**Configuration Settings:**

Name

Email

Default editor

Line Ending

**Three types of Config Settings:**

1. System: All users
2. Global: All repositories of the current users
3. Local: The current repository

**$ git config - -global user.name “Reema Garg”**

We use double quotes while writing Reema Garg because theres a space between Reema and Garg.

**$ git config - -global user.email** [**reemaaggrawal9@gmail.com**](mailto:reemaaggrawal9@gmail.com)

We do not use the double quotes because theres’s no space between emails.

**To connect the editor (example: MS Visual Code Editor) with the GIT BASH, use the below commands:**

terminal.integrated.shell.windows": "C:\\Program Files\\Git\\git-cmd.exe",

"terminal.integrated.shellArgs.windows": [

"--command=usr/bin/bash.exe",

"-l",

"-i"

]

**$ code**

Shows the editor showing the moving cursor and then redirects automatically to the MS Visual Code Editor.

**$ git config - -global core.editor “code - -wait”**

With the wait flag we tell the terminal window to wait until we close the new VS instance.

**$ git config - -global -e**

This will open our default editor i.e. MS visual code editor, containing all the basic information .

Close the default editor so that we can execute another command on GIT Bash.

**How GIT should handle end of lines?**

On windows: abc \r CARRIAGE RETURN

abc \n LINE FEED

TRUE

If we do not handle the end of lines properly, we are going to end with the issues down the road. To prevent these we have to configure a property called **core.autocrif** which is short for Carriage Return Line Feed.

**$ git config - -global core.autocrlf true**

For windows

**$ git config - -global core.autocrlf input**

For Mc/IOS.

**$ git config - -help**

**$ git config –h**

**$ vi old2.txt**

Opens the visual editor of linux. And we can add the text there. To get out of the editor, press **escape + shift + : + wq**

**$ git add file1.txt file2.txt**

The add command transfers the file from our personal workspace to the STAGING AREA.

To COMMIT the file , we have to first: transfer the file to the STAGING AREA and the COMMIT to the LOCAL REPOSITORY.

**$ git commit –m “Initial commit.”**

This commit command is used to finally transfer the file from the STAGING AREA to the LOCAL REPOSITORY.

This is basically, in simple language, THE INITIAL COMMIT stage.

**$ echo hello > file1.txt**

Echo command is used to create a file eg. file1.txt

So we write hello in file file1.txt

**$ echo hello > file2.txt**

Create another file file2.txt

**$ git status**

**$ git add .** or **$ git add file1.txt file2.txt**

After creating the files, which is in our workspace, we have to transfer this file into the staging area. For that, we used add command to move both the files in to the staging area.

**$ git status**

**$ git commit –m “initial commit”**

This commits the files to the local repository

**$ git status**

**$ echo world >> file1.txt**

This >> is used to append in the file. Using echo, we appended “world” in the file1.txt

(In the file file1.txt, “hello” was already written)

**$ git status**

**$ echo test >> file1.txt**

**$ git status**

Shows the file1.txt in red color, as the changes were made.

**$ git commit - -am “fixed the bug to prevent the users from signing up.”**

The commit - -am command directly copies the file from WORKING SPACE to the LOCAL REPOSITORY.

**$ git status**

Shows that theres nothing to commit, for now. As the files that were already present is committed already.

Now, lets say file2.txt is of no use. Remove that file from the WORKING SPACE.

**$ rm file2.txt**

Doing this, will only remove the file2.txt from the Working space. Now to check whether the file2.xt is still in the STAGING AREA, perform the following command:

**$ git ls-files**

It will list those files which are in STAGING AREA.

So, this shows that the files still exists in the staging area.

So, now to remove the file2.txt from staging area, we will use the add command again.

**$ git add file2.txt**

Now check whether the file is still in the staging area or not?

**$ git ls-files**

The file2.txt is removed from the staging area.

**$ git status**

The deleted file is green in color, shows that it is ready to COMMIT.

**$ git commit –m “removed the unused file2.”**

Shows 1 file changes, 1 file deleted.

**$ git status**

This shows that there’s nothing to commit for now, working tree clean.

To delete the file.txt from both WORKING AREA and the STAGING AREA collectively,following command is used:

**$ git rm file.txt**

Removes the file from both areas. Check using $ ls and $ git ls-files. And perform coomit using $ git comit –m “the unused filetxt is removed.”

**RENAMING OR MOVING FILES:**

We just have only single file file1.txt, for now.

**$ ls**

file1.txt

**$ mv file1.txt main.js**

The ”mv” command is basically used for moving the data.

Also, used for renaming the file1.txt to main.js and moving the files from file1.txt to main.js, altogether.

**$ git status**

Shows the file1.txt and main.js in red color.

Because, we have to add these changes to the STAGING AREA. That the file1.txt is renamed as main.js and all the data is moves in main.js

For that run, add command

**$ git add file1.txt**

Approves that the file1.txt is deleted in the STAGING AREA

**$ git add main.js**

Approves that the file main.js is the new file name in STAGING AREA.

**$ git status**

It shows:

Renamed: file1.txt -> main,js

**To directly move and rename the file name:**

**$ git mv main.js file1.js**

It shows:

Renamed: file1.txt -> file1.js

**$ git commit –m “Refactored code.”**

Now finally committed the changes to the LOCAL REPO.

It shows:

1 file changed, 0 insertions(+), 0 deletions(-)

Means, 1 file is changed from main.js to file1.js

There are no new insertion of the data (like hello, world, test) in the file.

There are no new deletion of any data from the file.

**IGNORING THE USELESS FILES:**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ mkdir logs**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ echo hello > logs/dev.log**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git ls-files**

file1.js

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 8 commits.

(use "git push" to publish your local commits)

Untracked files:

(use "git add <file>..." to include in what will be committed)

logs/

nothing added to commit but untracked files present (use "git add" to track)

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ echo logs/ > .gitignore**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ code .gitignore**

After we press(code .gitignore), the visual editor opens and mention the files that we don’t want to see and the files that we want to ignore.

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 8 commits.

(use "git push" to publish your local commits)

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore

nothing added to commit but untracked files present (use "git add" to track)

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git add .gitignore**

warning: LF will be replaced by CRLF in .gitignore.

The file will have its original line endings in your working directory

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 8 commits.

(use "git push" to publish your local commits)

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: .gitignore

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git commit -m "Add gitignore."**

[master 03a28a9] Add gitignore.

1 file changed, 3 insertions(+)

create mode 100644 .gitignore

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 9 commits.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ ls**

file1.js logs/

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git ls-files**

.gitignore

file1.js

**made the another directory to understand how to ignore such file, permanently:**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ mkdir bin**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ echo hello > bin/app.bin**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 9 commits.

(use "git push" to publish your local commits)

Untracked files:

(use "git add <file>..." to include in what will be committed)

bin/

nothing added to commit but untracked files present (use "git add" to track)

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git add .**

warning: LF will be replaced by CRLF in bin/app.bin.

The file will have its original line endings in your working directory

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 9 commits.

(use "git push" to publish your local commits)

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: bin/app.bin

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git commit -m "add bin."**

[master cd011d8] add bin.

1 file changed, 1 insertion(+)

create mode 100644 bin/app.bin

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ code .gitignore**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 10 commits.

(use "git push" to publish your local commits)

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git add .**

warning: LF will be replaced by CRLF in .gitignore.

The file will have its original line endings in your working directory

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 10 commits.

(use "git push" to publish your local commits)

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: .gitignore

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git commit -m "include bin/ in gitignire."**

[master aa4a79d] include bin/ in gitignire.

1 file changed, 2 insertions(+), 1 deletion(-)

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ echo helloworld > bin/app.bin**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 11 commits.

(use "git push" to publish your local commits)

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: bin/app.bin

no changes added to commit (use "git add" and/or "git commit -a")

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git ls-files**

.gitignore

bin/app.bin

file1.js

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git rm -h**

usage: git rm [<options>] [--] <file>...

-n, --dry-run dry run

-q, --quiet do not list removed files

--cached only remove from the index

-f, --force override the up-to-date check

-r allow recursive removal

--ignore-unmatch exit with a zero status even if nothing matched

--pathspec-from-file <file>

read pathspec from file

--pathspec-file-nul with --pathspec-from-file, pathspec elements are separated with NUL character

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git rm --cached bin/**

fatal: not removing 'bin/' recursively without -r

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git rm --cached -r bin/**

rm 'bin/app.bin'

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ ls**

bin/ file1.js logs/

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git ls-files**

.gitignore

file1.js

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 11 commits.

(use "git push" to publish your local commits)

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

deleted: bin/app.bin

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git commit -m "removed the bin directory that was accidently created."**

[master e2be786] removed the bin directory that was accidently created.

1 file changed, 1 deletion(-)

delete mode 100644 bin/app.bin

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 12 commits.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ echo test > bin/app.bin**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 12 commits.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git ls-files**

.gitignore

file1.js

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ echo test > logs/dev.log**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git status**

On branch master

Your branch is ahead of 'origin/master' by 12 commits.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

**To directly see the git status:**

**$ git status –s**

M file1.js

A file2.js

?? file3.js

M shows that the file1.js is modified (in the staging area)

A shows that file2.js is Added (in the staging area)

?? shows that file3.js is untracked (means still in your personal workspace, need to add in Staging area)

**Viewing the STAGED and UNSTAGED changes:**

**$ git diff - -staged**

diff --git a/file1.js b/file1.js

index badfb70..8636dbe 100644

--- a/file1.js

+++ b/file1.js

@@ -1,3 +1,5 @@

-hello

+hello world

world

test

+sky

+ocean

diff --git a/file2.js b/file2.js

new file mode 100644

index 0000000..f5e95e7

--- /dev/null

+++ b/file2.js

@@ -0,0 +1 @@

+sky

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

$ git config --global diff.tool vscode

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

$ git config --global difftool.vscode.cmd "code --wait --diff $LOCAL $REMOTE"

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git config --global -e**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git difftool**

ADMIN@DESKTOP-FLP7UME MINGW64 ~/Projects/Moon (master)

**$ git difftool --staged**

Viewing (1/2): 'file1.js'

Launch 'vscode' [Y/n]? Y

Viewing (2/2): 'file2.js'

Launch 'vscode' [Y/n]? n

This shows the visual editor showing the changes in file1.js

TO CHECK THE LIST OF FILES THAT ARE COMMITTED:

**$ git log**

**$ git log - -oneline**

**$ git log - -oneline - -reverse**

TO RESTORE THE CHANGES DONE/ TO RESTORE THE FILE IN STAGING AREA:

**$ git restore - -staged file1.txt**

TO UNDO THE LOCAL CHANGES OR CLEAN THE FILES THAT ARE IN OUR WORKING SPACE AND WE DONOT WANT THEM TO ADD TO THE STAGING AREA:

**$ git clean –fd**

**TO RECOVER THE DELETED FILE:**

**$ git restore - -source=HEAD~1 file1.js**