Git Workshop

Sobhan Ahmadian Moghadam

What is Git?

Git is a free and open source distributed version control system (VCS).

Web page

What is VCS?

Control of code changes over time.

What is VCS?

What should we do if we want manage writing of a book:

- Save a few drafts
  - Waste storage
  - The control is hard
- Use version control system

What is Git?

Git is a free and open source distributed version control system (VCS).

- Version Control System
- Manage Code History
- Track Changes

What is Github?

The largest and most advanced development platform in the world.
Web page

- Cloud Hosting
- Collaboration Provider
- Git Repository Hosting

Command-line interface

A text-based user interface (UI) used to view and manage computer files.
Users type commands in the command line interface to run tasks on a computer.

- Windows : Command Prompt
- Linux : GNOME Terminal

GNOME Terminal

-commands

- ‘**ls**’ : list files
- ‘**cd**’ : change directory
  - ‘**cd ..**’ back to upper directory
  - ‘**cd [dir name]**’ go to the directory
- ‘**touch [file name]**’ : make file
- ‘**rm [file name]**’ : remove file
- ‘**mkdir [dir name]**’ : make directory
- ‘**rmdir [dir name]**’ : remove directory

Command Prompt

-commands

- ‘**dir**’ : list files
- ‘**cd**’ : change directory
  - ‘**cd ..**’ back to upper directory
  - ‘**cd [dir name]**’ go to the directory
- ‘**cls**’ : clean cmd
- ‘**mkdir [dir name]**’ : make directory
- ‘**rmdir [dir name]**’ : remove directory
- ‘**echo [content] > [file name]**’ : make file
- ‘**del [file name]**’ : delete file

Install Git

-Linux

- Web page > Downloads > Linux/Unix
- `$ sudo apt-get install git`

Git Version

-Linux

```
$ git --version
```

Update Git

-Linux

```
$ sudo add-apt-repository -y ppa:git-core/ppa
$ sudo apt-get update
$ sudo apt-get install git -y
```

Install Git

-Windows

- Web page > Downloads > Windows
- Run .exe file
- Leave settings by default

Git Version

-Windows

```
$ git --version
```

Install Git

-macOS

- Web page > Downloads > macOS
- install Homebrew
- `$ brew install git`

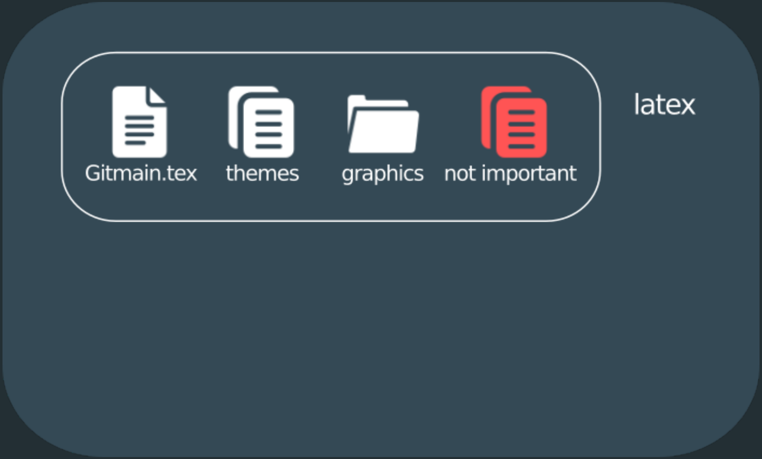
Git Version

-macOS

```
$ git --version
```

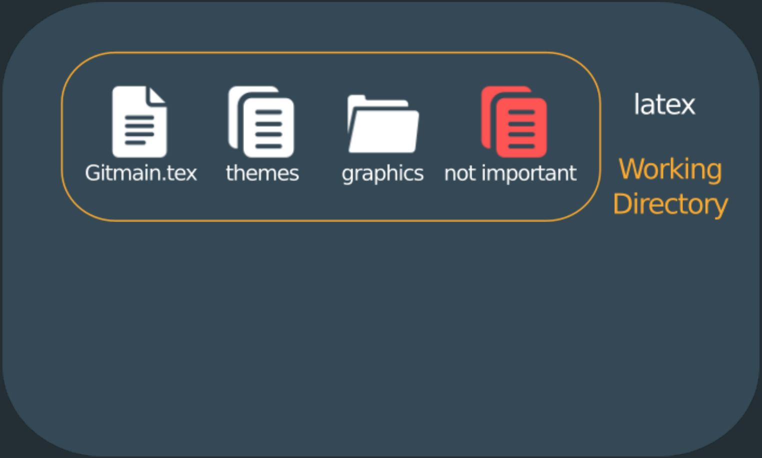
How Git Works?

-abstract example



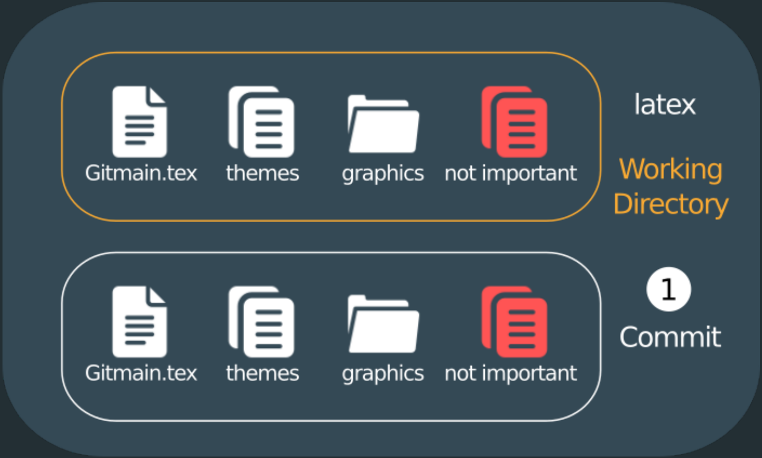
How Git Works?

-abstract example



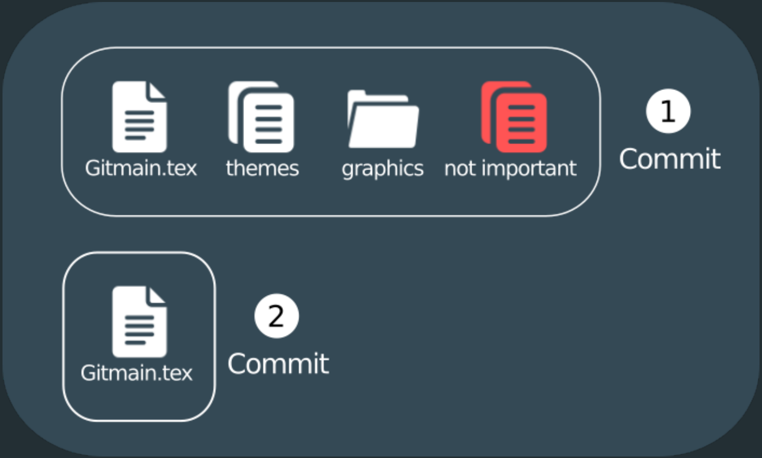
How Git Works?

-abstract example



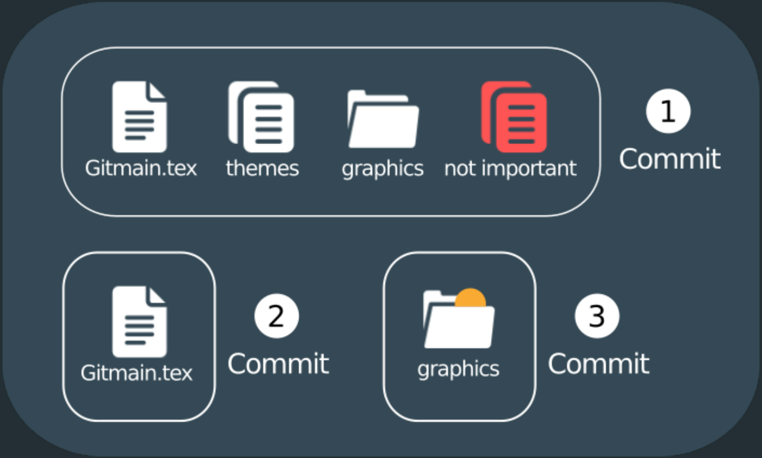
How Git Works?

-abstract example



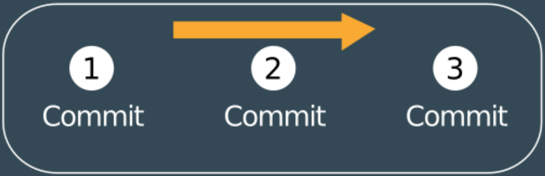
How Git Works?

-abstract example



How Git Works?

-abstract example



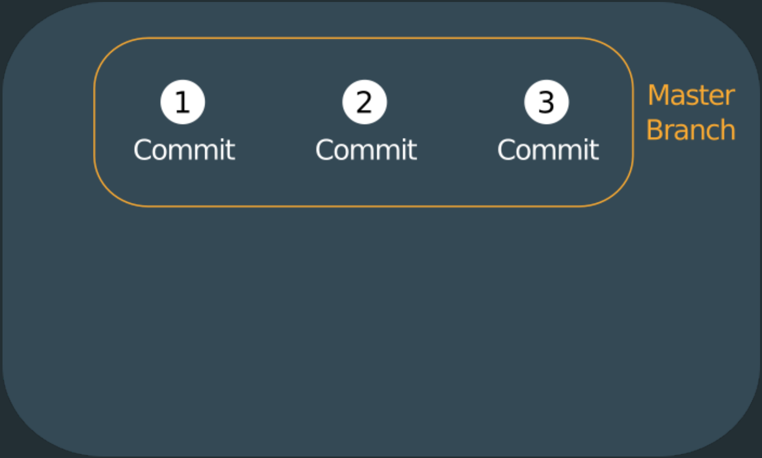
1 Commit

2 Commit

3 Commit

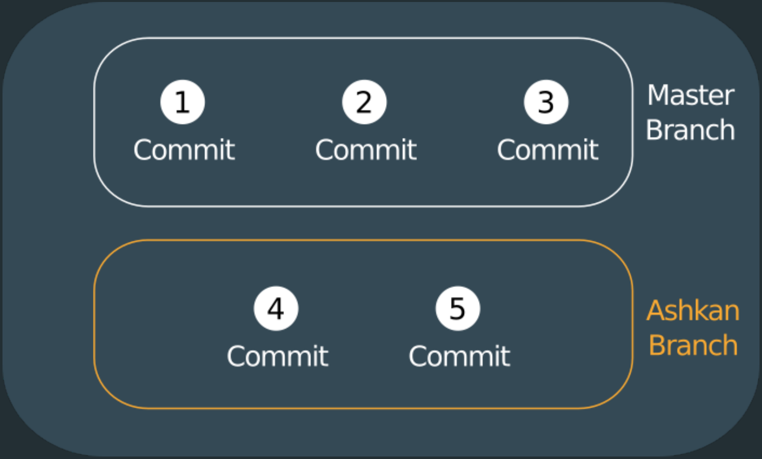
How Git Works?

-abstract example



How Git Works?

-abstract example



1 Commit

2 Commit

3 Commit

Master Branch

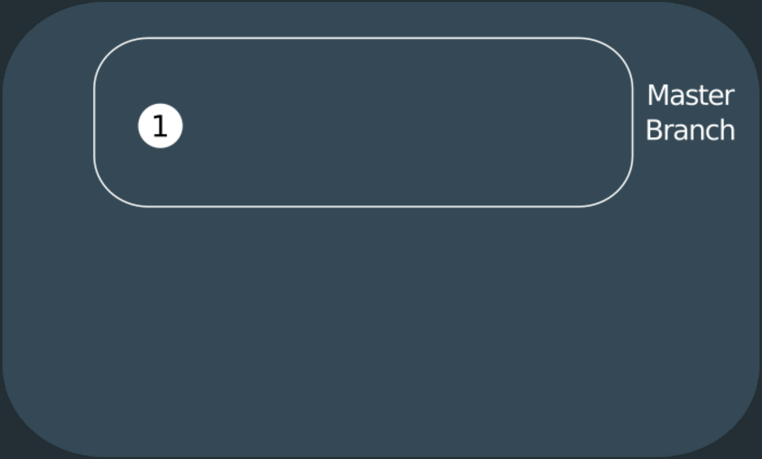
4 Commit

5 Commit

Ashkan Branch

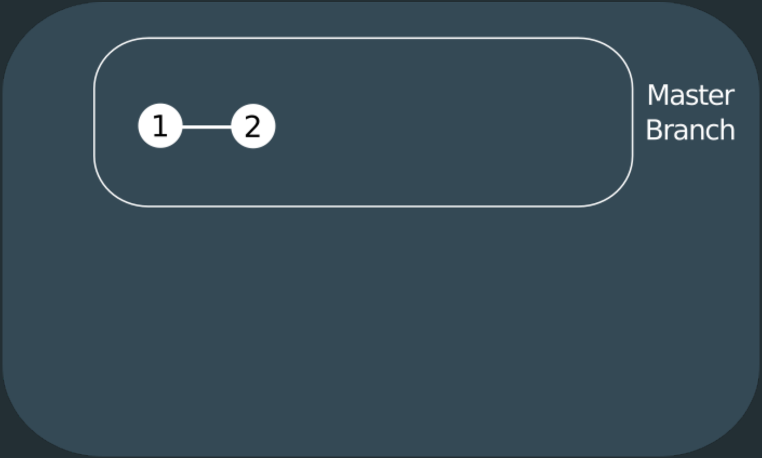
How Git Works?

-abstract example



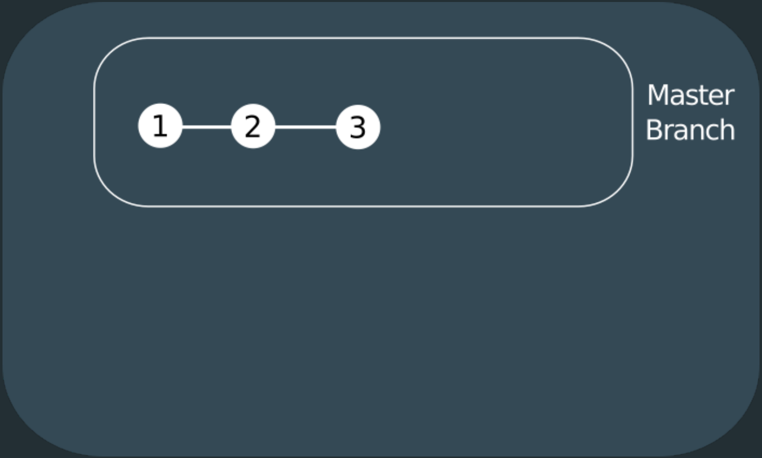
How Git Works?

-abstract example



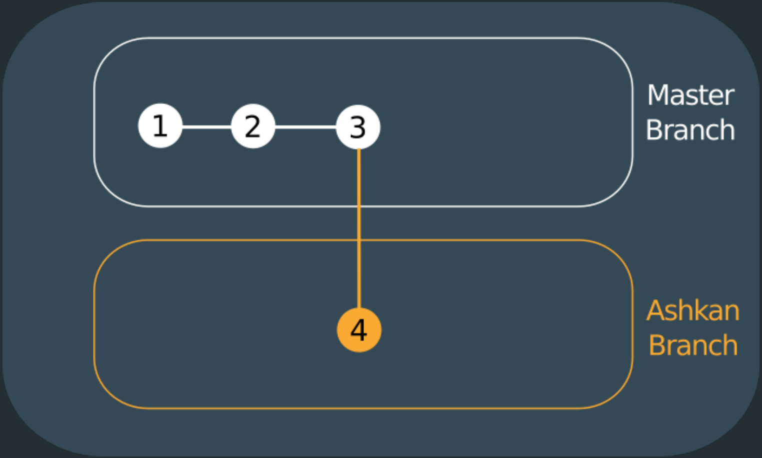
How Git Works?

-abstract example



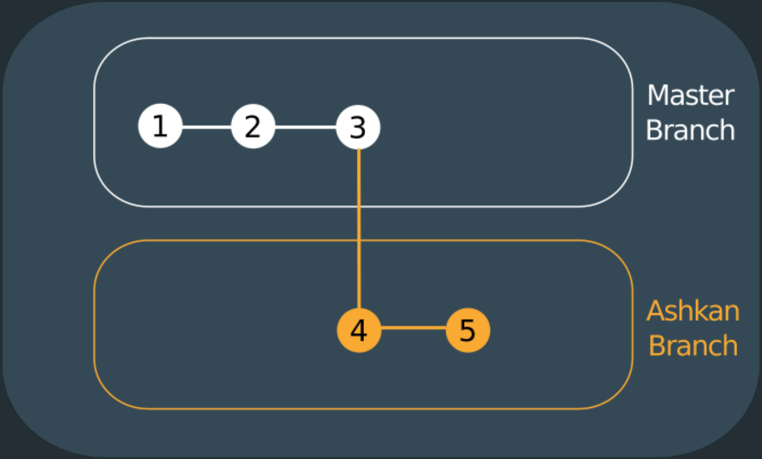
How Git Works?

-abstract example



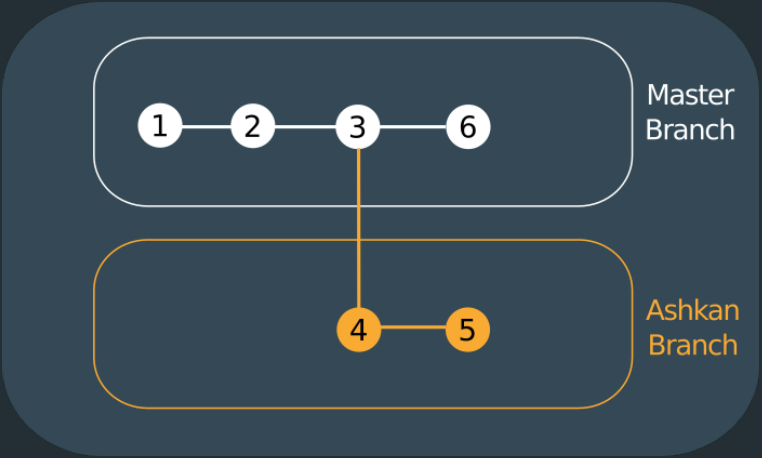
How Git Works?

-abstract example



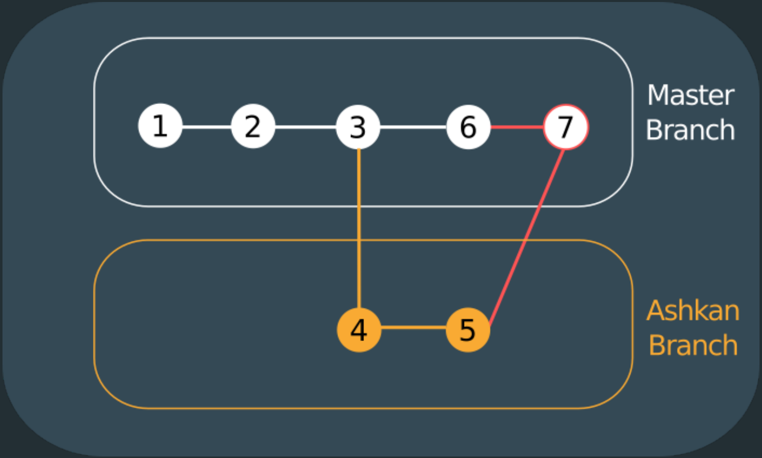
How Git Works?

-abstract example



How Git Works?

-abstract example



Git Commands

-status and initial

First check that git is installed.
Then check git status.
If there is no git repository, initial git.

```
# get git version
$ git --version

# get git status
$ git status

# initial git for current repository
$ git init
```

Git Commands

-configuration

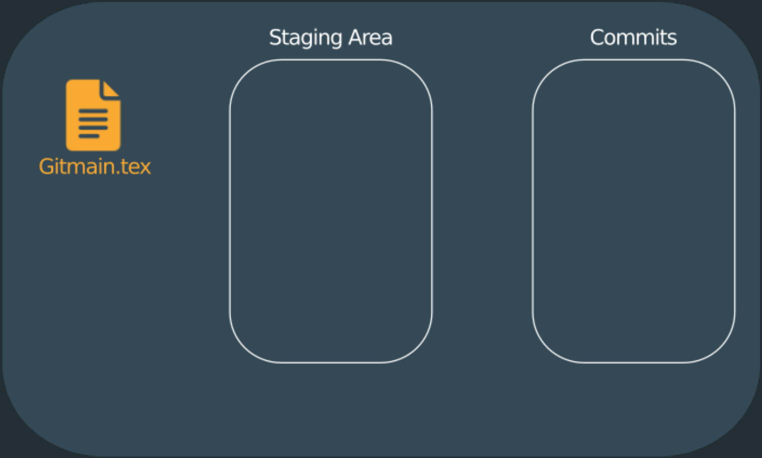
Set email and username (If you didn't do that before).

```
# show git configuration
$ git config --list

# set email and username
$ git config --global user.email "<email>"
$ git config --global user.name "<Your Name>"

# unset email and username
$ git config --global --unset user.name
$ git config --global --unset user.email
```

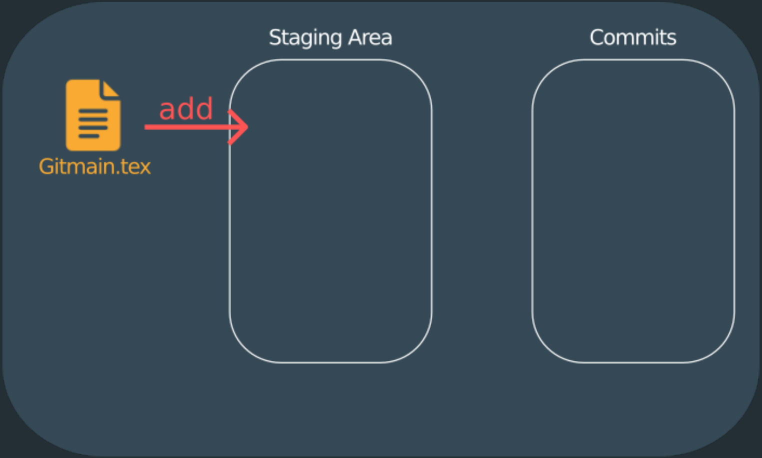
Staging

Gitmain.tex

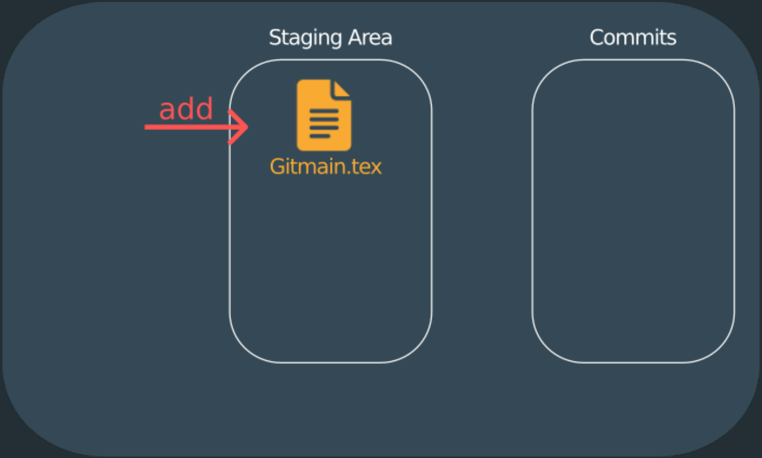
Staging



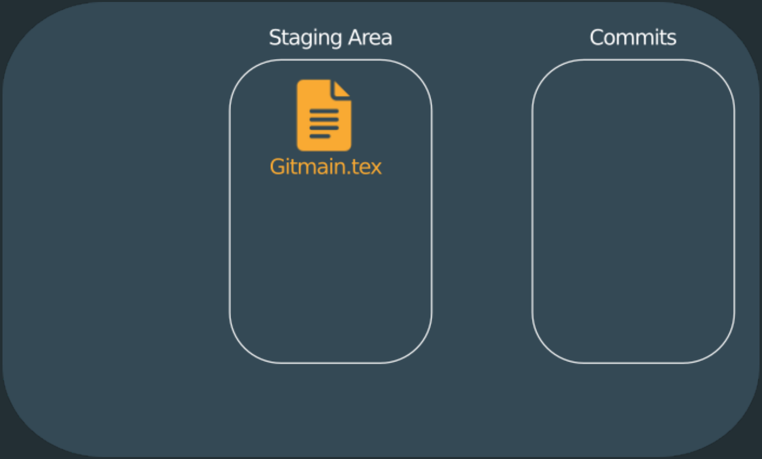
Staging



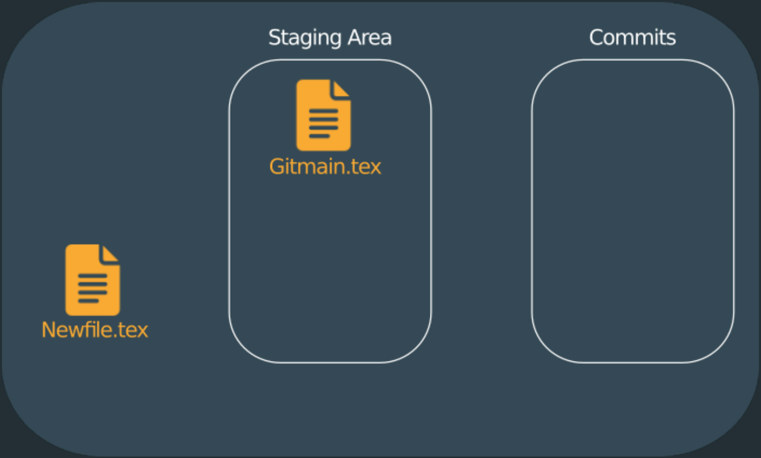
Staging



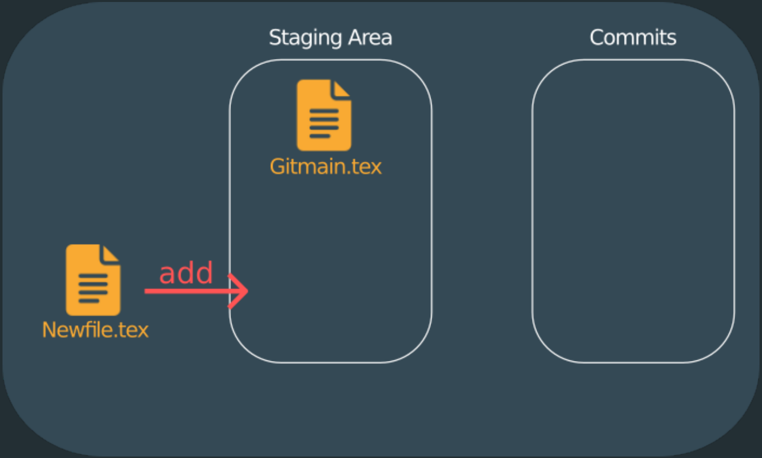
Staging



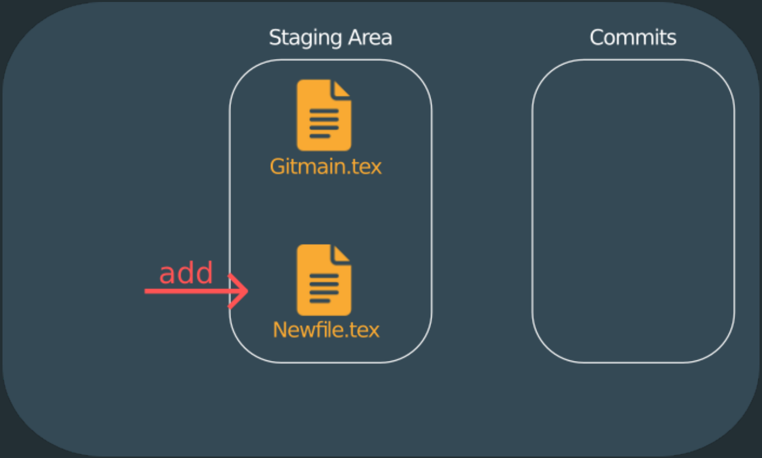
Staging



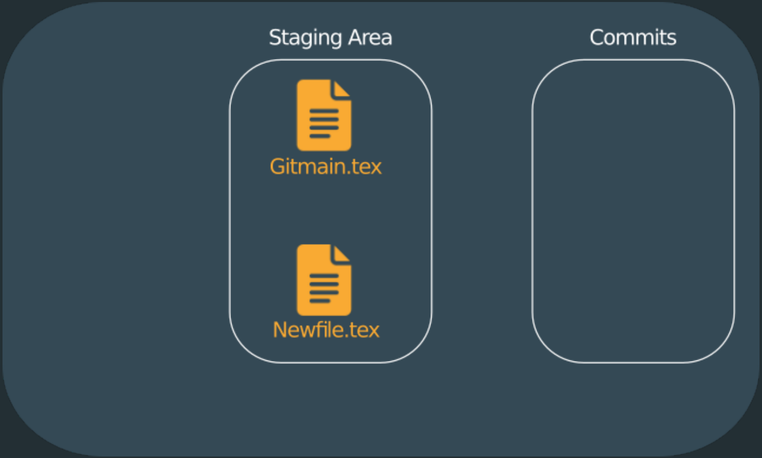
Staging



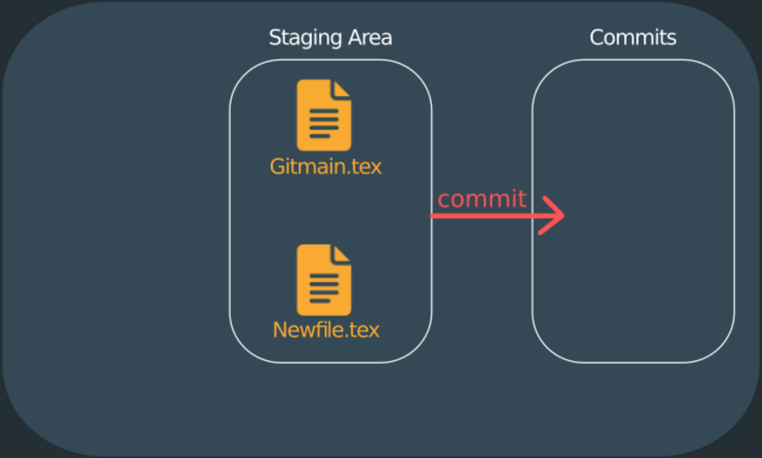
Staging



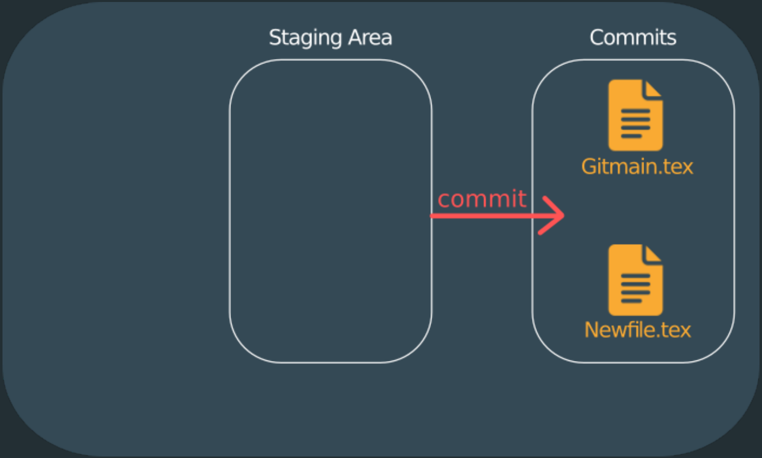
Staging



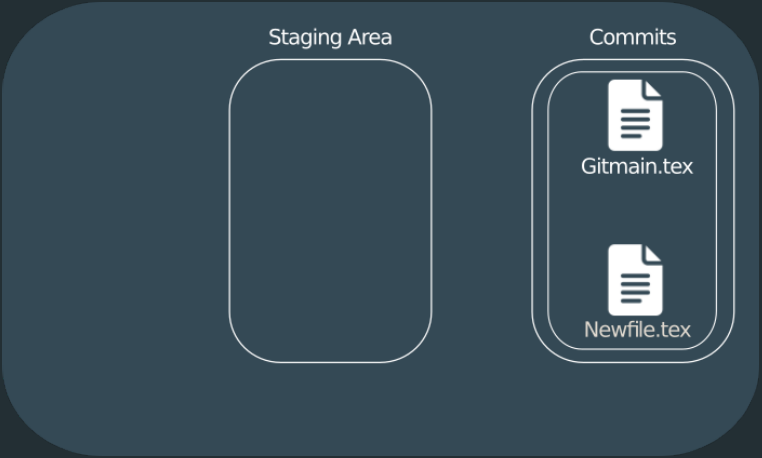
Staging



Staging



Staging



Git Commands

-stage and commit

```
# add file to staging area
$ git add <file name>

# add all changes to staging area
$ git add .

# remove file from staging area
$ git restore --staged <file name>

# commit changes
$ git commit -m "<message>"
```

Git Commands

-switch to a commit

```
# show commits in current branch
$ git log

# switch to a commit
$ git checkout <commit id>
```

Git Commands

-branches

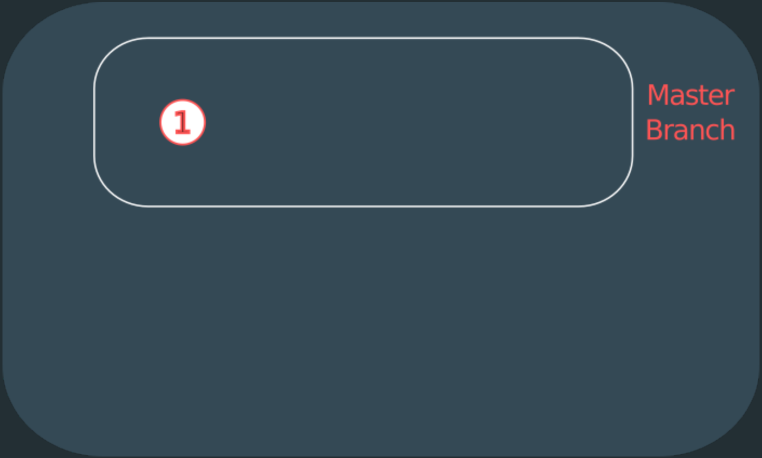
```
# show branches
$ git branch

# create a new branch
$ git branch <branch name>

# switch to a branch
$ git checkout <branch name>

# shortcut for create and move to a branch
$ git checkout -b <branch name>
```

Git Commands

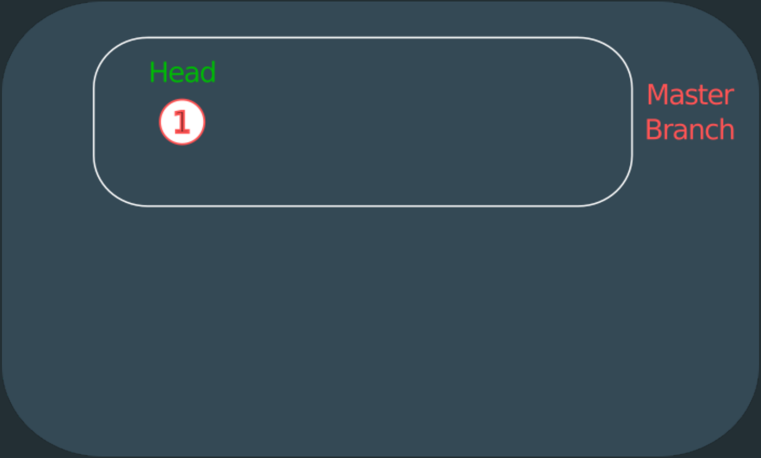
-merge branches

```
# merge a branch to current branch
$ git merge <branch name>
```

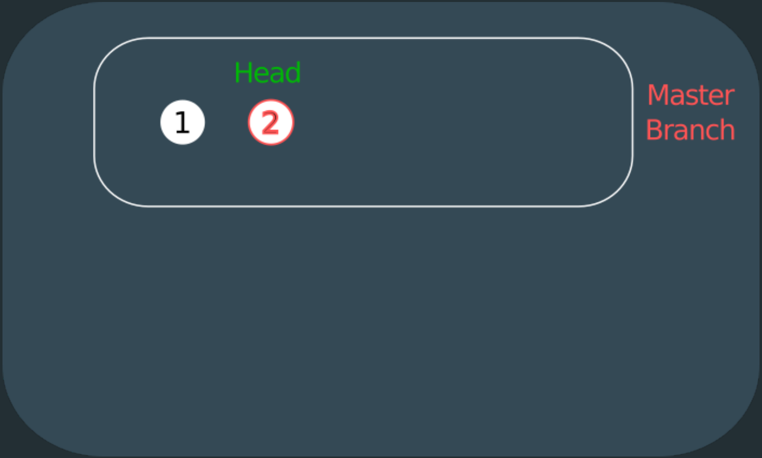
Head



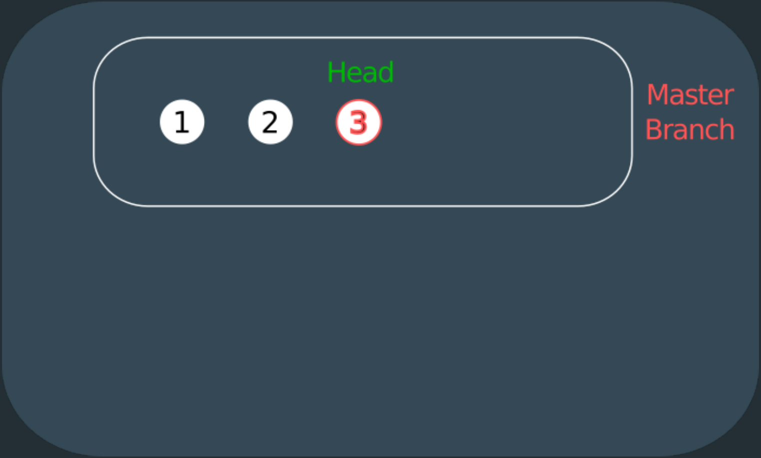
Head



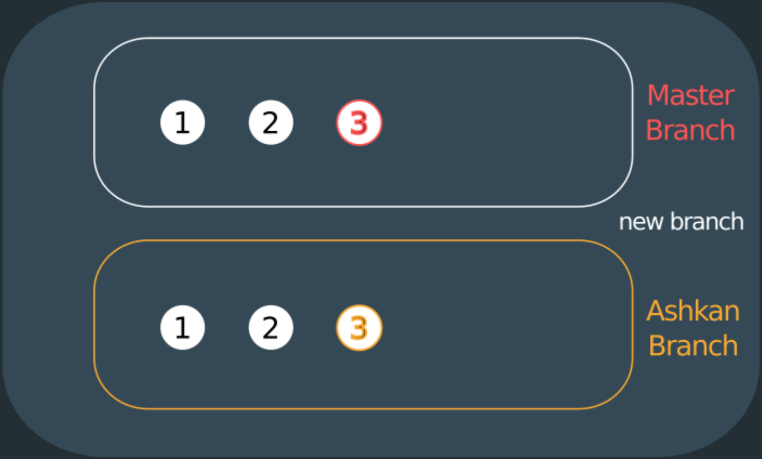
Head



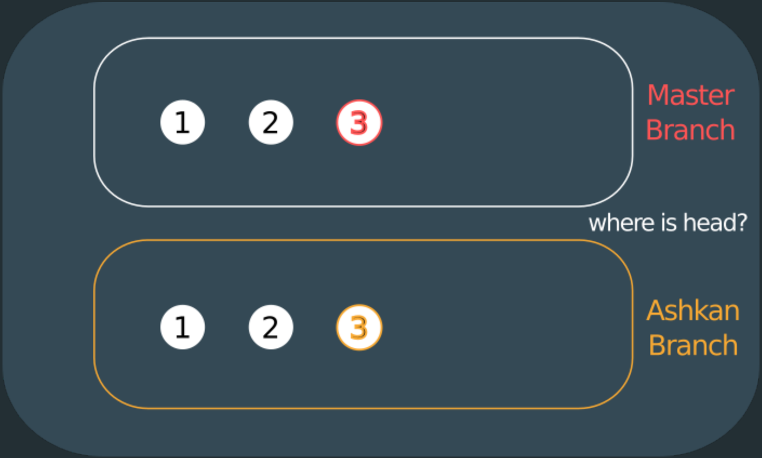
Head



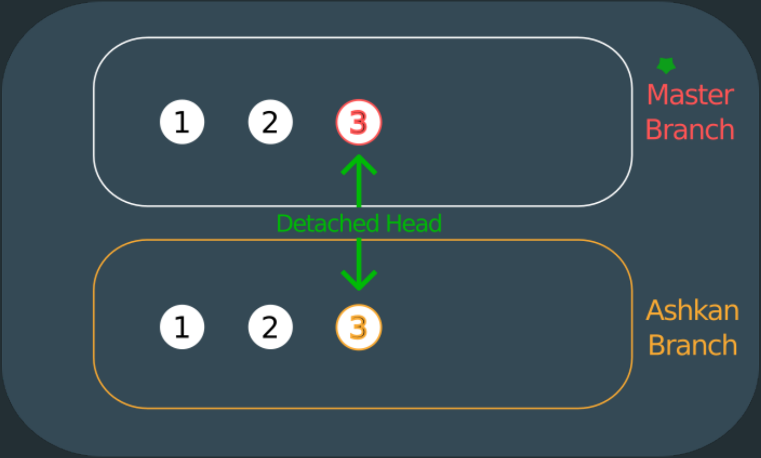
Head



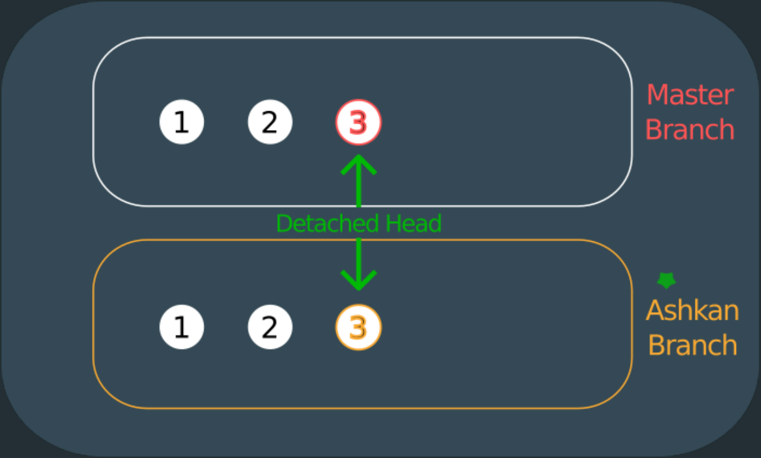
Head



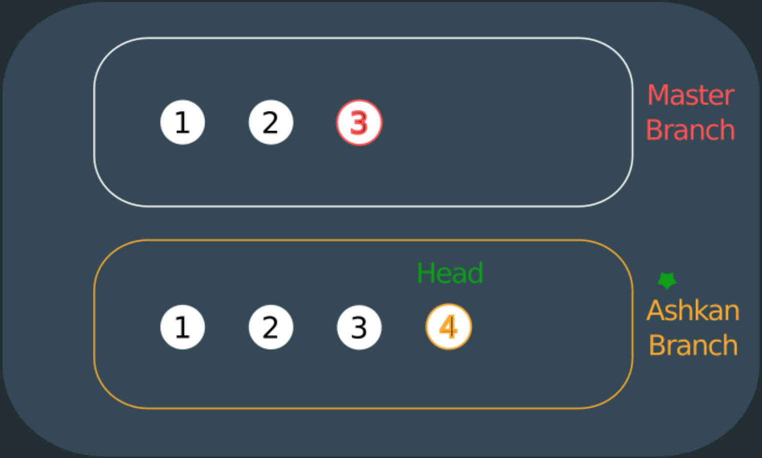
Head



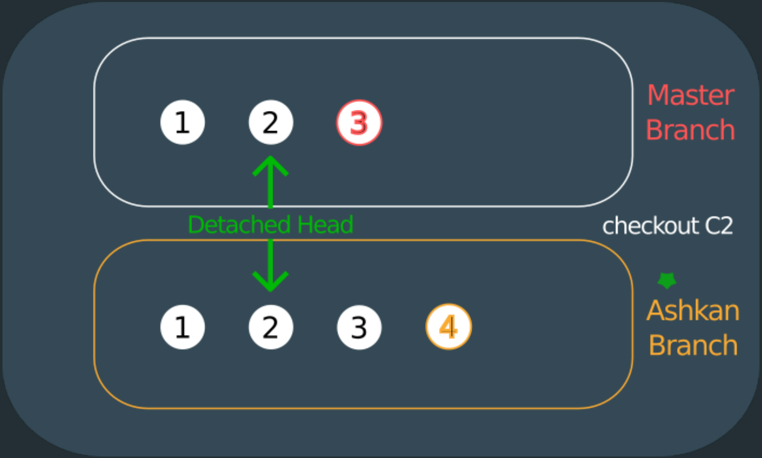
Head



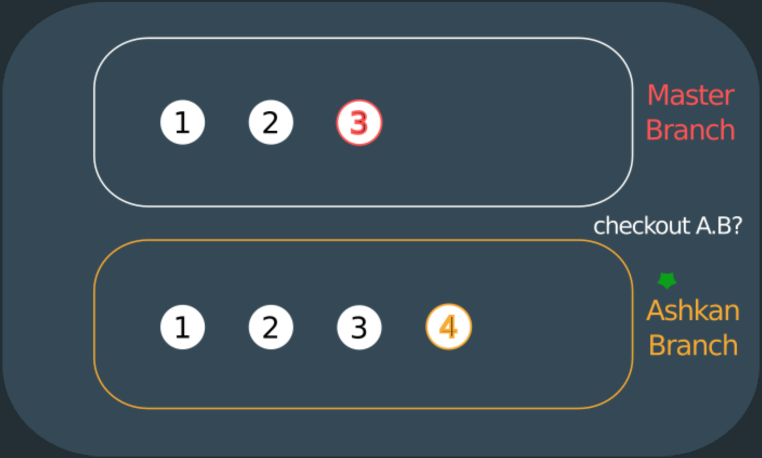
Head



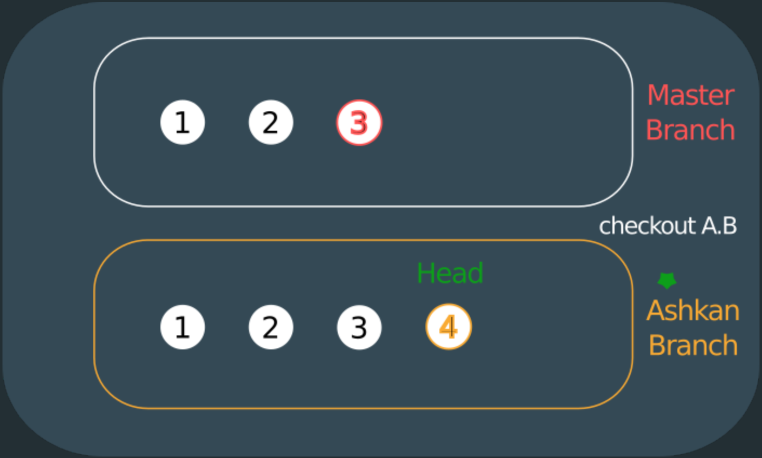
Head


1
2
3
Master Branch
Detached Head
checkout C2
1
2
3
4
Ashkan Branch

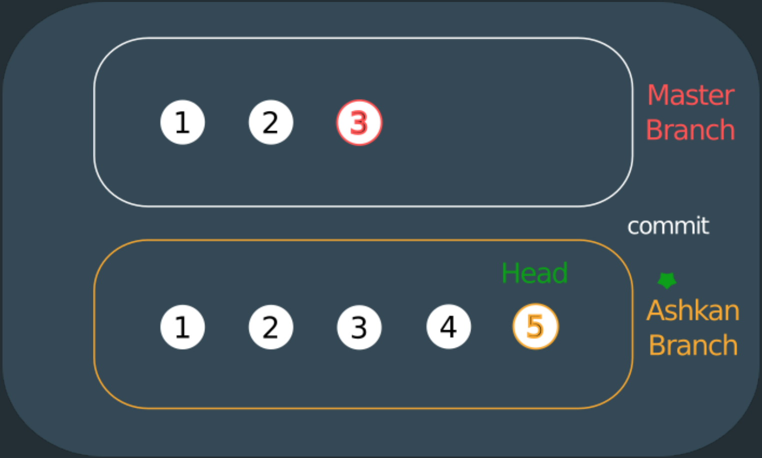
Head



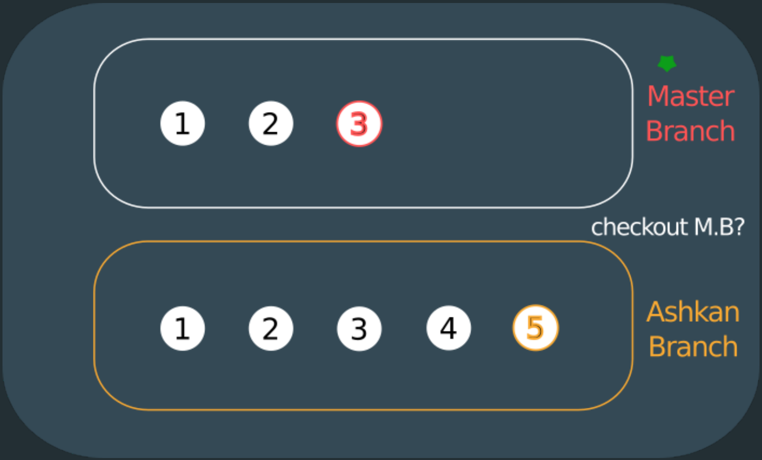
Head



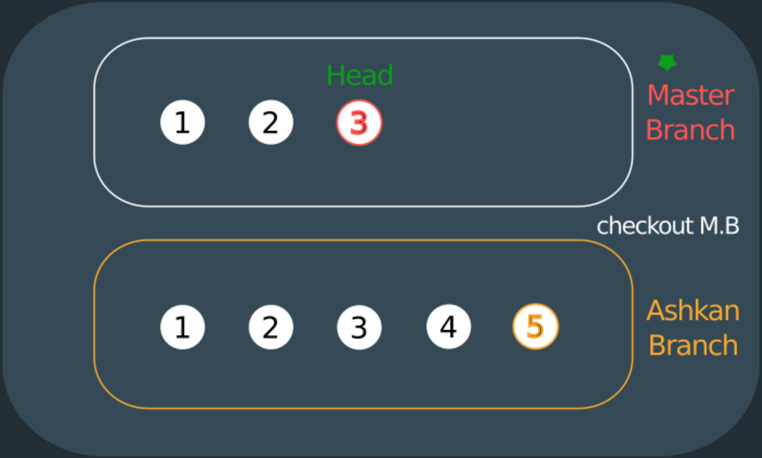
Head



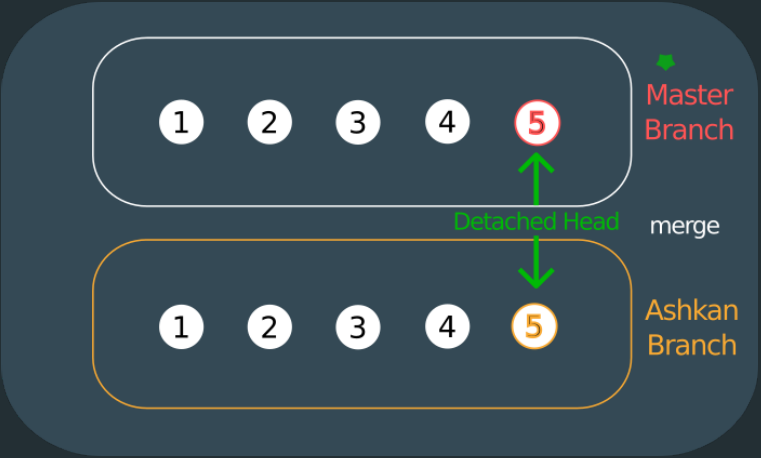
Head



Head



Head


1
2
3
4
5
Master
Branch
Detached Head
merge
1
2
3
4
5
Ashkan
Branch

Deleting Data

- Working Directory Files
- Unstaged Changes
- Staged Changes
- Latest Commit(s)
- Branches

Deleting Data

-Working Directory Files

- Move the file from working directory to trash
- `$ git rm <file name>`
- commit

Deleting Data

-Unstaged Changes

```
# remove unstaged changes in a file
$ git checkout <filename>
# or
$ git restore <filename>

# remove unstaged changes in all files
$ git checkout .
# or
$ git restore .
```

Deleting Data

-Unstaged Changes

```
# remove untracked file

# first check what will be removed
$ git clean -dn

# then force delete that
$ git chean -df
```

Deleting Data

-Staged Changes

```
# unstage staged file
$ git restore --staged <file name>

# delete unstaged change
$ git restore <file name>
```

Deleting Data

-Latest Commit(s)

```
# keep new files staged
$ git reset --soft HEAD~1

# keep new files unstaged
$ git reset HEAD~1

# remove new files
$ git reset --hard HEAD~1
```

Deleting Data

-Branches

```
$ git branch -D <branch name>
```

Gitignore

.gitignore file contains file that you don't want to track in git management.
you should create it in work directory.
each line one record.

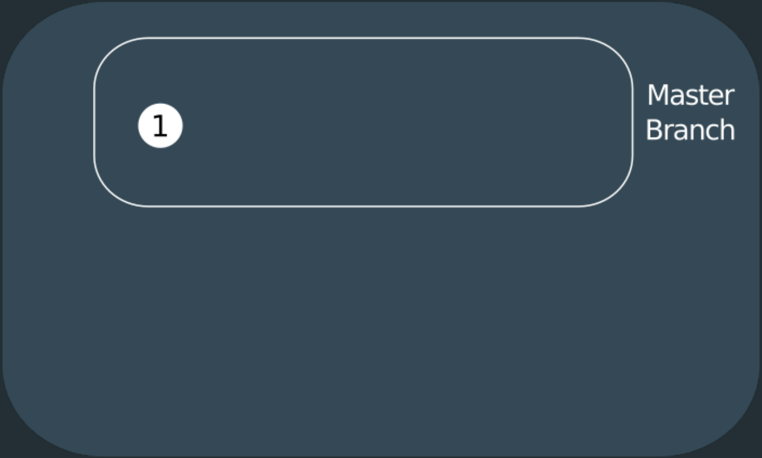
- '<filename>' ignores file
- '*.<extension>' ignore all files with a extension
- '!<file name>' remove file from being ignored
- '<dir>/*' ignores all files in directory

check this Web page for some gitignore files.

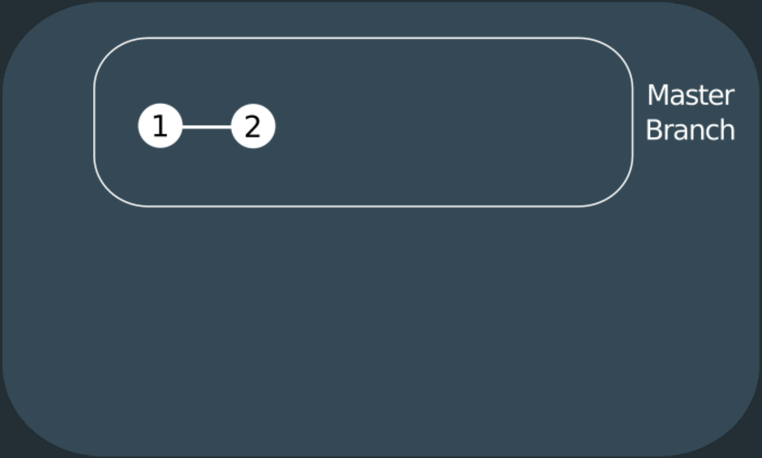
Restore Data

```
# check logs by following command
$ git reflog

# moving head to a commit
$ git checkout <commit>

# moving branch pointer to a commit
$ git branch -f <branch-name> <sha1-commit-hash>
```

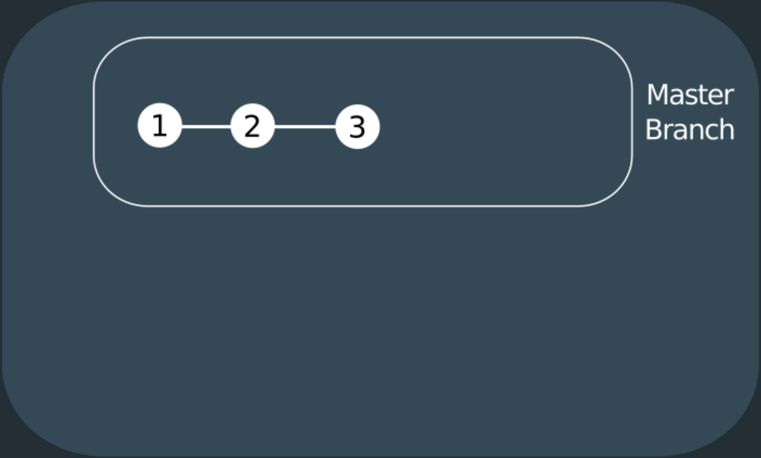
Fast Forward Merge



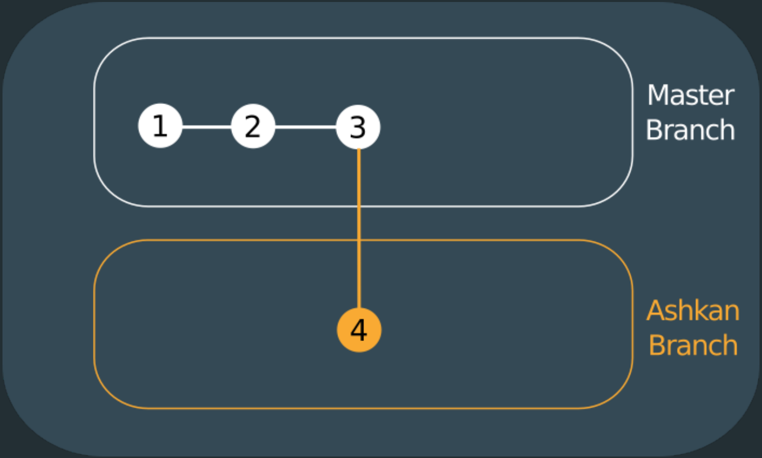
Fast Forward Merge



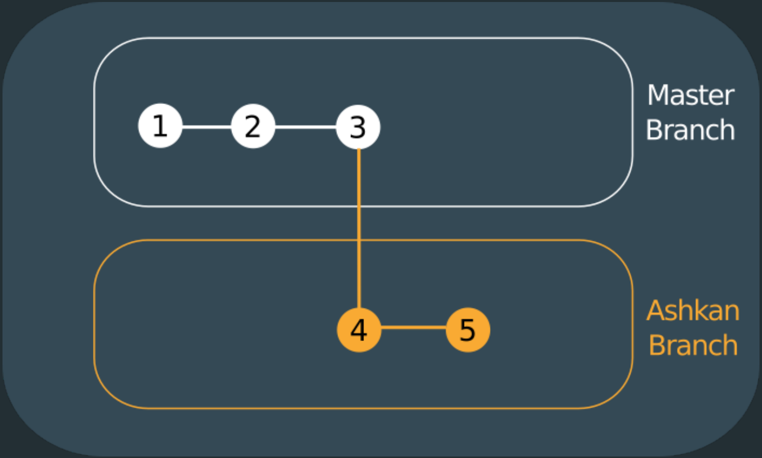
Fast Forward Merge



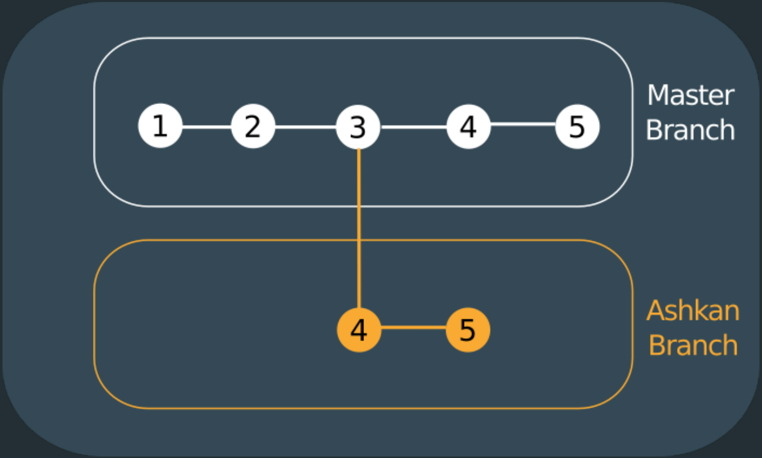
Fast Forward Merge



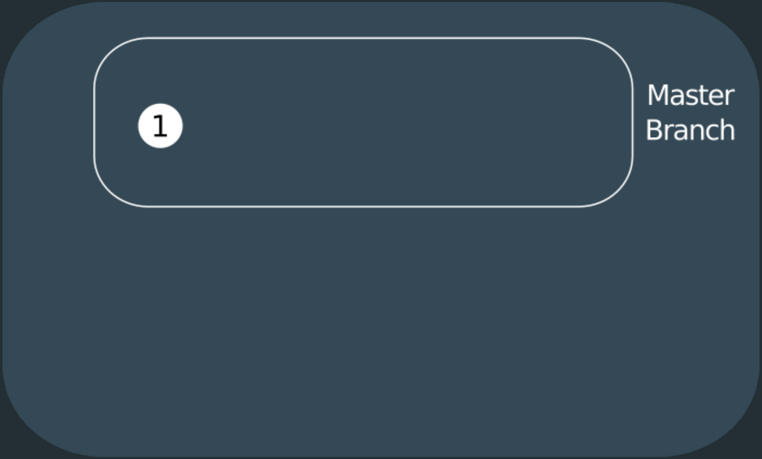
Fast Forward Merge



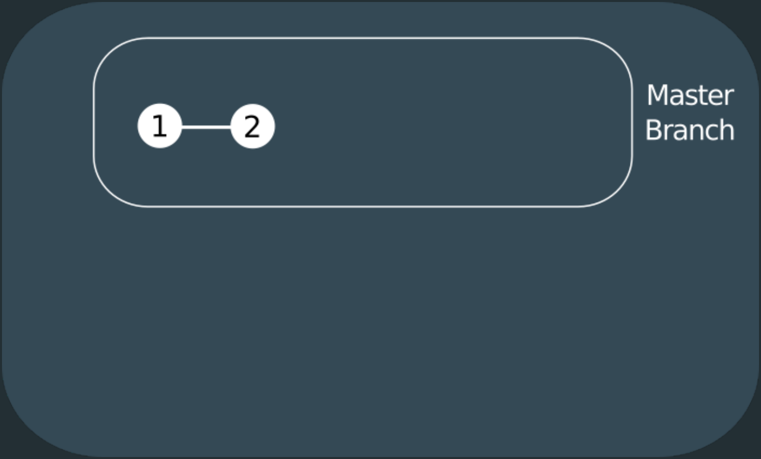
Fast Forward Merge



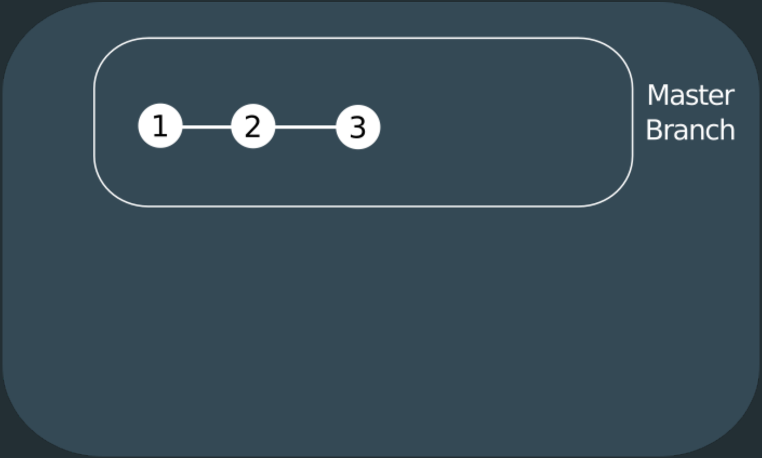
Merge



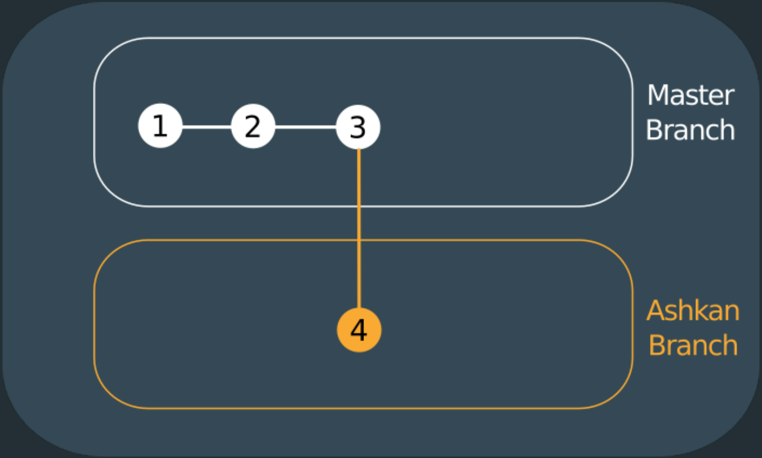
Merge



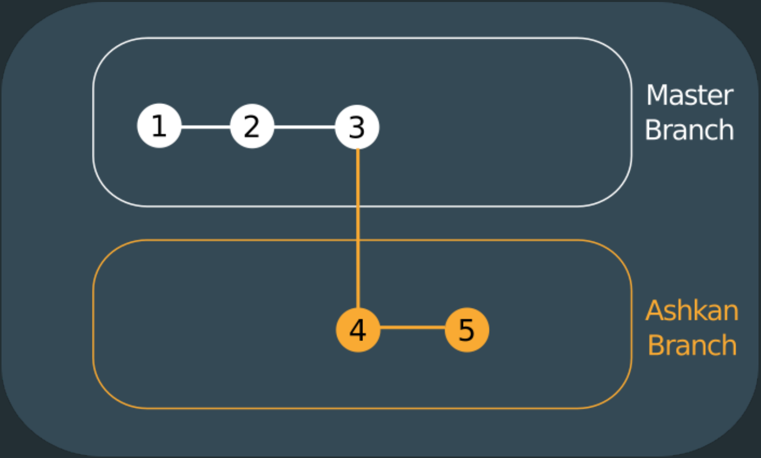
Merge



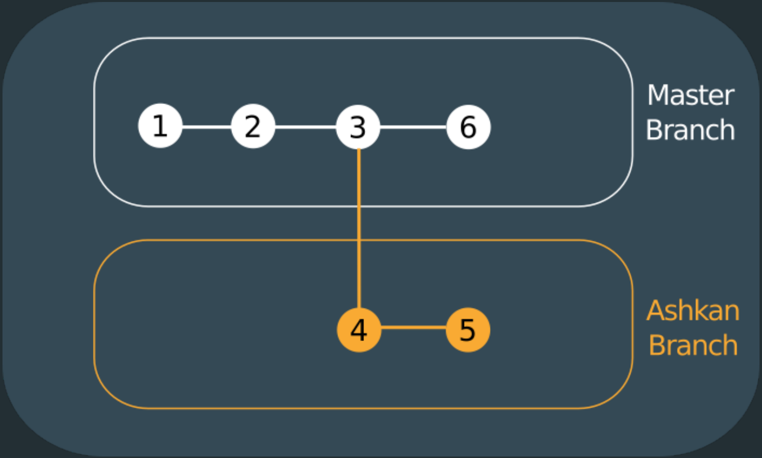
Merge



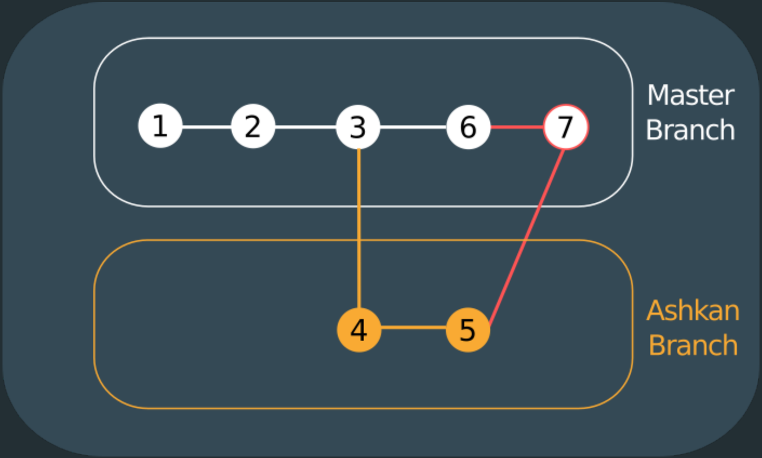
Merge



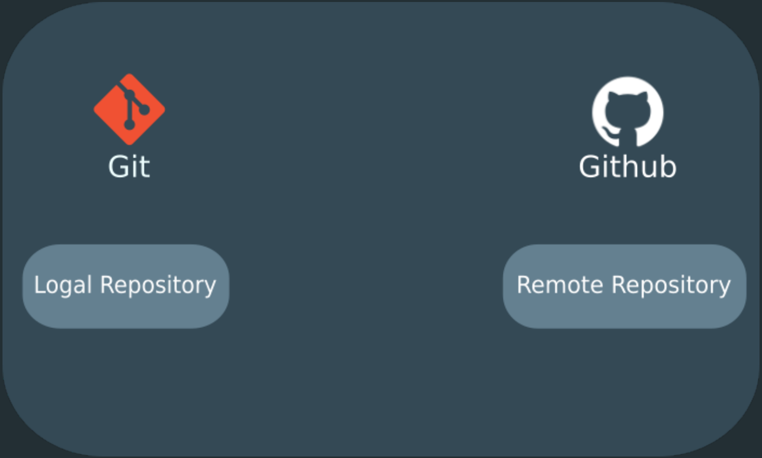
Merge



Merge



Github and Git



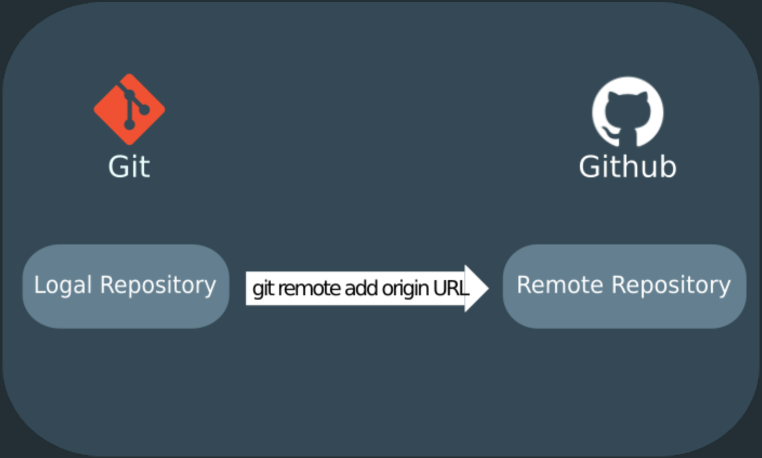
Git

Github

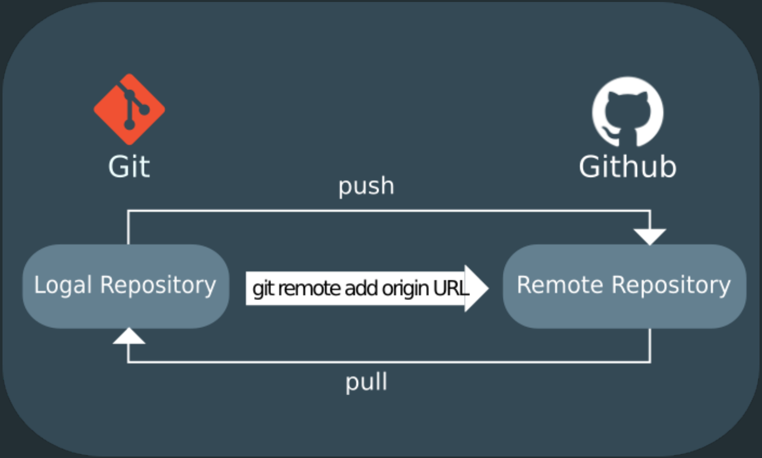
Logal Repository

Remote Repository

Github and Git



Github and Git



Github and Git

- push your local repository to remote repository

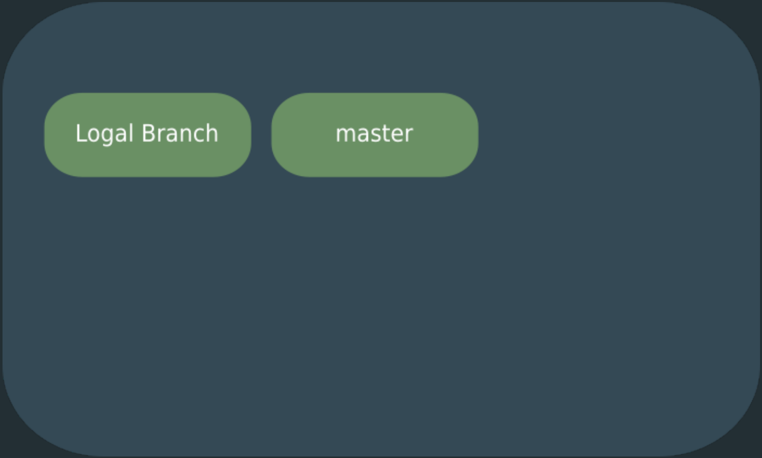
-
  ```
  # show local branches
  $ git branch

  # show local and remote branches
  $ git branch -a
  ```

Warning

in upcoming slides instead of local tracking branch should be remote tracking branch

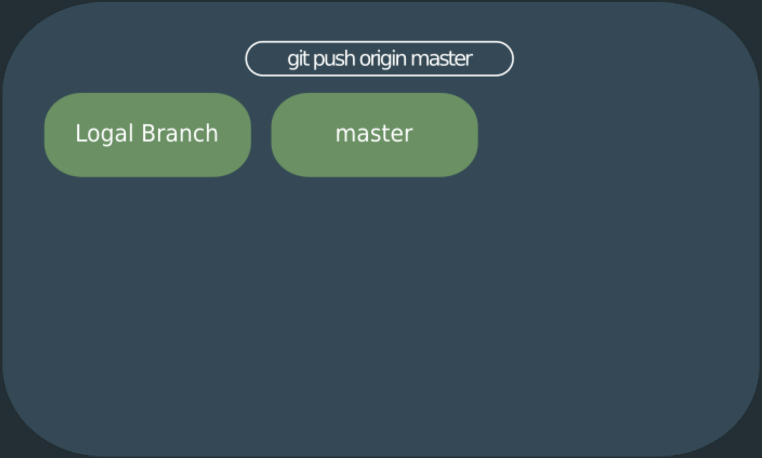
Github and Git

-branches



Github and Git

-branches



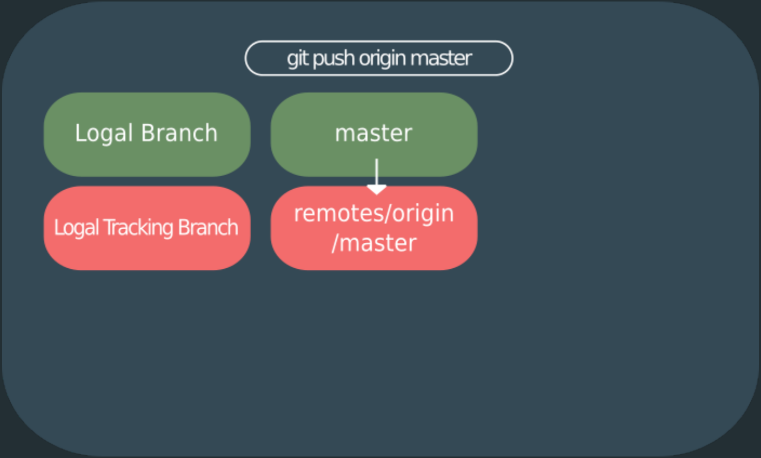
git push origin master

Logal Branch

master

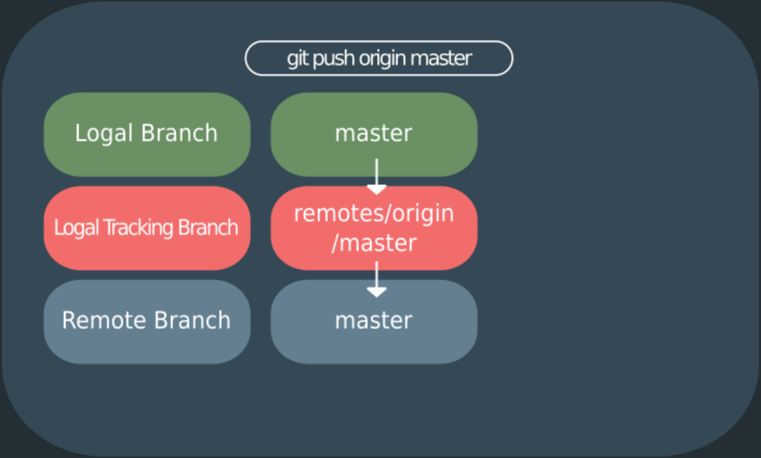
Github and Git

-branches



Github and Git

-branches



git push origin master

Logal Branch

master

Logal Tracking Branch

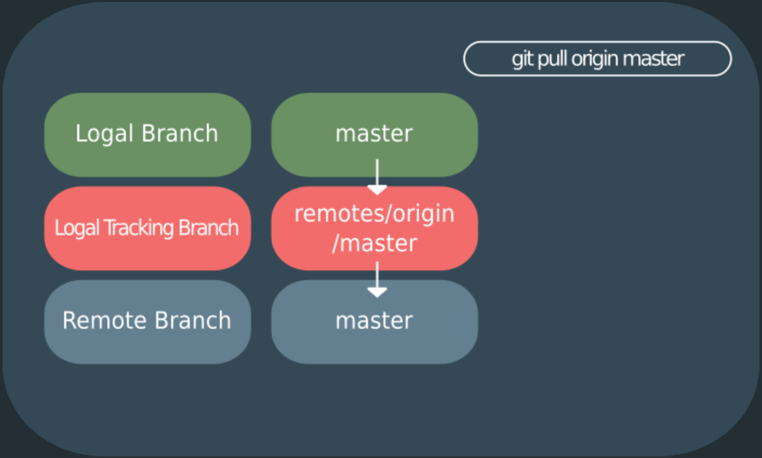
remotes/origin /master

Remote Branch

master

Github and Git

-branches



git pull origin master

Logal Branch

master

Logal Tracking Branch

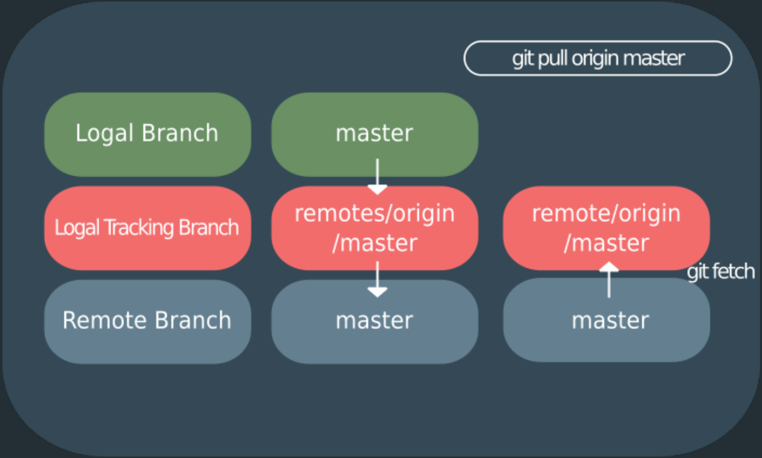
remotes/origin /master

Remote Branch

master

Github and Git

-branches



git pull origin master

Logal Branch

master

Logal Tracking Branch

remotes/origin /master

remote/origin /master

git fetch

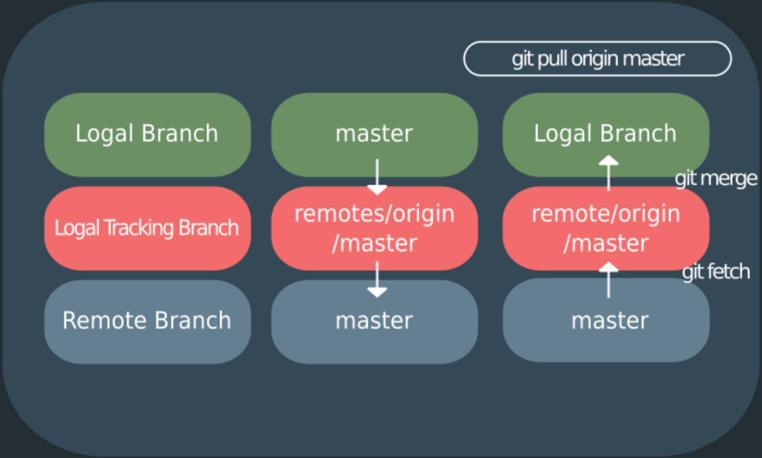
Remote Branch

master

master

Github and Git

-branches



git pull origin master

Logal Branch

master

Logal Branch

git merge

Logal Tracking Branch

remotes/origin /master

remote/origin /master

git fetch

Remote Branch

master

master

Github and Git

-add branch to remote repository

```
# show remote branches
$ git ls-remote

# sync remote branch and its local tracking branch
$ git fetch

# fast forward merge
$ merge <local tracking branch>

# fetch and merge
$ git pull <remote> <branch>
```

Branch type

- local branch
- remote branch
- remote tracking branch

Branch type

- local branch
- remote branch
- remote tracking branch
- local tracking branch

Branch type

-local tracking branch

Local reference to remote tracking branch

```
$ git push -u origing <branch>

# create a local tracking branch
$ git branch --track <name> <remote.t.b>

# show type of branches
$ git branch -vv
```