

**Name:** Shreyash Shete

**Roll No:** 322056

**Batch:** B4

**PRN No:** 22010439

### **AMD Practical No 5**

#### **AIM:**

**Installing Git, First-Time Git Setup, Getting a Git Repository. Write a study write up mentioning various git commands.**

#### **Theory:**

Git is a DevOps tool used for source code management. It is a free and opensource version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development. Linus Torvalds created Git in 2005 for the development of the Linux kernel.

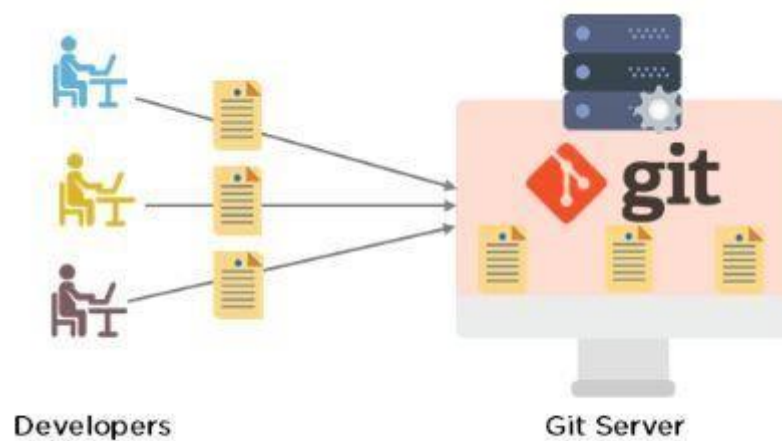
What is Git ?

Git is a version control system used for tracking changes in computer files. It is generally used for source code management in software development.

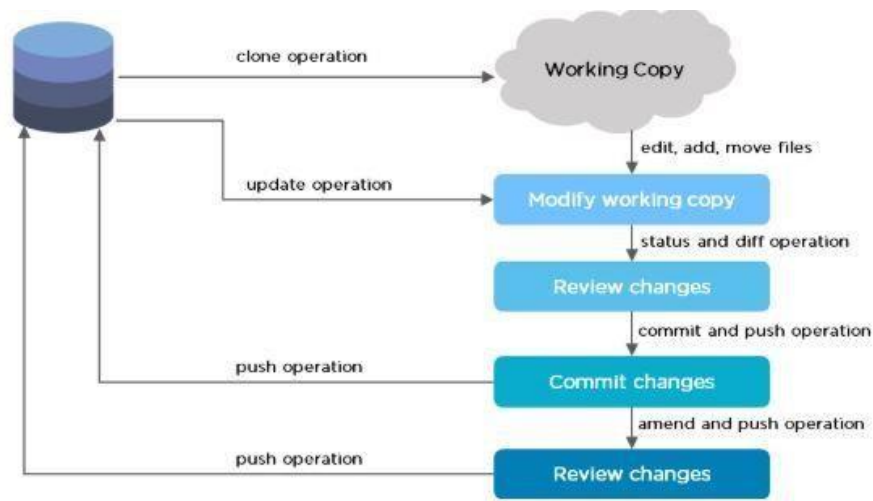
- Git is used to tracking changes in the source code
- The distributed version control tool is used for source code management
- It allows multiple developers to work together
- It supports non-linear development through its thousands of parallel branches

## Features of Git :

- Tracks history
- Free and open source
- Supports non-linear development
- Creates backups
- Scalable
- Supports collaboration
- Branching is easier
- Distributed development




## Git Workflow



The Git workflow is divided into three states:

- Working directory - Modify files in your working directory
- Staging area (Index) - Stage the files and add snapshots of them to your staging area
- Git directory (Repository) - Perform a commit that stores the snapshots permanently to your Git directory. Checkout any existing version, make changes, stage them, and commit.

**Git setup:**

 **git** --everything-is-local

Search entire site...

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

## Download for Windows

[Click here to download](#) the latest (**2.39.2**) **64-bit** version of **Git for Windows**. This is the most recent **maintained** build. It was released **23 days ago**, on 2023-02-14.

### Other Git for Windows downloads

**Standalone Installer**  
[32-bit Git for Windows Setup](#).  
[64-bit Git for Windows Setup](#).

**Portable ("thumbdrive edition")**  
[32-bit Git for Windows Portable](#).  
[64-bit Git for Windows Portable](#).

**Using winget tool**  
Install winget tool if you don't already have it, then type this command in command prompt or Powershell.  

```
winget install --id Git.Git --source winget
```

  
The current source code release is version **2.39.2**. If you want the newer version, you can build it from [the source code](#).

### Now What?

Git 2.39.2 Setup

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

reflect on the original authors' reputations.

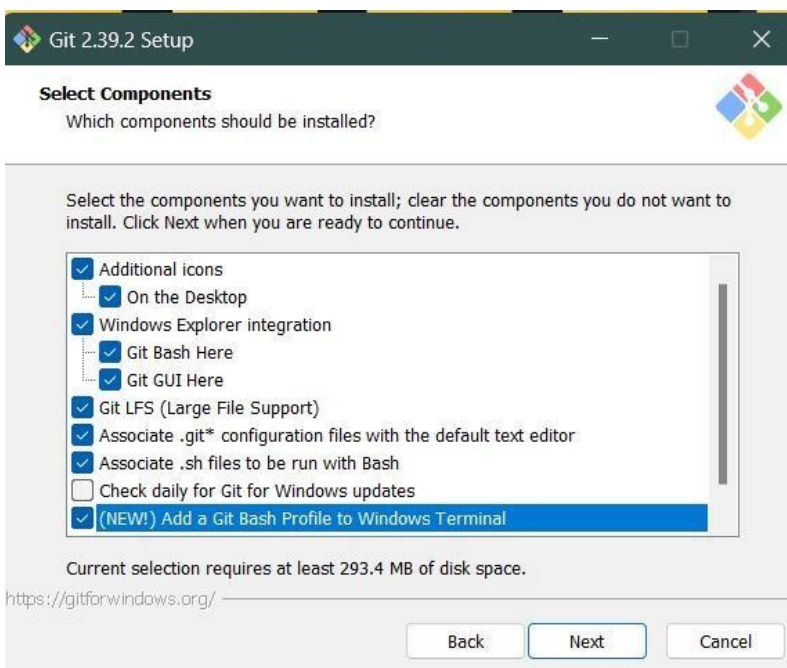
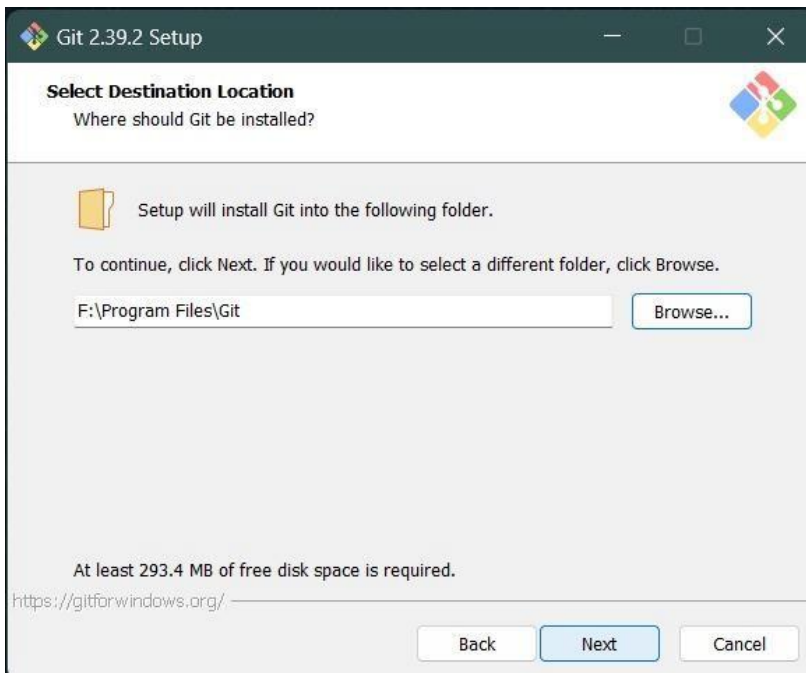
Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

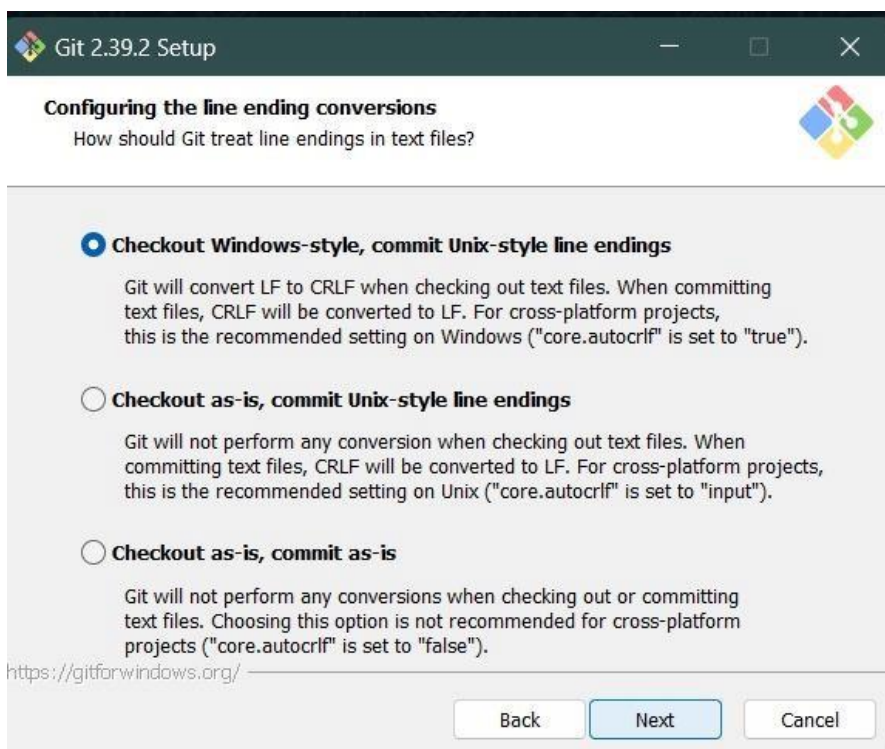
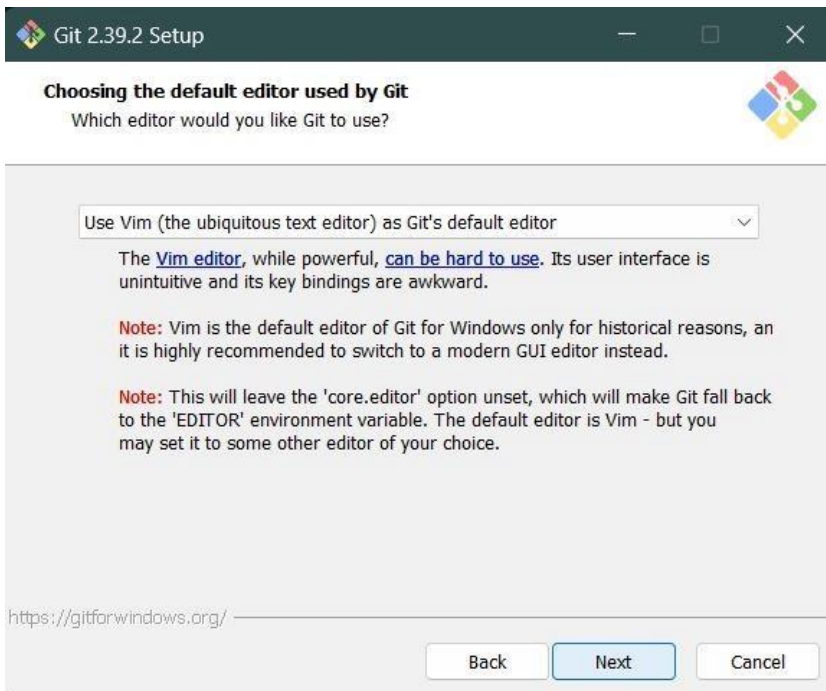
The precise terms and conditions for copying, distribution and modification follow.

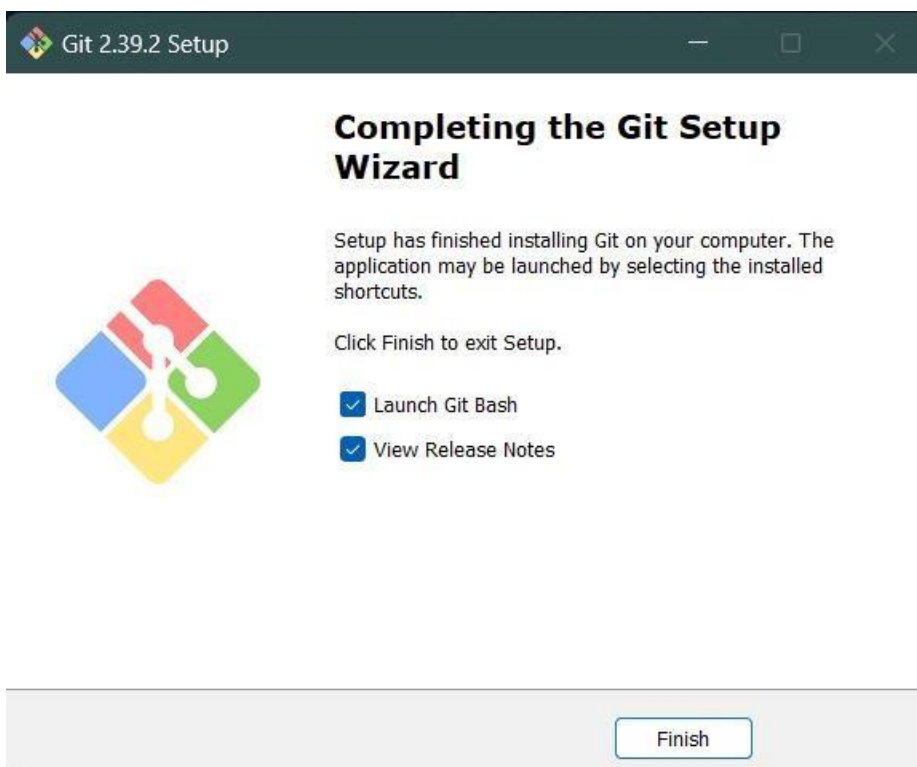
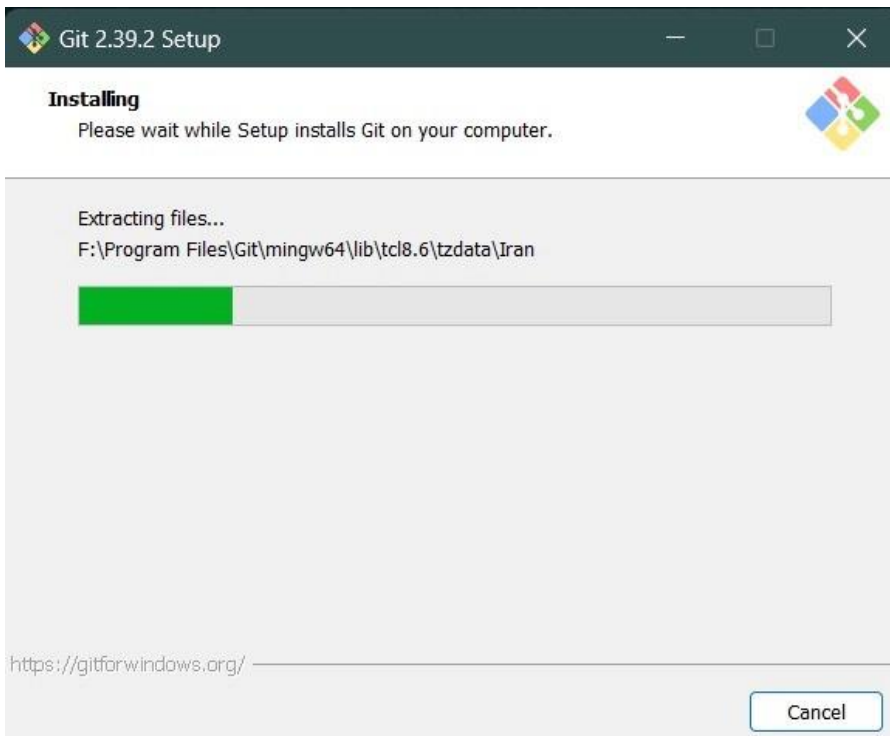
**TERMS AND CONDITIONS FOR COPYING,  
DISTRIBUTION AND MODIFICATION**

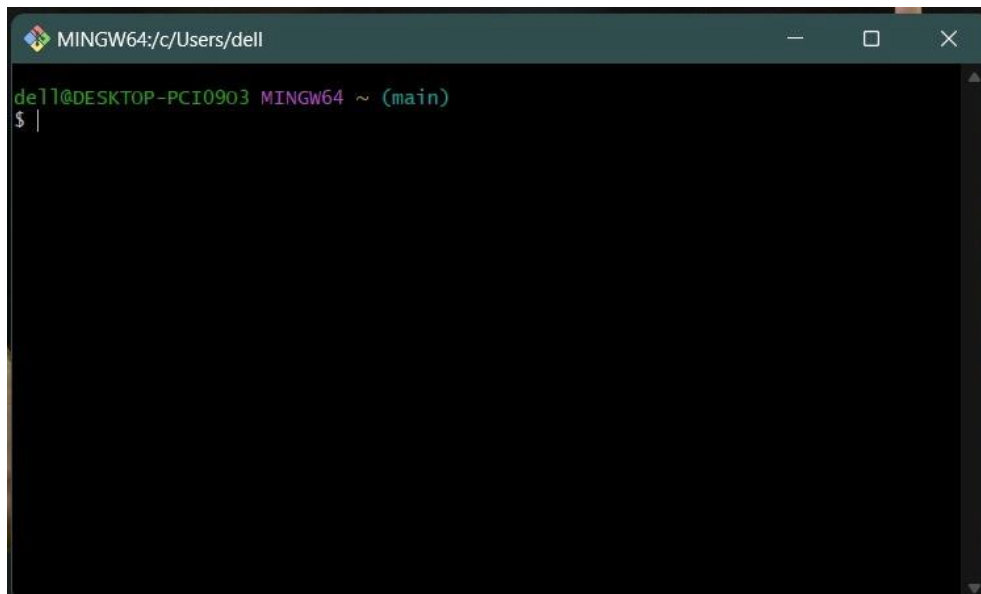
<https://gitforwindows.org/>

NextCancel



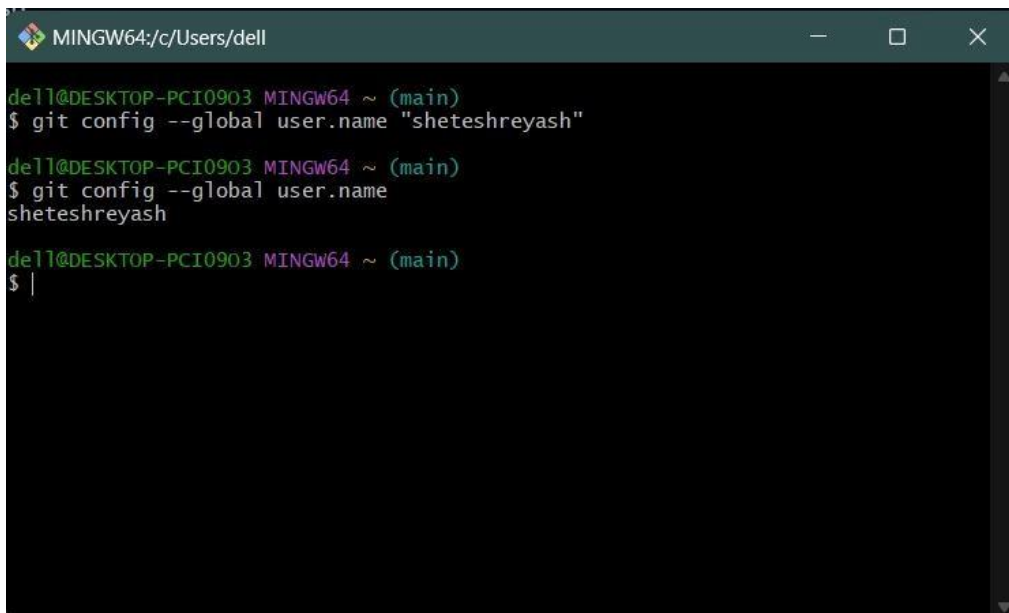






A terminal window titled "MINGW64:/c/Users/dell" with standard window controls. The prompt is "dell@DESKTOP-PCI0903 MINGW64 ~ (main)". The cursor is on the line "\$ |".

```
MINGW64:/c/Users/dell
dell@DESKTOP-PCI0903 MINGW64 ~ (main)
$ |
```



A terminal window titled "MINGW64:/c/Users/dell" with standard window controls. It shows the execution of the command "git config --global user.name 'sheteshreyash'". The prompt is "dell@DESKTOP-PCI0903 MINGW64 ~ (main)". The cursor is on the line "\$ |".

```
MINGW64:/c/Users/dell
dell@DESKTOP-PCI0903 MINGW64 ~ (main)
$ git config --global user.name "sheteshreyash"

dell@DESKTOP-PCI0903 MINGW64 ~ (main)
$ git config --global user.name
sheteshreyash

dell@DESKTOP-PCI0903 MINGW64 ~ (main)
$ |
```



```
MINGW64:/e/Parking management system
dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system
$ git init
Initialized empty Git repository in E:/Parking management system/.git/
dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system (master)
$
```

```
MINGW64:/e/Parking management system
dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system
$ git init
Initialized empty Git repository in E:/Parking management system/.git/
dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system (master)
$ git clone https://github.com/Rushikesh-Taskar/ParkingManagementSystem.git
Cloning into 'ParkingManagementSystem'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 25 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (25/25), 300.75 KiB | 61.00 KiB/s, done.
dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system (master)
$
```

```
MINGW64:/e/Parking management system/ParkingManagementSystem
dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system/ParkingManagementSystem (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md
        modified:   src/App.css

no changes added to commit (use "git add" and/or "git commit -a")
dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system/ParkingManagementSystem (main)
$ |
```

```
MINGW64:/e/Parking management system/ParkingManagementSystem
(use "git restore <file>..." to discard changes in working directory)
modified: README.md
modified: src/App.css

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

dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system/ParkingManagementSystem
m (main)
$ git add .

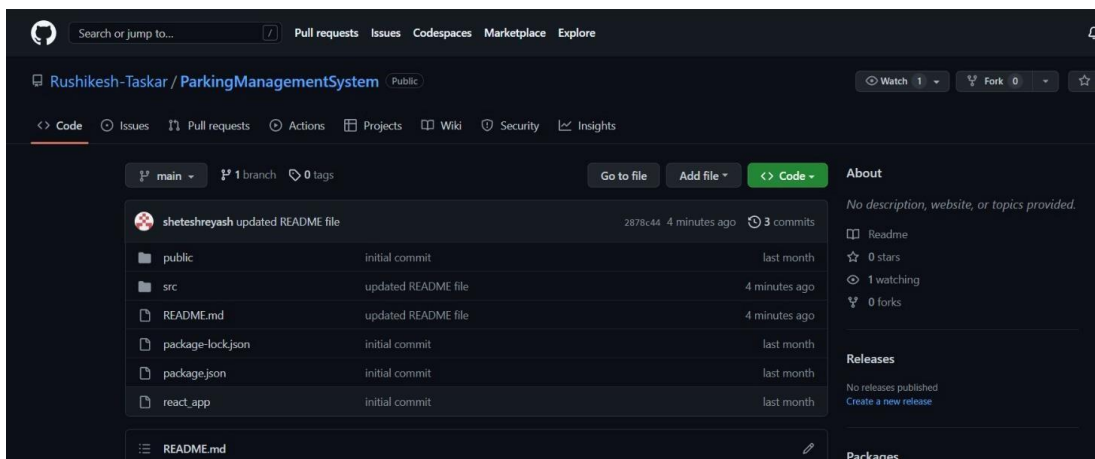
dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system/ParkingManagementSystem
m (main)
$ git commit -m "updated README file"
[main 2878c44] updated README file
2 files changed, 8 insertions(+), 4 deletions(-)
```

```
MINGW64:/e/Parking management system/ParkingManagementSystem
dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system/ParkingManagementSystem
m (main)
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 473 bytes | 473.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Rushikesh-Taskar/ParkingManagementSystem.git
f227aa9..2878c44 main -> main

dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system/ParkingManagementSystem
m (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

dell@DESKTOP-PCI0903 MINGW64 /e/Parking management system/ParkingManagementSystem
m (main)
$
```



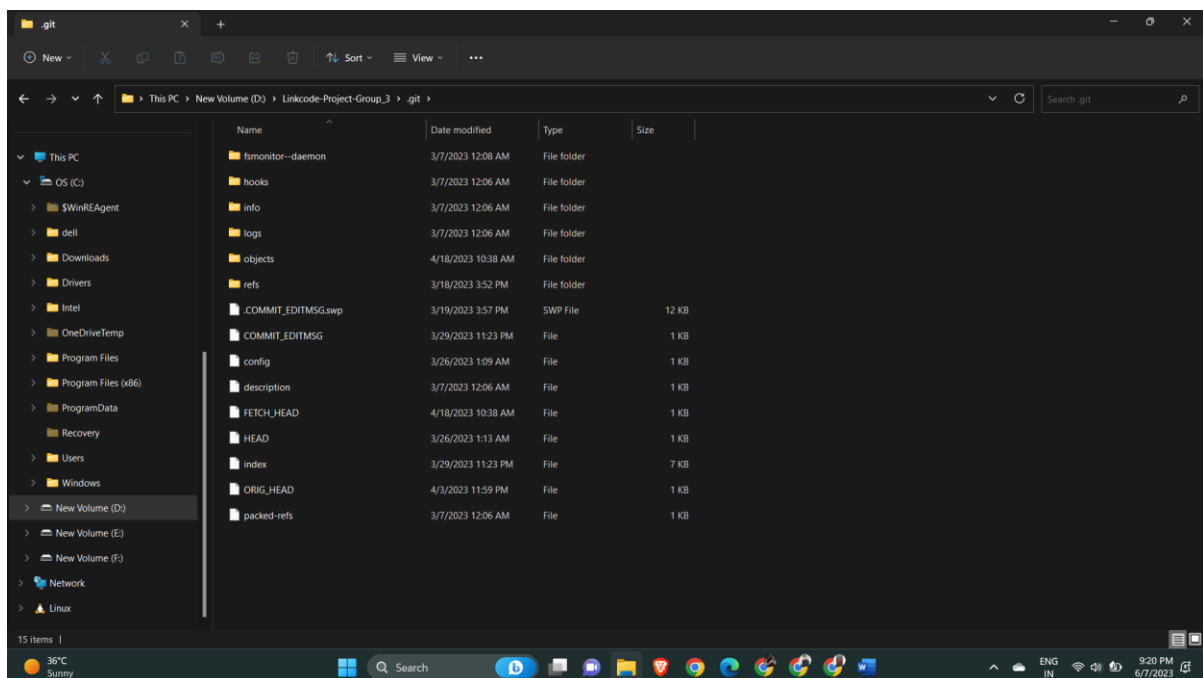
## My EPAM git work :

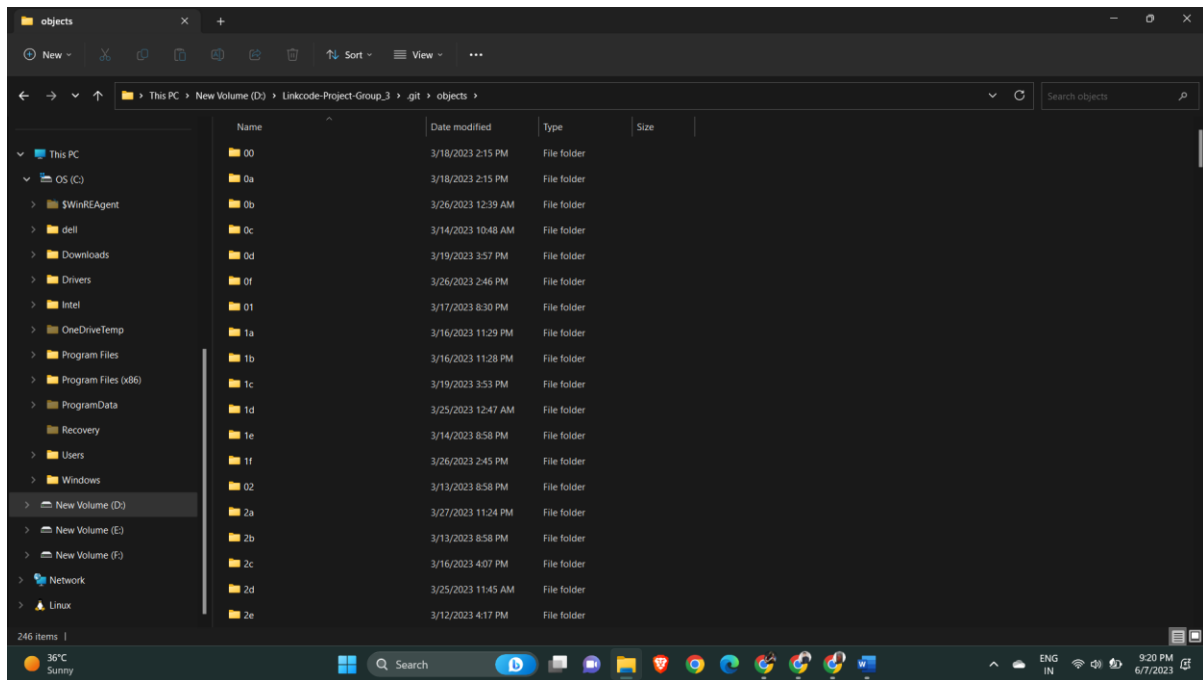
### 1) What's inside a .git folder ?

Commit, tree, blob



- `git show -s --pretty=raw d2d84`
- `git ls-tree 629eb`
- `git show abb54`



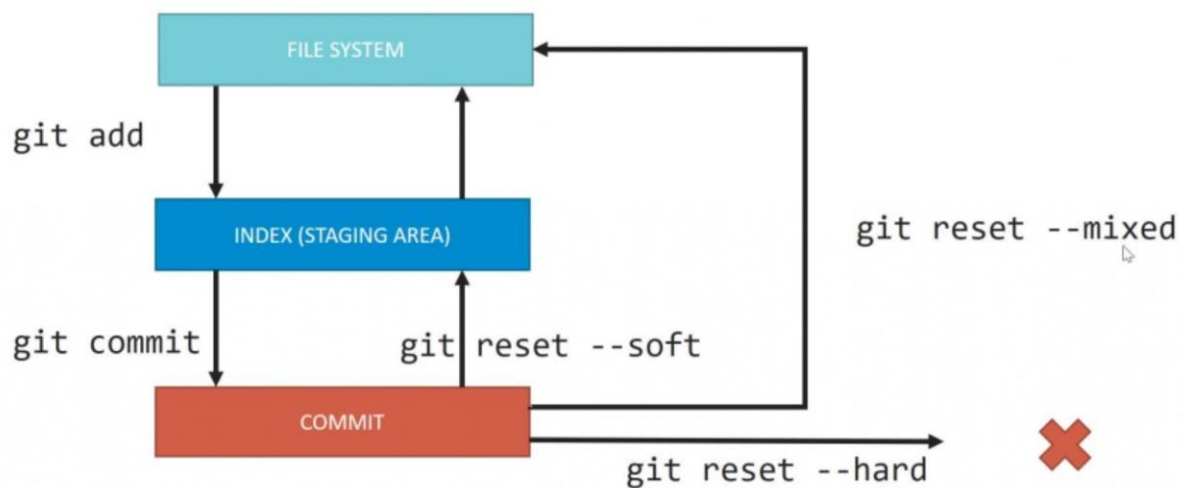


All data gathers here

Also gives the hash function of the file

## 2) Git reset :

Git reset



### 3) Git Revert :

#### Git revert

Undoing changes

Working directory

```
git checkout -- file.txt
git checkout .
git clean -xdf
```

Staging area (Index)

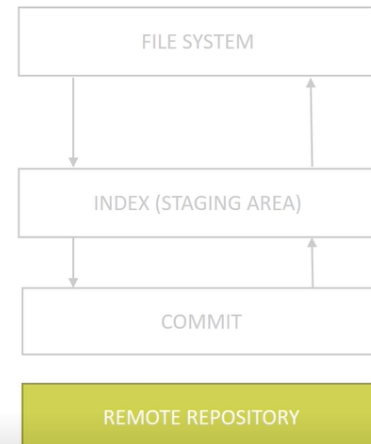
```
git reset -- file.txt
```

Local branch

```
git reset HEAD^^ (HEAD~2)
git commit --amend -m "commit message"
```

Remote repository

```
git revert <sha1>
```



### 4) .gitignore file :

#### .gitignore

.gitignore

```
# no .log files
*.log

# but do track error.log, even though you're
# ignoring .log files above
!error.log

# only ignore the TODO file in the current
# directory, not subdir/TODD
/TODD

# ignore all files in the build/ directory
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory
doc/**/*.pdf
```

## How to get rid of all unknown git files?

1/1 point

- ☐ git remove-all
- ☐ git clear
- ☒ git clean
- ☐ git purge



### Answer

Correct: That's right, don't forget flags, like -f or -xdf

You made a commit and forgot to add a couple of changes. The task is to add data to the already created commit. List all solutions.

1/1 point

- ☒ git commit --amend
- ☐ git update
- ☐ git add
- ☒ git gui, then I will select the amend last commit checkbox
- ☐ git reset --hard HEAD ^^



You want to delete the last commit by placing its contents in the file system. Which command will accomplish this task?

1/1 point

- ☐ git reset --hard HEAD~1
- ☐ git reset --hard HEAD^
- ☒ git reset --mixed HEAD~1
- ☐ git reset --soft HEAD~1



## What is the best to use to remove a commit from a remote repository?

1/1 point

☐ git delete -r

☐ git rerere

☒ git revert

☐ git push -f HEAD~1



## What will happen to the song if I call "git checkout - song.txt"?

1/1 point

☐ Nothing will happen. Checkout is for changing branches

☐ The file will be deleted from disk

☒ The contents of the file will be restored to the latest known git version

☐ We will go to the song.txt branch



## What needs to be added to .gitignore to hide all Microsoft Word files with the extension .doc?

1/1 point

☐ ~doc.

☐ /doc

☒ \*.doc

☐ !\*.doc

☐ doc

☐ !\*\$%



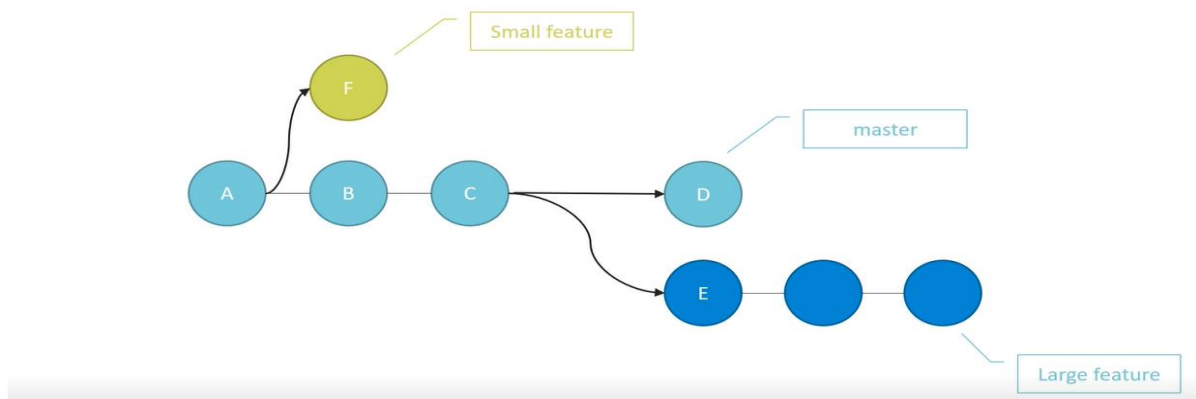
Submit

[Show answer](#)

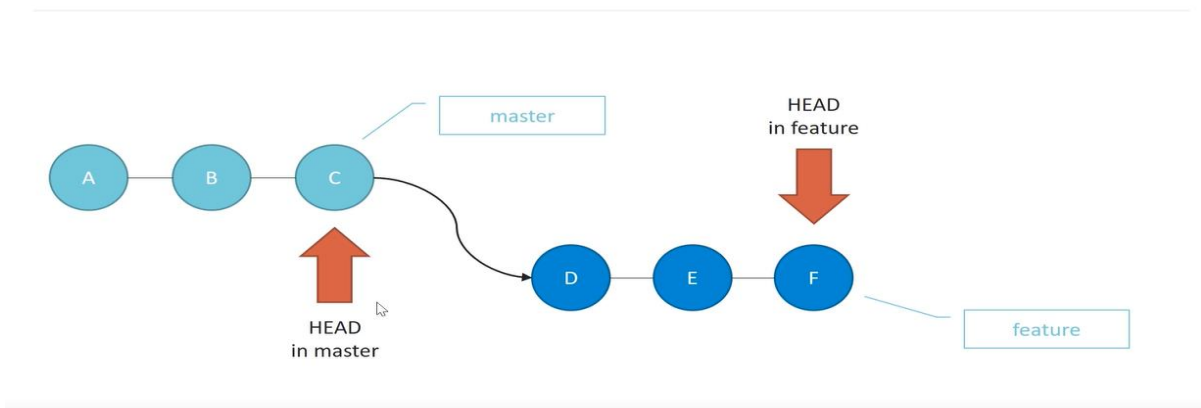
- Branching and Merge :

## Branching and merge

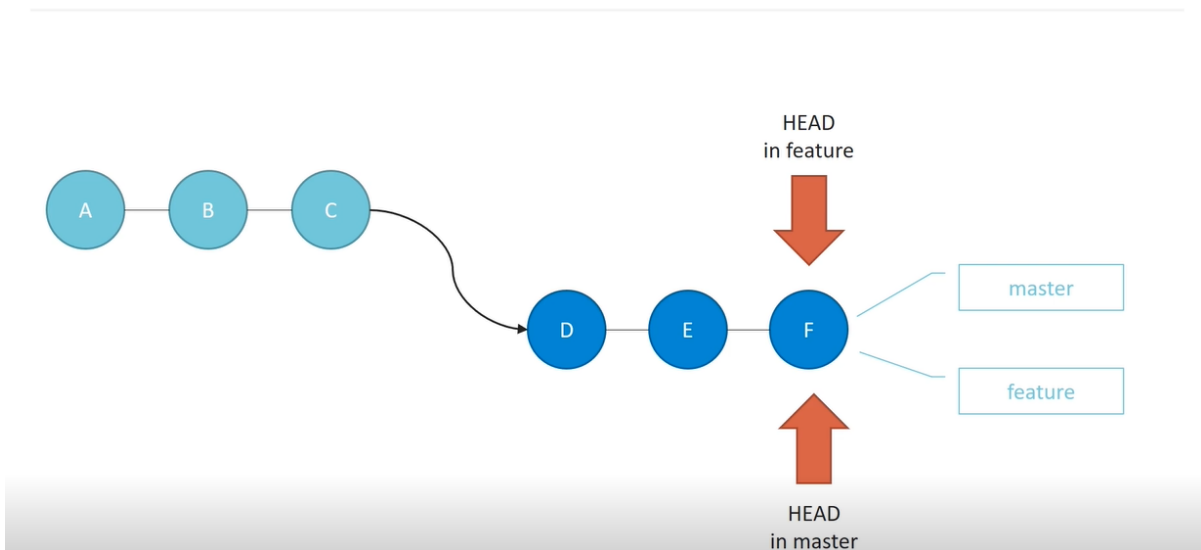
Branch concept



Fast-forward merge



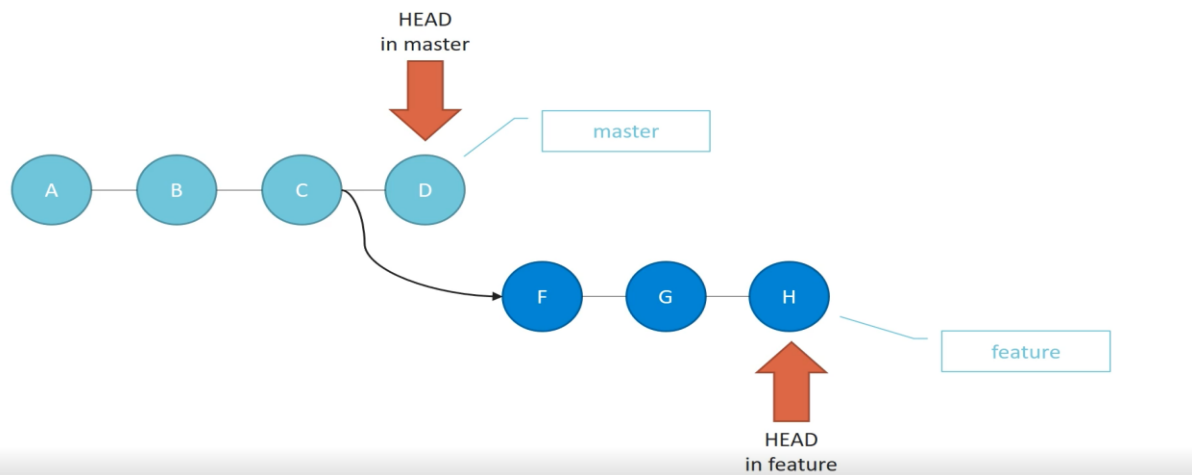
Fast-forward merge



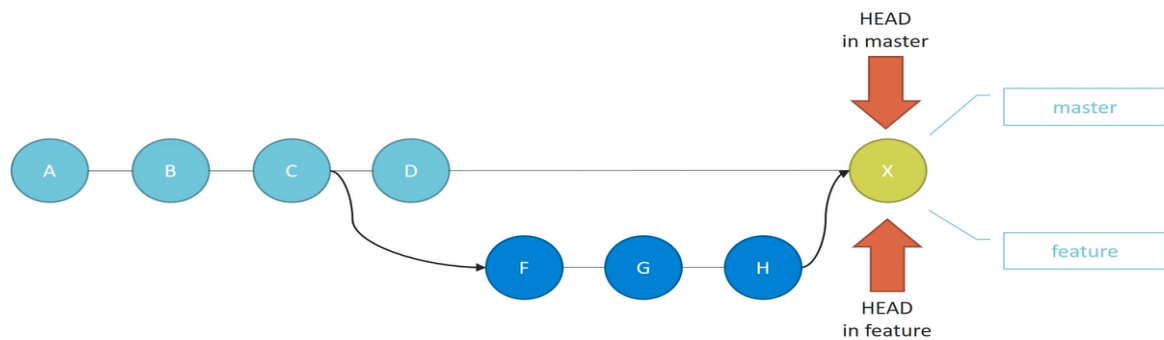


## Non fast-forward merge

Press **Esc** to exit full screen



## Non fast-forward merge



## Conflicts solving

### SOLVE CONFLICT

Abort merge

```
git merge --abort
```

Resolve by selecting version

```
git checkout --Xours --Xtheirs
```

Resolve manually

```
git diff
```

Undo merge

```
git revert 09fe472
```

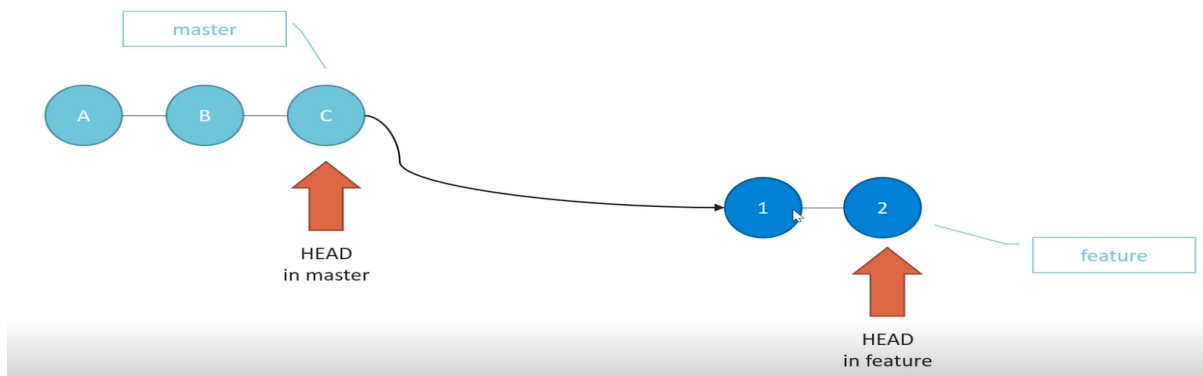
User merge tool

### AVOID CONFLICT

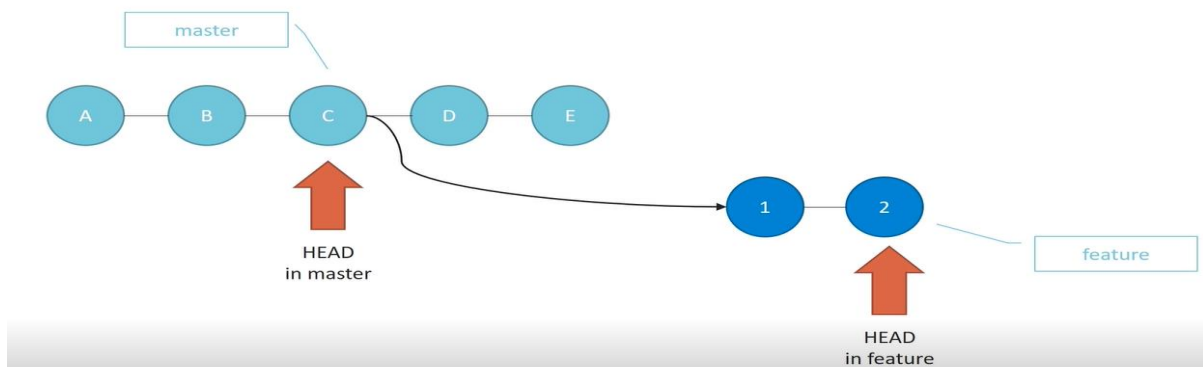
- Short commits
- No edits to whitespaces
- Merge often

- New concept :

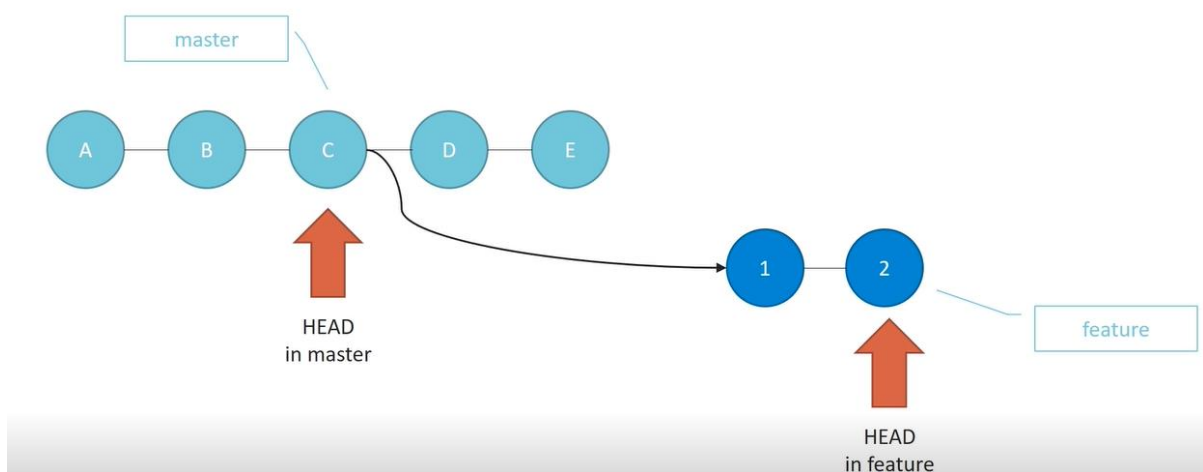
Rebase



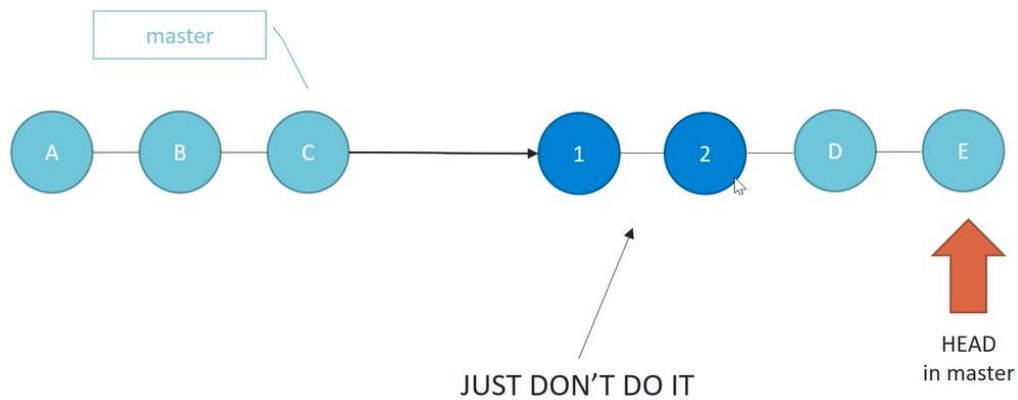
Rebase



Rebase

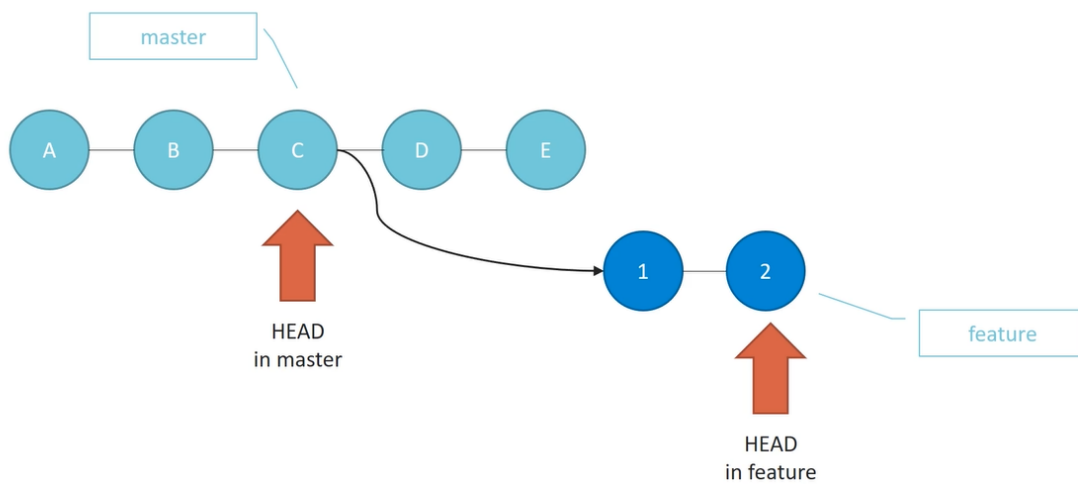


## Golden rule of Rebase



- Something new :

Cherry pick



## How much does branch occupy in the file system?

1/1 point

- ☐ Proportional to the size of the project. If the project is 10 MB in one branch, then when you create the second, it will become 20 MB.
- ☐ 1024 Kb
- ☒ 41 bytes



## How to view the list of branches in the repository?

1/1 point

- ☐ git show branch
- ☐ git branch show
- ☐ git checkout branch
- ☒ git branch --all

## How to exit vi saving changes?

1/1 point

- ☐ :q!
- ☒ :wq
- ☐ Ctrl+S
- ☐ Ctrl+C



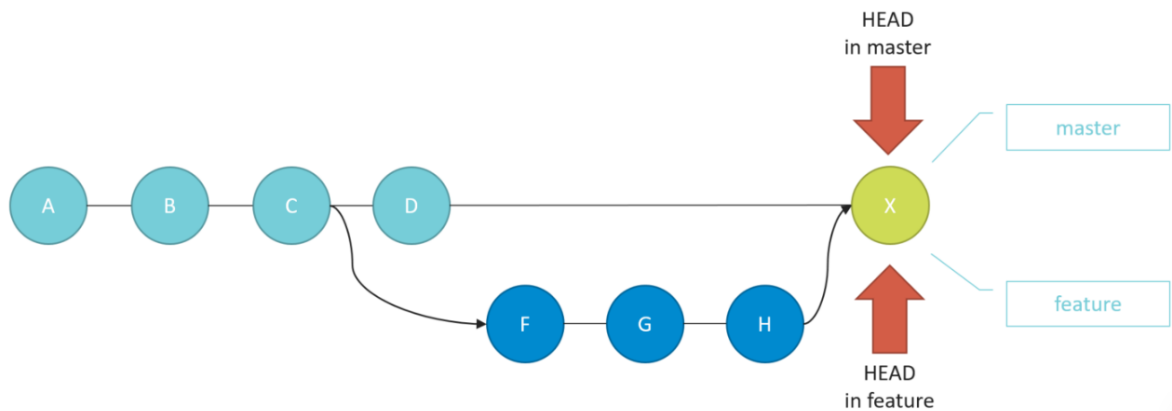
## How to resolve a conflict when merging branches (merge conflict)?

1/1 point

- ☐ Fix the contents of files with conflicts, make git merge --abort
- ☒ Fix the content of files with conflicts, make a commit of changes
- ☐ Use git reset --hard HEAD
- ☐ Make a new copy of the project via git clone. This copy is broken and will never work.



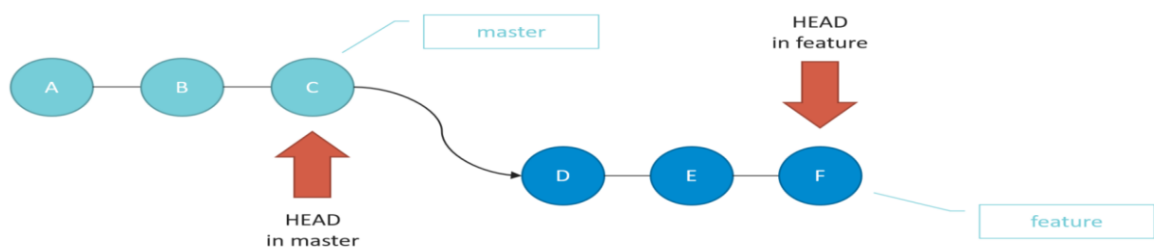
- ☐ Rebase
- ☐ Cherry-pick
- ☐ Fast-forward merge
- ☒ Non fast-forward merge



What will the HEAD pointer of the master branch refer to after we do the merge feature in master?

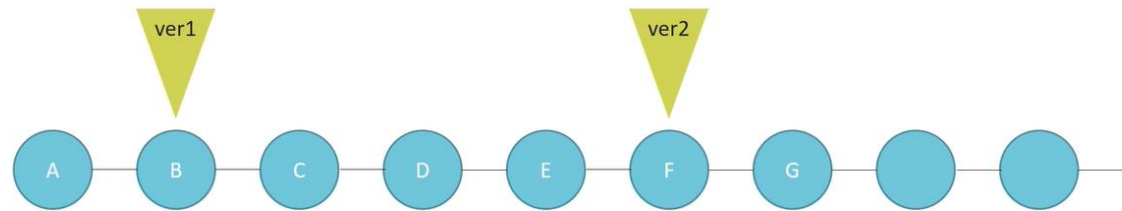
1/1 point

- ☐ A
- ☐ B
- ☐ C
- ☒ D
- ☐ E
- ☐ F
- ☐ New commit G



- Tags :

## Tags



Mark commit with tag  
`git tag ver1`

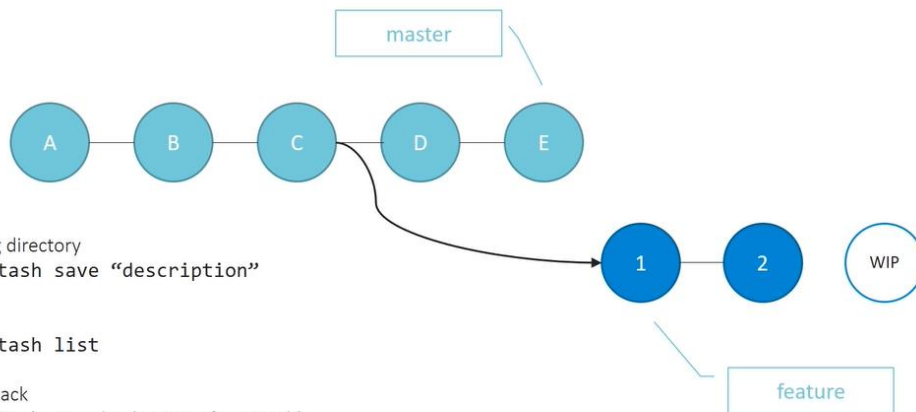
View tags  
`git tag -list`

Push  
`git push --tags`

Check it out  
`git checkout ver1`

- Stash :

## Stash



Save working directory  
`git stash save "description"`

View stashes  
`git stash list`

Bring them back  
`git stash pop` (and remove from stash)  
`git stash apply` (leave in stash)

Remove  
`git stash drop` (clear)

- Git remotes :

## Remotes



Add

```
git remote add <name> <url>
git remote add origin git@github.com:user/repo.git
```

View

```
git remote -v
git remote show <name>
```

**You are in the feature branch. The project manager asks you to urgently fix the defect in the master branch. You have unsaved changes that you are not ready to commit. What will be the procedure?**

1/1 point

- ☐ git clean -xdf, git checkout master, fixing the defect, returning to the feature branch
- ☒ git stash, git checkout master, fixing the defect, returning to the feature branch
- ☐ git add., git checkout master, fixing the defect, returning to the feature branch
- ☐ git checkout., git checkout master, fixing the defect, returning to the feature branch



**You want to mark the current commit with the tag "release1.0". What is the right command for it in the console?**

1/1 point

git tag release1.0



**What is the difference between git stash pop and git stash apply? Both will transfer the changes saved in stash back to the working directory, but what is the difference?**

1/1 point

- ☒ pop will delete stash after returning the changes saved in it, apply - will leave it
- ☐ no difference, this is alias. the commands are identical
- ☐ apply will remove stash after returning the changes saved in it, pop will leave it
- ☐ apply will apply stash to the project, pop - will list available stashes for application



**Which command can help to configure the git so that fetch and push go to another remote repository?**

1/1 point

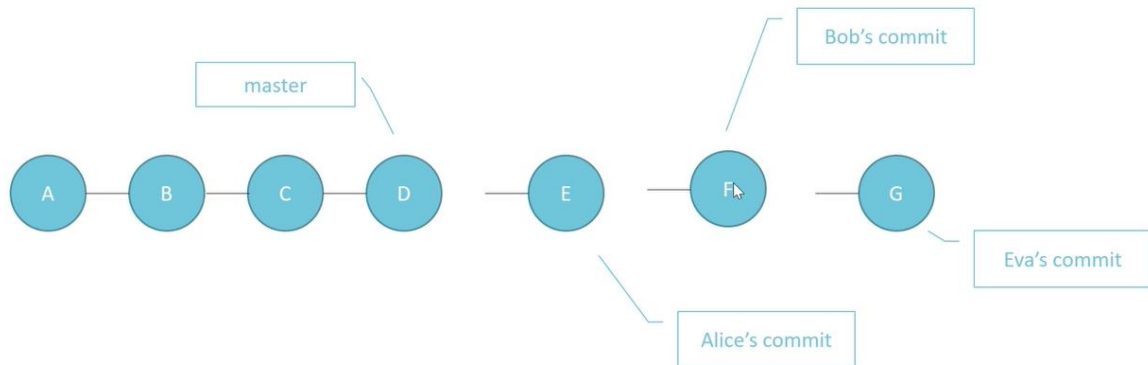
- ☒ git remote
- ☐ git repository
- ☐ git config
- ☐ that is imposible



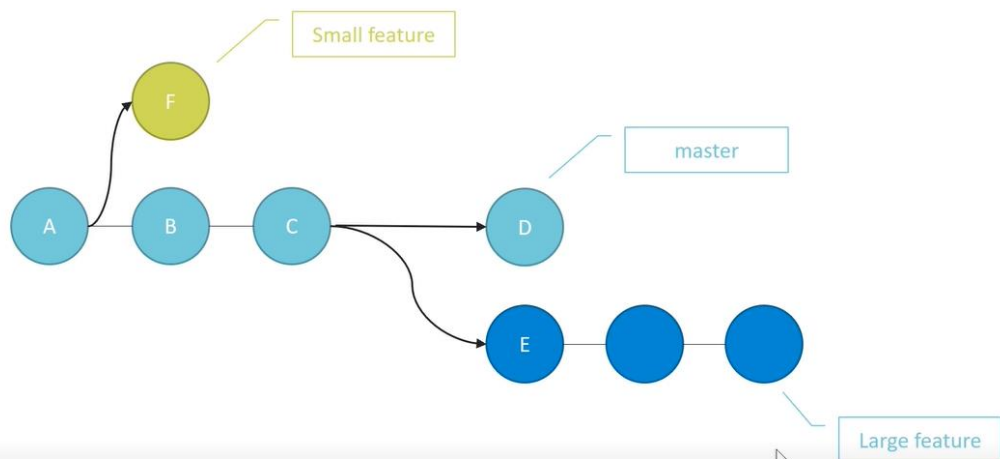


- Branching Strategies :

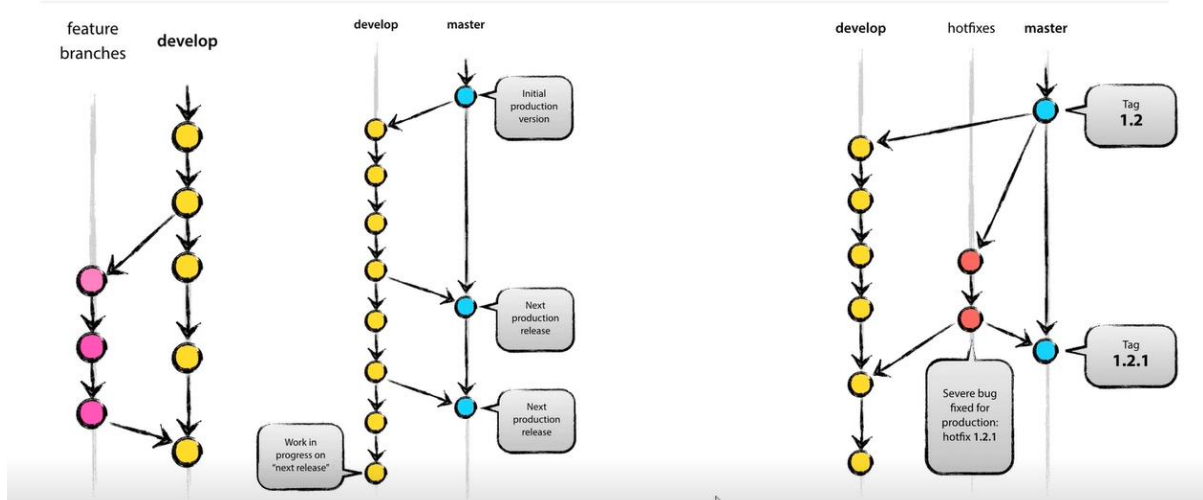
### Centralized strategy



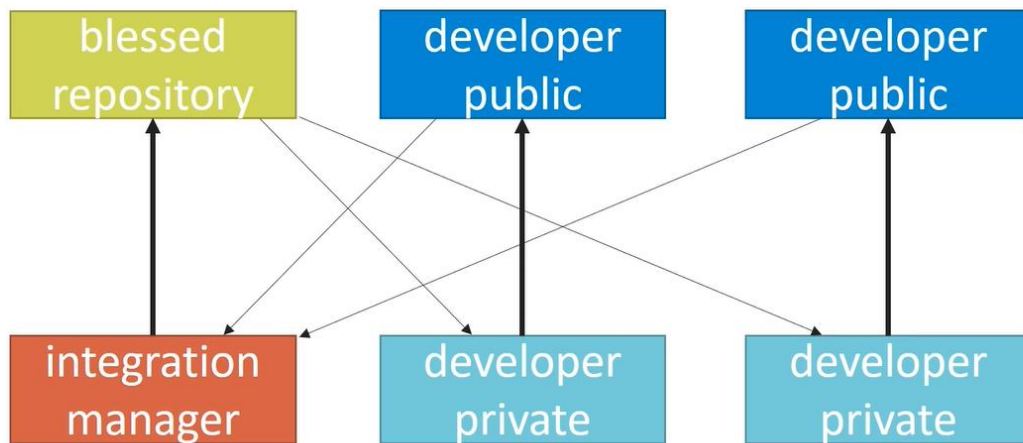
### Feature-branch workflow



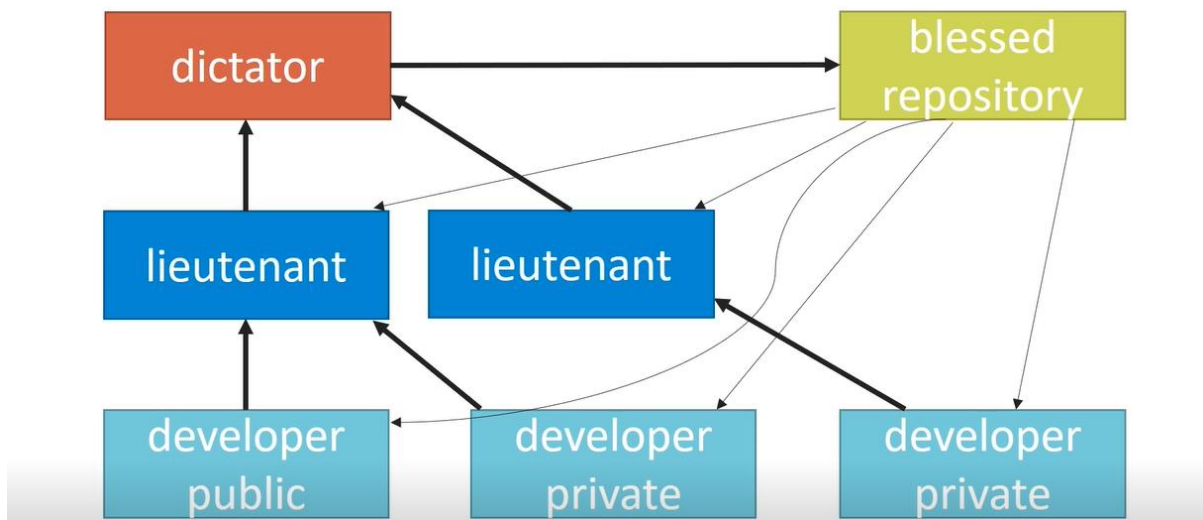
### Gitflow



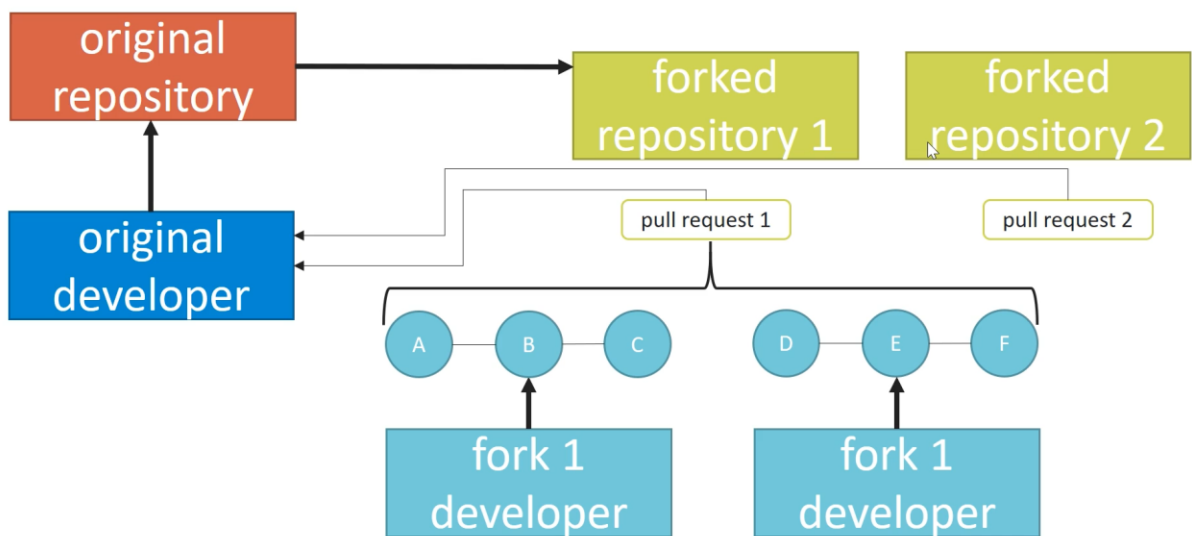
### Integration manager workflow



### Dictator and Lieutenants workflow



### Forking workflow



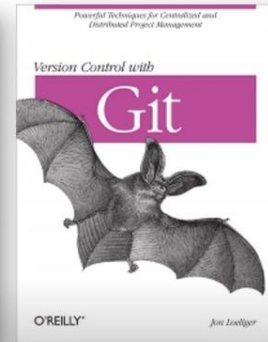
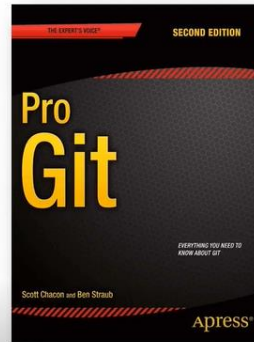
## Extras

### EXTRAS

- `git config --global user.name "Vitali Shulha"`
- `git config --global user.email "vitali_shulha@epam.com"`
- `git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe'"`
- `git blame`
- `git bisect`
- `git log --pretty=oneline`
- `git log --pretty=format:"%h %s" --graph`
- `git config --global alias.last 'log -1 HEAD'`
- `git last`
- `git log master..experiment`
- `git filter-branch --tree-filter 'rm -f passwords.txt' HEAD`
- `git rerere`
- `git submodule`

### READ MORE

- Pro Git by Scott Chacon and Ben Straub
- Version Control with Git by Jon Loeliger, Matthew McCullough



## Useful links

Training Presentation: [DevTestOps-Version\\_Control\\_with\\_Git.pdf](#)

Git official website: <https://git-scm.com/>

The book "ProGit", second edition, in Russian: <https://git-scm.com/book/ru/v2>

[< PREVIOUS](#)