



# Version Control with Git

# Agenda

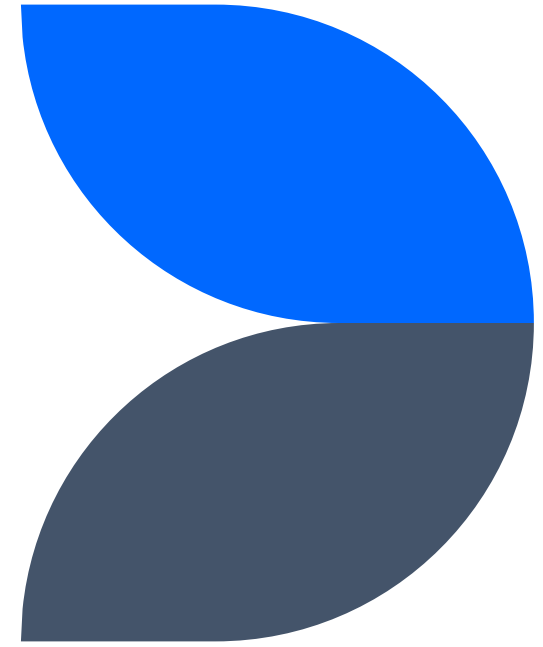
- **What** is version control and Git?
- How to **get** and **use** Git?
- Basic **commands**

# Acknowledgment

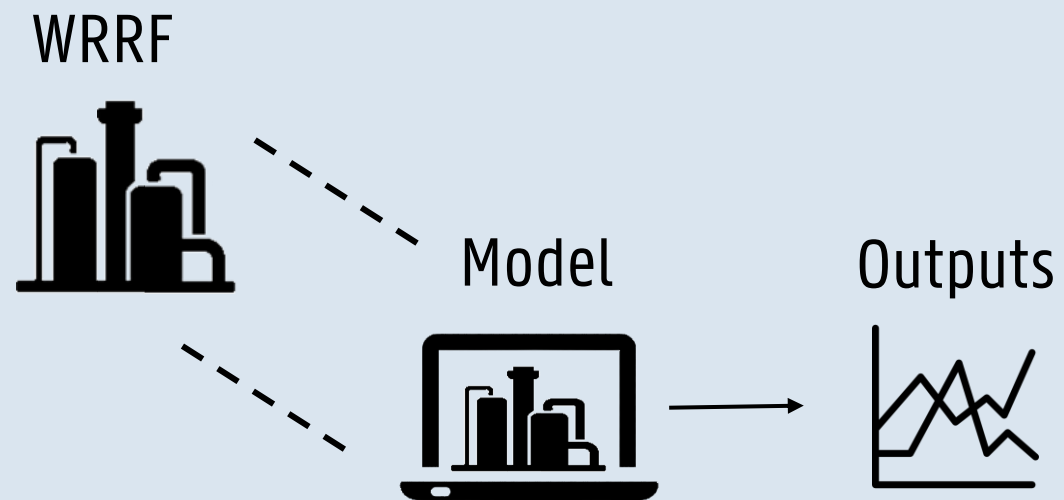
- Joris Meys
- Stijn Van Hoey
- Joris Van De Bossche
- Daan Van Hauwermeiren
- David Fernandes del Pozo
- Kensaku Matsunami
- Juan Pablo Gallo Molina
- Saba Daneshgar

# What is Version Control?

Why should you use it?

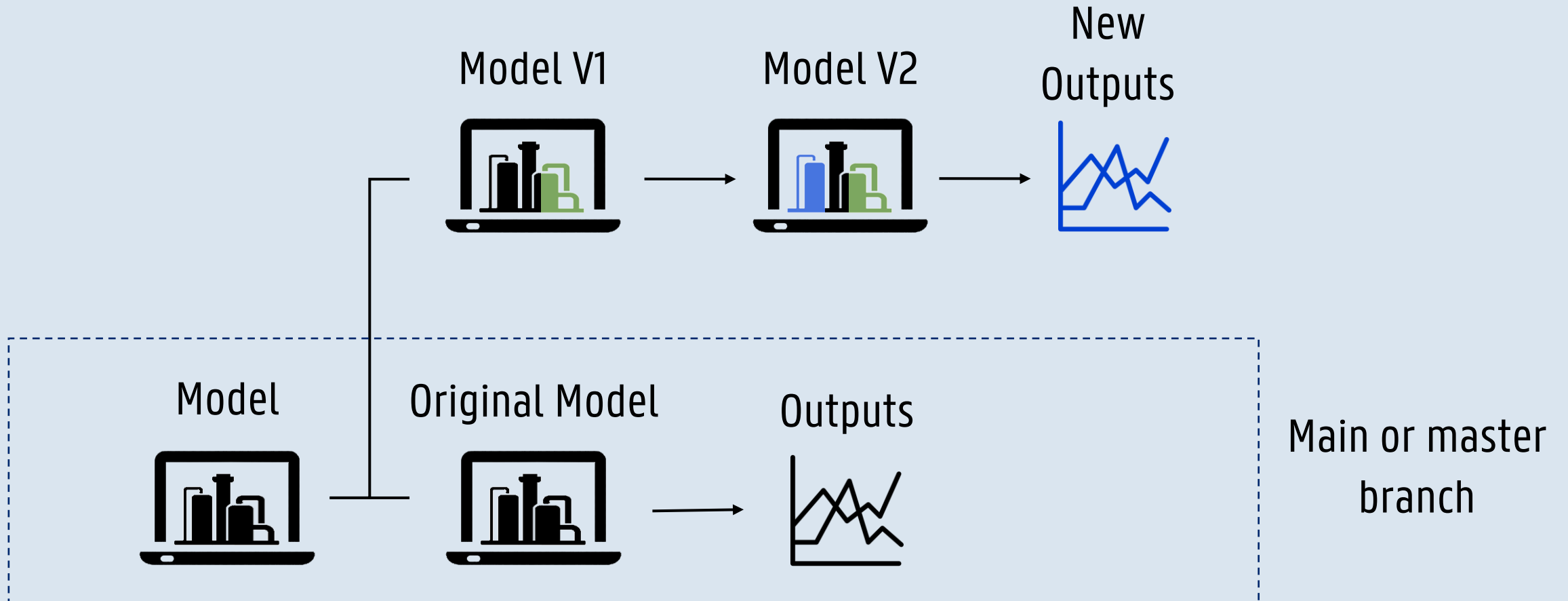


# Version Control



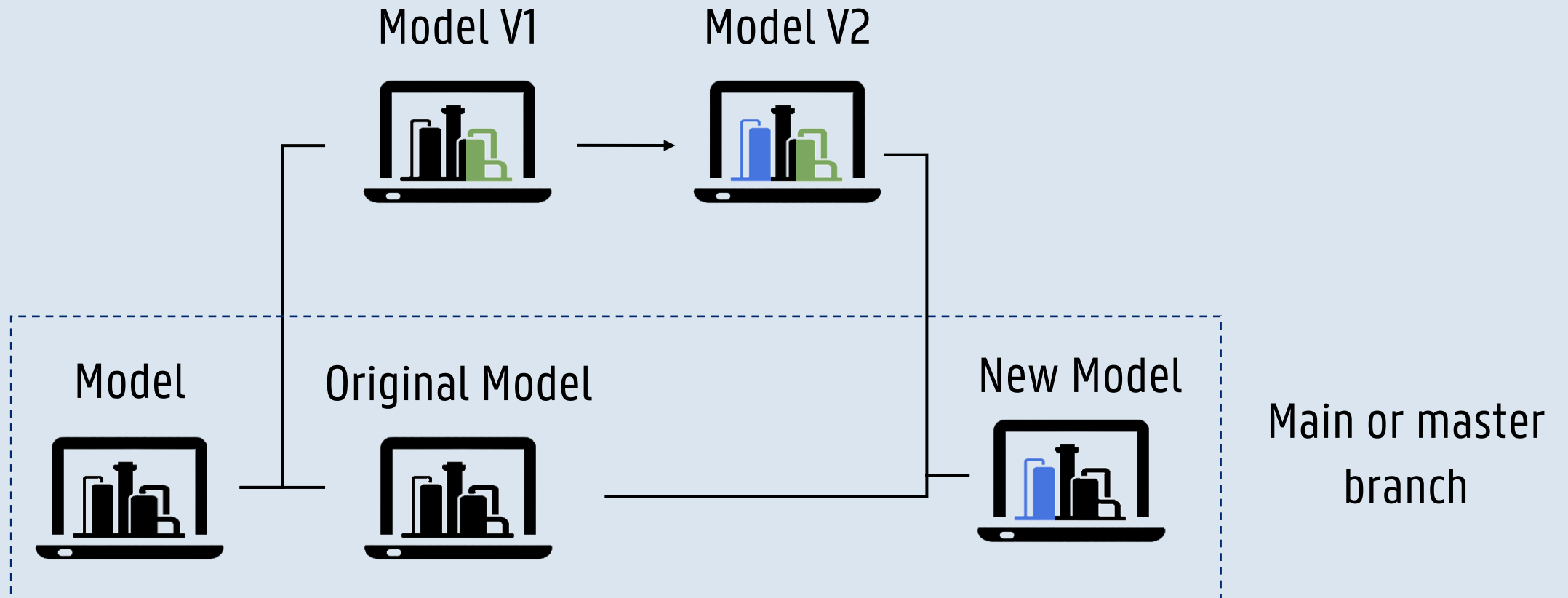
# Version Control

Create a branch



# Version Control

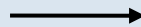
Merge a branch



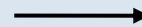
# Version Control

Restore previous versions

New Model



Model V2



Model V1



Original Model





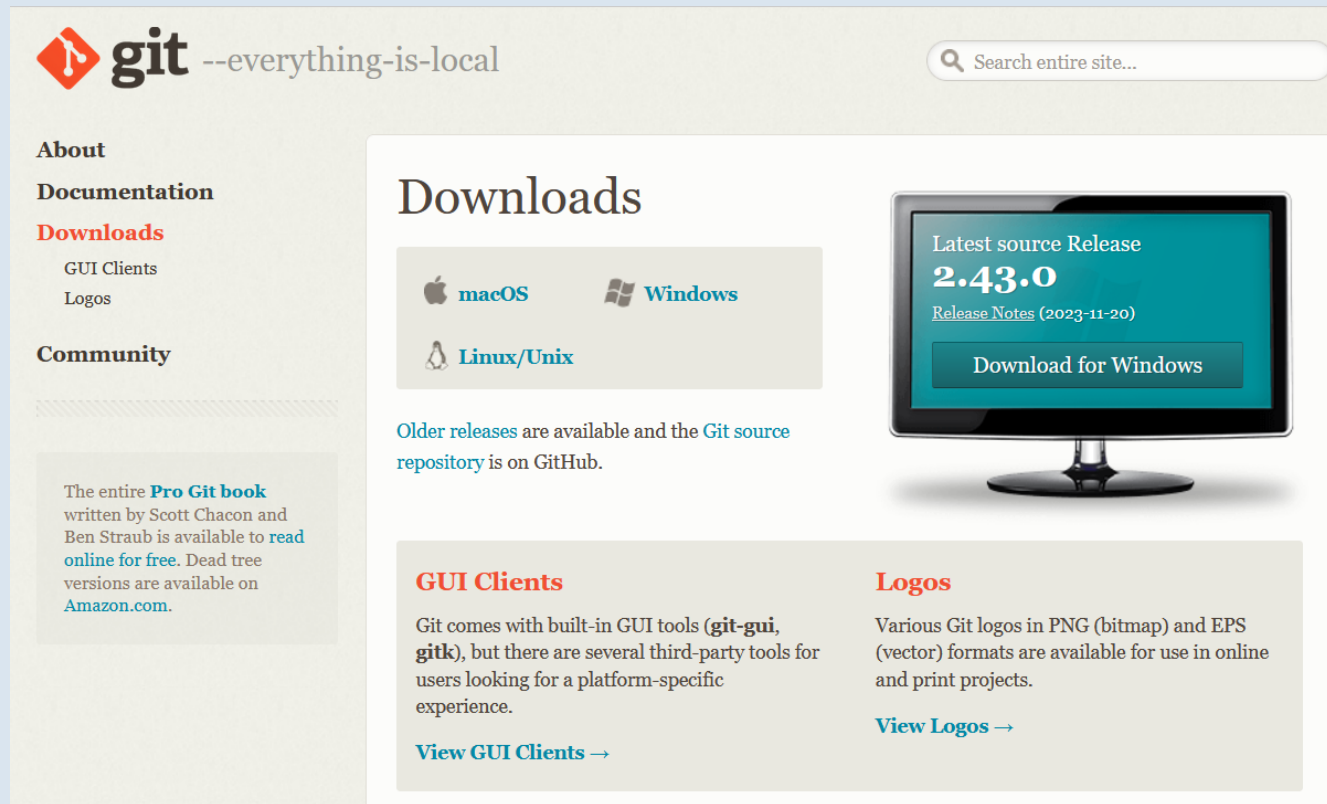
# What is Git?



Open-source and free  
Source Control Management (SCM)

# Git

<https://git-scm.com/>



The screenshot shows the Git website homepage. At the top left is the Git logo (a red diamond with a white branching diagram) followed by the text "git --everything-is-local". To the right is a search bar with the placeholder text "Search entire site...". Below the header, there is a left sidebar with navigation links: "About", "Documentation", "Downloads" (highlighted in red), "GUI Clients", "Logos", and "Community". The main content area features a "Downloads" section with a large heading. Below the heading, there are three platform-specific download buttons: "macOS" (with an Apple logo), "Windows" (with a Windows logo), and "Linux/Unix" (with a Linux logo). To the right of these buttons is a monitor graphic displaying the "Latest source Release 2.43.0" and a "Download for Windows" button. Below the download buttons, there is a text block stating "Older releases are available and the Git source repository is on GitHub." At the bottom of the page, there are two columns of text. The left column is titled "GUI Clients" and describes built-in GUI tools like "git-gui" and "gitk", along with third-party tools. The right column is titled "Logos" and describes various Git logos available in PNG and EPS formats. Both columns have a "View" link at the bottom.

**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](#).

## Downloads

macOS Windows Linux/Unix

Latest source Release  
**2.43.0**  
[Release Notes \(2023-11-20\)](#)  
Download for Windows

Older releases are available and the [Git source repository](#) is on GitHub.

### GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

### Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

# Git

Git work on any terminal on your computer

Git Bash

```
MINGW64~/c/Users/me/git
$ git clone https://github.com/git-for-windows/git
Cloning into 'git'...
remote: Enumerating objects: 500937, done.
remote: Counting objects: 100% (3486/3486), done.
remote: Compressing objects: 100% (1415/1415), done.
remote: Total 500937 (delta 2494), reused 2917 (delta 2071), pack-reused 497451
Receiving objects: 100% (500937/500937), 221.14 MiB | 1.86 MiB/s, done.
Resolving deltas: 100% (362274/362274), done.
Updating files: 100% (4031/4031), done.
$ cd git
$ git
$ git status
On branch main
Your branch is up to date with 'origin/main'.
nothing to commit, working tree clean
$
```

PowerShell

```
Windows PowerShell
PS C:\Users\jsoto> Get-WmiObject -Class Win32_BIOS -ComputerName .

SMBIOSBIOSVersion : 2.80
Manufacturer      : American Megatrends Inc.
Name               : 2.80
SerialNumber       : Default string
Version            : ALASKA - 1872089

PS C:\Users\jsoto> Get-WmiObject -Class Win32_ComputerSystem -Property UserName -ComputerName .

    __GENUS__ : 2
    __CLASS__ : Win32_ComputerSystem
    __SUPERCLASS__ :
    __DYNASTY__ :
    __RELPATH__ :
    __PROPERTY_COUNT__ : 1
    __DERIVATION__ : {}
    __SERVER__ :
    __NAMESPACE__ :
    __PATH__ :
    UserName : NONE-PC\jsoto
    PSComputerName :

PS C:\Users\jsoto> ps | sort -p ws | select -last 5

Handles NPM(K) PM(K) WS(K) CPU(s) Id SI ProcessName
-----
1679 82 218504 188808 1632 1 dm
719 92 174592 188808 258.09 15732 1 Discord
1429 110 148656 207816 12.80 12152 1 SearchApp
5198 200 155120 235592 406.94 10308 1 explorer
1206 81 352068 385992 455.55 13816 1 Telegram

PS C:\Users\jsoto>
```

CMD

```
Administrador: C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Windows\system32>cd..

C:\Windows>cd..

C:\>COMANDOS\INTERESANTES\CMD
```

# Git and GitHub

What is the difference?



Version control system

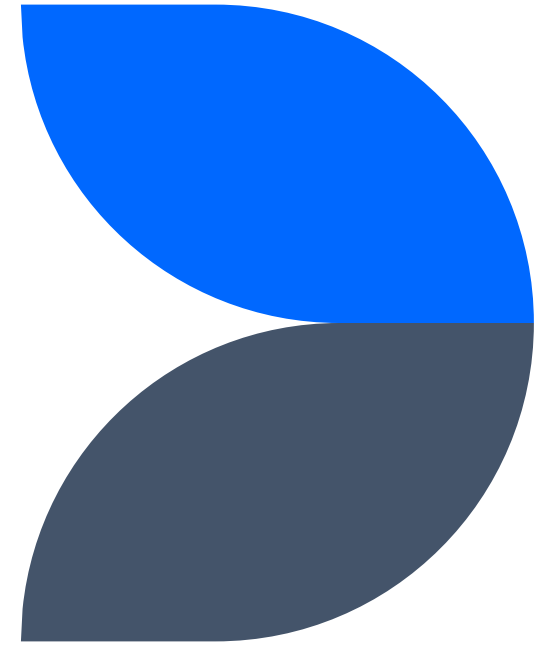


Back-up and sharing\*

\*One of many!

# Basic commands

On Git



# Working directory

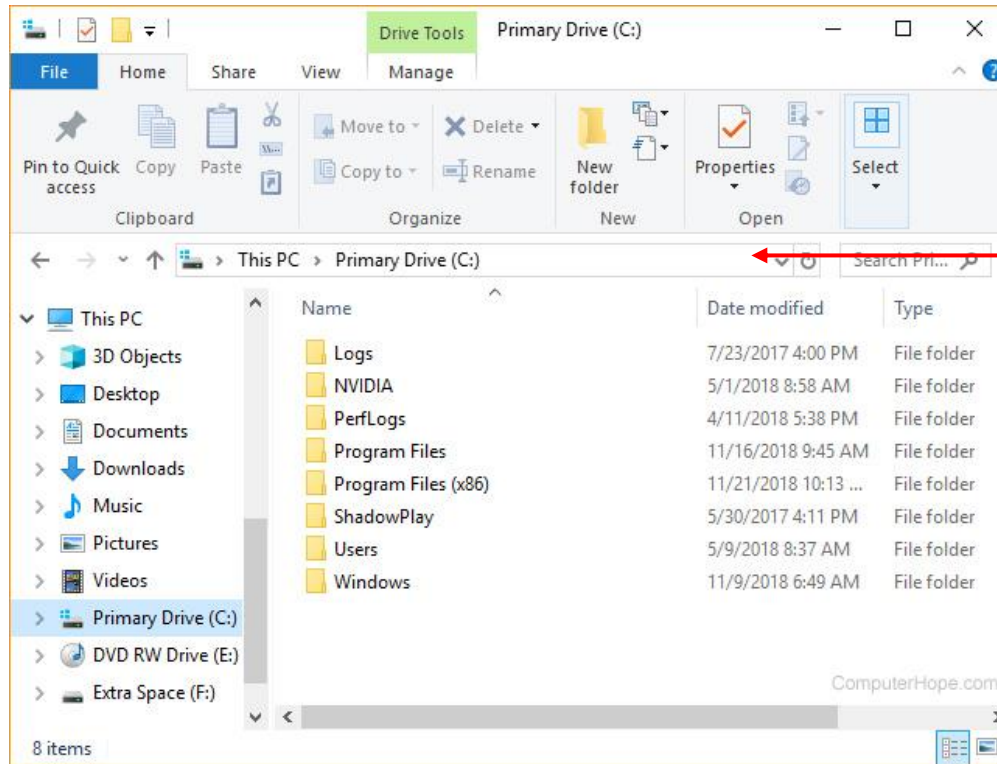
## Terminal

```
MINGW64 ~/c/Users/me/git
$ git clone https://github.com/git-for-windows/git
Cloning into 'git'...
remote: Enumerating objects: 500937, done.
remote: Counting objects: 100% (3486/3486), done.
remote: Compressing objects: 100% (1415/1415), done.
remote: Total 500937 (delta 2494), reused 2917 (delta 2071), pack-reused 497451
Receiving objects: 100% (500937/500937), 721.14 MiB | 1.86 MiB/s, done.
Resolving deltas: 100% (362274/362274), done.
Updating files: 100% (4031/4031), done.
MINGW64 ~/c/Users/me/git
$ cd git
MINGW64 ~/git (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.
nothing to commit, working tree clean
MINGW64 ~/git (main)
$ |
```

#cd stands for change directory  
cd <directory\_path>

# Working directory

## File explorer



Type “cmd” in the folder  
you want to work on

# Basic Git

## Start version controlling locally

```
# inside the directory, this creates the .git file  
$ git init
```

## Check status

```
# Check tracked and untracked files  
$ git status
```

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)  
$ git status  
On branch master  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    demofile  
  
nothing added to commit but untracked files present (use "git add" to track)
```



# Basic Git

## Get help on any command

```
# Documentation of any command $ git help <command>
$ git help init
```

### git-init(1) Manual Page

#### NAME

git-init - Create an empty Git repository or reinitialize an existing one

#### SYNOPSIS

```
git init [-q | --quiet] [--bare] [--template=<template-directory>]
          [--separate-git-dir <git-dir>] [--object-format=<format>]
          [-b <branch-name> | --initial-branch=<branch-name>]
          [--shared[=<permissions>]] [<directory>]
```

#### DESCRIPTION

This command creates an empty Git repository - basically a `.git` directory with subdirectories for `objects`, `refs/heads`, `refs/tags`, and template files. An initial branch without any commits will be created (see the `--initial-branch` option below for its name).

If the `$GIT_DIR` environment variable is set then it specifies a path to use instead of `./.git` for the base of the

# Basic Git

## Start tracking changes in files

```
# Add files to start tracking them $ git add <file_name>  
$ git add file1.txt  
$ git add -all  
$ git add .
```

# Basic Git

## Start tracking changes in files

```
Akash Jha@LAPTOP-LJJ1U61G MINGW64 ~/Desktop/Git/SetUp (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt
        file2.txt

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

Akash Jha@LAPTOP-LJJ1U61G MINGW64 ~/Desktop/Git/SetUp (master)
$ git add .

Akash Jha@LAPTOP-LJJ1U61G MINGW64 ~/Desktop/Git/SetUp (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt
        new file:   file2.txt
```

# What is commit?

Save a copy or milestone of the current status of your files

# Basic Git

## Commit current status of your files

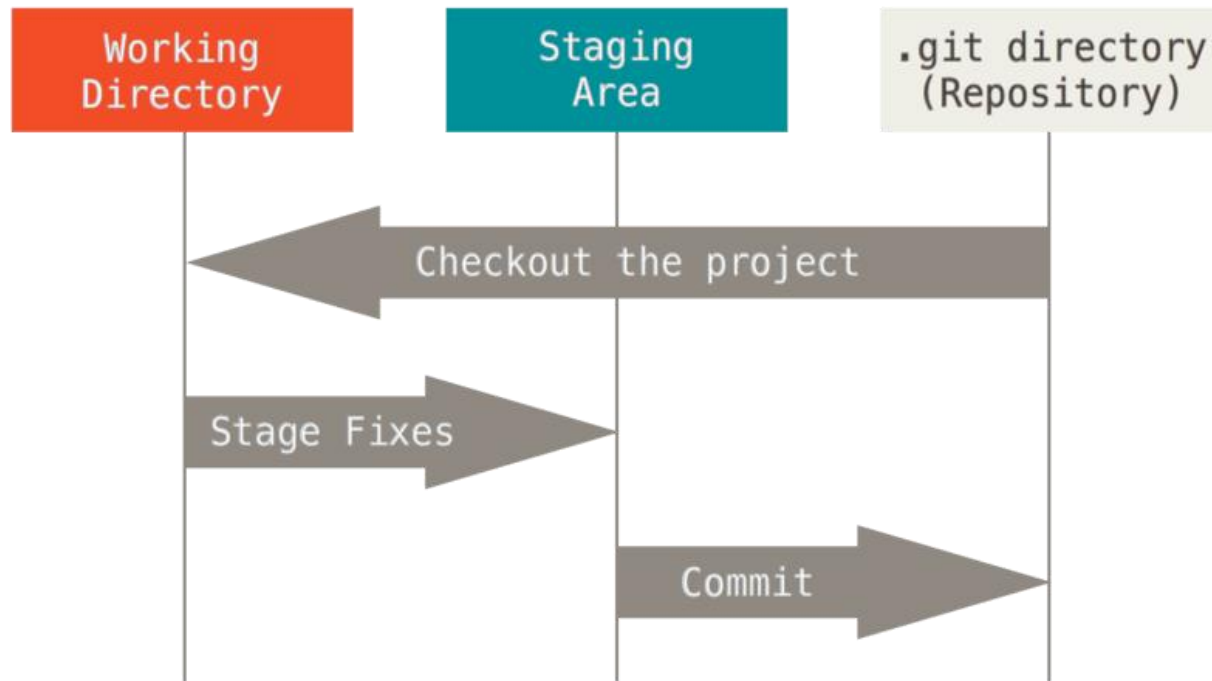
```
# Commit files $git commit -m (message) "<message>"  
$ git commit -m "Message"  
  
# Format for meaningful commits <kind of commit>: <file touched if only one>: <brief description>.  
<mention or resolve related issue with "working on issue #number" or "resolve issue #number">  
$ git commit -m "feat: file1.py: new function to print print(). Resolve issue #69"  
$ git commit -m "doc: improving documentation in many files. Used "parameters" instead of "inputs" "
```

## Check differences between versions

```
# Check differences between files or commits  
$ git diff
```

# Basic Git

## Git areas



# Basic Git

## Add and remove from staging area

```
# Add files to start tracking them $ git add <file_name>  
$ git add file1.txt
```

```
# Remove from staging area  
$ git restore --staged <name>
```

```
# Files in the stage are committed and those in the working directory are not
```

“

“Remember, anything that is committed in  
Git can almost always be recovered...  
**However, anything you lose that was never  
committed is likely never to be seen again.”**

ProGit

”



# Basic Git

## Check commit list

```
# Check commit list with commit number and description --oneline (make it shorter)
$ git log --oneline
```

## Check log

```
deepapandey@Deepas-MacBook-Air RealTime_Foodies % git log --oneline
23c49c7 (HEAD -> main, heroku/main) Updating in heroku
c8c75b0 Updating in heroku
6e963dd (origin/main, origin/HEAD) Productuin build
fc8a730 completed the realtime part
019df82 updated
914815d added login reg property
ab37a7d database setup and backend setup
5e29018 almost completed the UI Part
cf1d0ff updated the UI part
64b4909 first commit
```

# Basic Git

## Restore commit

```
# Restore commit from the commit id $ git reset <commit_id>  
$ git reset "23c49c7"
```

## Clone a repository

```
# Clone an existing repository $ git clone <source>  
$ git clone https://gitlab.com/datinfo/PeePyPoo.git
```

# What is a branch?

“A pointer to a snapshot of your changes in the files” -Git

# Basic Git

## Create a branch

```
# Create a branch $ git branch <branch_name>  
$ git branch Model_V1
```

## Show existing branches and current position

```
# Show existing branches and highlight the branch you are working on  
$ git branch
```

# Basic Git

## Switch branch

```
# Switch to the branch you are going to work on $ git switch <branch_name>  
$ git switch Model_V1
```

## Merging branches

```
# Combining the changes in two branches $ git merge -m "<message>"  
$ git merge -m "New unit process process_v1"
```

## Delete branch

```
# Delete the branch you have merged $ git branch -d <branch_name>  
$ git branch -d Model_V1
```



**Thank you**