Git Architecture

References:

1. <https://indepth.dev/becoming-a-git-pro-part-1-internal-git-architecture/>
2. <https://medium.com/@willhayjr/the-architecture-and-history-of-git-a-distributed-version-control-system-62b17dd37742>

# Introduction

The regular cycle of Git: checkout > modify > commit.

## Git as a Folder

When you run git init in a folder, Git creates the .git directory.

mkdir git-playground && cd git-playground  
git init  
ls .git

The output is the following:

    Directory: D:\Projects\git-playground\.git  
  
Mode                 LastWriteTime         Length Name  
----                 -------------         ------ ----  
d-----        02-11-2020  10:21 AM                hooks  
d-----        02-11-2020  10:21 AM                info  
d-----        02-11-2020  10:21 AM                objects  
d-----        02-11-2020  10:21 AM                refs  
-a----        02-11-2020  10:21 AM            130 config  
-a----        02-11-2020  10:21 AM             73 description  
-a----        02-11-2020  10:21 AM             23 HEAD

This is where Git stores all your commits and other relevant information to manipulate these commits.

When you clone a repository:

1. Git clones this single directory into your folder
2. Creates remote-tracking branches for each branch in the cloned repository
3. Creates and checks out an initial branch that is specified by the HEAD file.

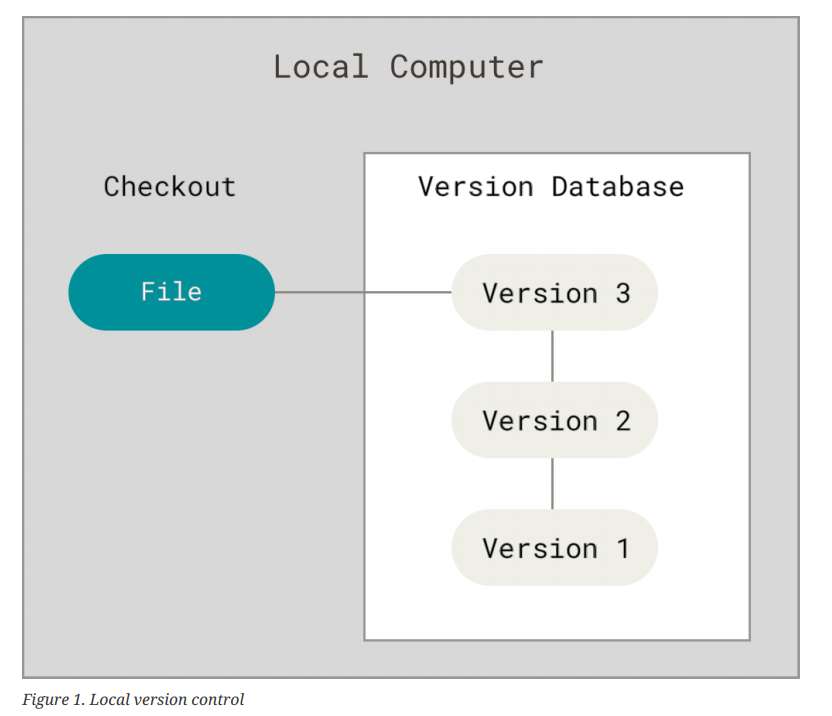
"cloning a repository is essentially just copying `.git` directory from the other location."

## Git as a Database

Git is a simple key-value data store, where a value is put into the repository and a key is retrieved by which the corresponding value can be accessed.

# Getting Started

## Local Version Control Systems



One of the most popular VCS tools was a system called RCS(Revision Version Control System), which is still distributed with many computers today. RCS works by keeping patch sets (that is, the differences between files) in a special format on disk; it can then re-create what any file looked like at any point in time by adding up all the patches.

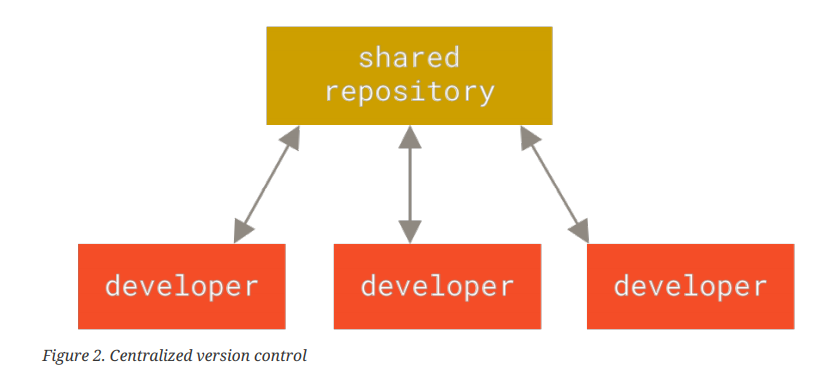
RCS manages multiple revisions of files. RCS automates the storing, retrieval, logging, identification, and merging of revisions.

## Centralized Version Control Systems

The next major issue that people encounter is that they need to collaborate with developers on

other systems. To deal with this problem, Centralized Version Control Systems (CVCSs) were

developed.



However, this setup also has some serious downsides. The most obvious is the single point of failure that the centralized server represents. If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they’re working on. If the hard disk the central database is on becomes corrupted, and proper backups haven’t been kept, you lose absolutely everything — the entire history of the project except whatever single snapshots people happen to have on their local machines. Local VCS systems suffer from this same problem — whenever you have the entire history of the project in a single place, you risk losing everything.

## Distributed Version Control Systems

In DVCS, clients don’t just check out the latest snapshot of the files; rather, they fully the repository, including its full history.

## What is Git ?