Git notes

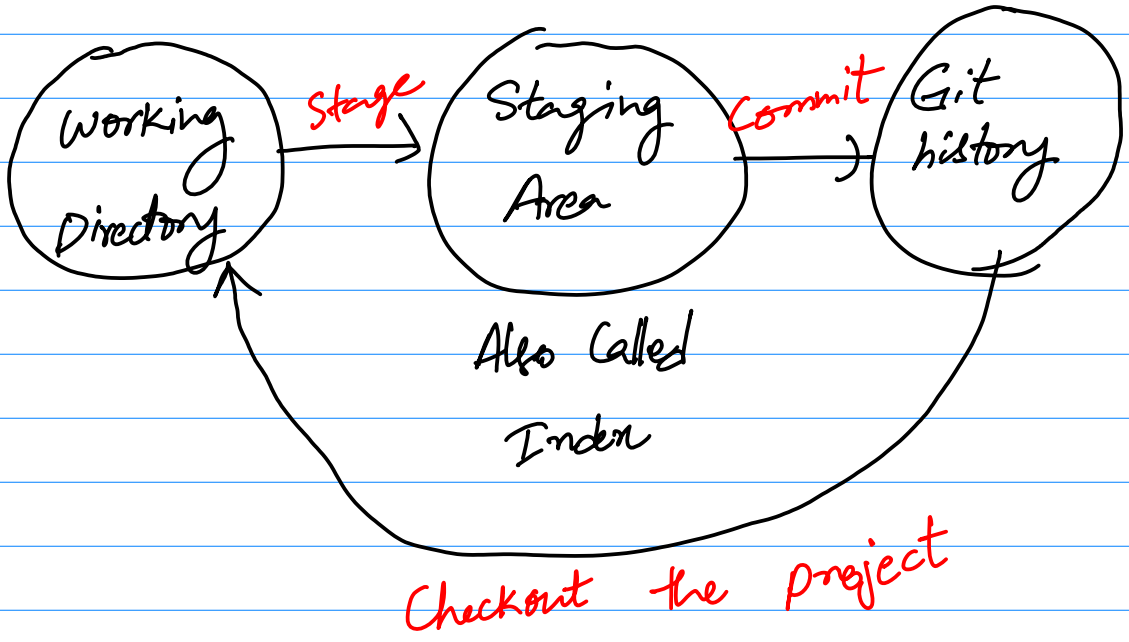
Created in 2005

For each version (commit) git keeps a snapshot of all the files in the corresponding repo

---

SHA-1 Checksums — 40 hexa decimal digits
Git keeps checksums of the files. This is how it tracks changes in the files.

The 3 states



1 Git Setup

`git config`

```
git config --global user.name "Rahul"
git config --global user.email "rahul.pyd@gmail"

git config --global core.editor vim
```

Checking your setting.

```
git config --list
```

Getting help

```
git help <verb>
git <verb> --help
man git-<verb>
```

② Basic git Commands

```
git init

git add <file names>

git commit -m "<message>"

git clone <repo path>

git clone http://.....git

git clone git://....git
```

→ this uses SSH

```
git status
git status -s (or --short)
```

.gitignore

Standard glob Patterns

/Todo — avoid recursitivity
Todo/ — treat Todo as a directory
!Todo — exceptions

what is changed in the working dir. but not yet staged
git diff.

What changes are staged
git diff --staged or
git diff --cached

Skipping staging before committing

git commit -a -m "<msg>"

This will automatically stage all the unstaged changes

Removing files

rm PROJECT.md

git rm PROJECT.md

This stages the file removal. The file is no more tracked.
If the file is already staged use -f

git rm -f PROJECT.md

I need to try this

git rm --cached README
We want the file to be the working directory but do not want to be tracked. We however have staged the file.

Moving files

```
git mv file_from file_to
```

The above is same as

```
mv file_from file_to
git rm file_from
git add file_to
```

(3) History

```
git log

git log -p -2        Show diff of the last 2 commits

git log --stat

git log --pretty=oneline

git log --pretty="format: %h %s"

git log --pretty="format: %h %s" --graph
```

Filtering Logs

```
git log --since=2.weeks
git log --since="2 weeks ago"
git log --since="2008-01-15"

git log -S <some string>
```

filters

```
-n
--since , --after
--until , --before
--author
--committer
--grep
-S
```

④ Undoing things

```
git commit --amend
```

This lets us do 2 things

① Make changes to the commit contents

② Make changes to the commit message

Unstaging staged changes

```
git reset HEAD <file>
```

git status command readily provides us the above command

Note: git reset --hard .... is a unsafe command

Undo changes in a modified file

```
git checkout -- <file>
```

Again git status command provides us the above command

Note: The changes lost from the above command are LOST FOREVER

There are some other methods to the same effect as above method but with an option to recover

① Stashing

② Branching

A note on data recovery: Anything that is committed can almost always be recovered. Eg. Deleted branches or commits that are overwritten with --amend

Anything we lose that was not committed can almost never be recovered.

(5) Working with remote repositories

How to write a good commit message

1. Separate subject from body with a blank

Sometimes the body is so simple that body is not required

2. Limit subject line to 50 chars

3. Capitalize the subject line

4. Do not end the subject line with a period

5. Use imperative mood in subject line

Remove deprecated methods ✓

Removed deprecated methods ✗

6. Wrap body at 72 chars

7. Use body to explain what and why vs how