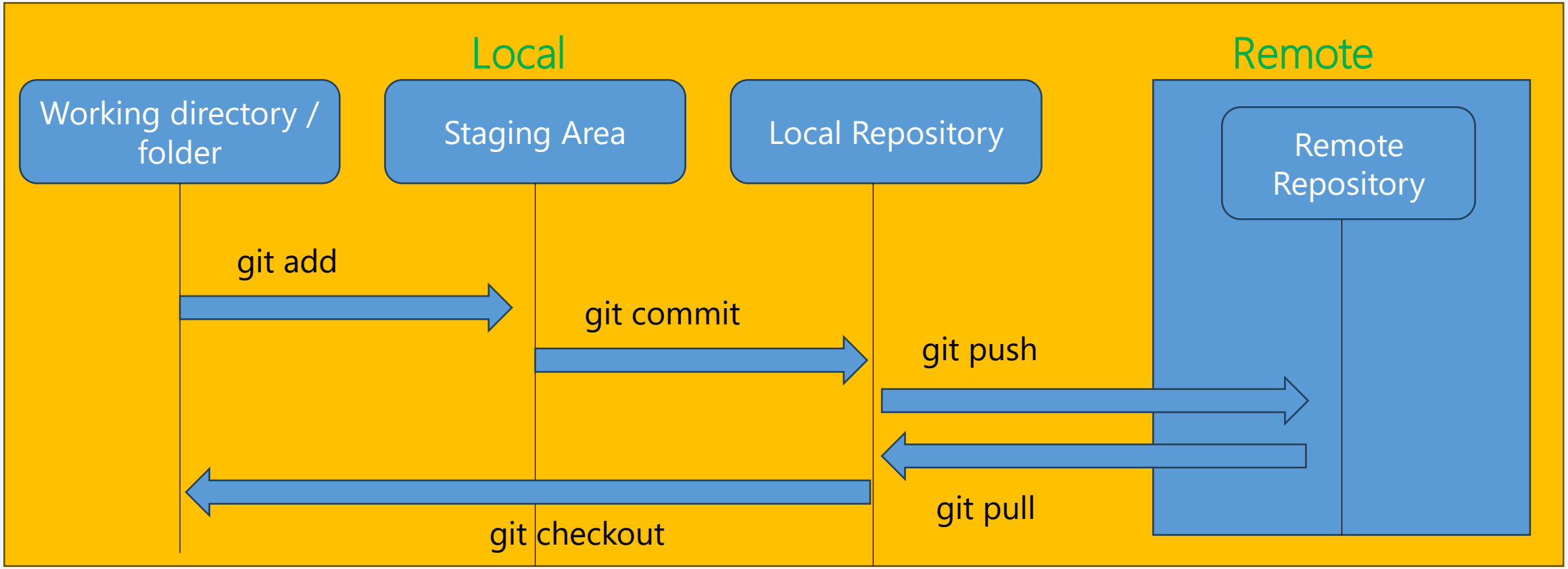
Welcome! This is GIT Cheat Sheet!

Want hands on for GIT just go through these slides!

GIT OVERVIEW



GIT internal components are managed by Plumbing commands while Porcelain are mostly user friendly commands which are used day to day.

GIT COMMANDS

1. **Initialize a GIT repository => git init (not creates the hidden .git folder which we can use for configuration purpose)**

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Test
$ git status
fatal: not a git repository (or any of the parent directories): .git

vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Test
$ git init
Initialized empty Git repository in C:/Users/vinay/Documents/Vinay Interview Prep/Test/.git/

vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Test (master)
$ ls -a
./  ../  .git/
```

2. **Clone repository present on github/bitbucket => git clone https://github.com/vinaykelkar/GIT_From_Scratch.git**

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Test (master)
$ git clone https://github.com/vinaykelkar/GIT_From_Scratch.git
Cloning into 'GIT_From_Scratch'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 12 (delta 2), reused 8 (delta 2), pack-reused 0
Receiving objects: 100% (12/12), 467.26 KiB | 1.28 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

GIT COMMANDS

1. **Setting up username and password in your .git/config file => git config –global user.name “username”**

2. **Checking what all configuration is set => git config --list**

```
linuxmint@linuxmint21:~/Desktop/Devops/git/GIT_From_Scratch/.git$ git config --global user.name "vinaykVMWare"
linuxmint@linuxmint21:~/Desktop/Devops/git/GIT_From_Scratch/.git$ git config --list
credential.helper=store
user.name=vinaykVMWare
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=https://github.com/vinaykelkar/GIT_From_Scratch.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.main.remote=origin
branch.main.merge=refs/heads/main
```

GIT COMMANDS

1. **Check which branch you are on => git branch**

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/GIT_From_Scratch (main)
$ git branch
  basics
* main
```

2. **Check which remote repository you are going to push and the pull the data from along with their URLs => git remote –v**

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/
GIT_From_Scratch/.git (GIT_DIR!)
$ git remote -v
origin  https://github.com/vinaykelkar/GIT_From_Scratch.git (fetch)
origin  https://github.com/vinaykelkar/GIT_From_Scratch.git (push)
```

GIT COMMANDS

1. **Pull Data from the main branch in your local repo => git pull origin**

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/GIT_From_Scratch (main)
$ git pull origin
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 645 bytes | 37.00 KiB/s, done.
From https://github.com/vinaykelkar/GIT_From_Scratch
   d334f06..0a4b17e  main       -> origin/main
Updating d334f06..0a4b17e
Fast-forward
 Understanding Git/Check History of The project.rtf | 70173 ++++++++++++++++
 Understanding Git/Git stash.txt                    |    15 +
 .../Link Local Project to GITHUB Repo.txt          |    22 +
 Understanding Git/Setting Up GIT.rtf               | 21108 +++++
 Understanding Git/Tracking Changes Via Example.rtf | 81862 ++++++++++++++++++++
 5 files changed, 173180 insertions(+)
 create mode 100644 Understanding Git/Check History of The project.rtf
 create mode 100644 Understanding Git/Git stash.txt
 create mode 100644 Understanding Git/Link Local Project to GITHUB Repo.txt
 create mode 100644 Understanding Git/Setting Up GIT.rtf
 create mode 100644 Understanding Git/Tracking Changes Via Example.rtf
```

2. **Create branch and checkout to that particular branch => git branch <branchname> ; git checkout <branchname>**

GIT COMMANDS

1. View untracked files => git status

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/GIT_From_Scratch (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        GIT Commands/

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

2. Add untracked files into staging area in order to push them to further to remote repo => git add . OR git add <file name>

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/GIT_From_Scratch (gitPPT)
$ git add .

vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/GIT_From_Scratch (gitPPT)
$ git status
On branch gitPPT
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   GIT Commands/GIT Simplified.pdf
```

GIT COMMANDS

1. **Commit the staging area changes to local repository => git commit –m “Type down meaningful message”**

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/GIT_From_Scratch (gitPPT)
$ git commit -m "Adding git cheatsheet version 1"
[gitPPT 68a2ae8] Adding git cheatsheet version 1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 GIT Commands/GIT Simplified.pdf
```

2. **Push the commit to remote repository via the branch you created => git push –u origin <branchname>**

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/GIT_From_Scratch (gitPPT)
$ git push -u origin gitPPT
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 115.28 KiB | 19.21 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'gitPPT' on GitHub by visiting:
remote:      https://github.com/vinaykelkar/GIT_From_Scratch/pull/new/gitPPT
remote:
To https://github.com/vinaykelkar/GIT_From_Scratch.git
 * [new branch]      gitPPT -> gitPPT
branch 'gitPPT' set up to track 'origin/gitPPT'.
```

GIT Branch

1. **Check list of all remote branches available in the github repository => git branch --remote**

```
linuxmint@linuxmint21:~/Desktop/Devops/git/GIT_From_Scratch$ git branch --remote
  origin/HEAD -> origin/main
  origin/basics
  origin/gitPPT
  origin/main
linuxmint@linuxmint21:~/Desktop/Devops/git/GIT_From_Scratch$
```

2. **Delete the local branch once the work is done => git branch –d <branchname> OR git branch –D <branchname> (This is forceful deletion)**

```
linuxmint@linuxmint21:~/Desktop/Devops/git/GIT_From_Scratch$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
linuxmint@linuxmint21:~/Desktop/Devops/git/GIT_From_Scratch$ git branch -d basics
warning: deleting branch 'basics' that has been merged to
         'refs/remotes/origin/basics', but not yet merged to HEAD.
Deleted branch basics (was 78ee0db).
```

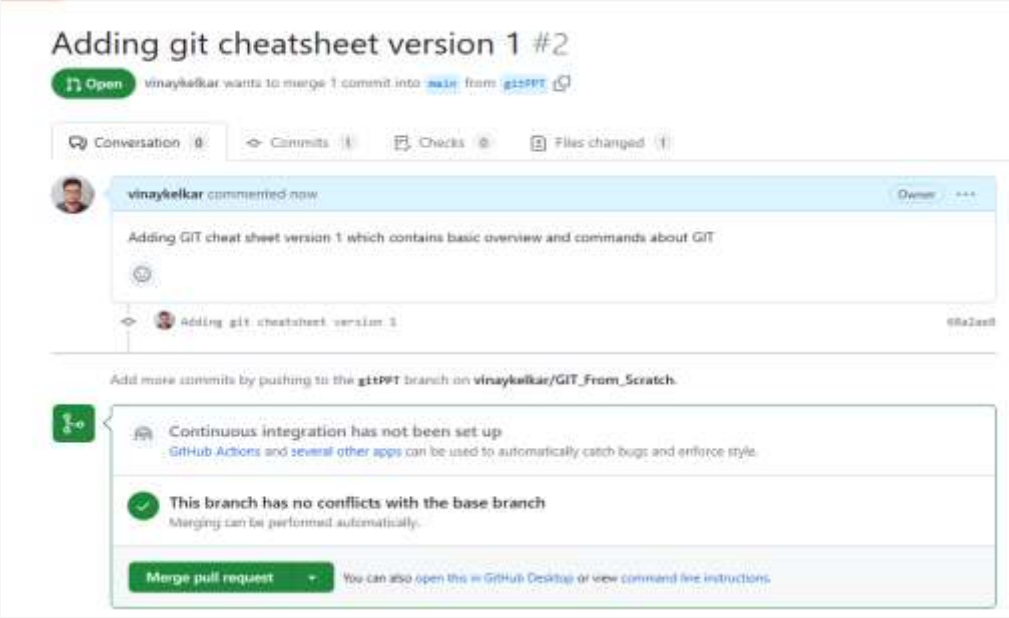
3. **Delete remote branch once the work is done => git push --delete origin <branchname>**

```
linuxmint@linuxmint21:~/Desktop/Devops/git/GIT_From_Scratch$ git branch
* main
  test
linuxmint@linuxmint21:~/Desktop/Devops/git/GIT_From_Scratch$ git push --delete origin test
To https://github.com/vinaykelkar/GIT_From_Scratch.git
 - [deleted]         test
```

GIT BRANCH and PULL REQUEST

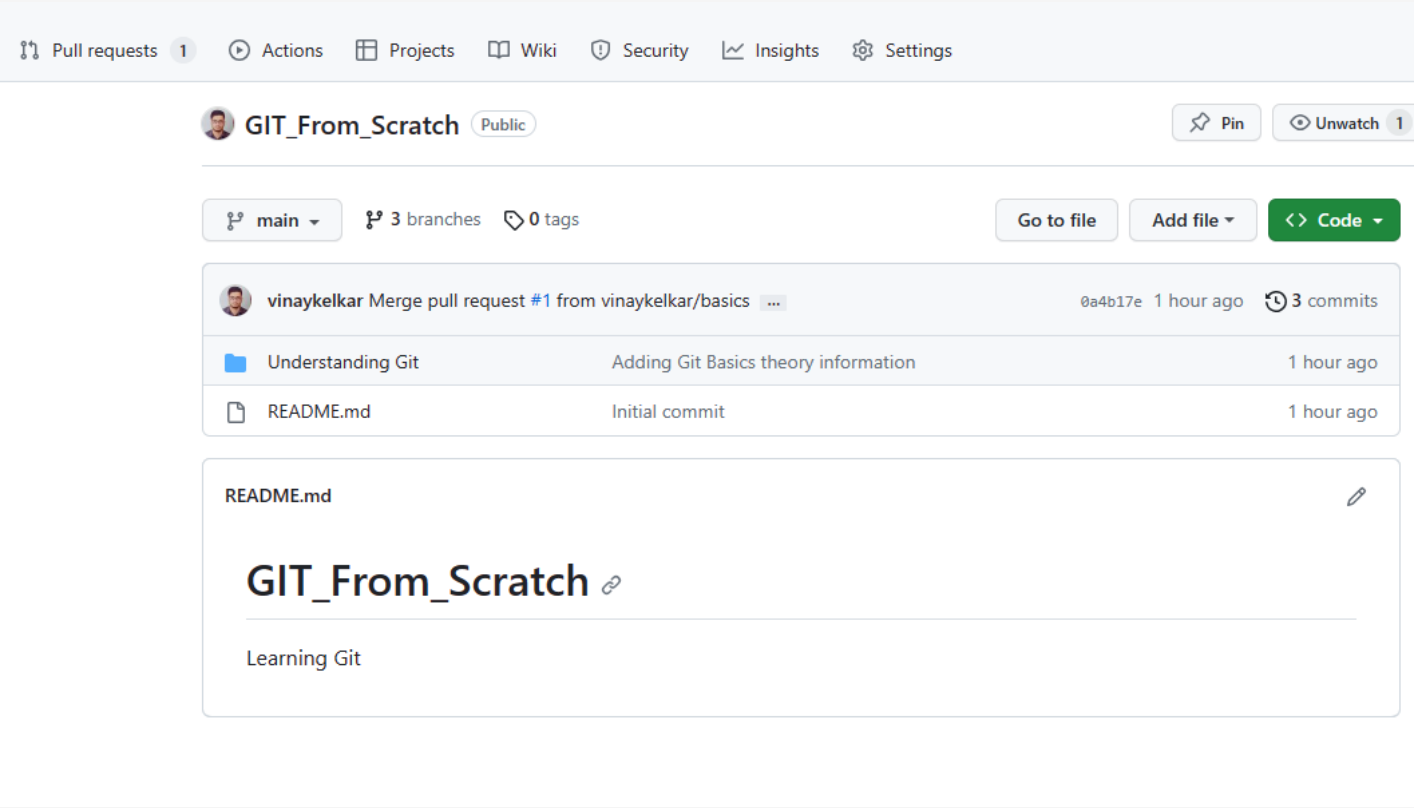
We create branch every time we need to push or work on our own individual changes. Imagine in the production environment everyone starts pushing the code to *main* branch?

It will be a mess, because CI/CD is enabled on the main/master branch!! Hence, always create a branch and push the code/data via that branch to your remote repository. Once you have pushed the code, create a PR (Pull Request), which will later be verified by the owner of the repository and if everything is fine, it will be merged with *main* branch.



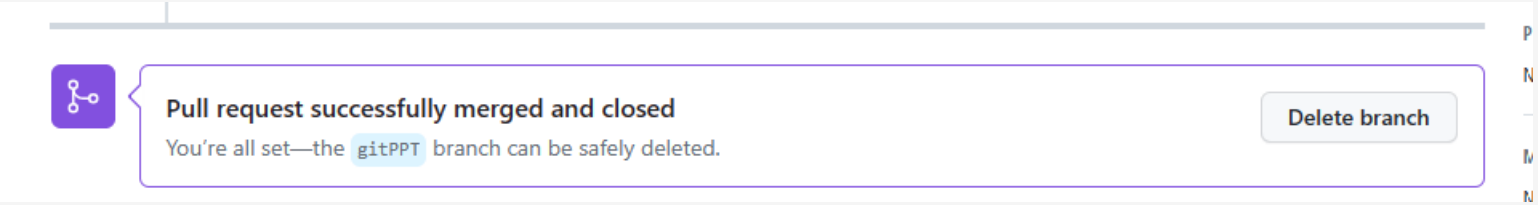
GIT BRANCH and PULL REQUEST

Once the Pull request (PR) is submitted. The owner of the repo or code reviewer, verifies and accepts the request to merge PR. As you can see on top left hand corner, Pull request is currently showing as 1.

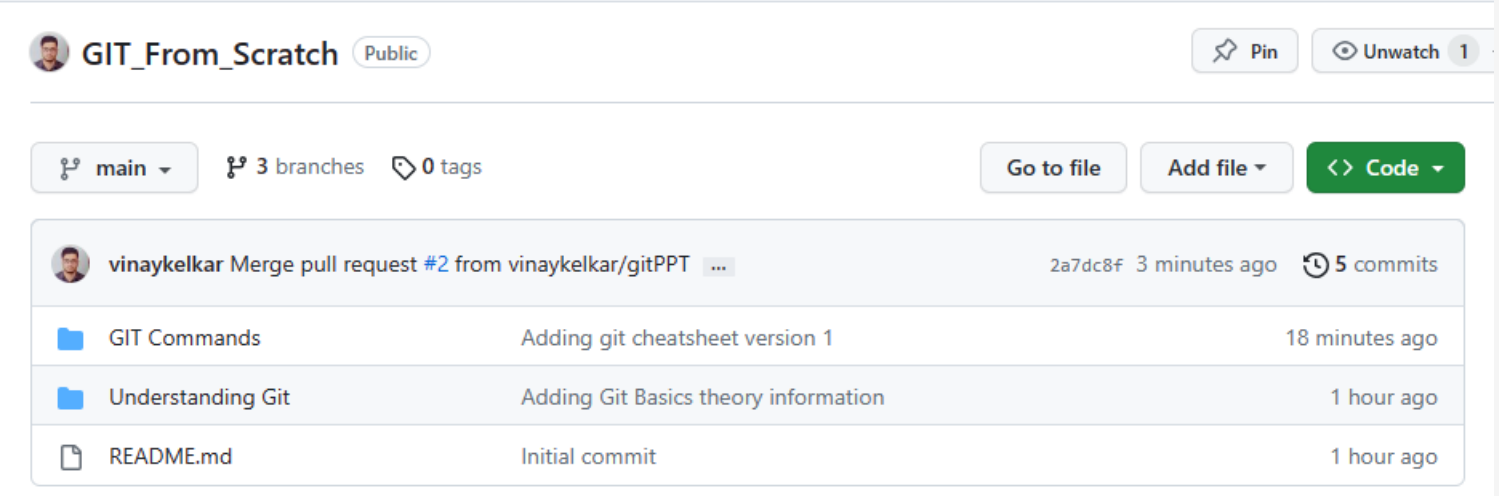


GIT BRANCH and PULL REQUEST

Once the owner/code reviewer says all good, the branch and data under is merged with main branch.



As you can see below now, the code/data is added and we see the new folder "Git commands" which was not present as per previous slide.



GIT Log and Reset

1. Check list of all commits done/performed in the repository => git log OR git log –oneline

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/GIT_From_Scratch (main)
$ git log --oneline
123301c (HEAD -> main, origin/main, origin/HEAD) Merge pull request #4 from vinaykelkar/test
aa19148 adding temp file for testing
5991bc4 Merge pull request #3 from vinaykelkar/basics
78ee0db (origin/basics) adding test file for testing purpose
2a7dc8f Merge pull request #2 from vinaykelkar/gitPPT
68a2ae8 (origin/gitPPT, gitPPT) Adding git cheatsheet version 1
0a4b17e Merge pull request #1 from vinaykelkar/basics
19eebf1 (basics) Adding Git Basics theory information
d334f06 Initial commit
```

git log will show you detailed description like time, username, message passed with commit and complete hashid

2. If you want to go back to particular commit, you can use => git reset <hashid>

```
vinay@DESKTOP-Q1C8F9A MINGW64 ~/Documents/Vinay Interview Prep/Git from Scratch/GIT_From_Scratch (main)
$ git reset 123301caefd24f21e4d1985e2b0ec43705f2c072
```

3. If you want to check every reference that git repo has starting from clone (commits, branch switches) => git reflog