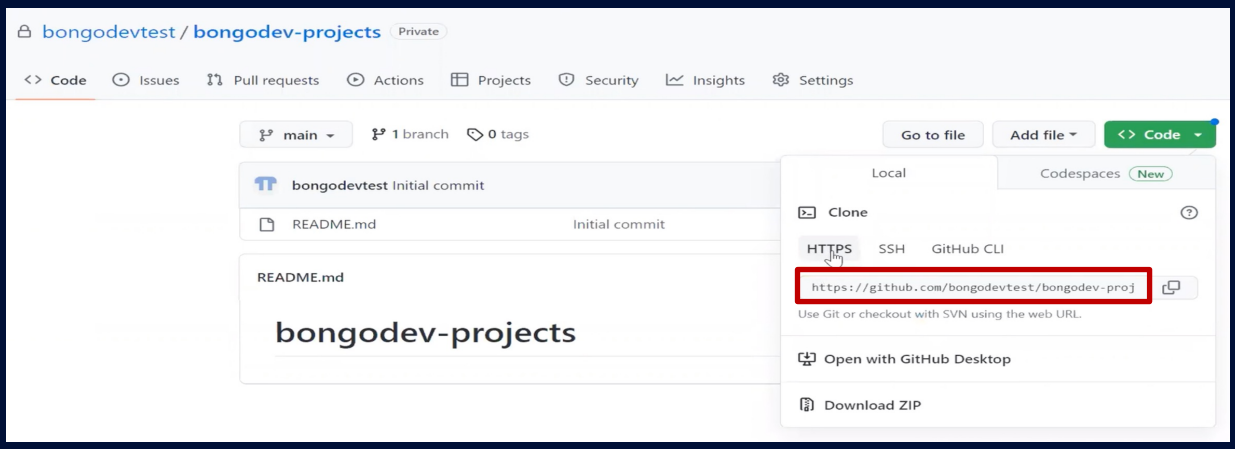
$ Create a repositori in the GitHub first
$ Copy the repository URL



$ Go to your local project folder using terminal, then run the command
$ git remote add origin <URL_copied_from_github>
$ commit the changes (git add . , git commit -am <message>) if required
$ git push origin master

আমার কোম্পানি বা বন্ধুর GitHub এ একটা repository আছে। আমি কিভাবে সেই repository তে collaborate /কাজ করবো?

$ Copy the repository URL



$ Come to your computer's any location, then type
$ git clone <repository_URL>
$ change directory: cd <the new folder name created by git clone>
$ git remote --v

```
$ git remote --v
origin  https://github.com/bongodevtest/bongodev-projects.git (fetch)
origin  https://github.com/bongodevtest/bongodev-projects.git (push)
```

$ create a new branch: git branch <branch_name>
$ switch to the new branch: git switch <branch_name>
$ add files, codes, then commit
$ Finally, git push –u origin <branch_name>

আমি ৫টা আলাদা প্রজেক্টস GitHub এ আপলোড করবো। আমি কি একটা repository তে আপলোড করবো নাকি ৫ টার জন্য আলাদা আলাদা repository তৈরী করে আপলোড করবো?

Please always create new repository and upload projects there to separate your projects. If you upload all projects under a single folder/repository, your projects will not be noticed by employers/others.

Good Luck!