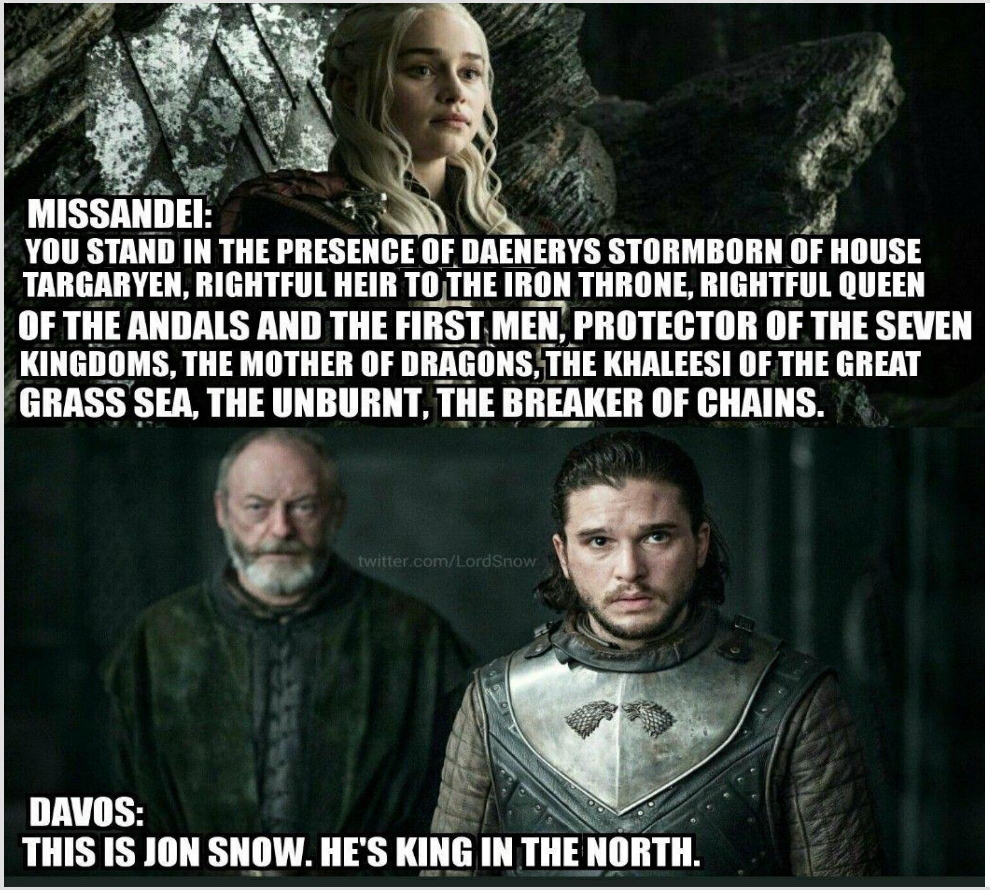
01

S&T

OSC-Open Source Community

Powered by:
mahmoud mohamed

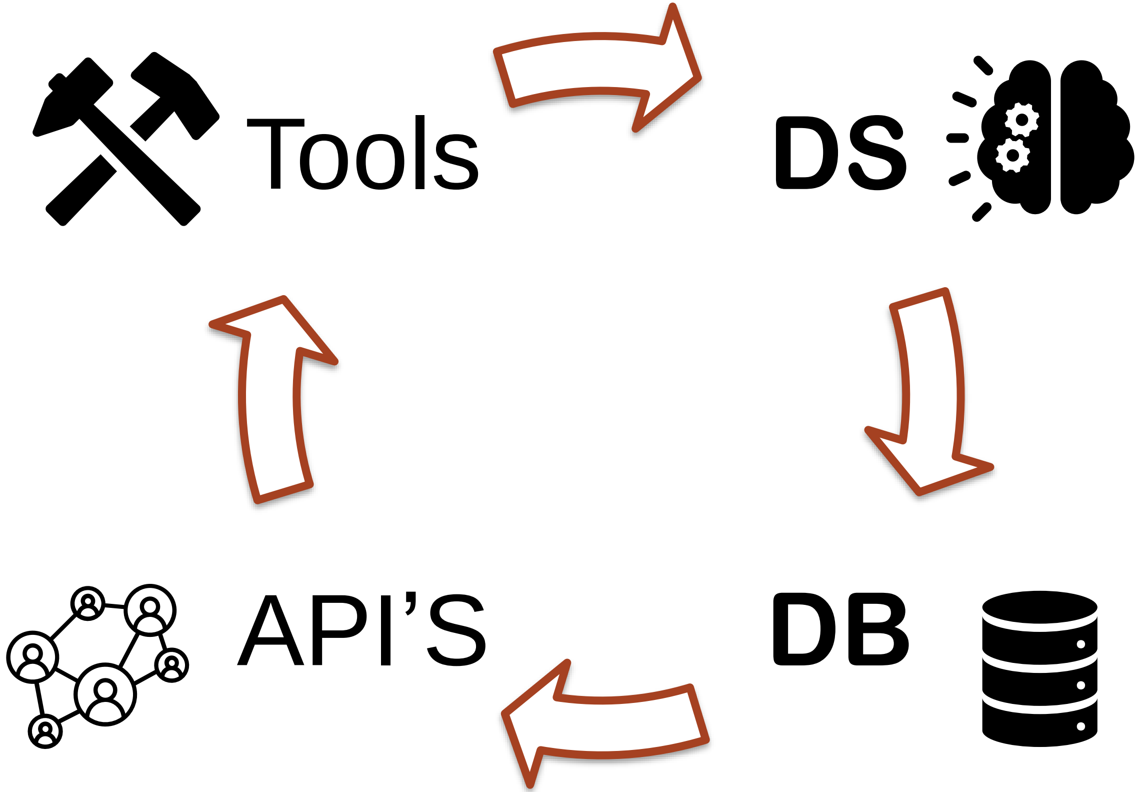

MISSANDEI:
YOU STAND IN THE PRESENCE OF DAENERYS STORMBORN OF HOUSE TARGARYEN, RIGHTFUL HEIR TO THE IRON THRONE, RIGHTFUL QUEEN OF THE ANDALS AND THE FIRST MEN, PROTECTOR OF THE SEVEN KINGDOMS, THE MOTHER OF DRAGONS, THE KHALEESI OF THE GREAT GRASS SEA, THE UNBURNT, THE BREAKER OF CHAINS.
twitter.com/LordSnow
DAVOS:
THIS IS JON SNOW. HE'S KING IN THE NORTH.

Who???

Your turn

- **Big family**

I'm ready.



content

Tools

Tools

Git + Git Hub

Vs code

CV

Linked in

Data Structures

PS

Data Base

EF

API'S

MVC

Agenda

- Git History
- Why Git
- The Basic workflow of Git
- Git Common Commands
- GitHub
- Hands On

Let’s Go

You
136 contributions in the last year
Contribution settings
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan
Mon
Wed
Fri
Learn how we count contributions.
Less
More
Vs The guy she told you not to be worried about

1953

1983

Richard Stallman

Importance of Open Source

Freedom

Contribute & enhance your resume

No malicious code

Learning

Don’t reinvent the wheel

OSS Examples

CakePHP
git
php
openmoko
mongoDB
MySQL

1969

Linus Torvalds

1992

21 Linux

10

2002

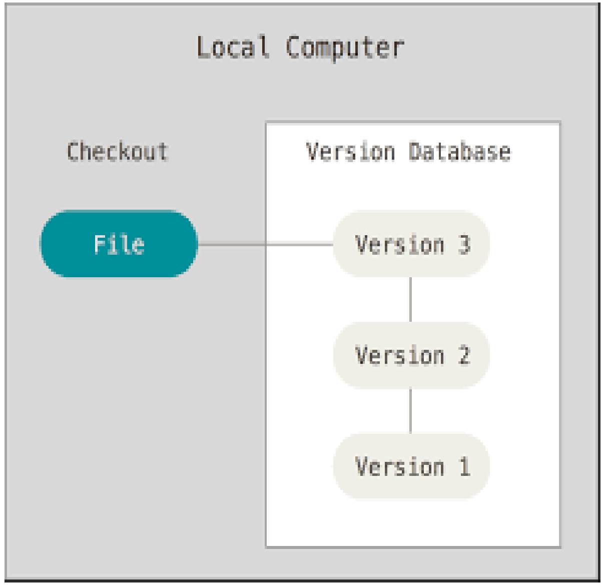
BitKeeper
Scalable Version Control

2005

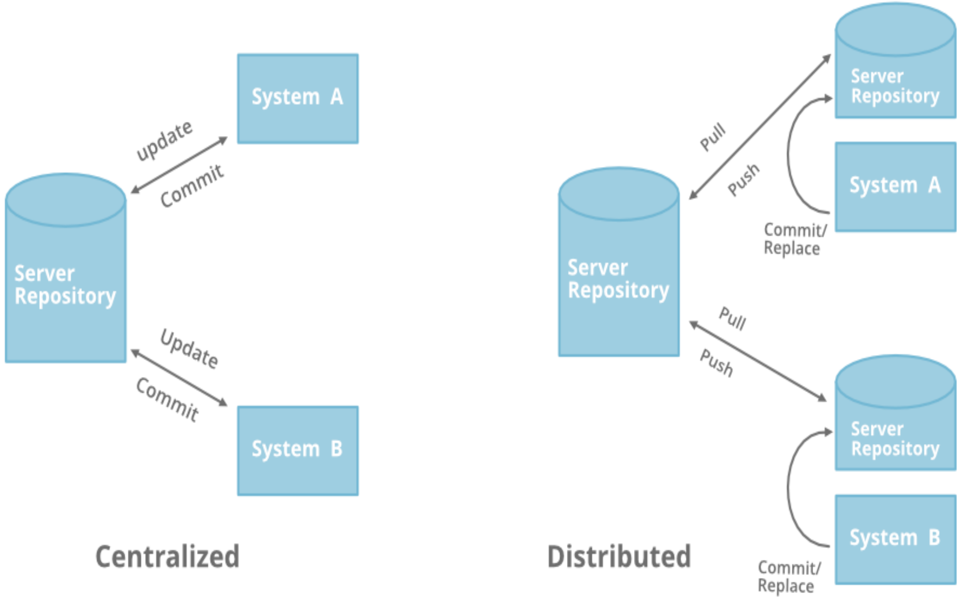
git

Version Control Overview

Version Control Systems 1975



local



git ? ?

Jerk: contemptible or foolish person

What is Git?

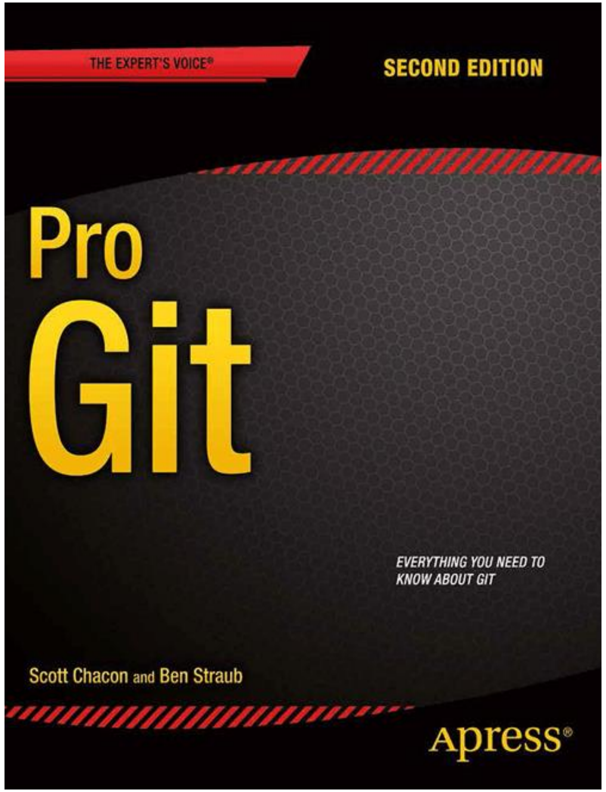
Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development.

Why Learn Git?

- **Tracking code changes**
- **Tracking who made changes**
- **Coding collaboration**
- Snapshots Not Differences
- Speed
- Simple design

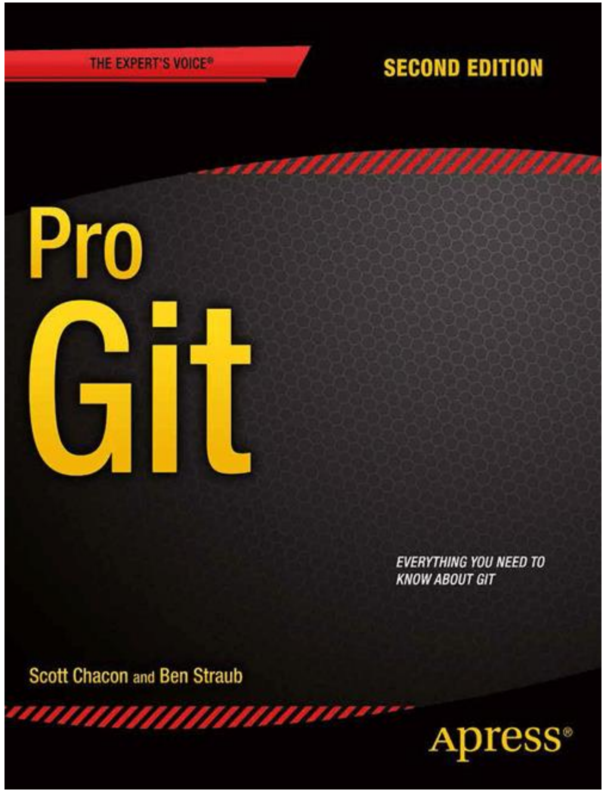
And it's not limited to just that, it can do more.


THE EXPERT'S VOICE®
SECOND EDITION
Pro Git
EVERYTHING YOU NEED TO KNOW ABOUT GIT
Scott Chacon and Ben Straub
Apress®

Why Learn Git?

- **Tracking code changes**
- **Tracking who made changes**
- **Coding collaboration**
- Snapshots Not Differences
- Speed
- Simple design

And it's not limited to just that, it can do more.


THE EXPERT'S VOICE®
SECOND EDITION
Pro Git
EVERYTHING YOU NEED TO KNOW ABOUT GIT
Scott Chacon and Ben Straub
Apress®

Incremental VCS

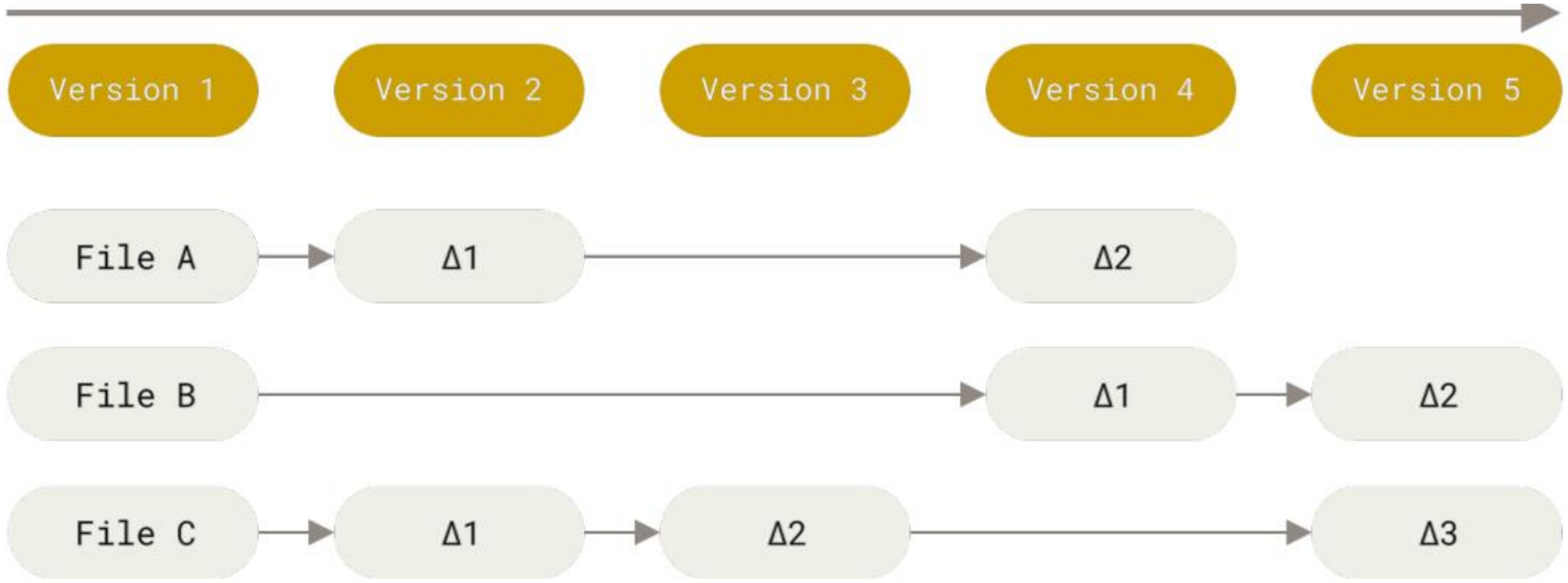


Figure 4. Storing data as changes to a base version of each file

snapshots VCS

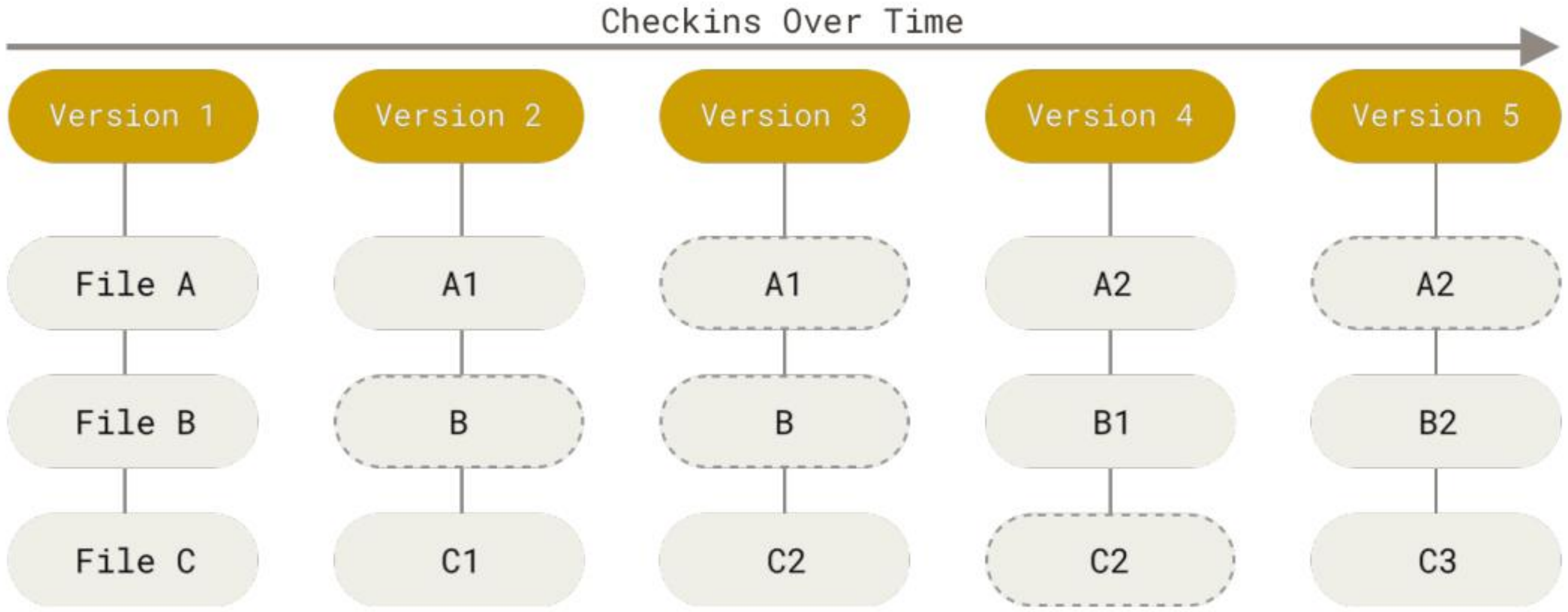


Figure 5. Storing data as snapshots of the project over time

What we need in Git??!!

TRACK EVERY THING

OS INDEPENDENT

UNIQUE ID

TRACK HISTORY

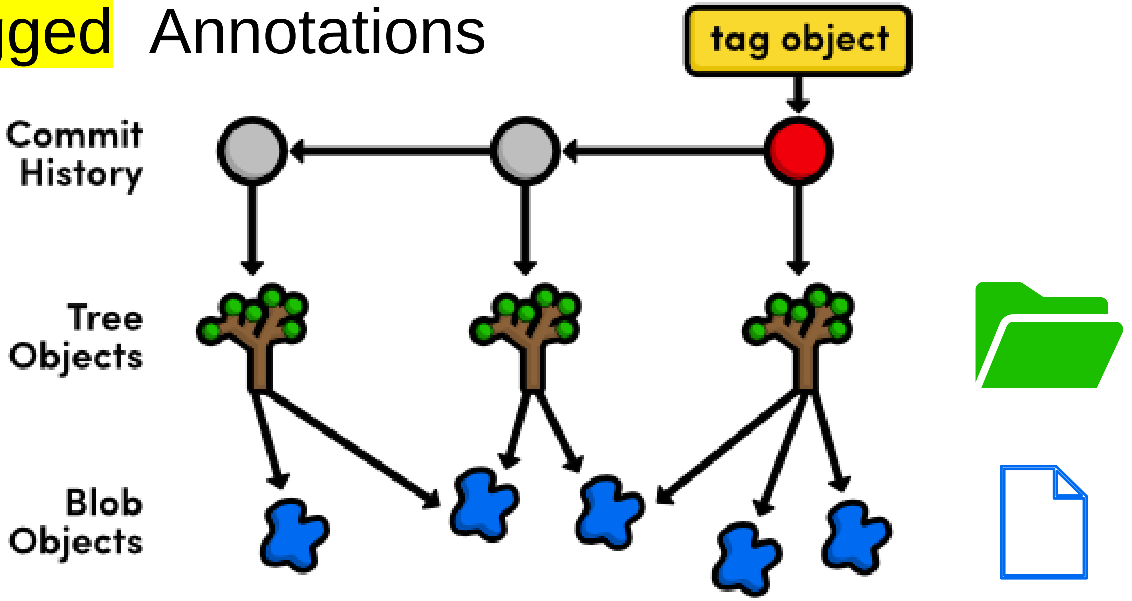
Track every thing

Git objects

(1) File becomes Blob why Blob contains content + meta data

(2) Folder becomes Tree why Tree contains content + meta data

(3) Commit Tagged Annotations



What we need in Git??!!

TRACK EVERY THING

OS INDEPENDENT

UNIQUE ID

TRACK HISTORY

OS INDEPENDENT

- Add .git hidden folder to WD

What we need in Git??!!

TRACK EVERY THING

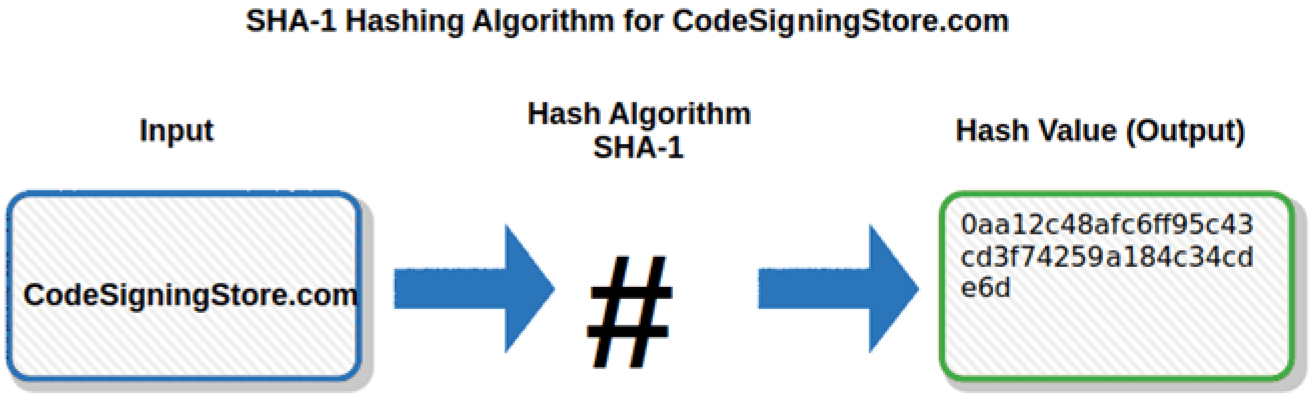
OS INDEPENDENT

UNIQUE ID

TRACK HISTORY

UNIQUE ID

- Make for each object Hashed value famous :- SHA-1


SHA-1 Hashing Algorithm for CodeSigningStore.com
Input
Hash Algorithm SHA-1
Hash Value (Output)
CodeSigningStore.com
#
0aa12c48afc6ff95c43cd3f74259a184c34cde6d

Not only content but we add[{Blob || tree} size content null char]
40 char

What we need in Git??!!

TRACK EVERY THING

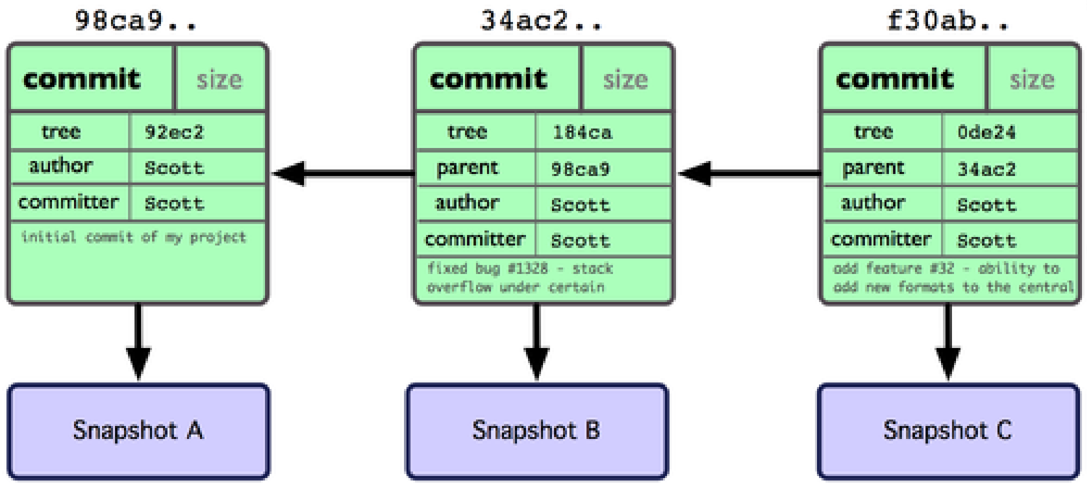
OS INDEPENDENT

UNIQUE ID

TRACK HISTORY

TRACK HISTORY

➢ Compare between hash value to new file and old



Linked list

What we need in Git??!!

TRACK EVERY THING

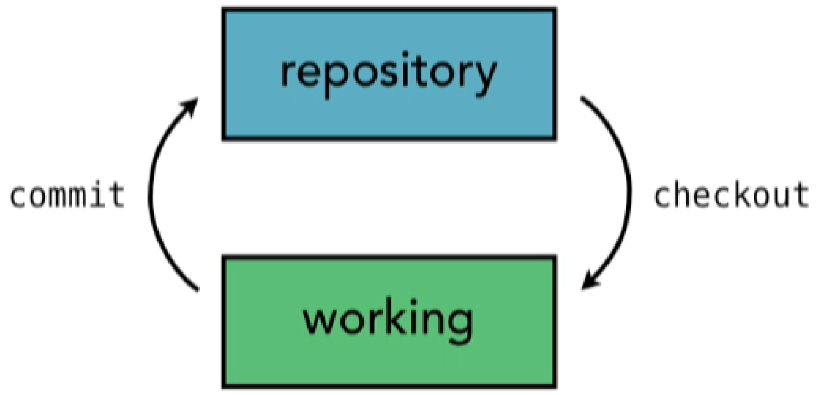
OS INDEPENDENT

UNIQUE ID

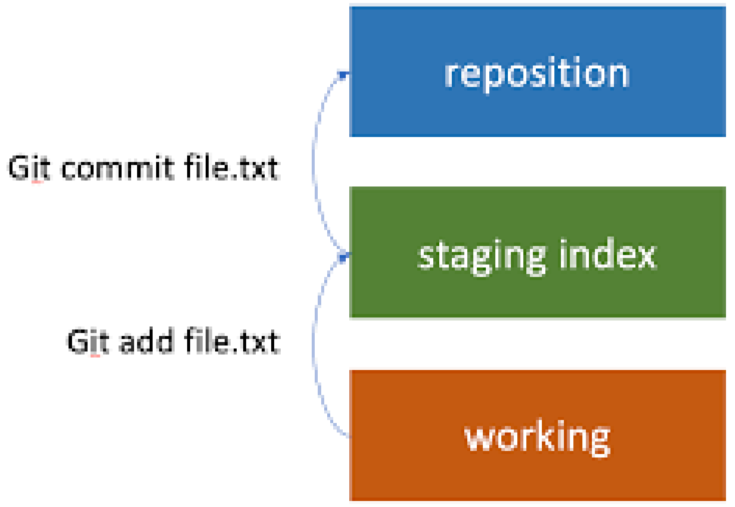
TRACK HISTORY

Architecture

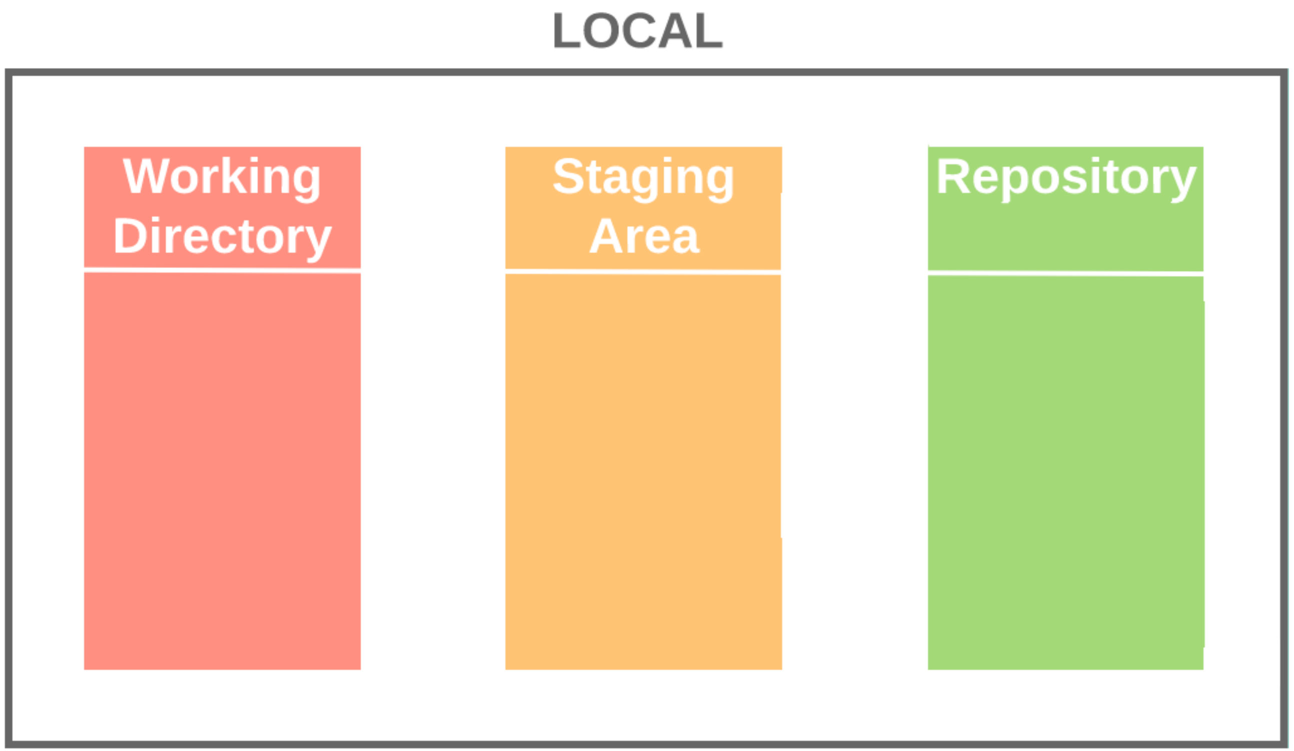
2 Tree

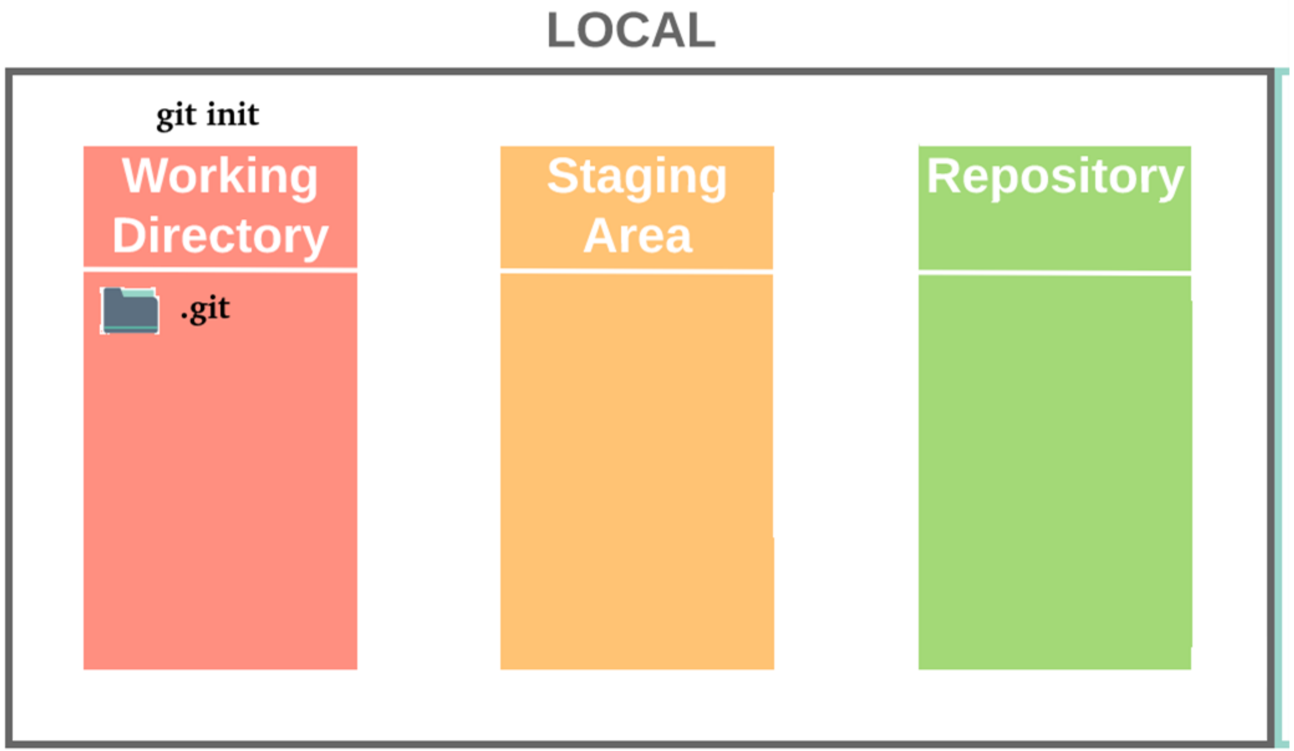

repository
commit
checkout
working

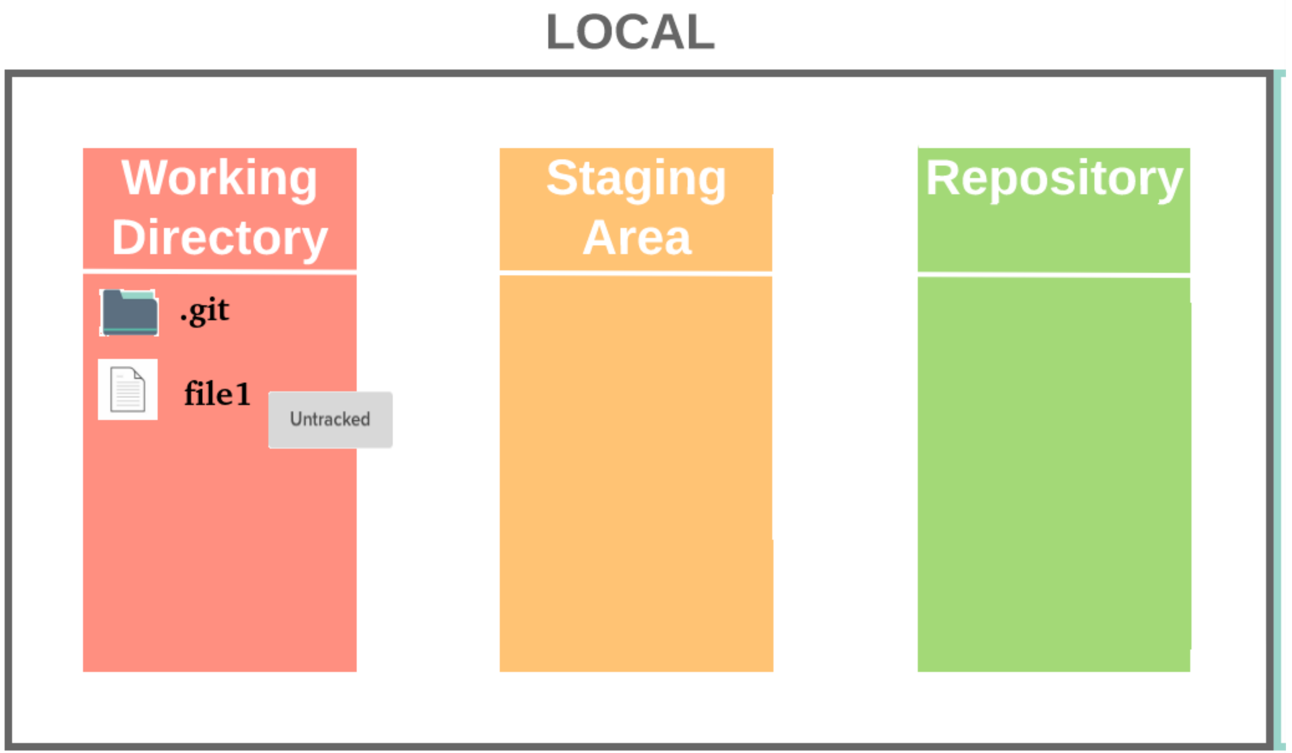
3 Tree

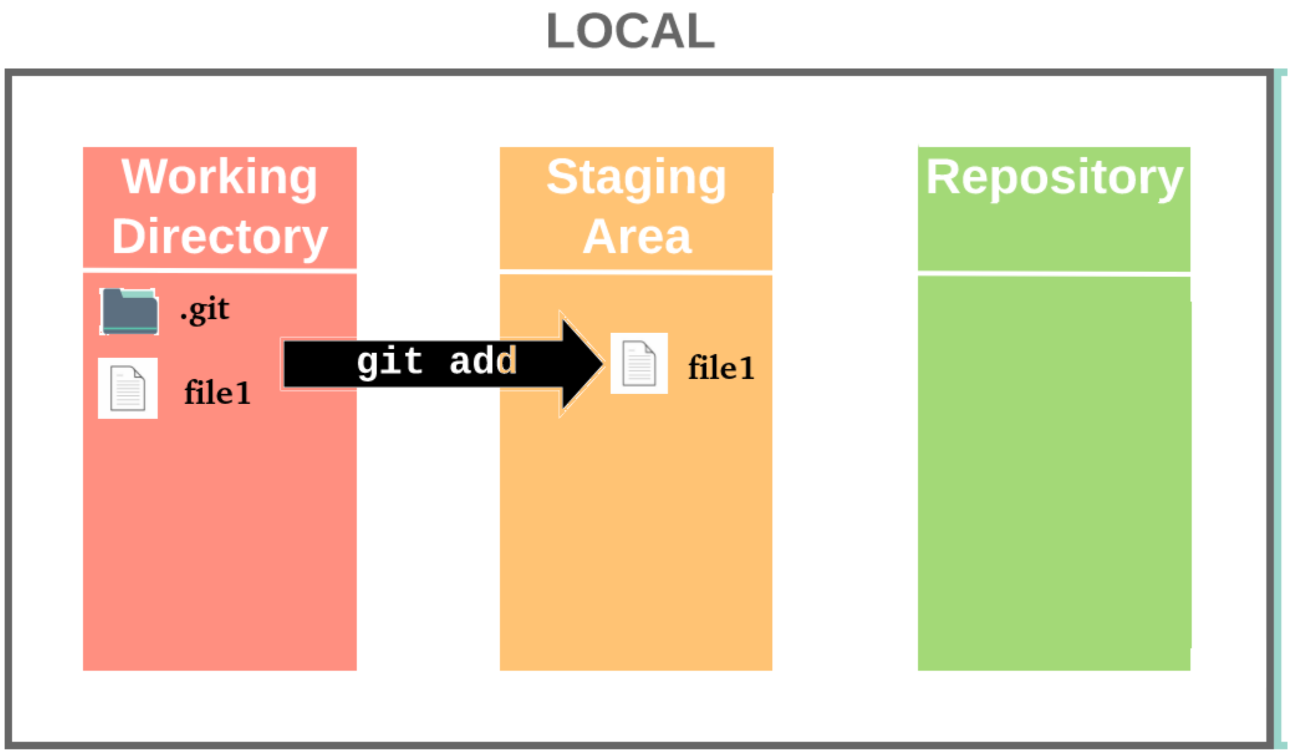

reposition
Git commit file.txt
staging index
Git add file.txt
working

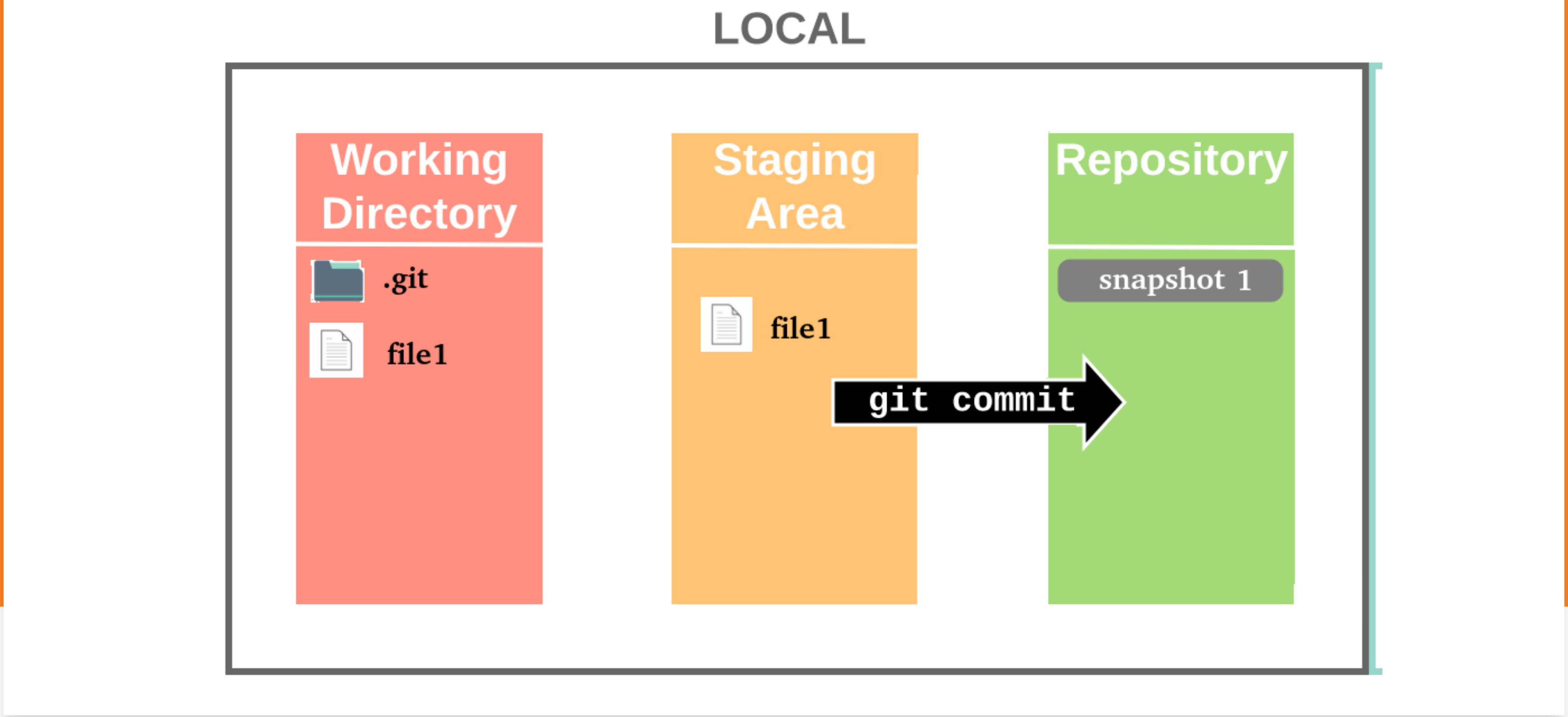
The Basic Workflow of Git

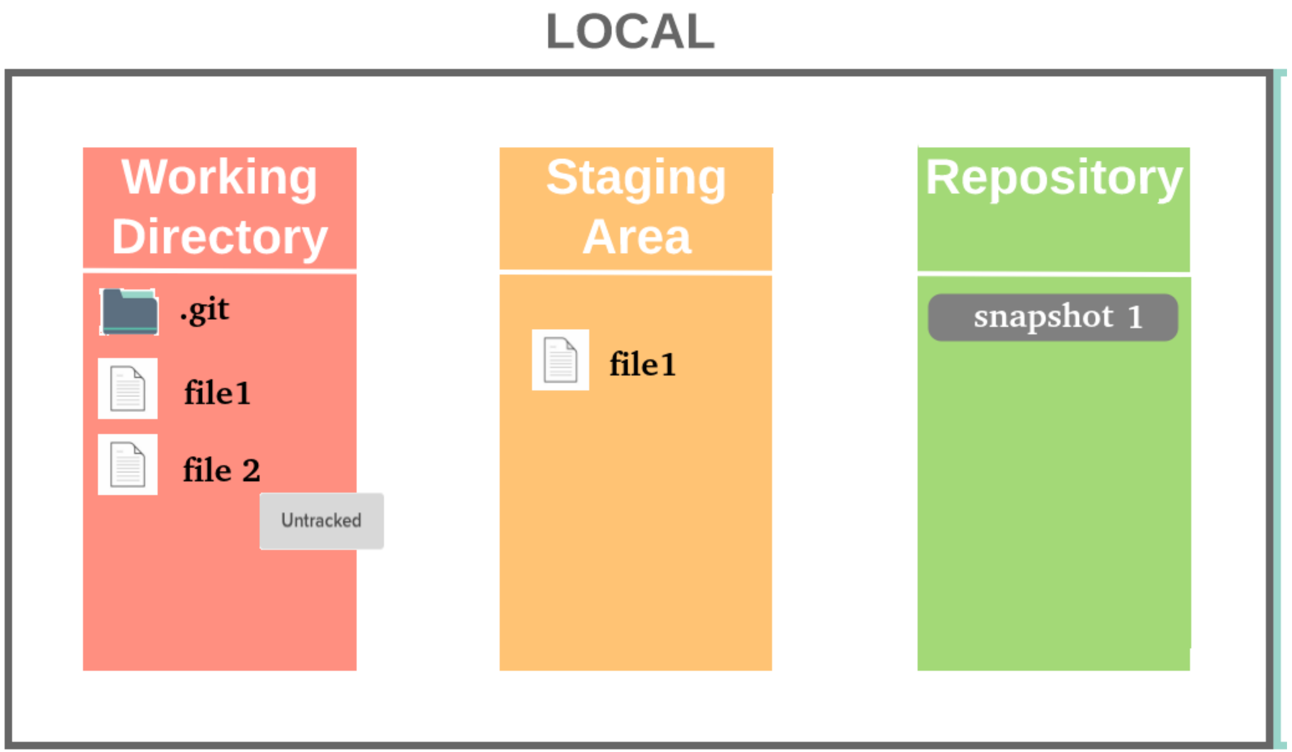

LOCAL
Working Directory
Staging Area
Repository

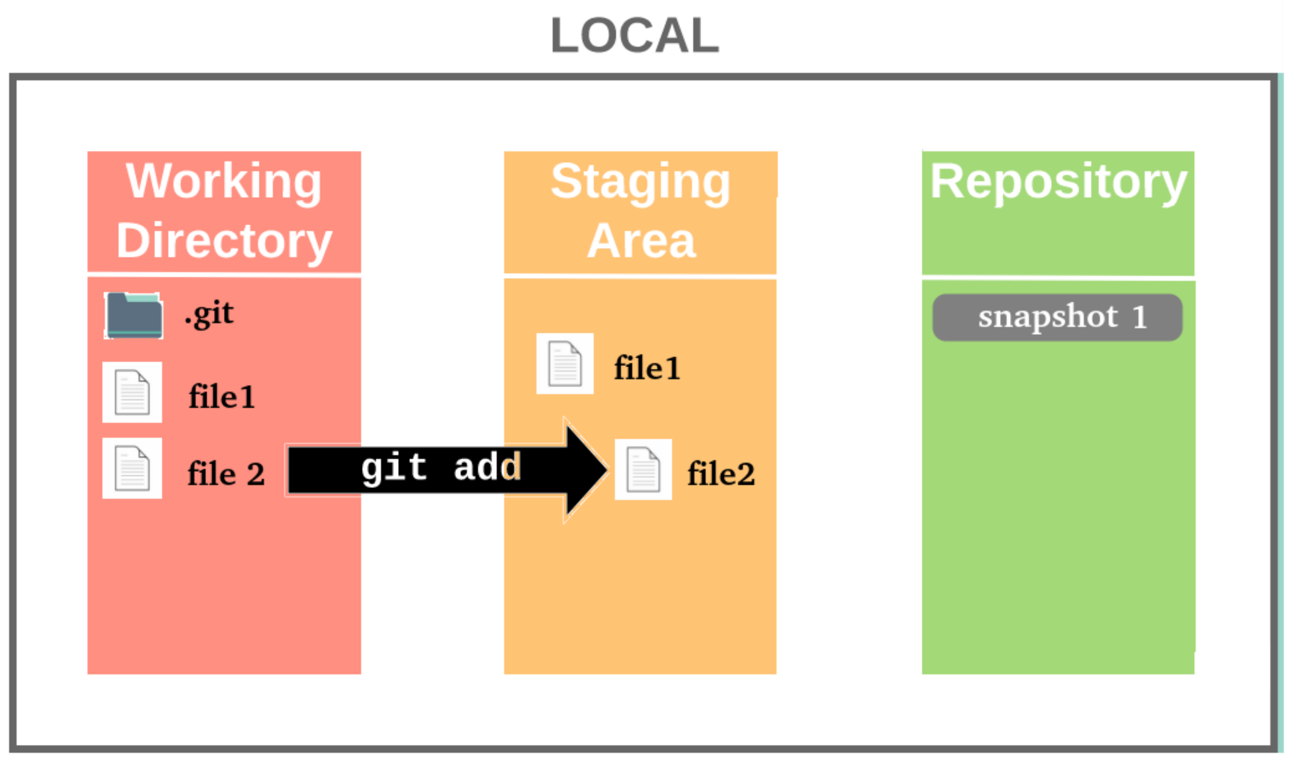

LOCAL
git init
Working Directory
.git
Staging Area
Repository


LOCAL
Working Directory
.git
file1
Untracked
Staging Area
Repository

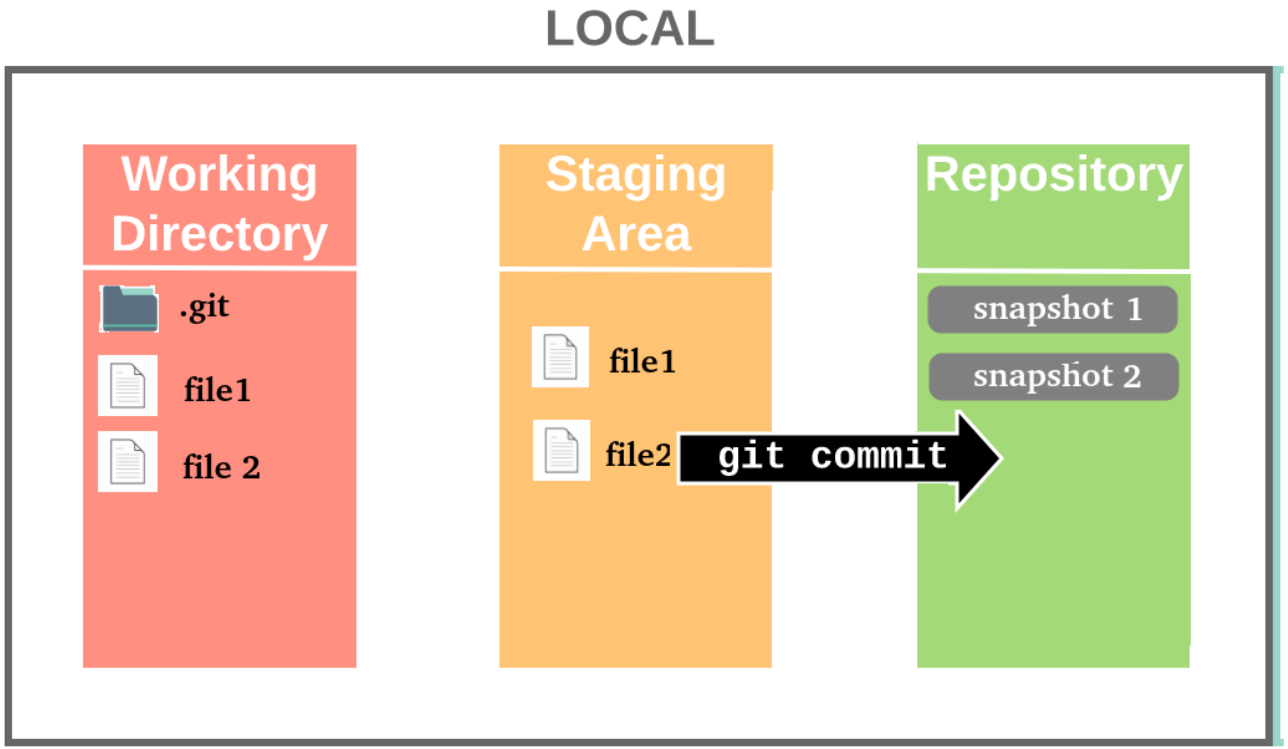

LOCAL
Working Directory
.git
file1
git add
Staging Area
file1
Repository


LOCAL
Working Directory
.git
file1
Staging Area
file1
git commit
Repository
snapshot 1

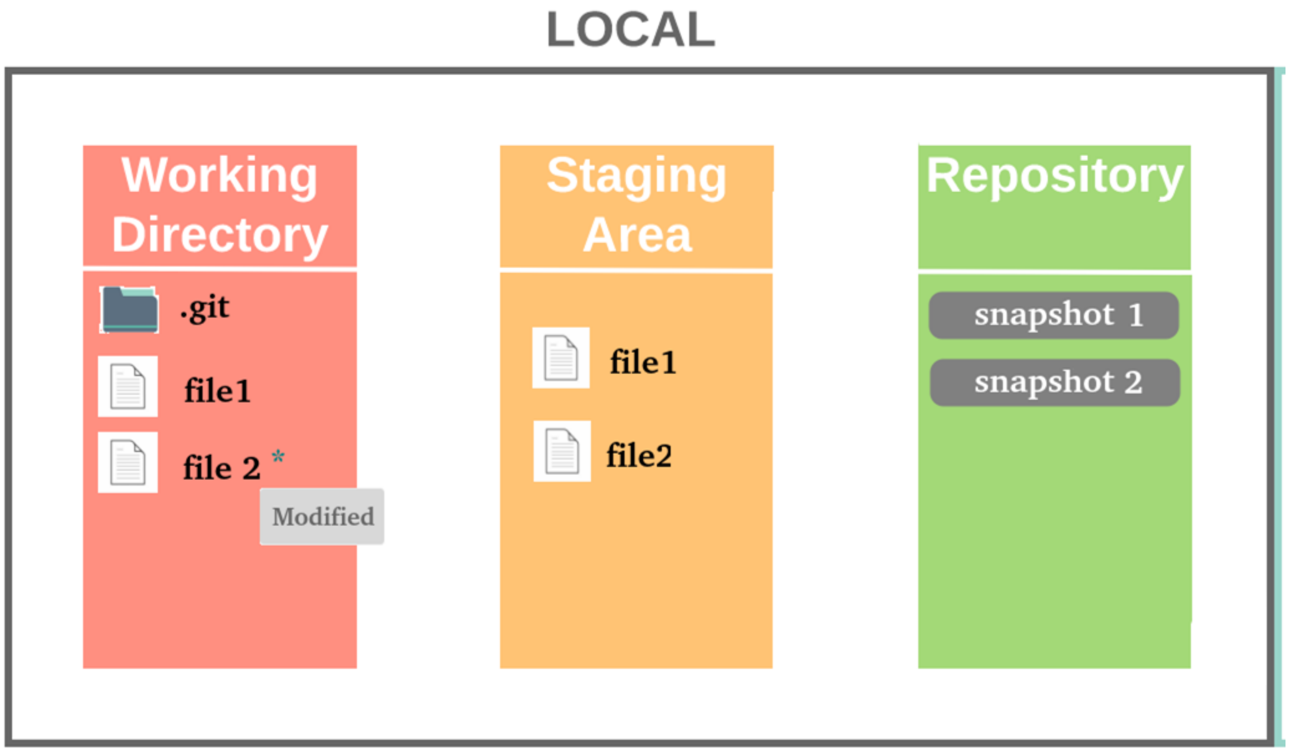

LOCAL
Working Directory
.git
file1
file 2
Untracked
Staging Area
file1
Repository
snapshot 1



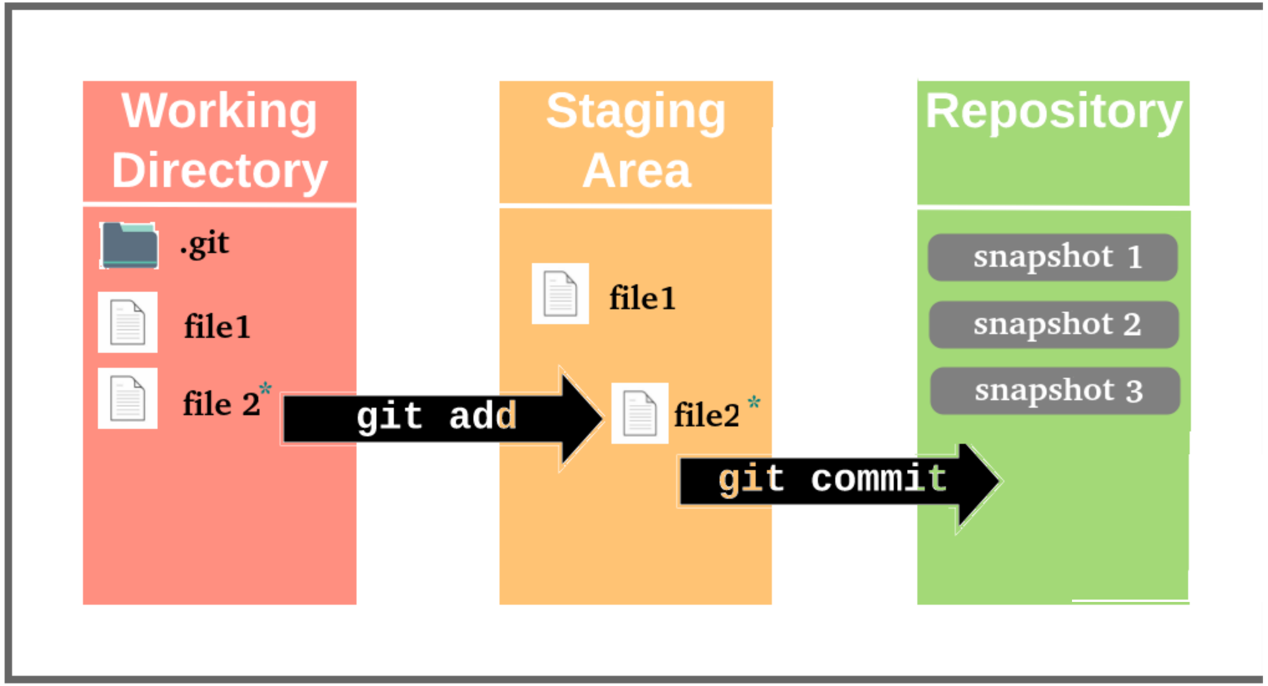
LOCAL
Working Directory
.git
file1
file 2
git add
Staging Area
file1
file2
Repository
snapshot 1


LOCAL
Working Directory
.git
file1
file 2
Staging Area
file1
file2
git commit
Repository
snapshot 1
snapshot 2

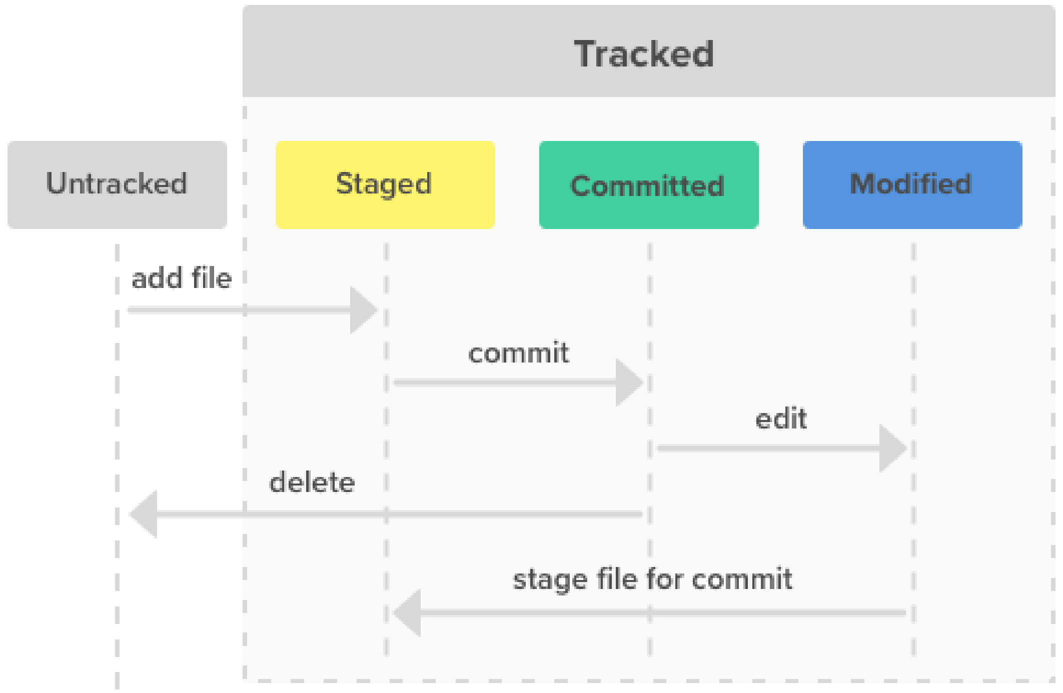
What if you modify a tracked file?



LOCAL


Working Directory
.git
file1
file 2*
git add
Staging Area
file1
file2 *
git commit
Repository
snapshot 1
snapshot 2
snapshot 3

Git Files Status



Git Commit

Commit

ID
Author & Email
Date/Time
Message
Complete snapshot

Let’s try it now

Git Common Commands

How to Write Git Commands?

Git commands format is

 git <command> [<args>]

Examples:

- git init
- git add main.cpp
- git commit -m “add main.cpp”

Managing Repository

add, remove, status, commit, clean

status

Describe what’s going on the repository

- Detailed status: status
- Brief status: status -s

add

Moves untracked changes to staging area

- Add a file/directory: add <filename,dir name>
- Add all files: add . or add —all
- Add files with patterns: add <pattern>, Ex: git add *.txt: adds all txt files

remove

Removes added files/directories from staging area to untracked

- Remove newly added files: rm –cached
- Remove newly added directories: rm -r –

  cached

restore

Moves changes from staging area to untracked

- restore file initial state: restore <filename>
- move to untracked: restore —staged

  <filename>

clean

Remove files that are untracked

- Removes any files that aren’t in staging

  area: clean

commit

Takes a snapshot of working directory

- Commit with message: commit -m “message”

log

Watch your timeline

- Detailed history: log
- Brief history: log –oneline
- Graph history: log –graph
- Git reflog show all flow of commits

diff

View changes in detail

- Detailed changes to file: diff <file-name>
- Brief changes to file: diff –stat <file-name>
- Brief changes to all: diff –stat <commit-code>

amend

Changes last commit

- Change message: commit –amend -m “new message”
- Change content: commit –amend –no-edit

checkout

Travel through your timeline

- One file checkout <filename>
- Whole directory: checkout <commit-code>

reset

Travel through the past

- Go to previous commit but keep changes

  reset <commit-code>

- Go to previous commit and ignore changes:

  reset <commit-code> –hard

What’s .gitignore?

A gitignore file specifies intentionally untracked files that Git should ignore.

Files already tracked by Git are not affected

Why .gitignore?

Untracked files aren’t the same, we can split them to two categories:

- Files you want to share, you might add to staging area
- Files you won’t share, you will never add to staging area → (why should git watch this?) That’s the neet part it shouldn’t

How to .gitignore?

Each line will describe either :

- A file name ex: main.cpp

OR

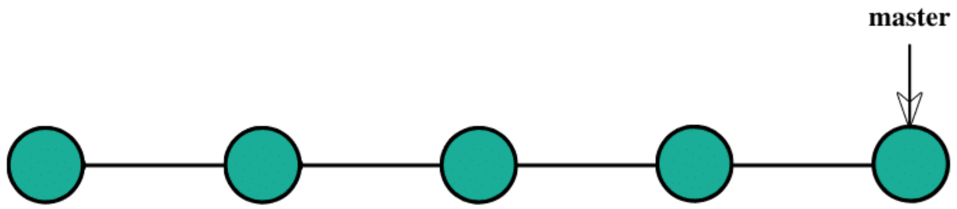
- A pattern ex: *.

git push
git add .

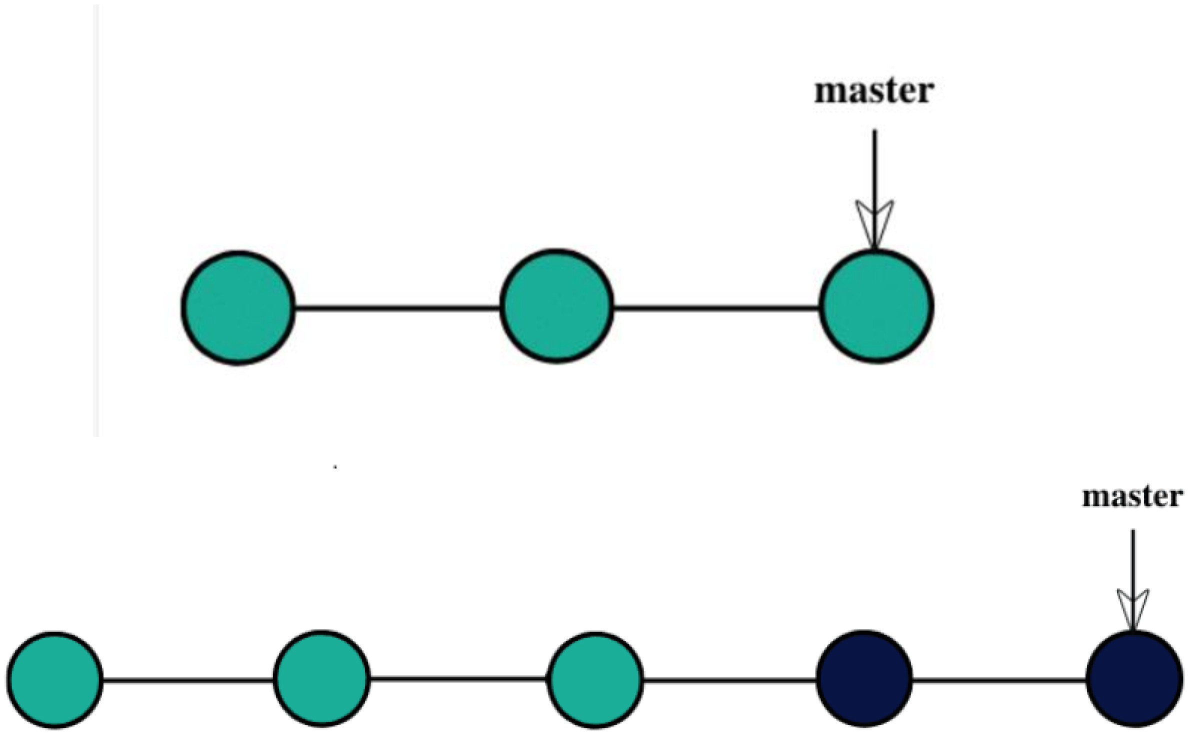
GIT PUSH --FORCE
PROBLEM SOLVED
makeameme.org

Break

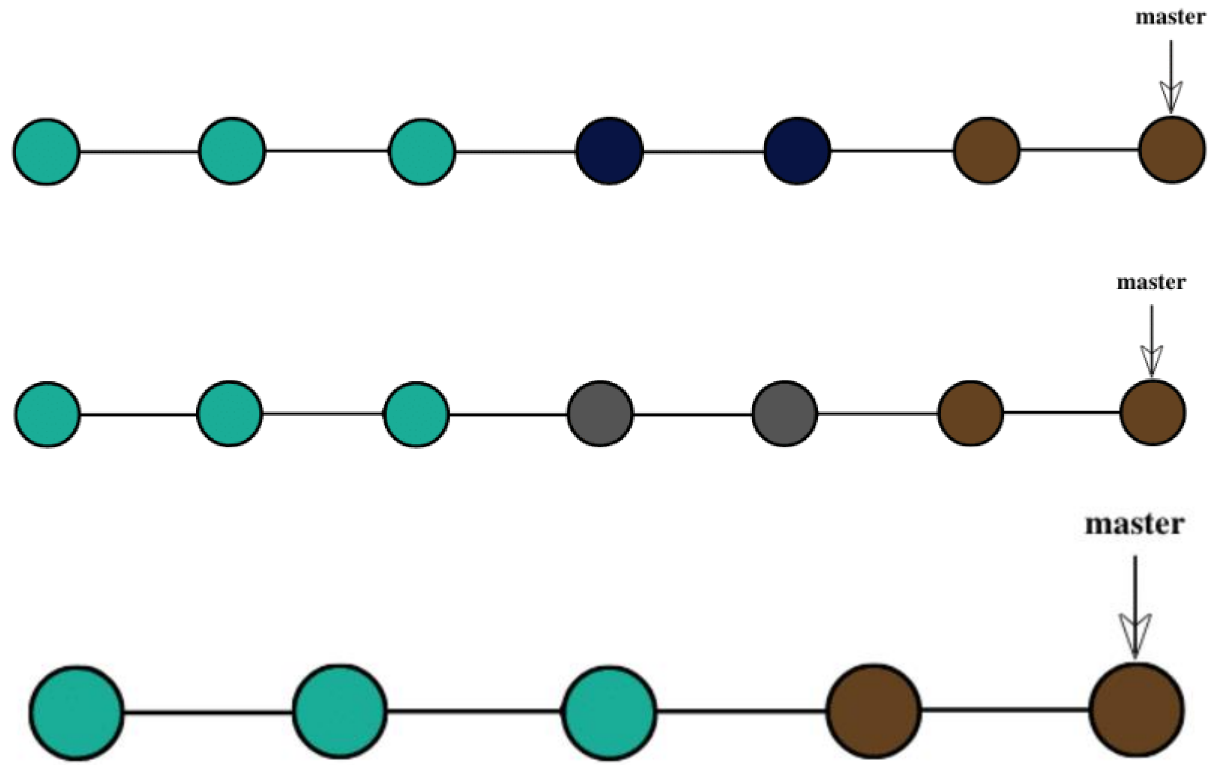
Git Branches

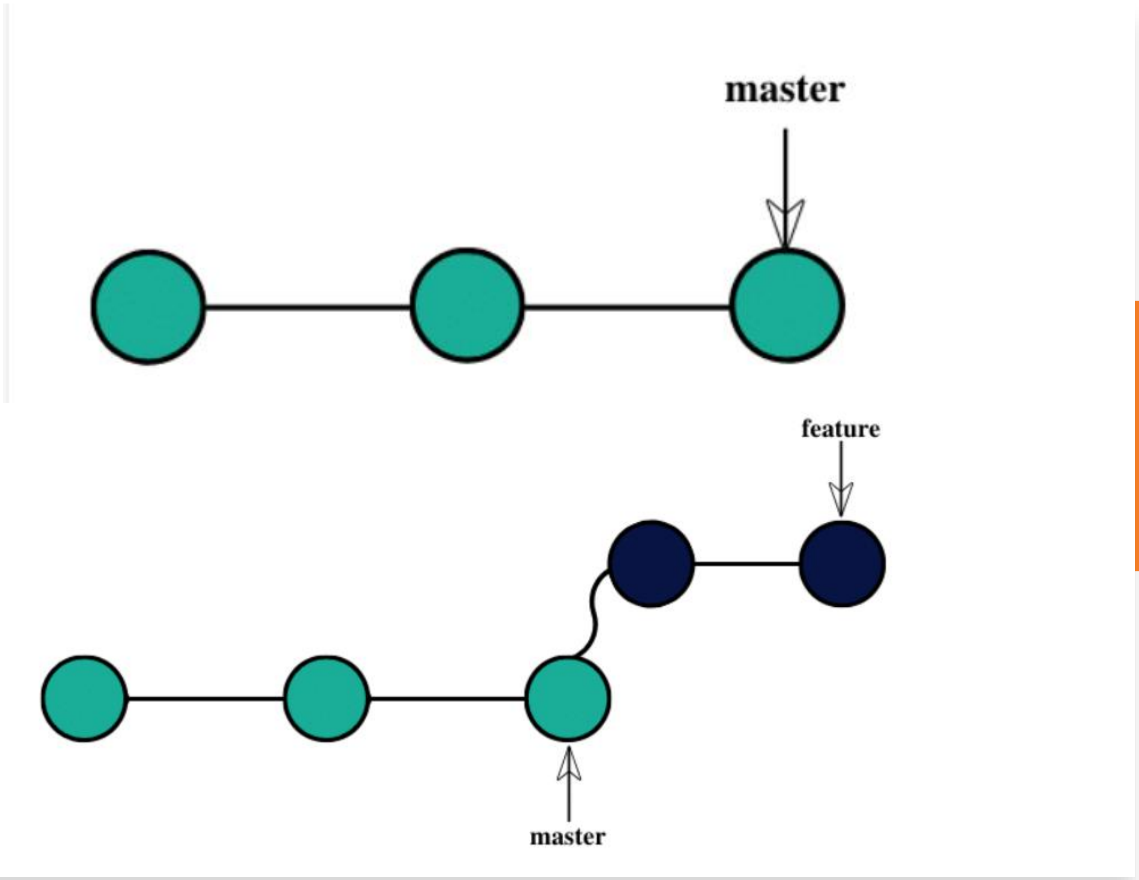


What is a Branch in Git?

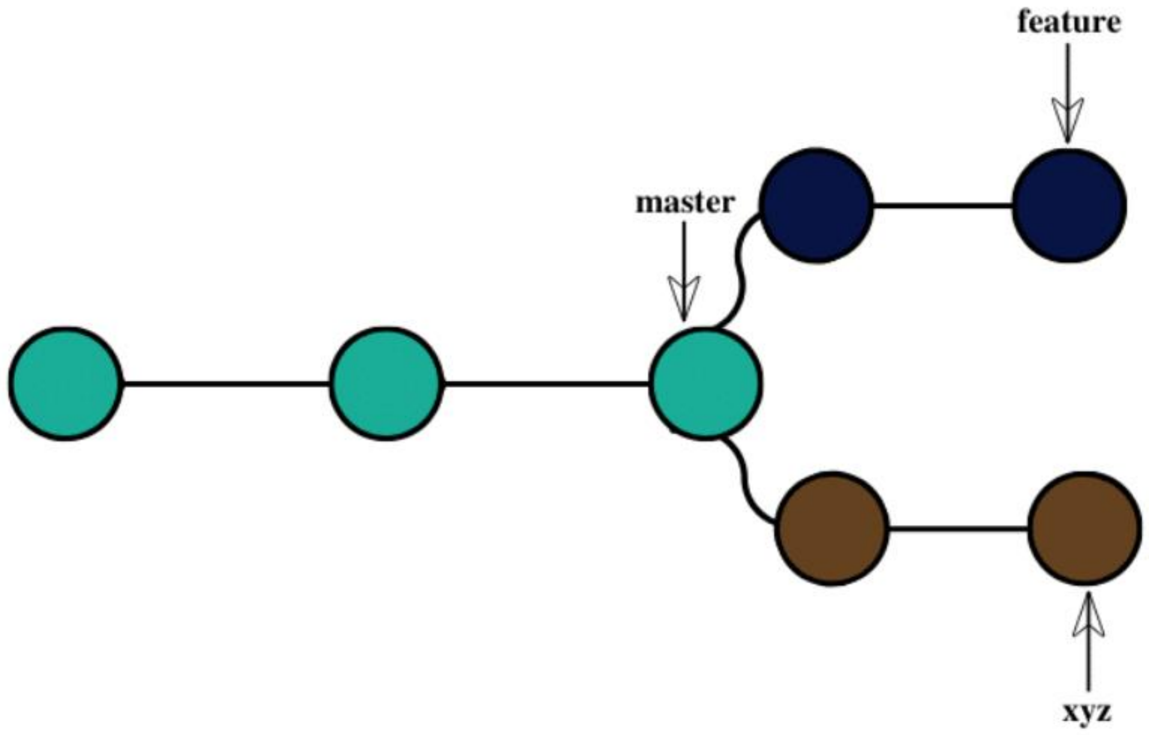


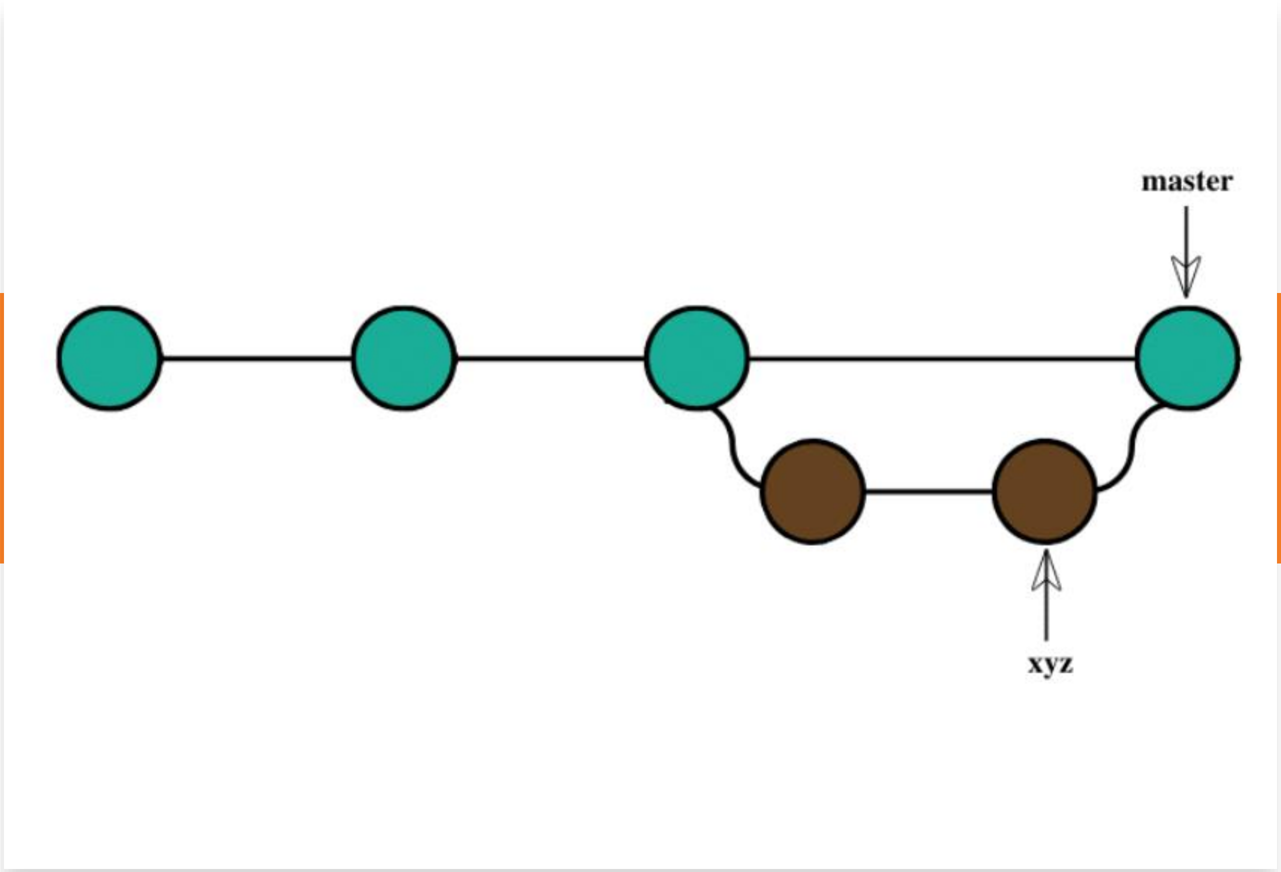
Project Development through linear development

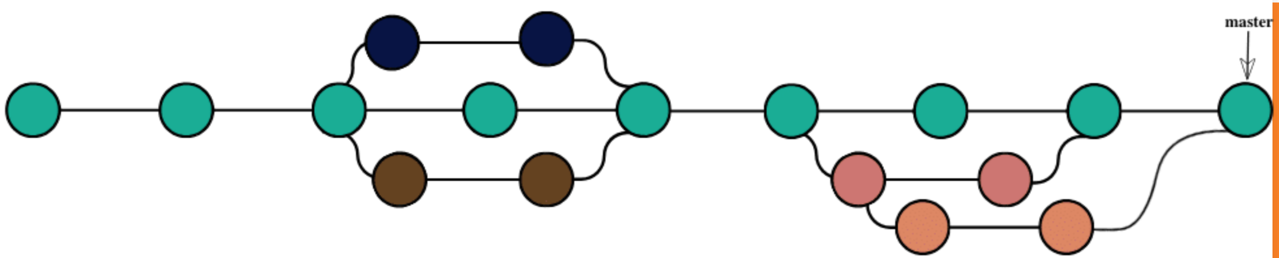

master
master
master



Developing the project through branching


feature
master
xyz


master
xyz


master

Branches Commands

Let’s try it now

Recap

- **create :** git branch <branch-name>
- **display:** git branch
- **delete:** git branch -d <branch-name>
- **move:** git checkout <branch-name>
- **rename:** git branch -m <branch-name>

Merging

Combining your work

Straight forward

- `Merge <branch-name>`

Resolve Conflict

Merge <branch-name>

If changes wasn’t straight forward git’ll ask you to resolve the changes and create a new commit

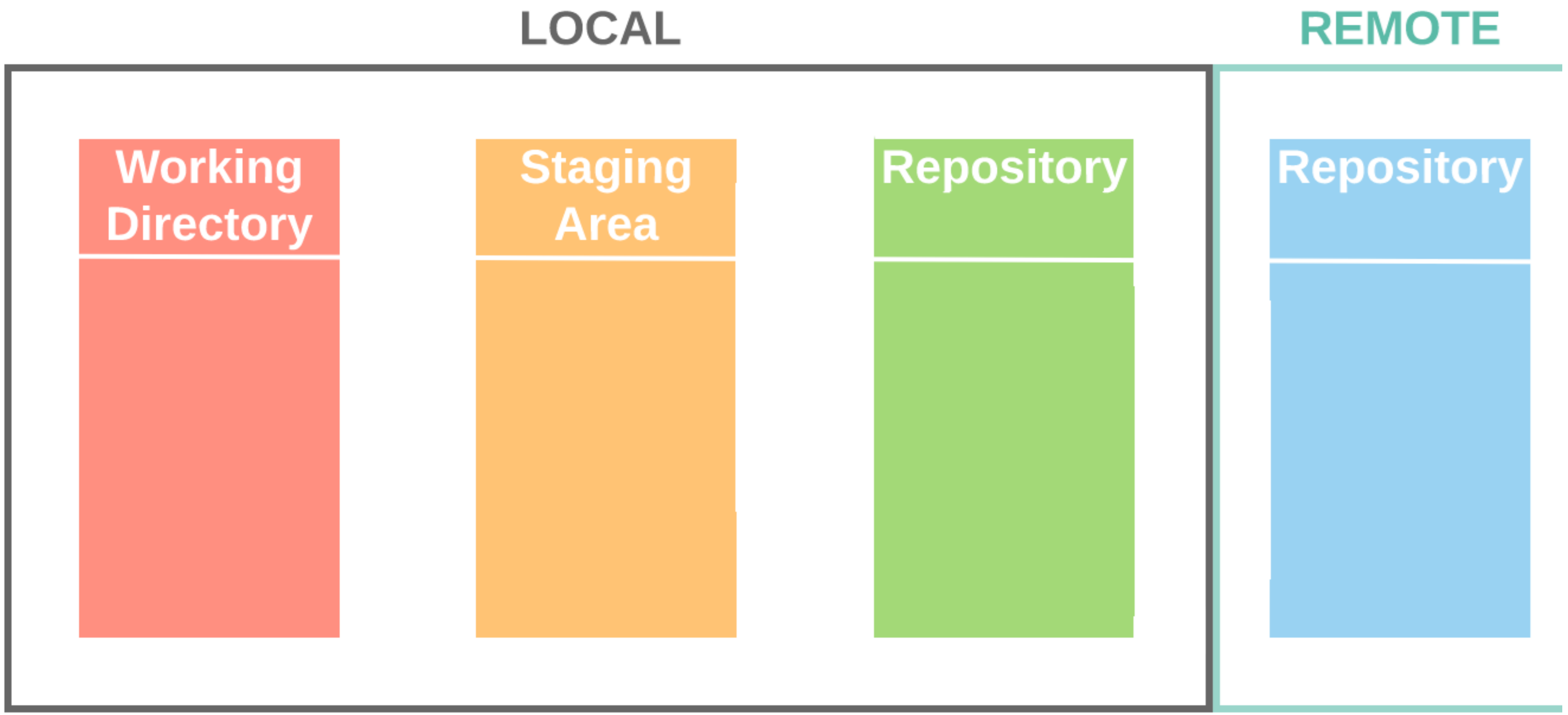
Remote Repository

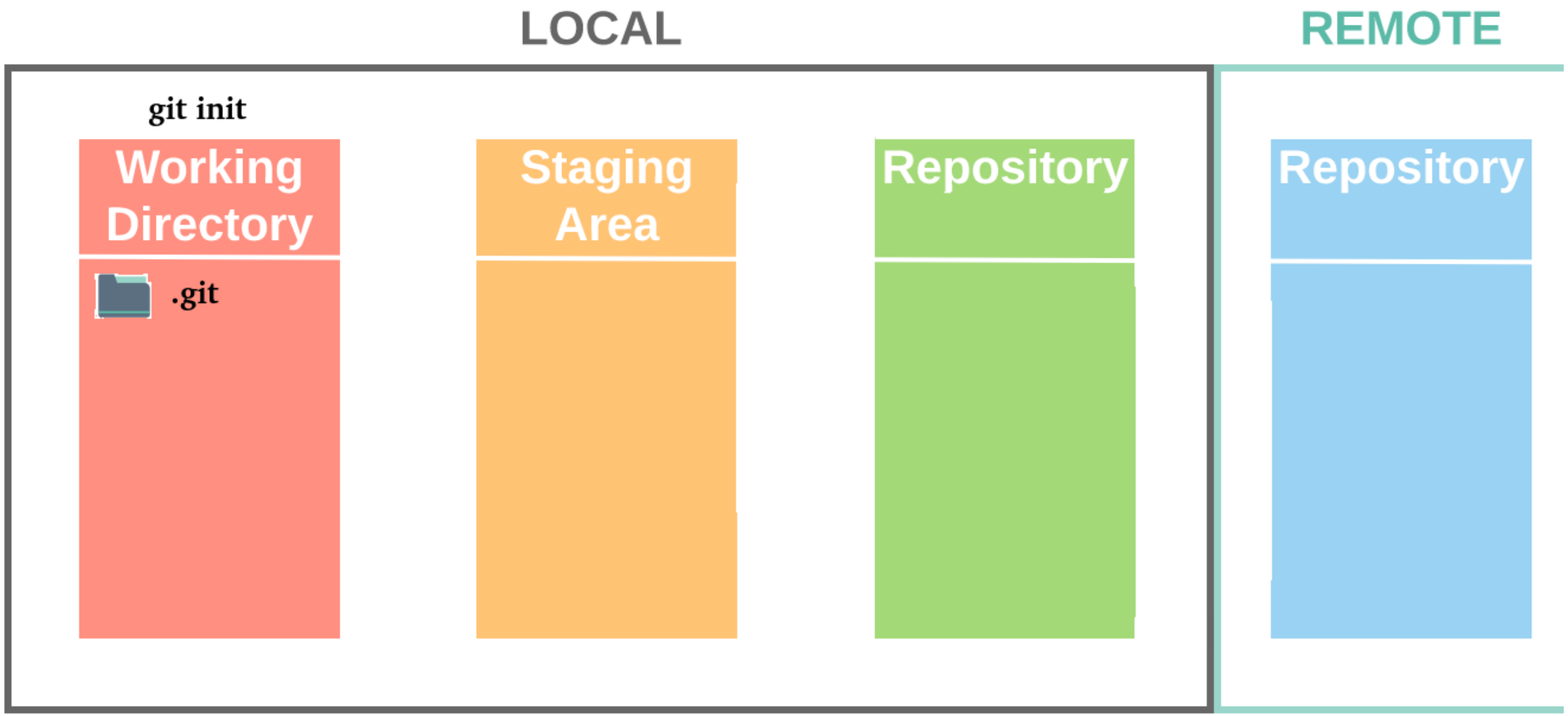
Sharing your development

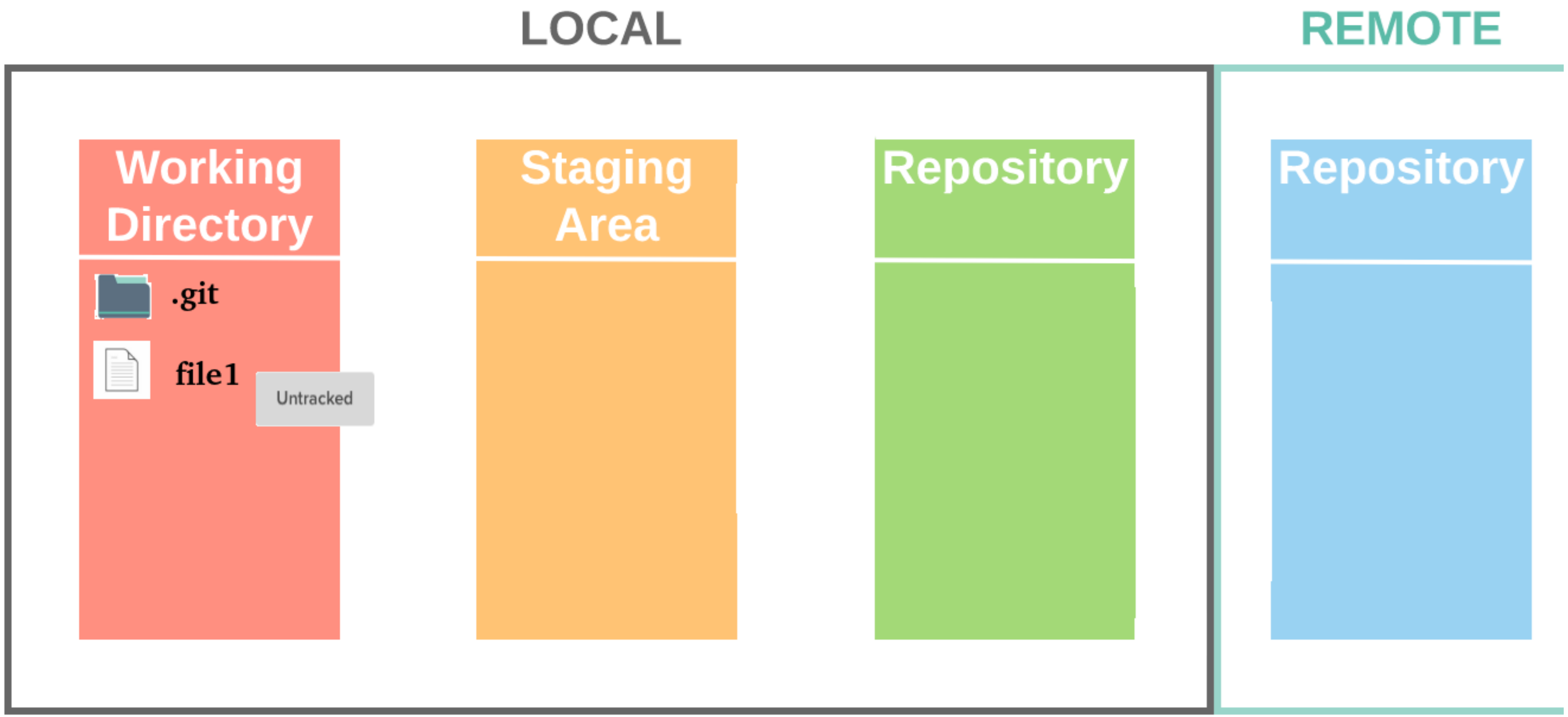
There is more and more

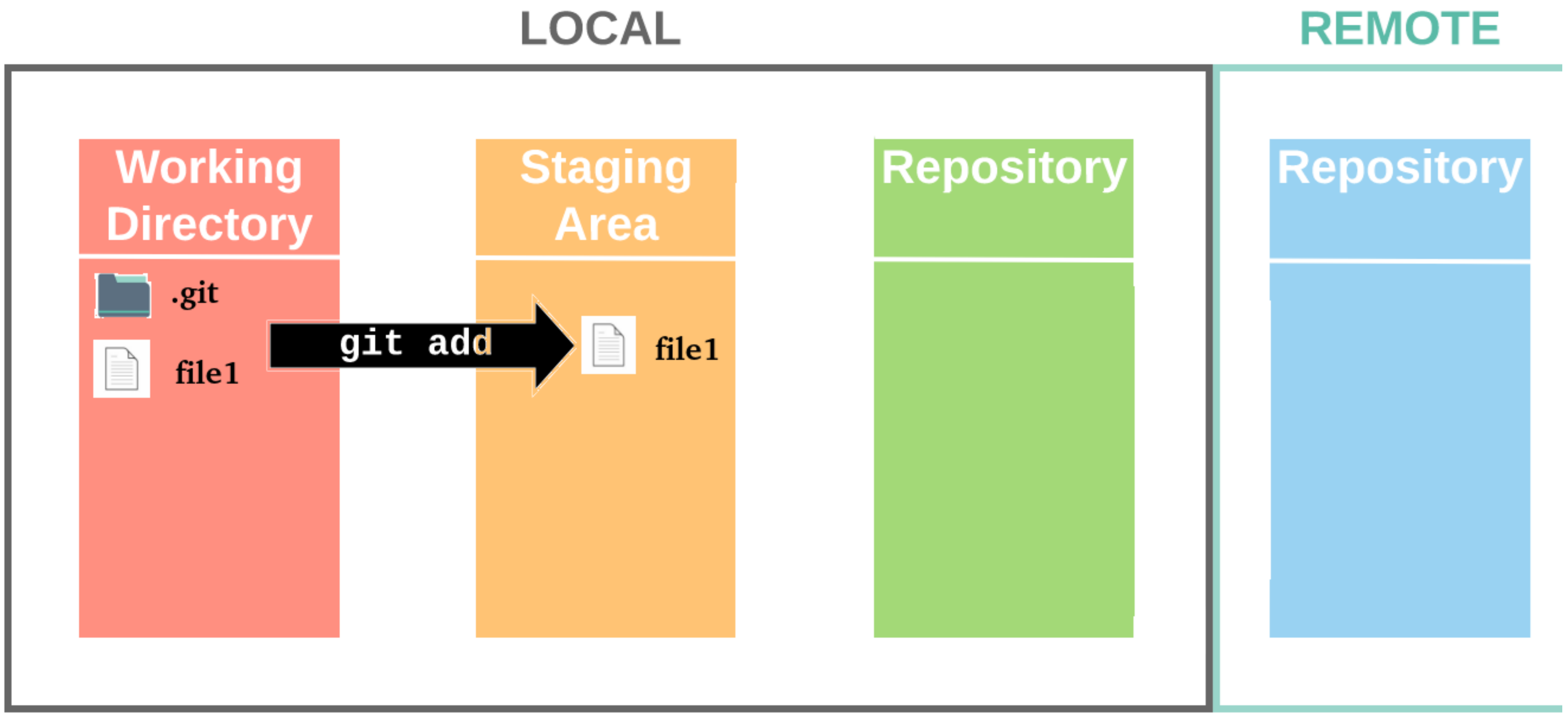
GitHub

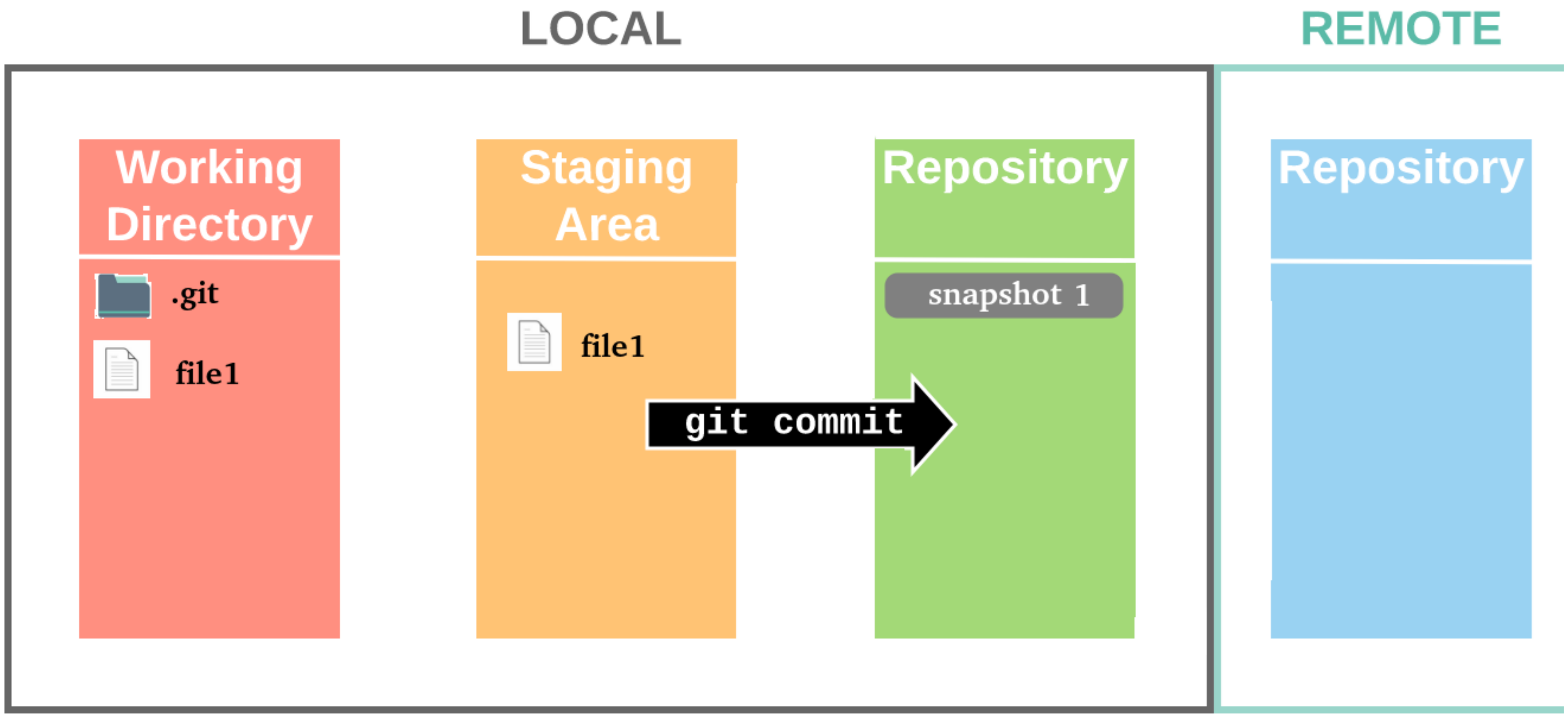
The Basic Workflow of Github

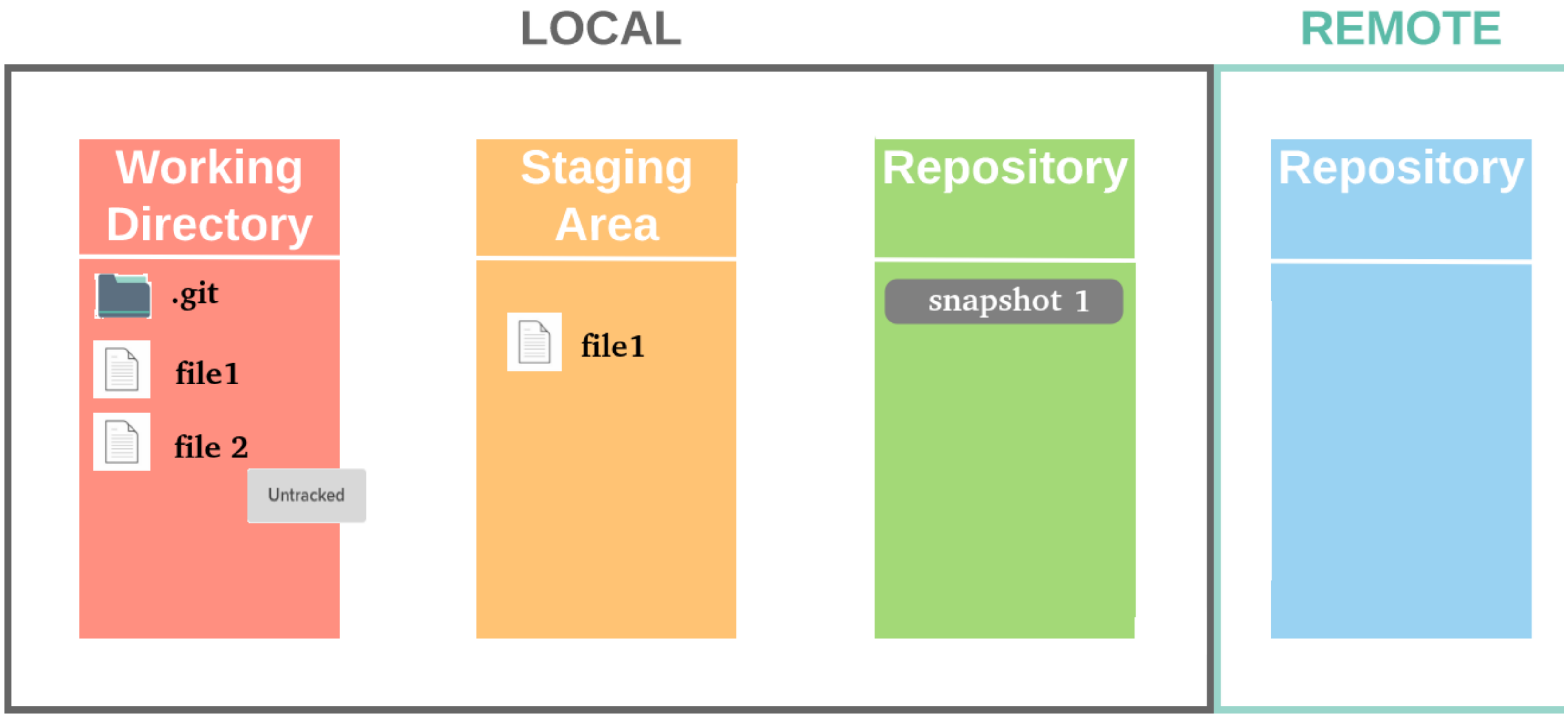

LOCAL
REMOTE
Working Directory
Staging Area
Repository
Repository

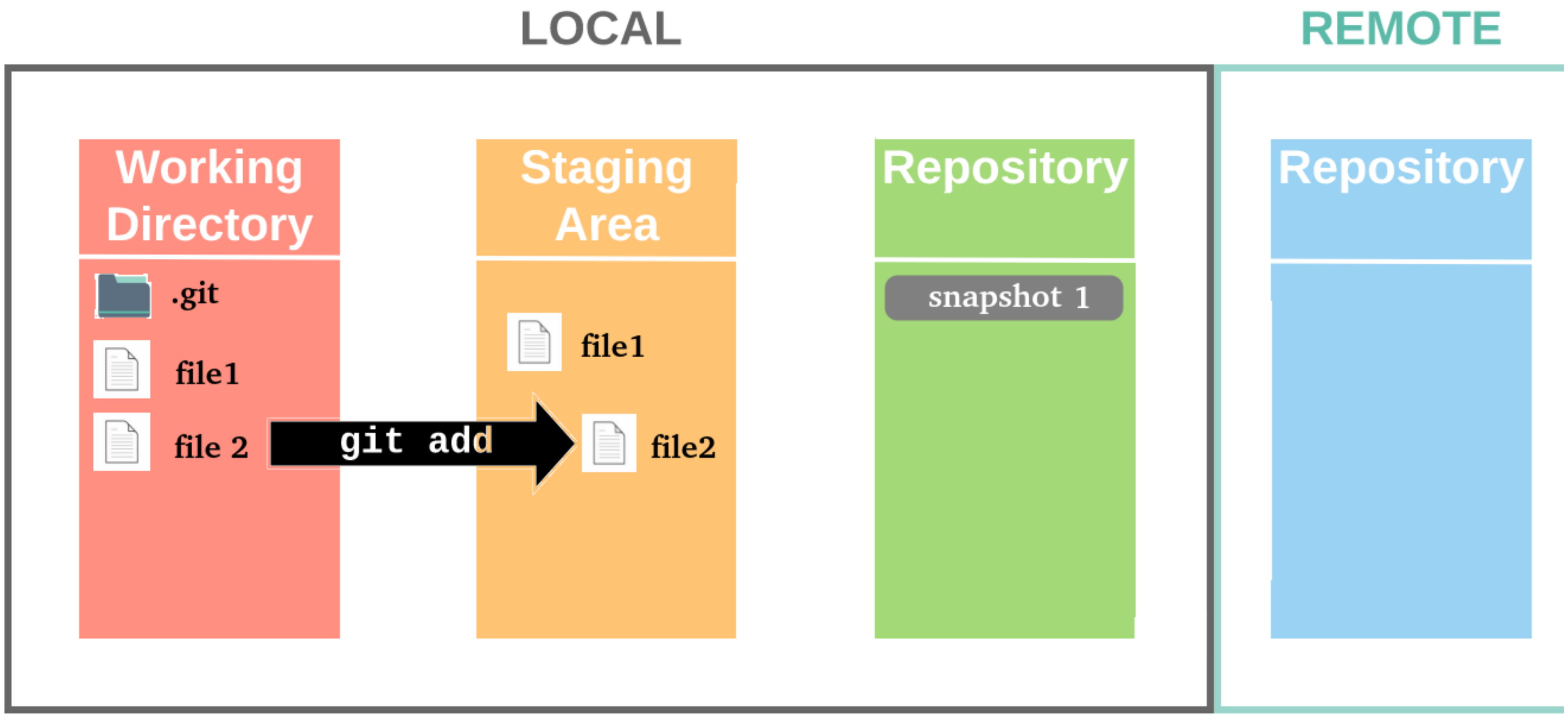

LOCAL
REMOTE
git init
Working Directory
.git
Staging Area
Repository
Repository

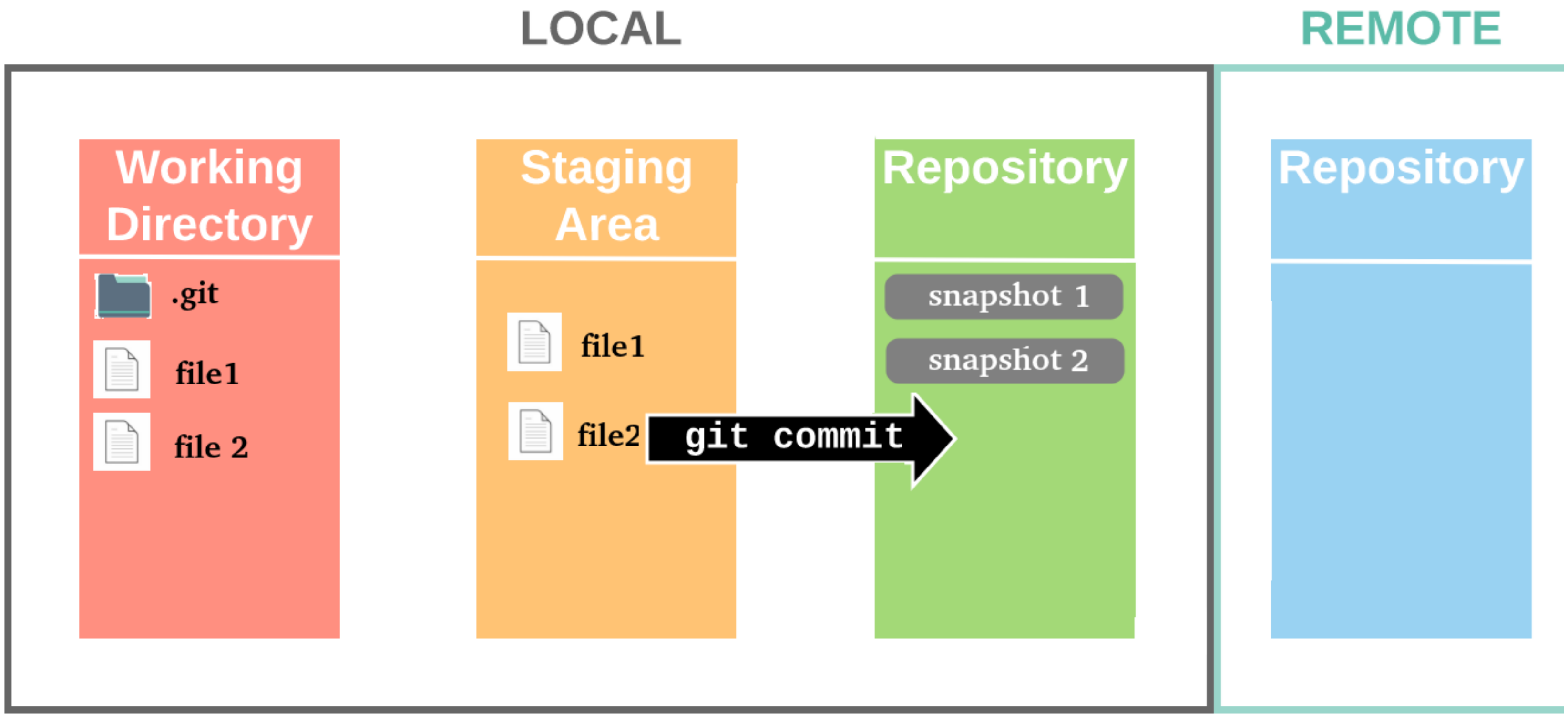

LOCAL
REMOTE
Working Directory
Staging Area
Repository
Repository
.git
file1
Untracked

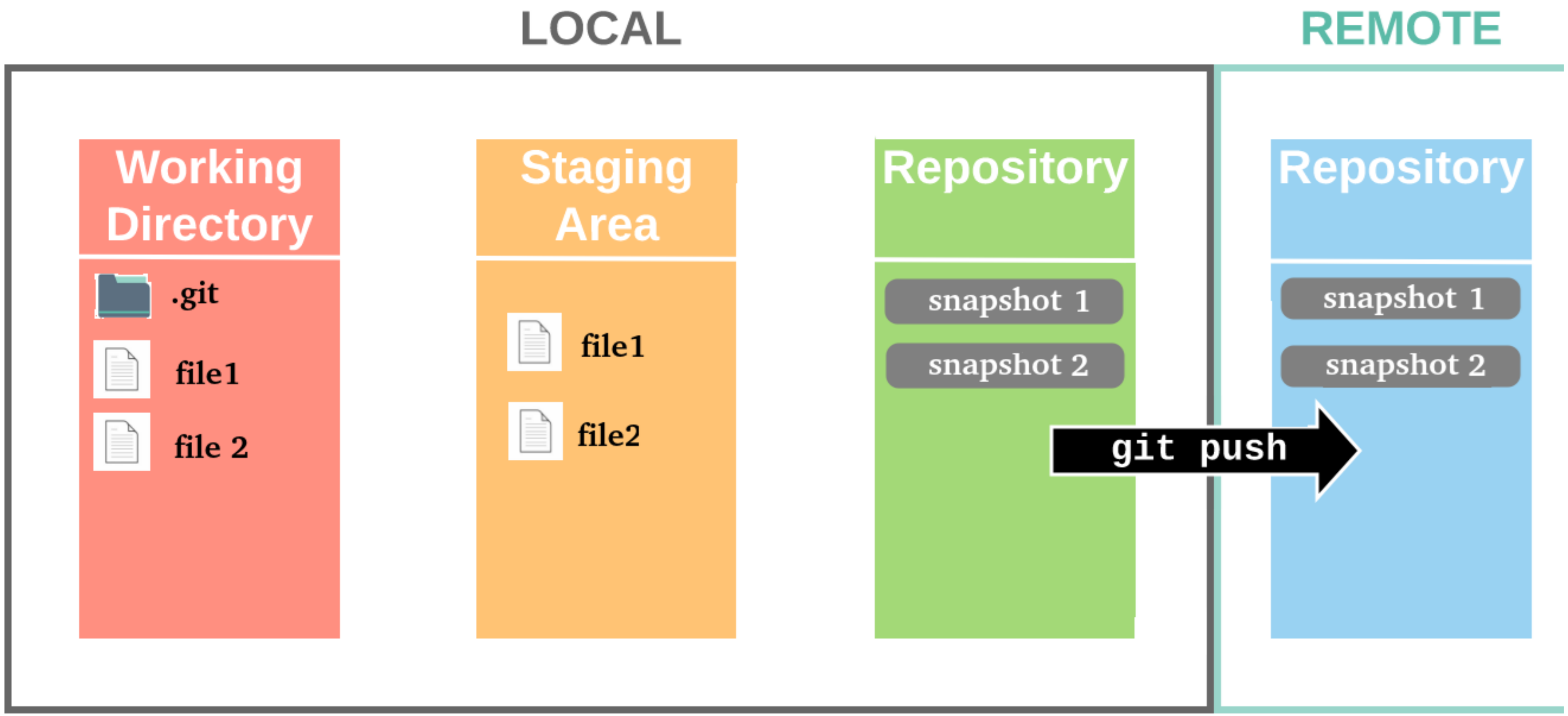

LOCAL
REMOTE
Working Directory
Staging Area
Repository
Repository
.git
file1
git add
file1


LOCAL
REMOTE
Working Directory
.git
file1
Staging Area
file1
git commit
Repository
snapshot 1
Repository

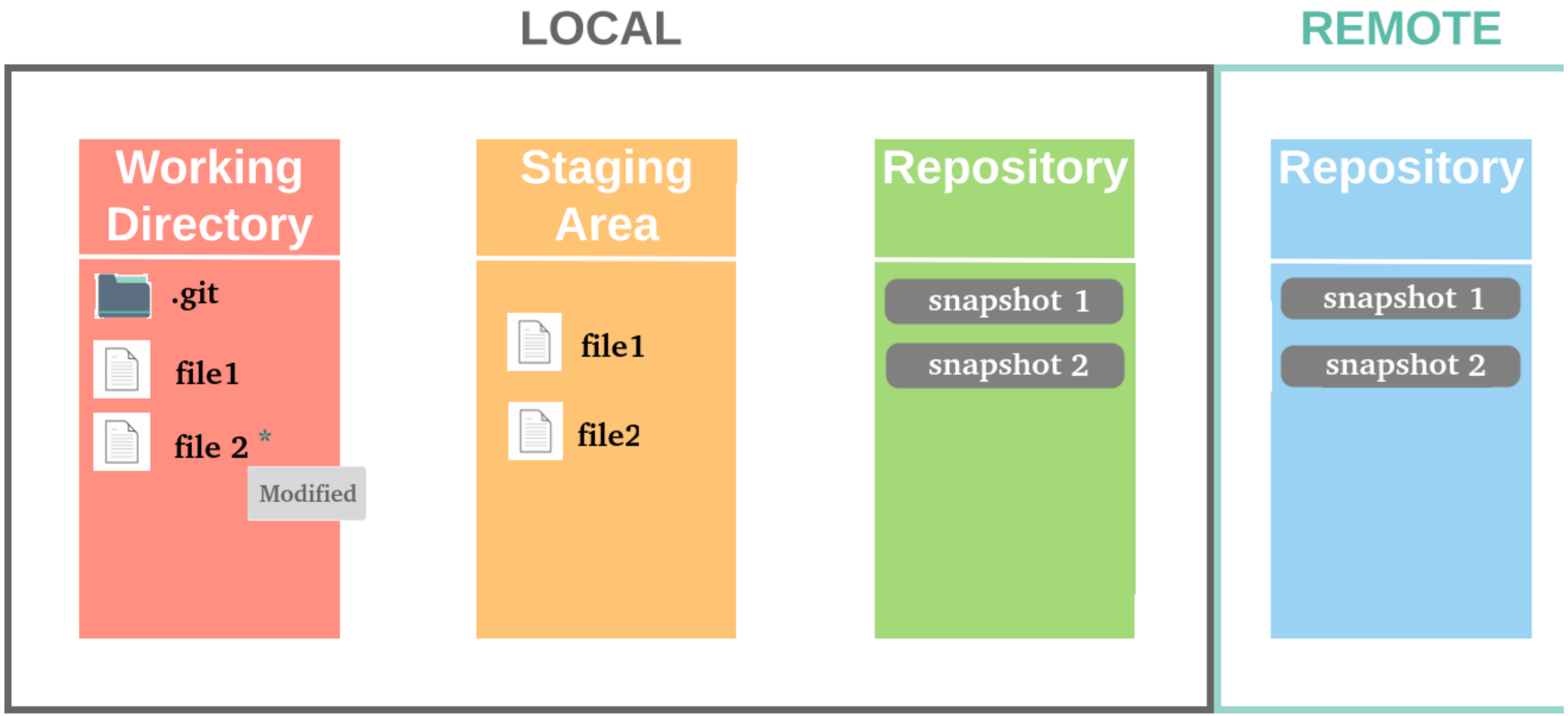

LOCAL
REMOTE
Working Directory
.git
file1
file 2
Untracked
Staging Area
file1
Repository
snapshot 1
Repository

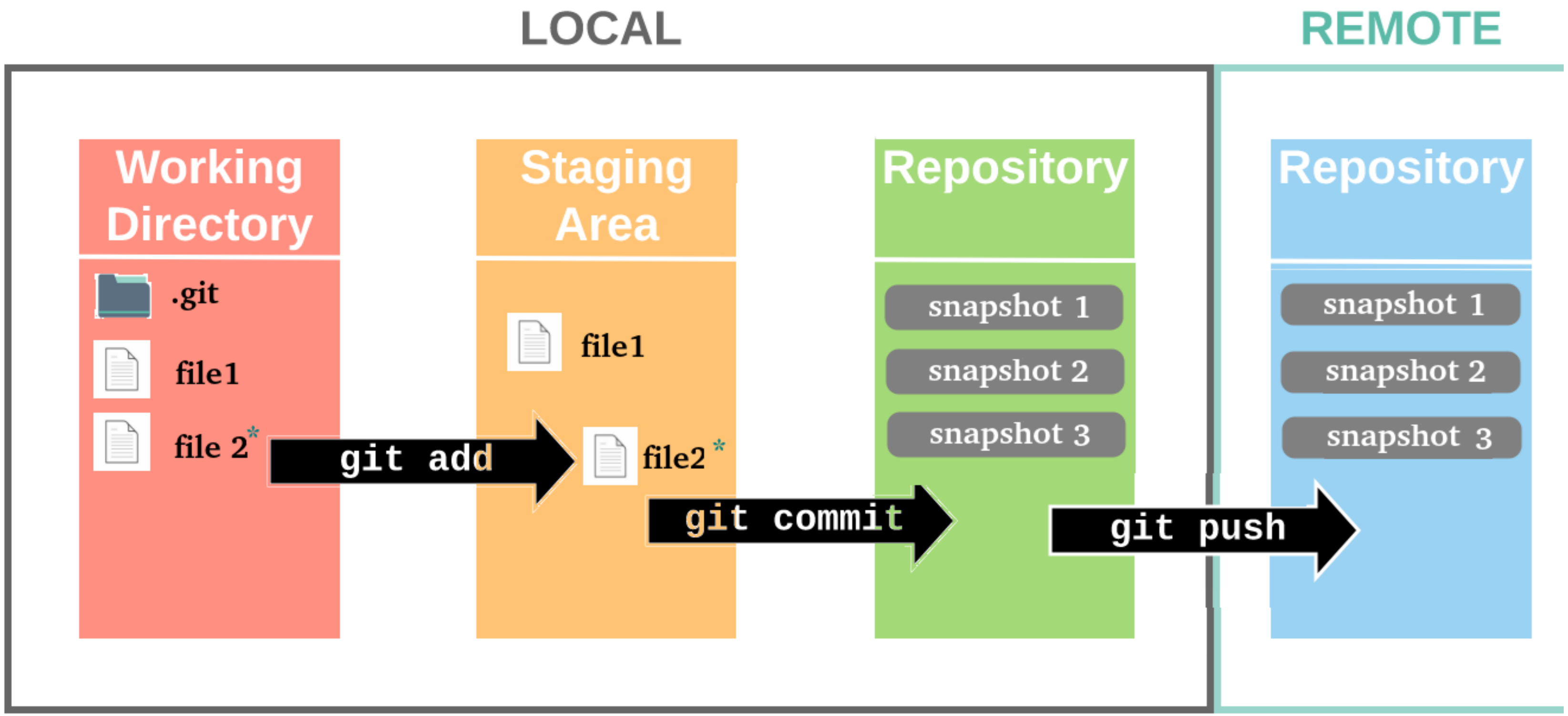

LOCAL
REMOTE
Working Directory
.git
file1
file 2
git add
Staging Area
file1
file2
Repository
snapshot 1
Repository


LOCAL
REMOTE
Working Directory
.git
file1
file 2
Staging Area
file1
file2
git commit
Repository
snapshot 1
snapshot 2
Repository


LOCAL
REMOTE
Working Directory
.git
file1
file 2
Staging Area
file1
file2
Repository
snapshot 1
snapshot 2
git push
Repository
snapshot 1
snapshot 2

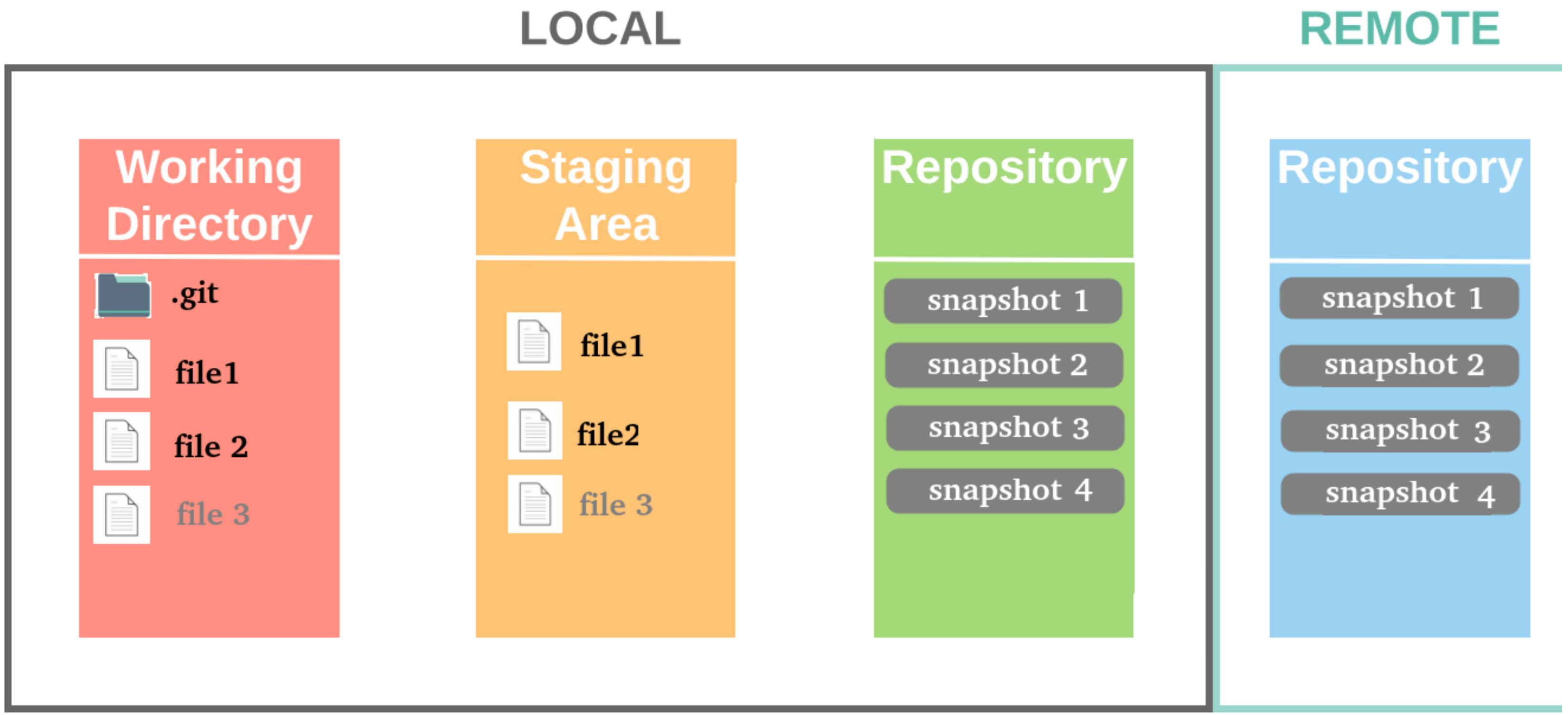
What if you modify a tracked file?

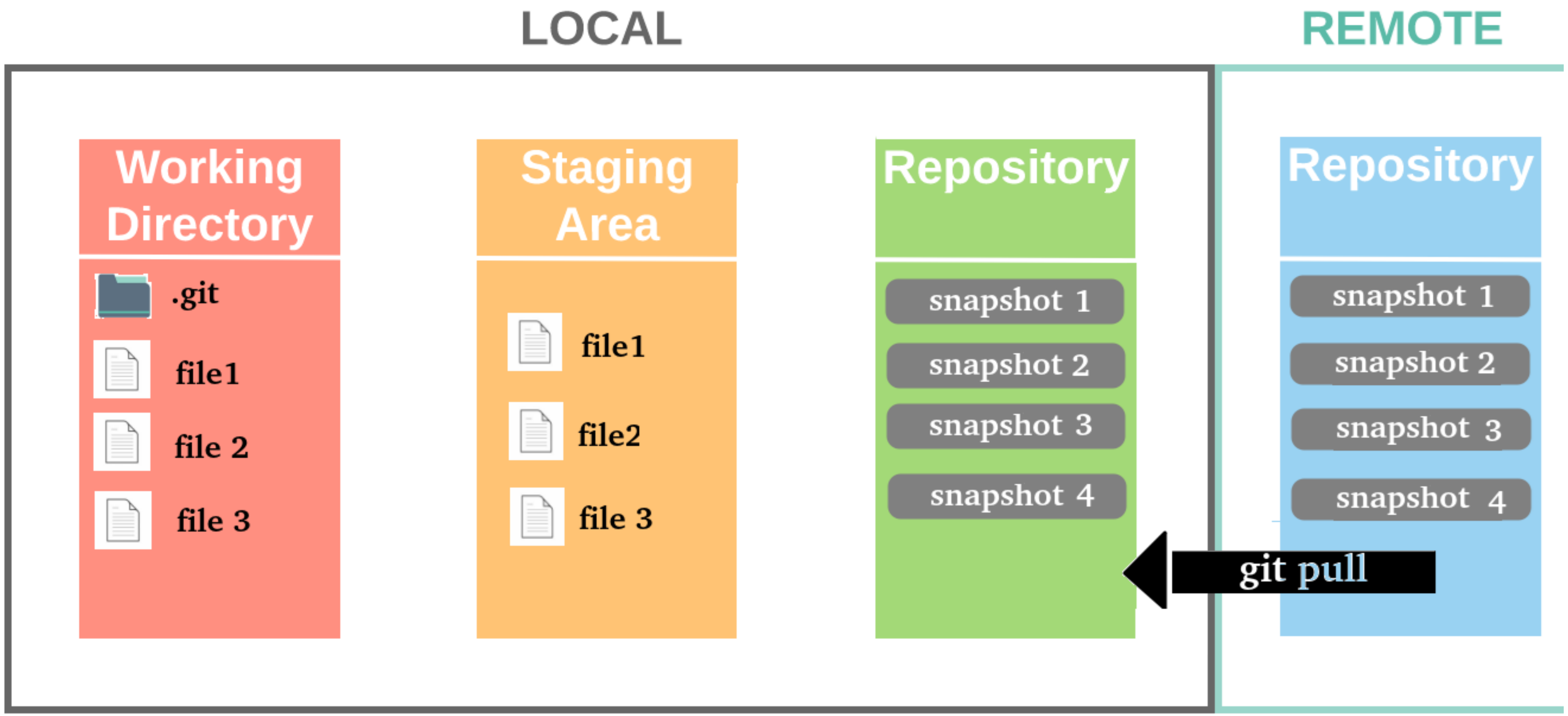


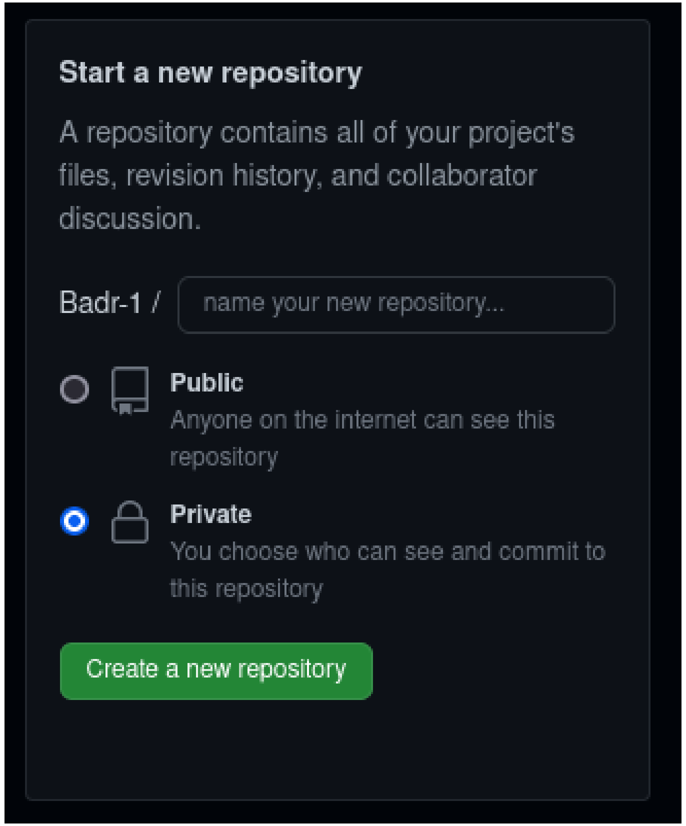

LOCAL
REMOTE
Working Directory
.git
file1
file 2*
git add
Staging Area
file1
file2 *
git commit
Repository
snapshot 1
snapshot 2
snapshot 3
git push
Repository
snapshot 1
snapshot 2
snapshot 3

What if your repo is not up-to-date?

Your Friend:




LOCAL
REMOTE
Working Directory
.git
file1
file 2
file 3
Staging Area
file1
file2
file 3
Repository
snapshot 1
snapshot 2
snapshot 3
snapshot 4
Repository
snapshot 1
snapshot 2
snapshot 3
snapshot 4
git pull


Start a new repository

A repository contains all of your project's files, revision history, and collaborator discussion.

Badr-1 / name your new repository...

( ) Public
Anyone on the internet can see this repository

(●) Private
You choose who can see and commit to this repository

Create a new Repository on GitHub

New Repository

- `echo "# demo" >> README.md`
- `git init`
- `git add README.md`
- `git commit -m "first commit"`
- `git branch -M main`
- `git remote add origin URL`
- `git push -u origin main`

Existing Repository

- `git remote add origin https://github.com/Badr-1/demo.git`
- `git branch -M main`
- `git push -u origin main`

Cloning

Getting a remote repository to your local repository

Fetching

Getting a remote repository changes to your local repository

Pulling

Fetching changing and merging to local repository

Cloning and download

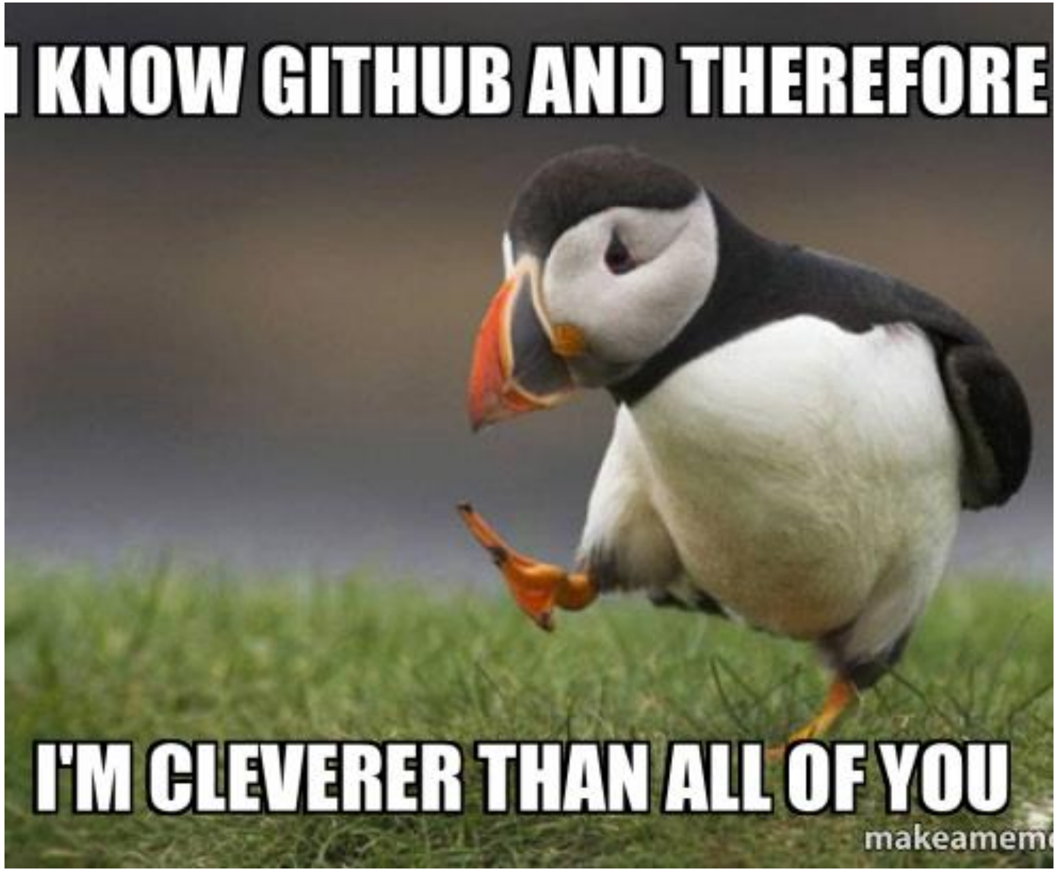
Getting a remote repository to your local repository

Cloning

CLONE OUR FIRST SESSION

ALL??

- Alias commands
- What diff HTTPS,SSH
- Forking
- Contributions
- Git internals
- Rebasing vs merging
- Using GUI
- ….
- ….



Youtube videos

[(133) Free software, free society: Richard Stallman at TEDxGeneva 2014 - YouTube]

[(133) The mind behind Linux | Linus Torvalds | TED – YouTube]

[(133) What is Git? (Arabic) – YouTube]

[(133) Git and GitHub | شخبط وانت متطمن - YouTube]

Books

-pro git

-Git Internals by Scott Chacon

To watch

[(133) Git Unleashed – YouTube]

[(133) Git Internals - Intro Video – YouTube]

[(133) Git for Professionals Tutorial - Tools & Concepts for Mastering Version Control with Git - YouTube]

OSC-Open Source Community