GIT

Pamela Solano

Goals

This note should be give you the **introduction** to perform basic **distributed revision control** for **your own/team projects**.

References

Pro Git book, written by Scott Chacon and Ben Straub. 2014.

(Chacon and Straub 2014)

https://git-scm.com/book/en/v2

Target

- No experience in *Version Control*
- Version control system but not *GIT*
- *GIT GUI*'s users (visual tools) not *command-line*.

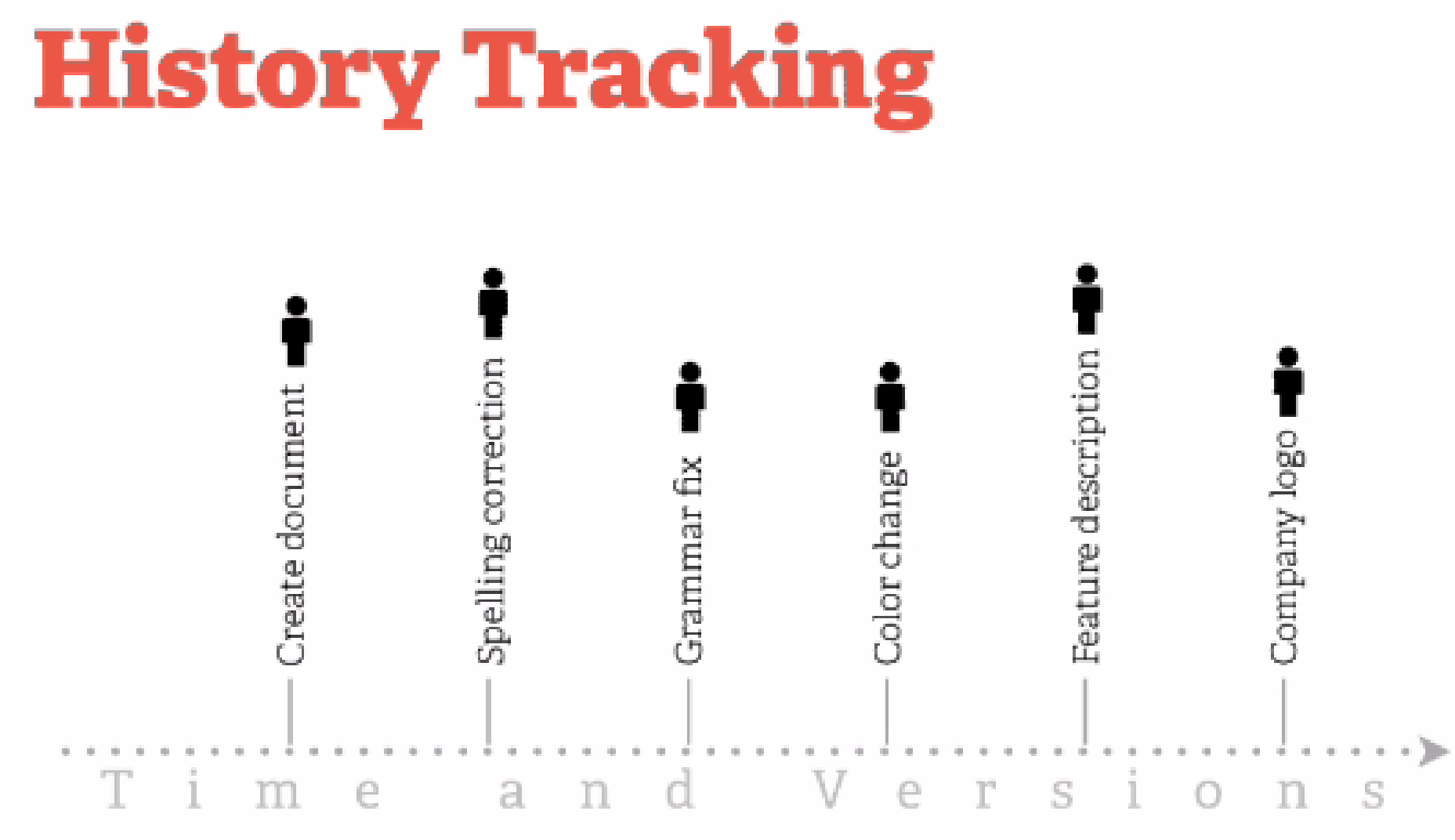
Your job and daily task

- Software developer
- Designer
- Someone who writes code

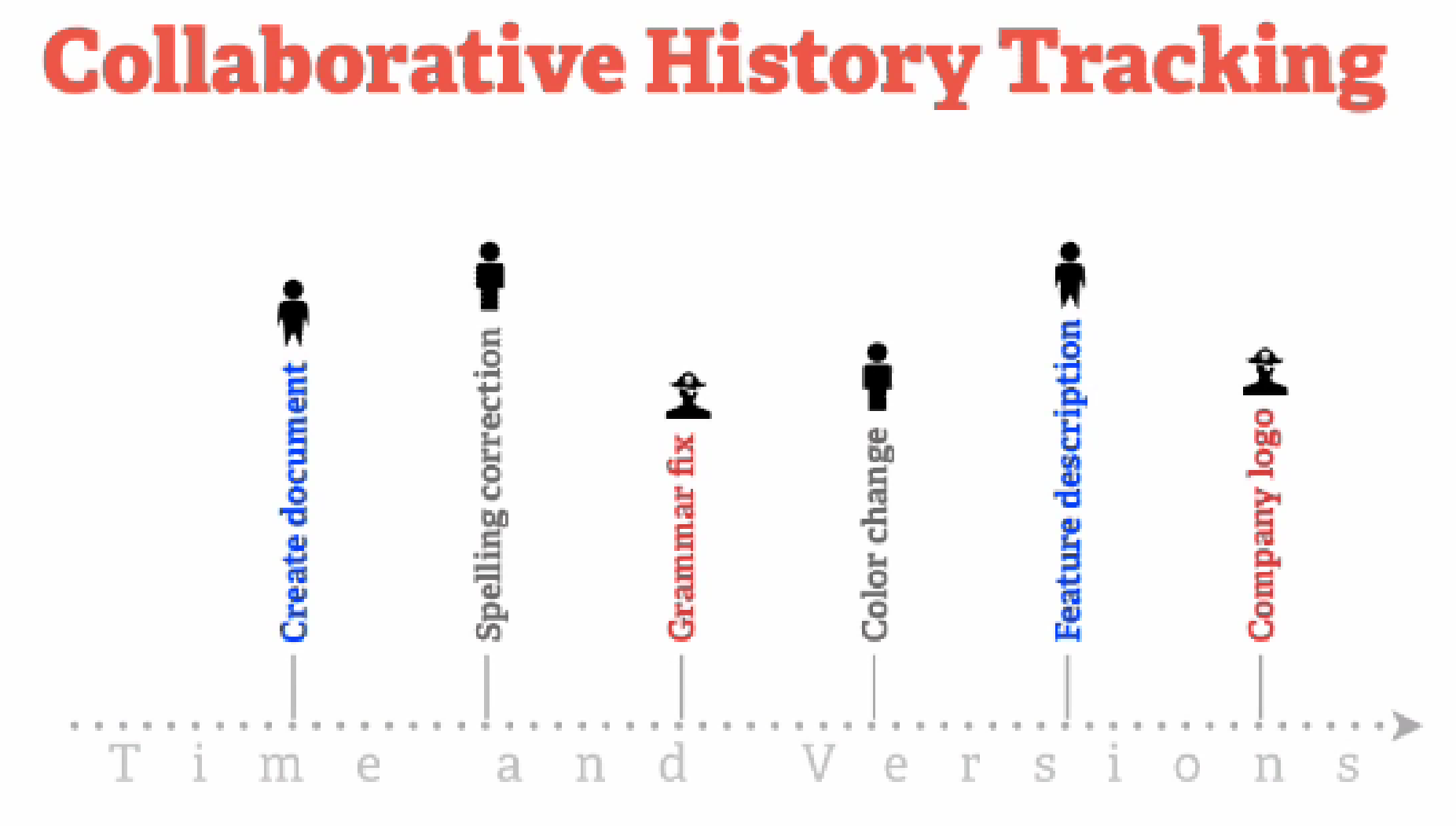
your task

- *Create* things
- *Save* things
- *Edit* things (correction-request of modification)
- Save the thing *again* and *again*

Personal changes

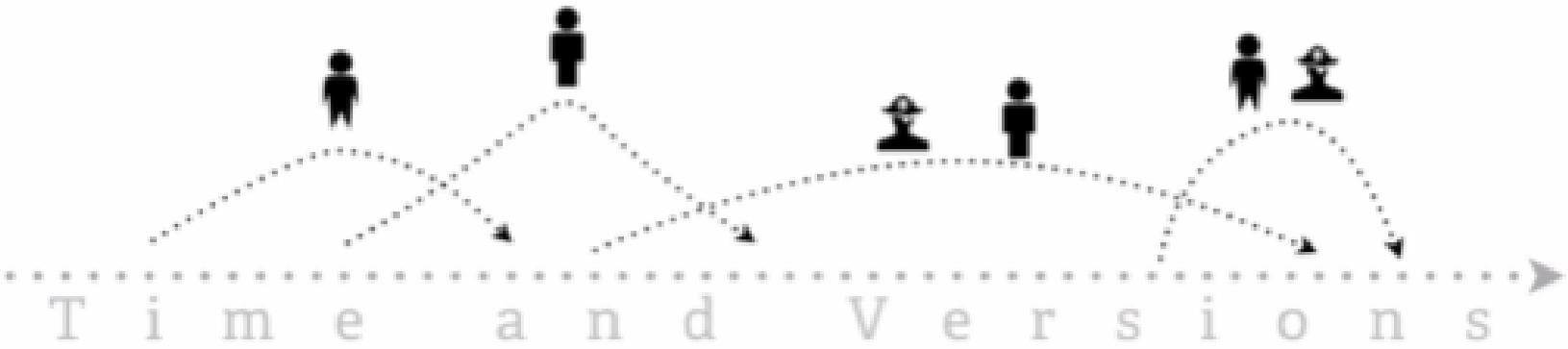


Team work


Collaborative History Tracking
Create document
Spelling correction
Grammar fix
Color change
Feature description
Company logo
Time and Versions

Non-Sequential Overlaping

Collaborative History Tracking



Track modifications

! Important

When-Why-What-Who are the goals *VERSION CONTROL*

Why GIT?

- GIT is a fast and modern
  - implementation of *version control*
- GIT provides a *history* of content changes
- Individual bases, tracking each file's content piece (graphics design, programs and code)

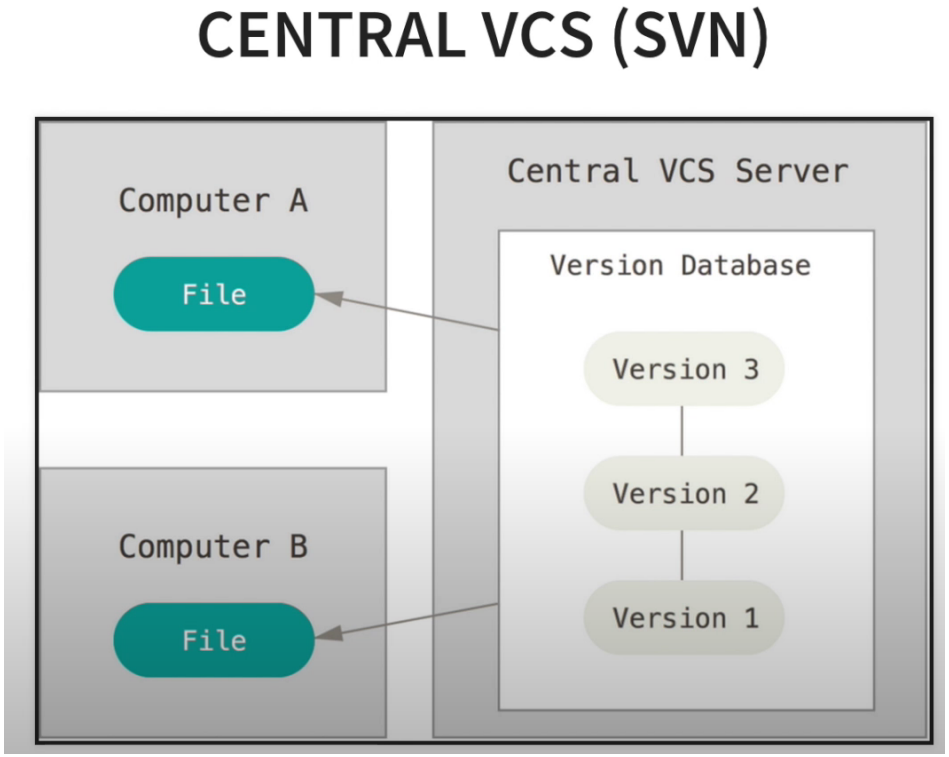
Why GIT? continuation

- GIT facilitates **collaborative changes** to files.
- Not just one person bringing modification and for all the team at the same time.
- People simultaneous change files, working at the same time about an idea.
- GIT is easy to use for
  - any type of knowledge worker.

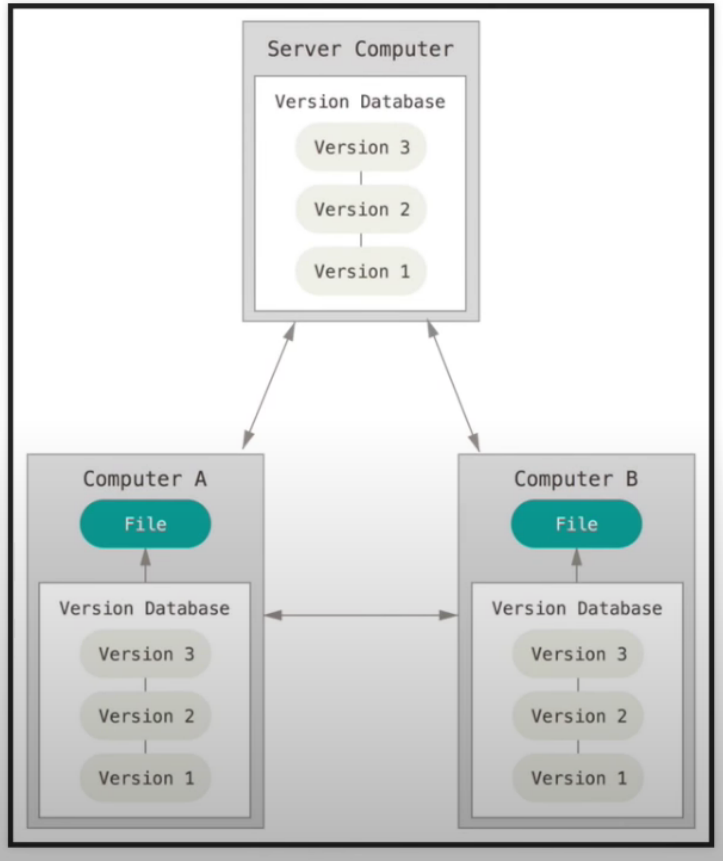
Types of version control

What is the difference between *CENTRAL* and *DISTRIBUTED* Version Control System.

Without GIT: SVN



DISTRIBUTED VCS (GIT)



GIT

Version control

- Local GIT
  - Simple commands.
- *Distributed*
  - Connectivity not required
  - Simple software
- *Easy* $\rightarrow$ commands can be learnt *progressively*.

Summary

Important

with GIT you have a way **how to track** your/team project progress

Tip

To fully understand the depth and power of Git you need to understand two simple ideas (LOCAL-REMOTE).

Get going with git

- How to install
- Setup
- Configure
- Make your first use of the command line

Installing GIT

Official GIT's homepage

https://git-scm.com/

Opening GIT

Important

GIT Bash

check version

Tip

$ git - -version

Configuration

- Configure your *username* and *email*

Note

$ git config - -global user.name “Ana Devops”

Note

$ git config - -global user.email “anadevops@gmail.com”

Help

! Important

```
$ git help config
```

! Important

```
$ git config - -help
```

- Google is your friend

Example

Developer Ana :

- She is working on her new project in a file called clients.txt

Creating a new repository

Note

```
$ git init project
```

- If the directory exists use only

Note

```
$ git init
```

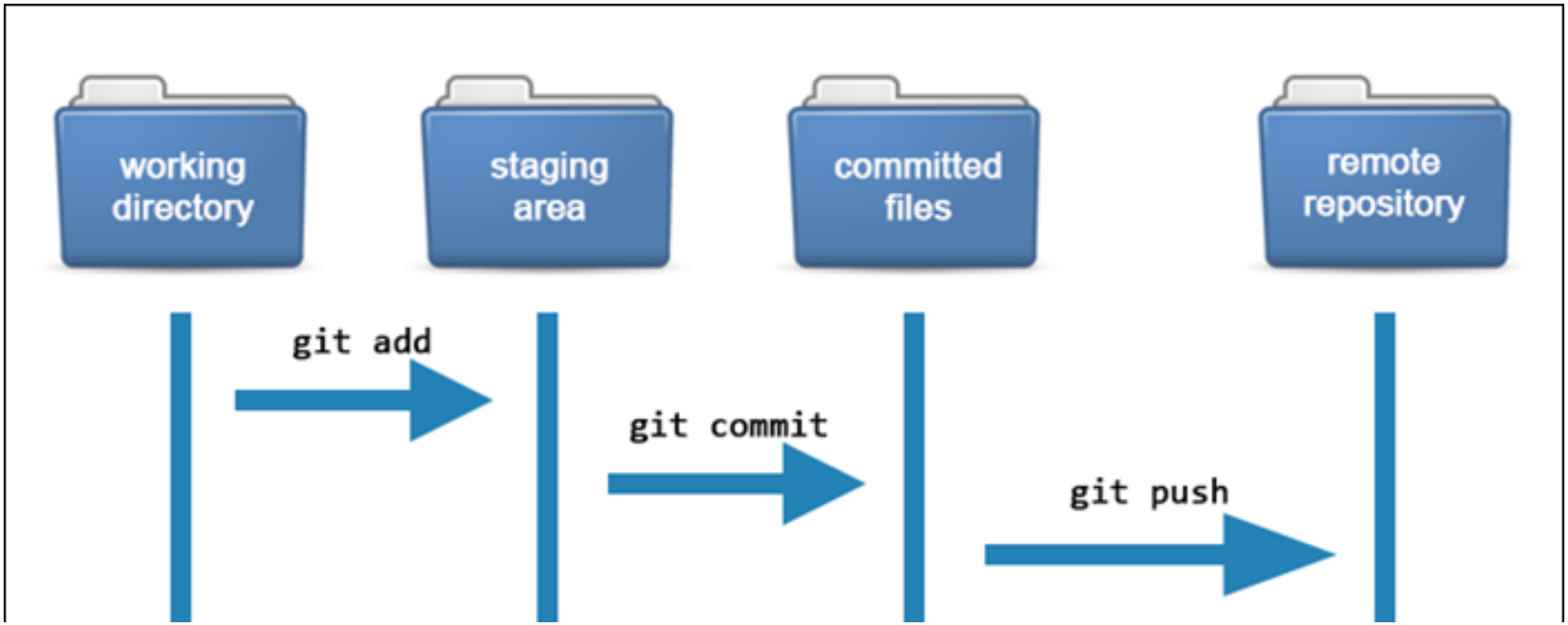
Note

```
$ cd project
```

In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents (master)
$ git init Pam
Initialized empty Git repository in C:/Users/Pamela/Documents/Pam/.git/

Pamela@PC1022350854 MINGW64 ~/Documents (master)
$ cd Pam

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git status
On branch master

No commits yet

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

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

Git workflow



Tracking-Staged clients.txt

Start tracking *holding zone*. Zone ready to be committed

Note

```
$ git add clients.txt
```

Note

```
$ git status
```

In color *green*

In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git add Clients.txt

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Clients.txt
```

unTracking - unstage clients.txt

Note

- $ $ git rm –cached clients.txt
- $ git restore - -staged clients.txt

Note

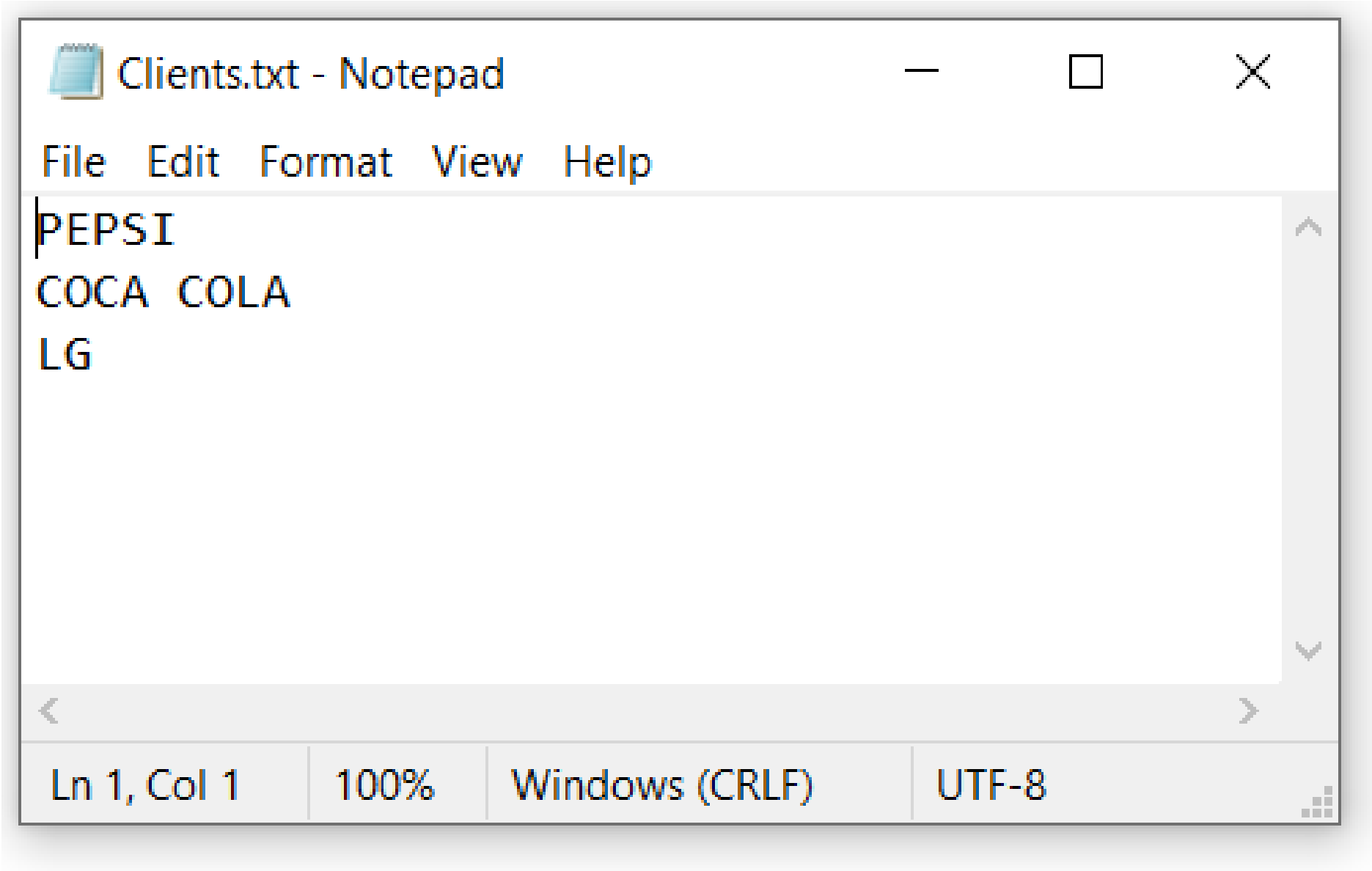
$ git status

In color *red*

In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git rm --cached Clients.txt
rm 'Clients.txt'

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git status
On branch master

No commits yet

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

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

Making changes



Commit command

Remember: put in stage area with add before commit

Note

```
$ git add clients.txt
```

Note

```
$ git commit -m "first commit"
```

Commit is the keyword that permanently logs changes

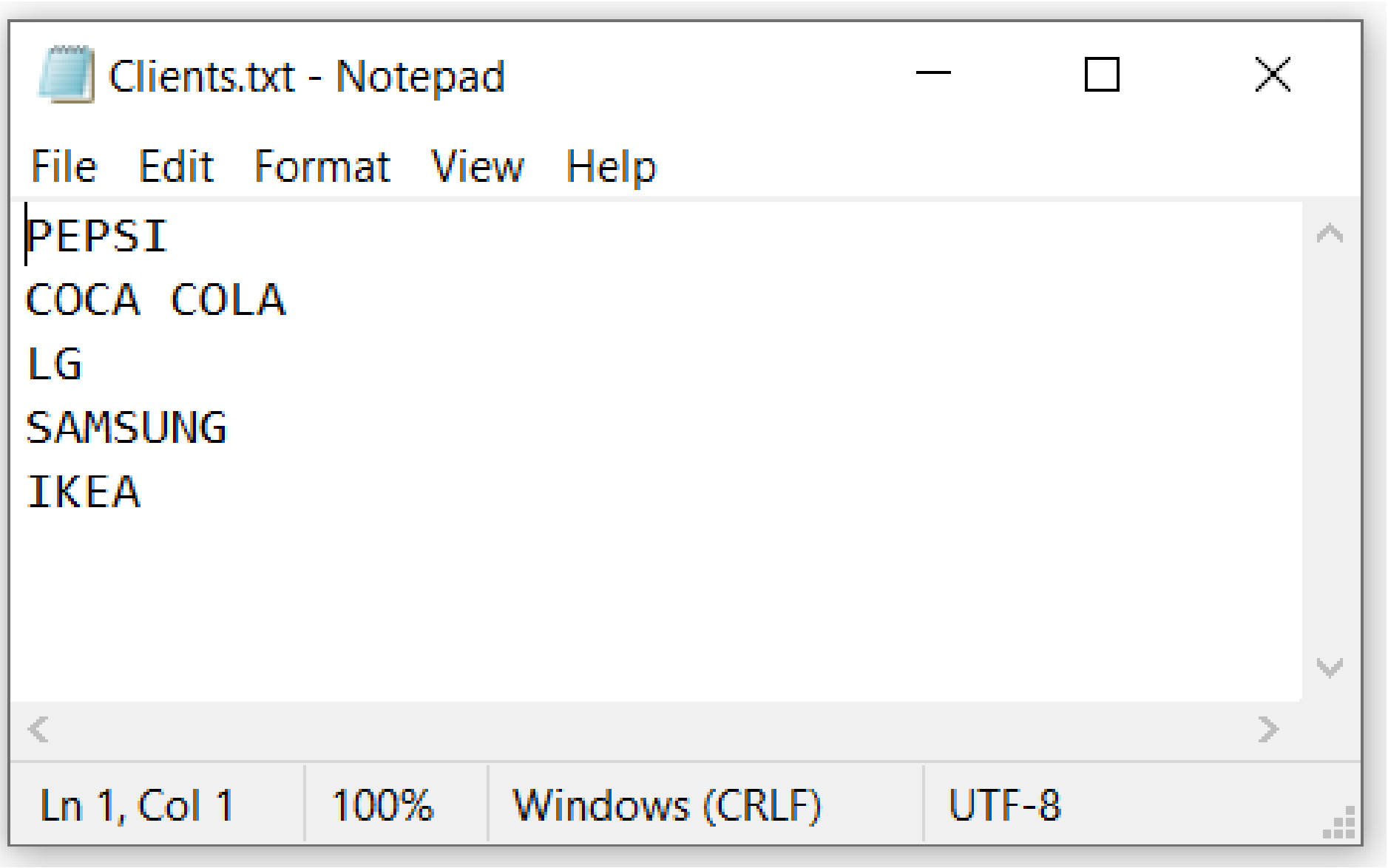
In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git add Clients.txt

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Clients.txt


Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git commit -m "Including Clients"
[master (root-commit) 4a671af] Including Clients
 1 file changed, 3 insertions(+)
 create mode 100644 Clients.txt
```

Exercise

- Make 3 commits
- Use $ git diff to see differences when relevant

In your Git Bash



In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git add Clients.txt

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git commit -m "new clients"
[master 5032123] new clients
 1 file changed, 3 insertions(+), 1 deletion(-)

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git log --oneline
5032123 (HEAD -> master) new clients
4a671af Including Clients
```

Return to the previous commit

Note

$ git log - - oneline

Note

$ git revert ID

write :qa! (VIM's command)

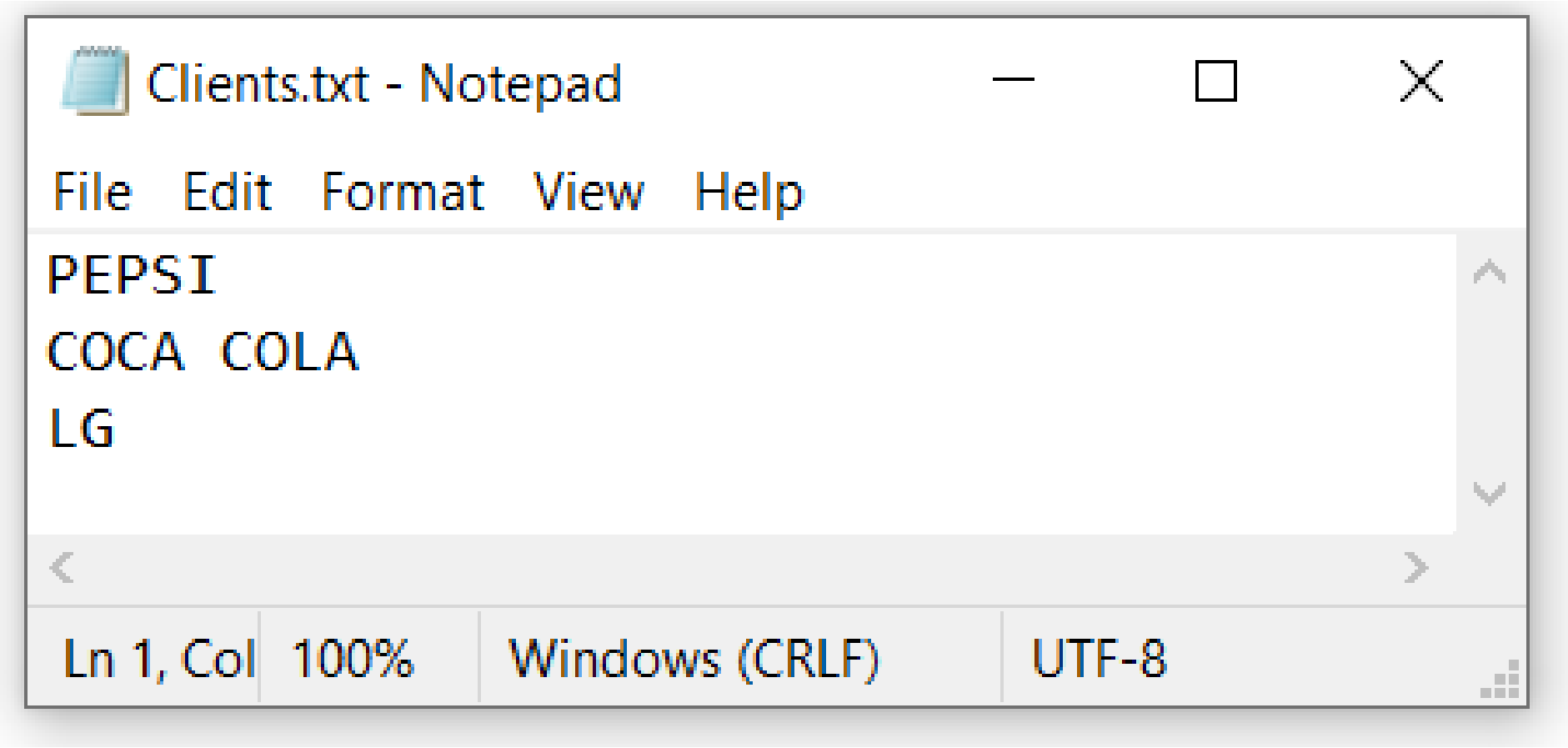
Note

$ git log - - oneline

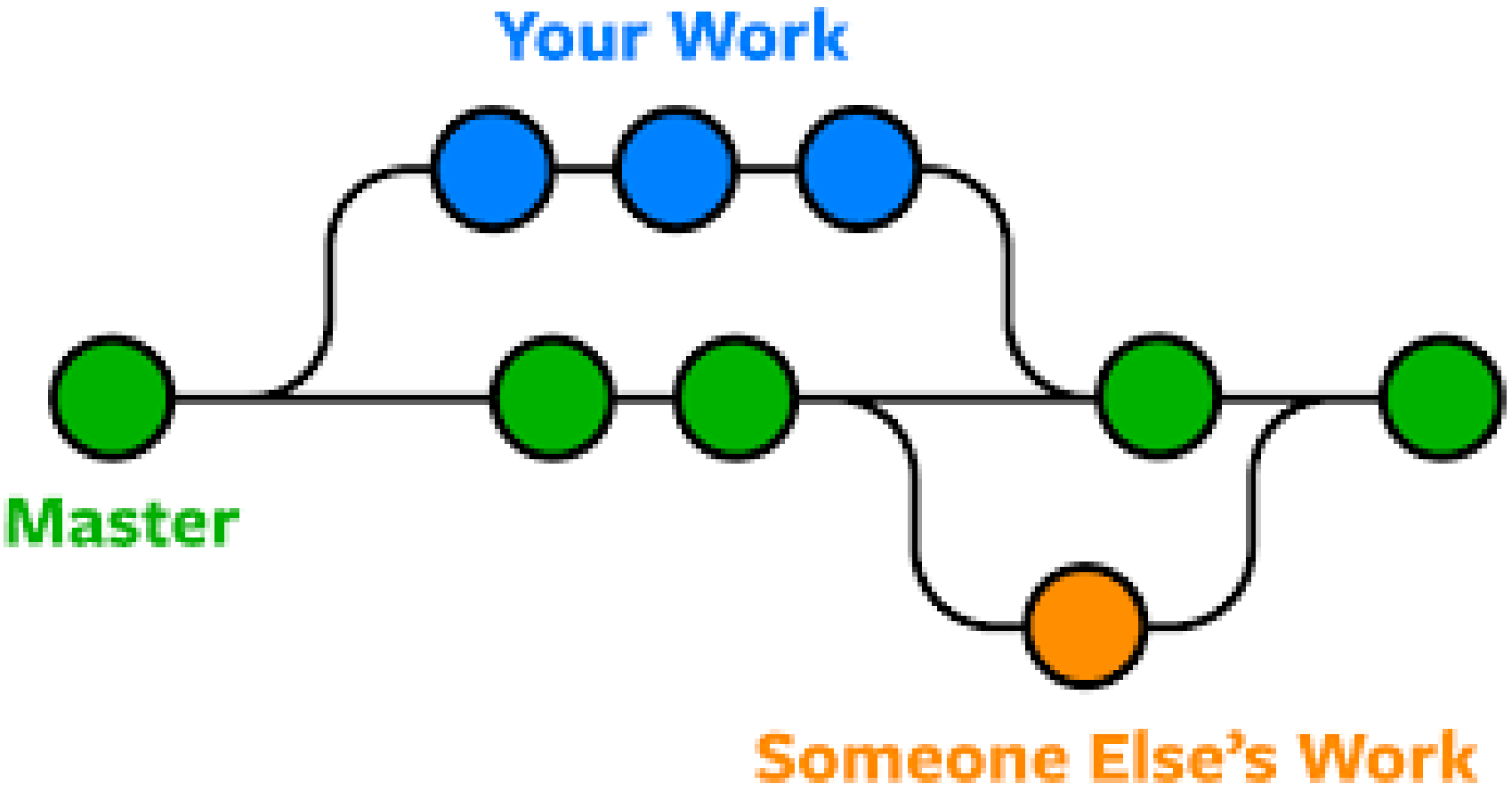
In your Git Bash

```
[master 2687c16] Revert "new clients"
 1 file changed, 1 insertion(+), 3 deletions(-)

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git log --oneline
2687c16 (HEAD -> master) Revert "new clients"
5032123 new clients
4a671af Including Clients
```

File Clients.txt



Our objective



Branches

A **Branch** is a version of your repository. An independent line of development.

> **Note**
>
> $ git branch potential

into branch potential: checkout

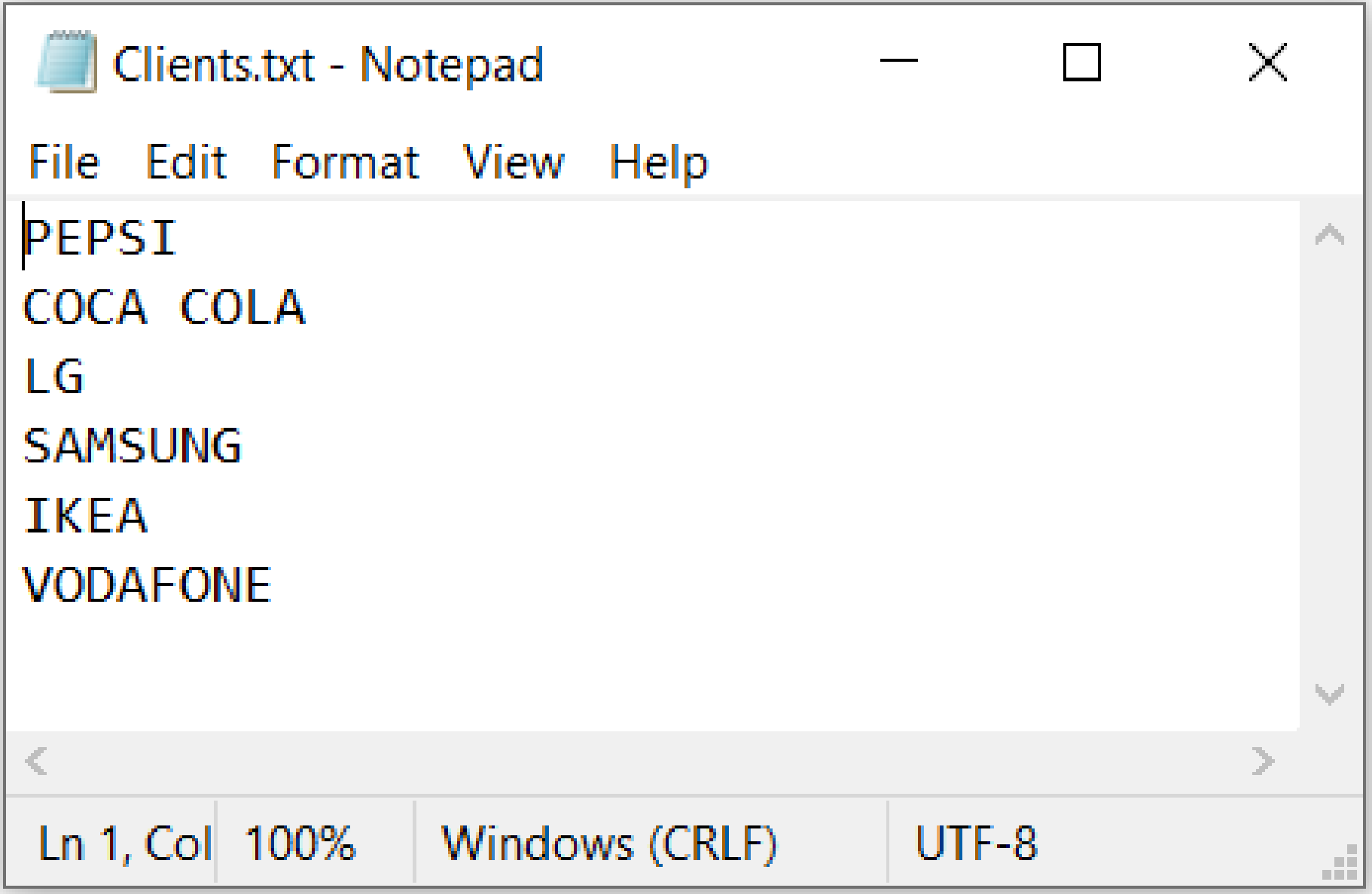
> **Note**
>
> $ git checkout potential

> **Note**
>
> $ git branch -l

In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git branch potential

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git checkout potential
Switched to branch 'potential'

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (potential)
$ git branch -l
  master
* potential
```

Changes in Clients.txt



In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/Pam (potential)
$ git branch -l
  master
* potential

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (potential)
$ git add Clients.txt

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (potential)
$ git diff
diff --git a/Clients.txt b/Clients.txt
index f00cf49..5934292 100644
--- a/Clients.txt
+++ b/Clients.txt
@@ -1,3 +1,6 @@
 PEPSI
 COCA COLA
-LG
\ No newline at end of file
+LG
+SAMSUNG
+IKEA
+VODAFONE
\ No newline at end of file

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (potential)
$ git commit -m "potential clients"
On branch potential
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:   Clients.txt
```

Branch vs Master

Make changes

Note

- $ git add .
- $git commit -m “potential clients”

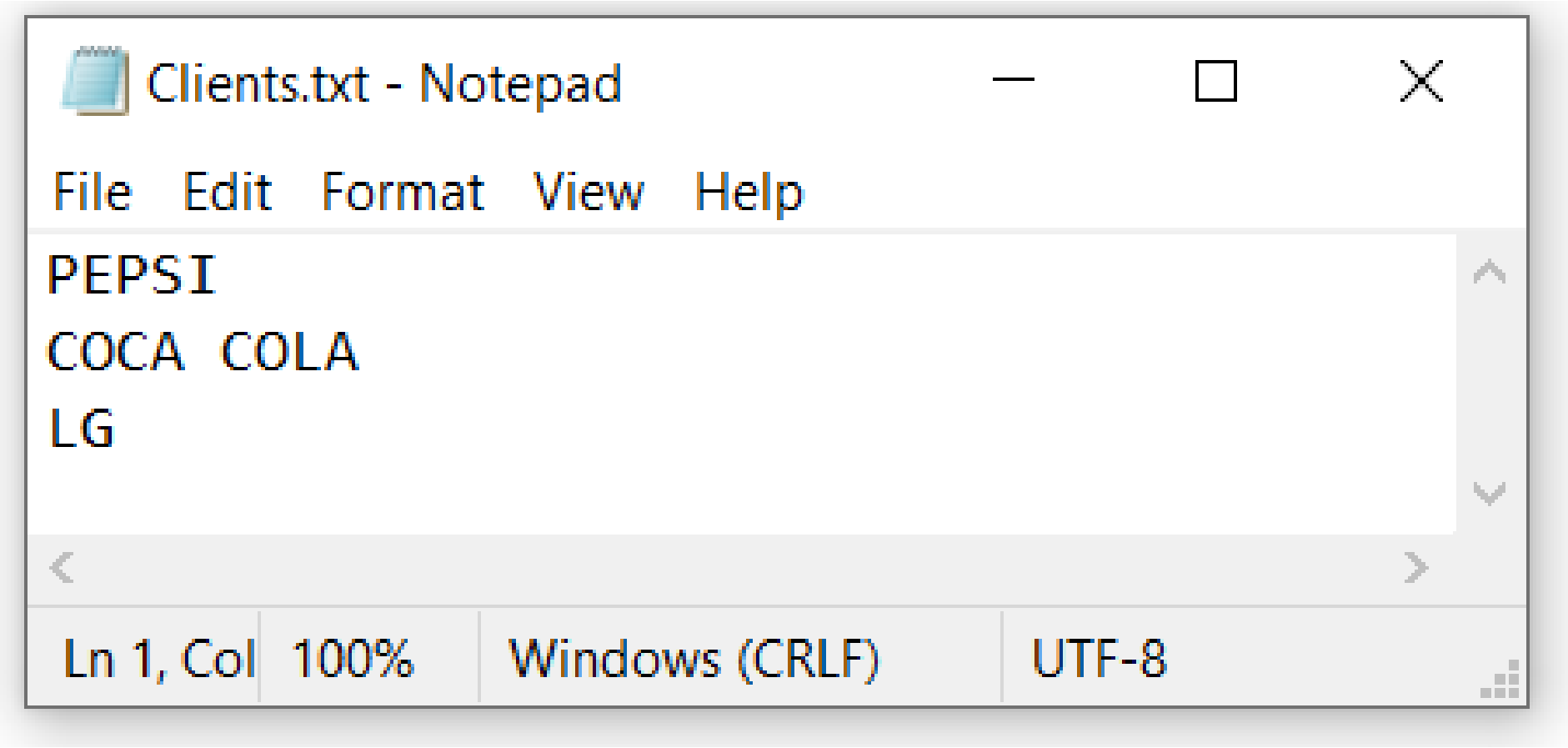
Master branch still old version

- $ git checkout master

In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/Pam (potential)
$ git checkout master
Switched to branch 'master'
M       Clients.txt

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git status
On branch master
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:   Clients.txt

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

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git diff
diff --git a/Clients.txt b/Clients.txt
index f00cf49..5934292 100644
--- a/Clients.txt
+++ b/Clients.txt
@@ -1,3 +1,6 @@
 PEPSI
 COCA COLA
-LG
\ No newline at end of file
+LG
+SAMSUNG
+IKEA
+VODAFONE
```

File Clients.txt in MASTER



Merge branches

- Put the changes from **potential** into **master**

Note

```
$ git checkout master
```

Note

```
$ git merge potential
```

In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git merge potential
Already up to date.
```

Deleting potential branch

Note

```
$ git branch -d potential
```

Note

```
$ git branch -l
```

In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git branch -d potential
Deleted branch potential (was 2687c16).

Pamela@PC1022350854 MINGW64 ~/Documents/Pam (master)
$ git branch -l
* master
```

Webservices

REMOTE repositories

GitLab

Plan, organize, and track team progress using Scrum, Kanban, SAFe, and other Agile methodologies.

GitHub

Projects is an adaptable, flexible tool for planning and tracking work on GitHub.

REMOTE repository

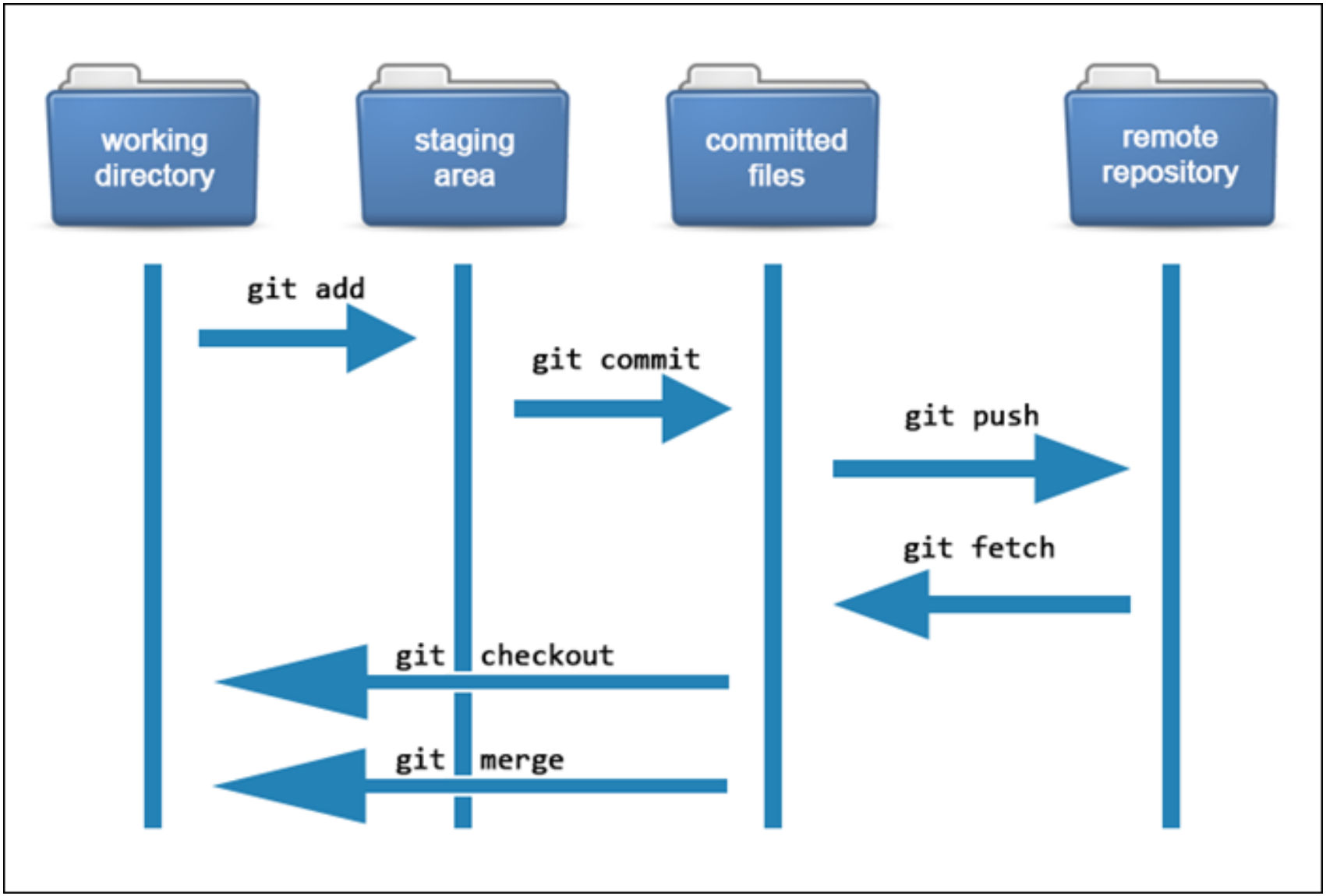
GitHub - *GitLab* your free WEBSERVICES

Commit all your changes and **push** your **branch** to REMOTE

Note

- $ git remote add origin URL
- $ git push -u origin master

GIT workflow details



Tracking an existing remote project

Note

```
$ git clone URL
```

Command that clones all the files from the repository and includes in my working directory

- The **url** from GitLab or GitHub

Tracking: Cloned project

- Make changes in the code:
  - add, commit, diff, status
- **Push**

> **! Important**
>
> *REMEMBER*: Multiple developers

PULL any changes that have been made since the last time that we cloned the repository

- $ git pull origin master
- $ git push origin master

In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents (master)
$ git clone https://github.com/pchiroque/NewTest.git
Cloning into 'NewTest'...
warning: You appear to have cloned an empty repository.
```

In your Git Bash

```
Pamela@PC1022350854 MINGW64 ~/Documents/NewTest (main)
$ git add README.md

Pamela@PC1022350854 MINGW64 ~/Documents/NewTest (main)
$ git commit -m "first commit"
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean

Pamela@PC1022350854 MINGW64 ~/Documents/NewTest (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 228 bytes | 228.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/pchiroque/NewTest.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Common commands

- $ git branch DOTHIS
- $ git checkout DOTHIS
- make modifications
- $ git add . and then $ git commit -m “changes”
- $ git checkout master
- $ git pull origin master
- $ git merge DOTHIS
- $ git push origin
- $ git branch -d DOTHIS

Wins-Benefits with GIT

- Focus on content.
- OPT in, not OPT-OUT: GIT's idea of the staging area.
- OPEN, not LOCKED, let people make contibution, **open source** project, free.
- DISTRIBUTED, not centralized.
- Exchange of code. WEBSERVICES like **GitHub** or **GitLab**.
- Focus on the PEOPLE, NOT TOOLS
- JOURNAL, NOT BACKUP. Version control system - historical changes
- ANYWHERE, NOT JUST ONLINE: GIT works entirely **OFFLINE** mode

Exercises

Explore commands

- $ git fetch
- What is difference between git fetch and git pull?

Aditional Documentation

https://git-scm.com/

https://education.github.com/git-cheat-sheet-education.pdf.
https://about.gitlab.com/images/press/git-cheat-sheet.pdf

https://ndpsoftware.com/git-cheatsheet.html

- Book Reference

Chacon, Scott, and Ben Straub. 2014. *Pro Git*. Apress.