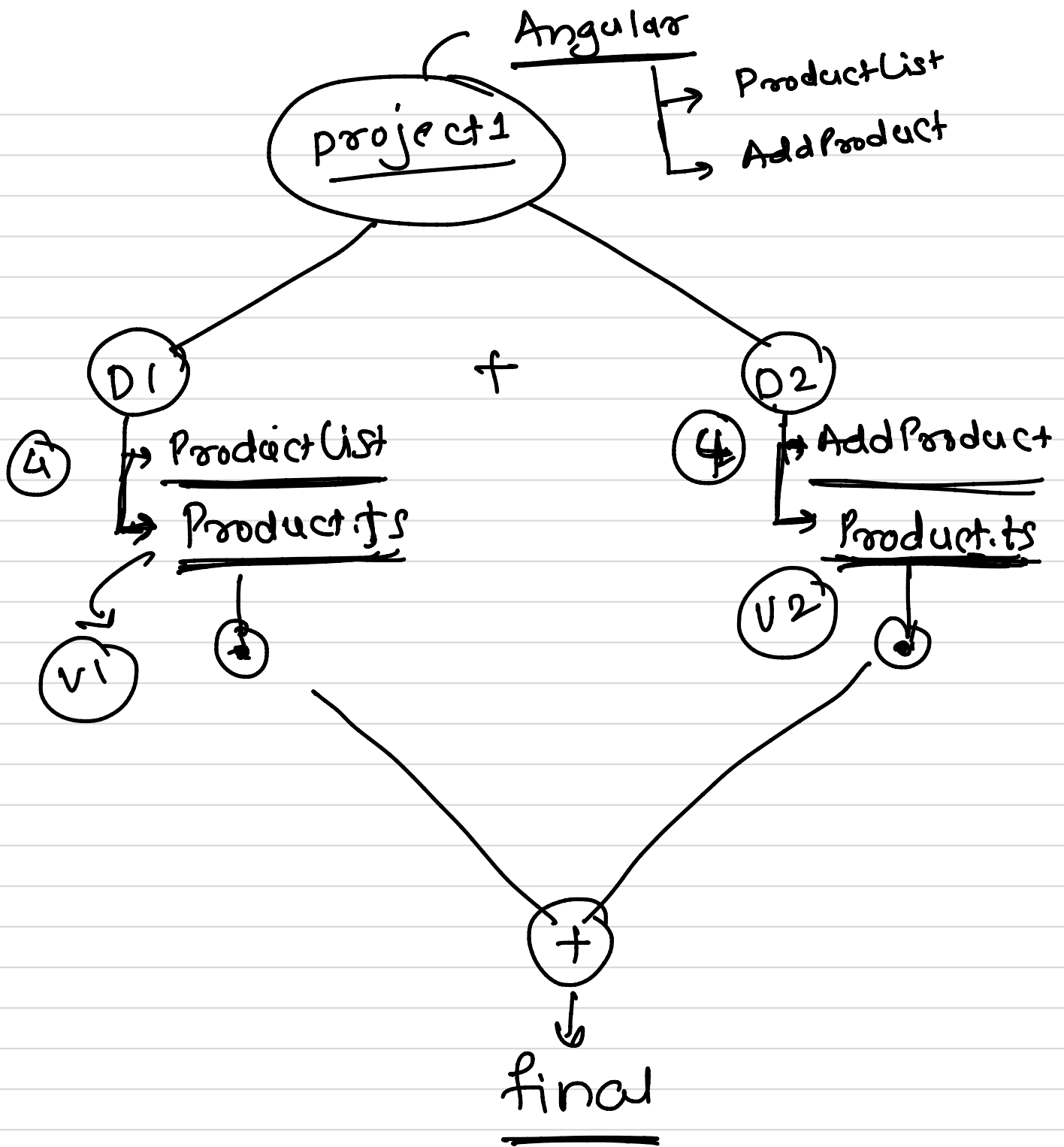


# Version Control System

# VCS - source code management (scm)

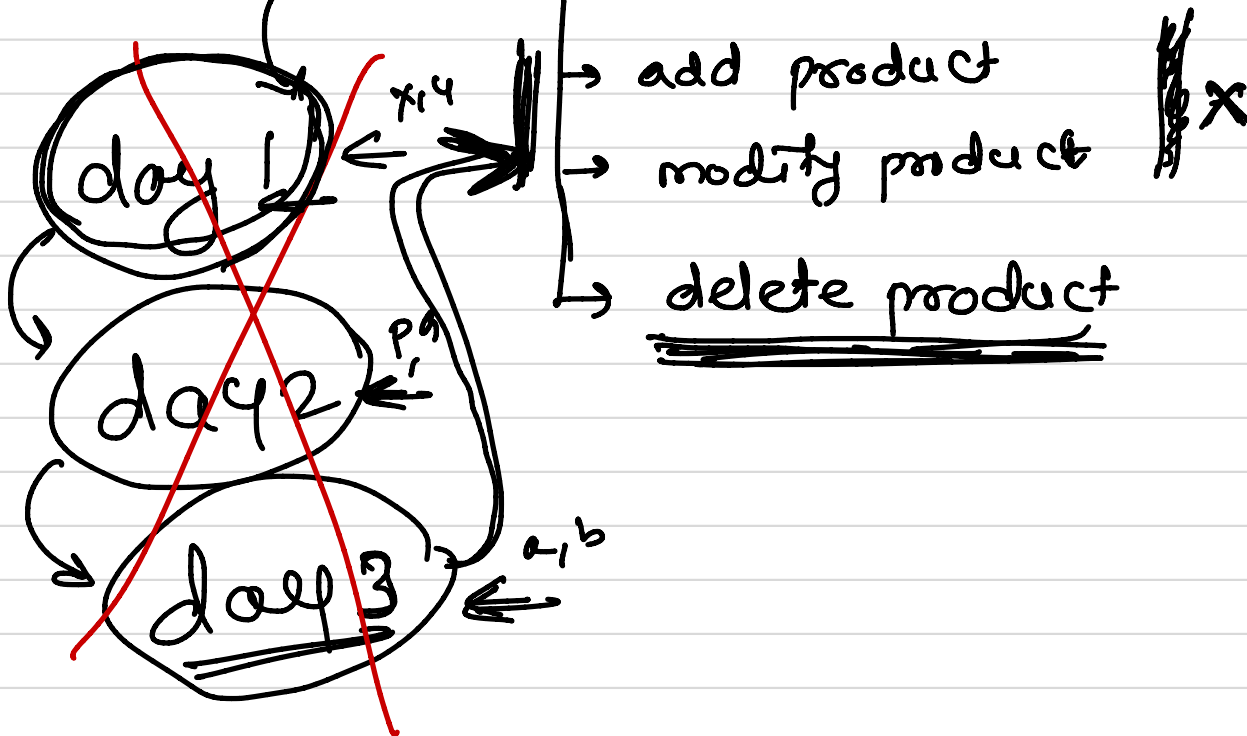
→ versions  
revisions

- also known as revision control or source code control system
- is the management of documents (source code)
- logical way to organize and control the revisions of source code
- tracks and provides control over the changes made in the code
- E.g.
  - ✓ CVS → Concurrent Versioning System
  - ✓ SVN → Subversion
  - ✓ Git →
  - ✓ Bazar →
- Types
  - ↳ • Client-Server
  - ↳ • Distributed





project

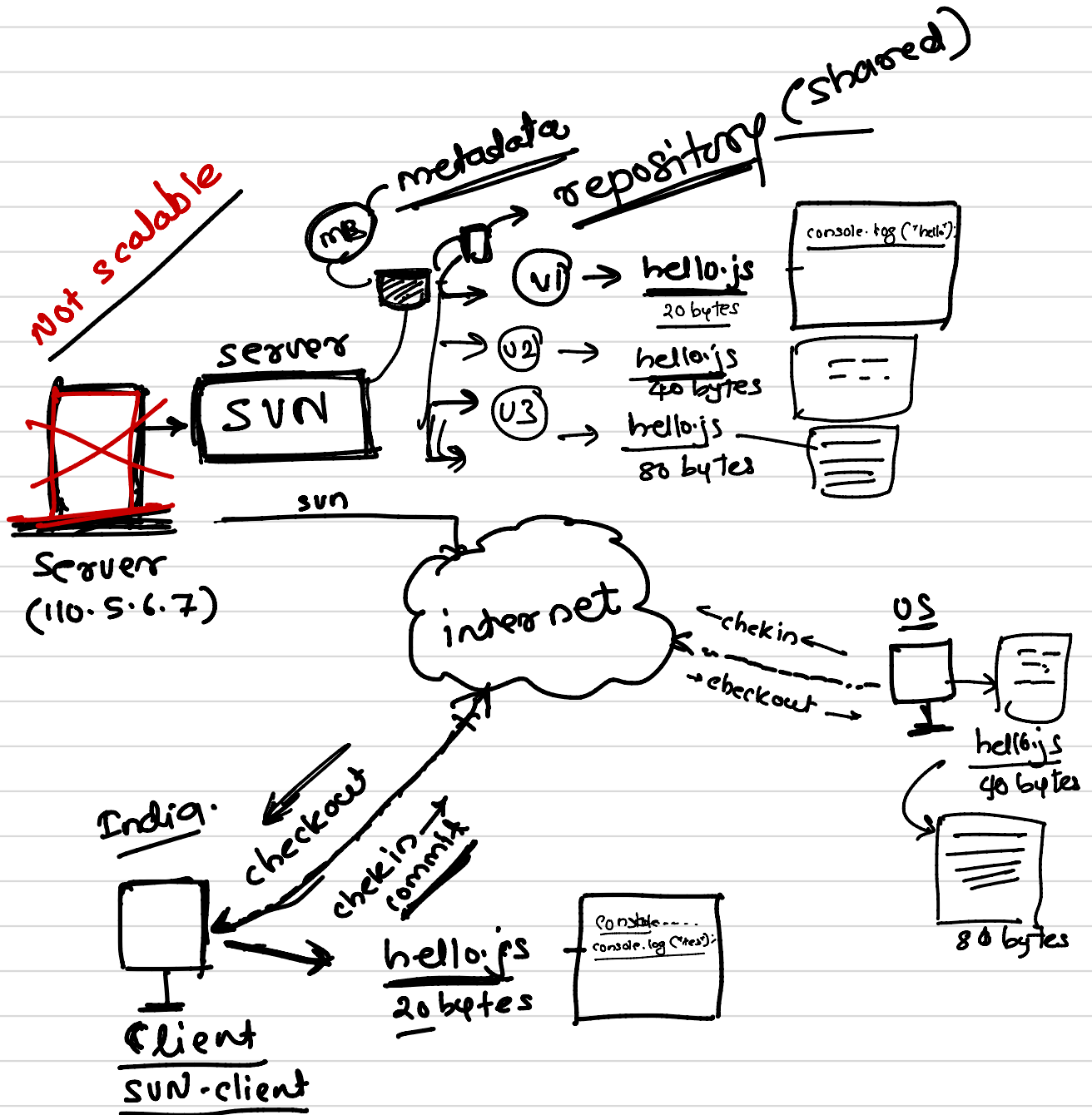
developer



# Client-Server model

- only one server maintains shared repository (repo) Directory which contains source code
- every developer sends the changes to the same repository
- disadvantages
  - ①  not scalable.
  - ②  dependency on the server
  - ③ client needs connectivity with the server to checkin the code
- E.g.
  - open source
    - ↳ CVS (Concurrent Version System)
    - ↳ SVN (Subversion)
  - Proprietary
    - ↳ AccuRev
    - ↳ Razor
    - ↳ TeamCity
    - ↳ Vault
    - ↳ Visual SourceSafe

client-server scm (sun)

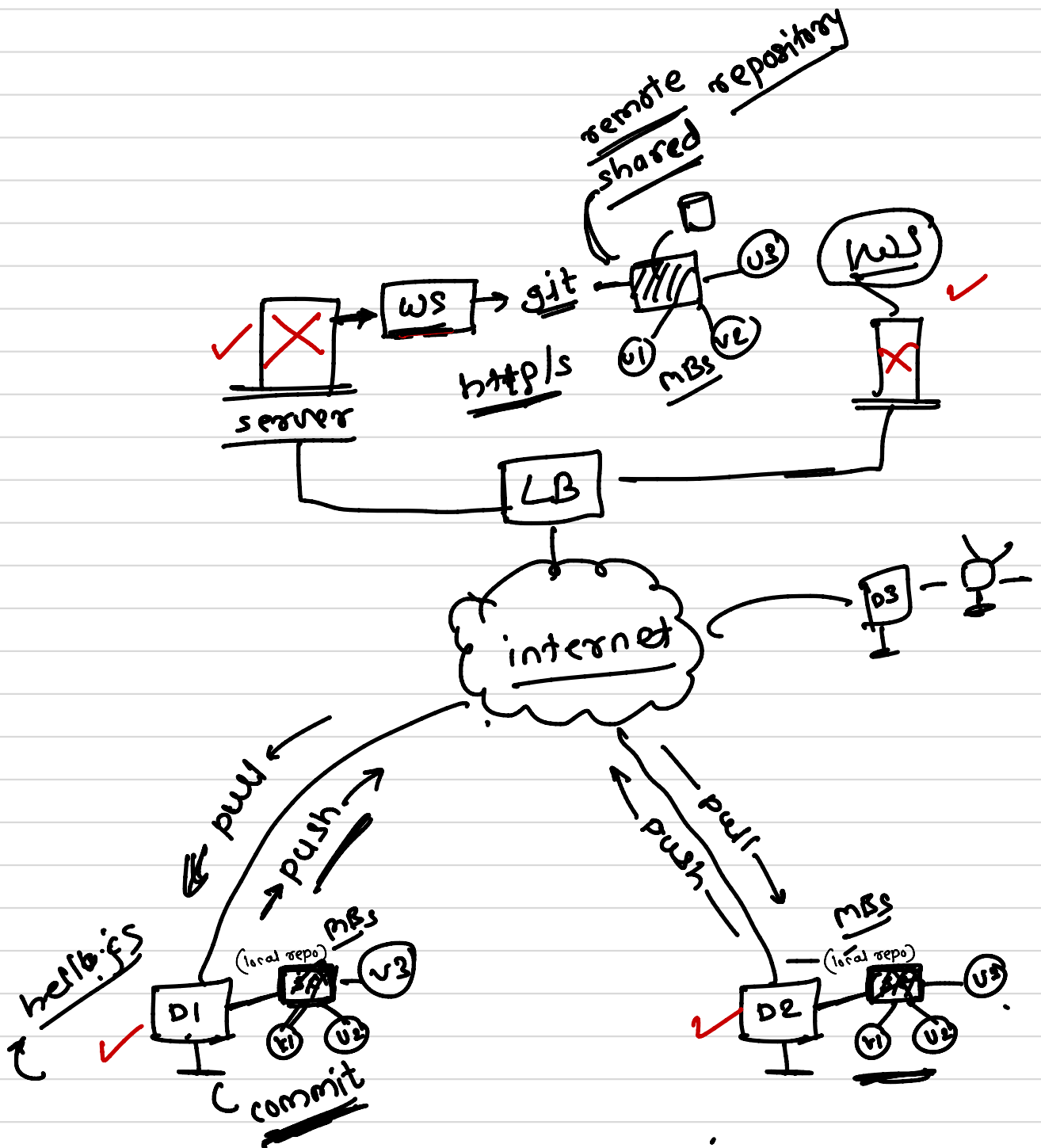


- ① a server will host scm server process
- ② a developer
  - ↳ checkout the latest code from server
  - ↳ checkin the new changes to the server

# Distributed model

- takes peer-to-peer approach to version control
- synchronizes repositories by exchanging patches from peer to peer
- there is no single server which maintains the code, rather user has a working copy and full change history

# Distributed scm (git)





# Distributed model – advantages

- allows users to work productively even when not connected to internet
- common operations like commit, version history etc. are faster because there is no need to communicate with server
- communication with server is necessary only when developer wants to share the changes with others
- allows private work, users don't need to publish the changes for early drafts
- working copies function effectively as backups
- permits centralized control of the release version of code

# Distributed model – disadvantages

- initial checkout of repository is slower than client-server version control system
- additional storage is required for every user

Git

# History

- Linux Kernel developers were using BitKeeper till 2005
- The copyright holder of BitKeeper, Larry McVoy, had withdrawn free use of the product after claiming that Andrew Tridgell had reverse-engineered the BitKeeper protocols
- Linus Torvalds wanted
  - To take Concurrent Versions System (CVS) as an example of what not to do
  - To support a distributed, BitKeeper-like workflow
  - To include very strong safeguards against corruption, either accidental or malicious
- But none of the available free systems met his needs

# History

- The development of Git began on 3 April 2005
- Torvalds announced the project on 6 April
- It became self-hosting as of 7 April
- The first merge of multiple branches took place on 18 April
- Torvalds achieved his performance goals on 29 April
- On 16 June Git managed the kernel 2.6.12 release
- Torvalds turned over maintenance on 26 July 2005 to Junio Hamano,  
a major contributor to the project

# Characteristics

- Strong support for non-linear development
- Distributed development
- Compatibility with existent systems and protocols
- Efficient handling of large projects
- Cryptographic authentication of history
- Toolkit-based design
- Pluggable merge strategies

↳ http / S

# Terminologies

- Repository

- Directory containing .git folder

- Object

- Collection of key-value pairs

- Blobs (Binary Large Object)

- Each version of a file is represented by blob ←
- A blob holds the file data but doesn't contain any metadata about the file
- It is a binary file, and in Git database, it is named as SHA1 hash of that file
- In Git, files are not addressed by names. Everything is content-addressed

local

remote (shared)

# Terminologies

## Clone

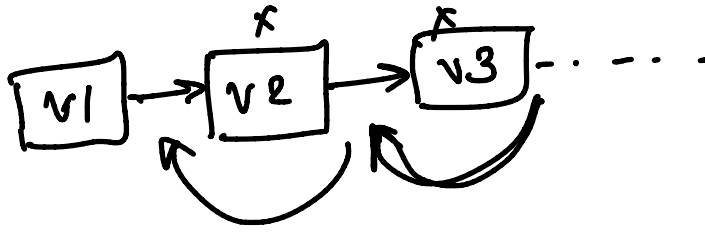
- Clone operation creates the instance of the repository
- Clone operation not only checks out the working copy, but it also mirrors the complete repository
- Users can perform many operations with this local repository
- The only time networking gets involved is when the repository instances are being synchronized

## Pull

- Pull operation copies the changes from a remote repository instance to a local
- The pull operation is used for synchronization between two repository instances



# Terminologies



## ✓ Push →

- Push operation copies changes from a local repository instance to a remote
- This is used to store the changes permanently into the Git repository

## • HEAD →

↳ commit

- HEAD is a pointer, which always points to the latest commit in the branch
- Whenever you make a commit, HEAD is updated with the latest commit
- The heads of the branches are stored in **.git/refs/heads/** directory

## • Commits

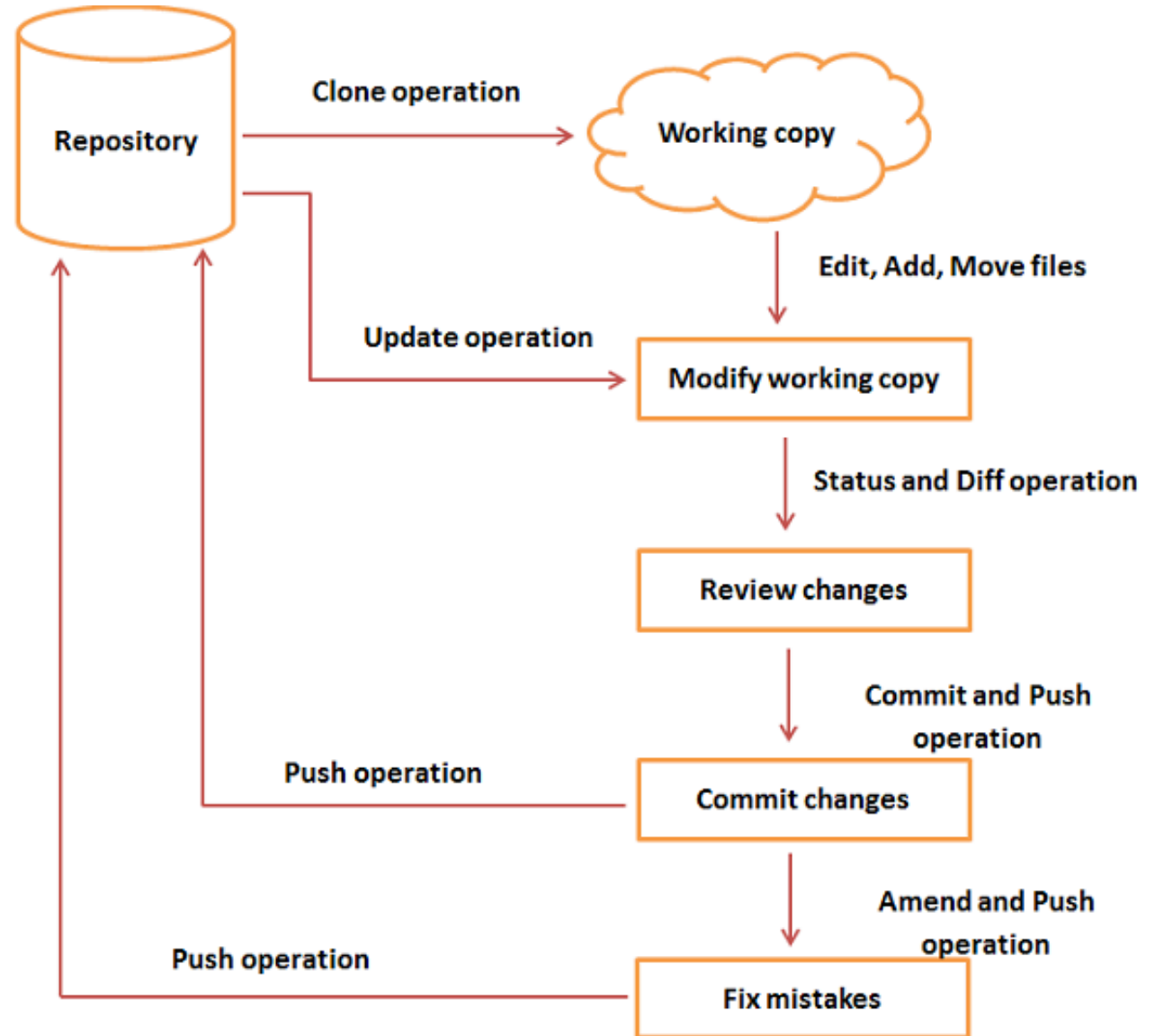
- Commit holds the current state of the repository.
- A commit is also named by **SHA1** hash
- A commit object as a node of the linked list
- Every commit object has a pointer to the parent commit object
- From a given commit, you can traverse back by looking at the parent pointer to view the history of the commit

# Terminologies

## • Branches

- Branches are used to create another line of development
- By default, Git has a master branch
- Usually, a branch is created to work on a new feature
- Once the feature is completed, it is merged back with the master branch and we delete the branch
- Every branch is referenced by HEAD, which points to the latest commit in the branch
- Whenever you make a commit, HEAD is updated with the latest commit

# Life Cycle



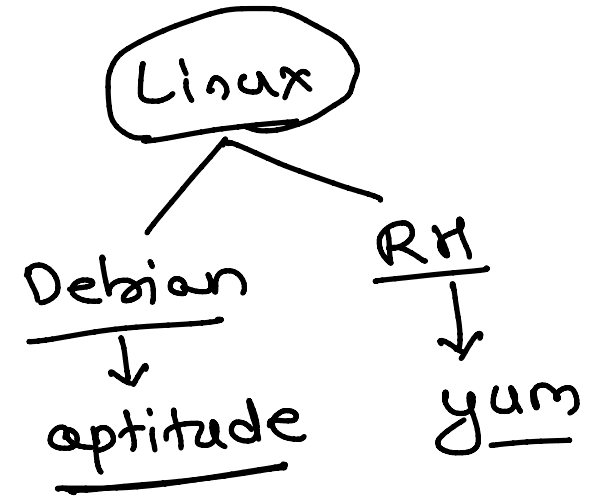
# Installation

- On Ubuntu (Debian)

sudo apt-get install git

on RH based

sudo yum install git



# First time setup

> git config --global --list

> git config --global user.name <user name>

> git config --global user.email <user email>

> git config --global core.editor <editor>

> git config --global merge.tool vimdiff

# Basic Commands

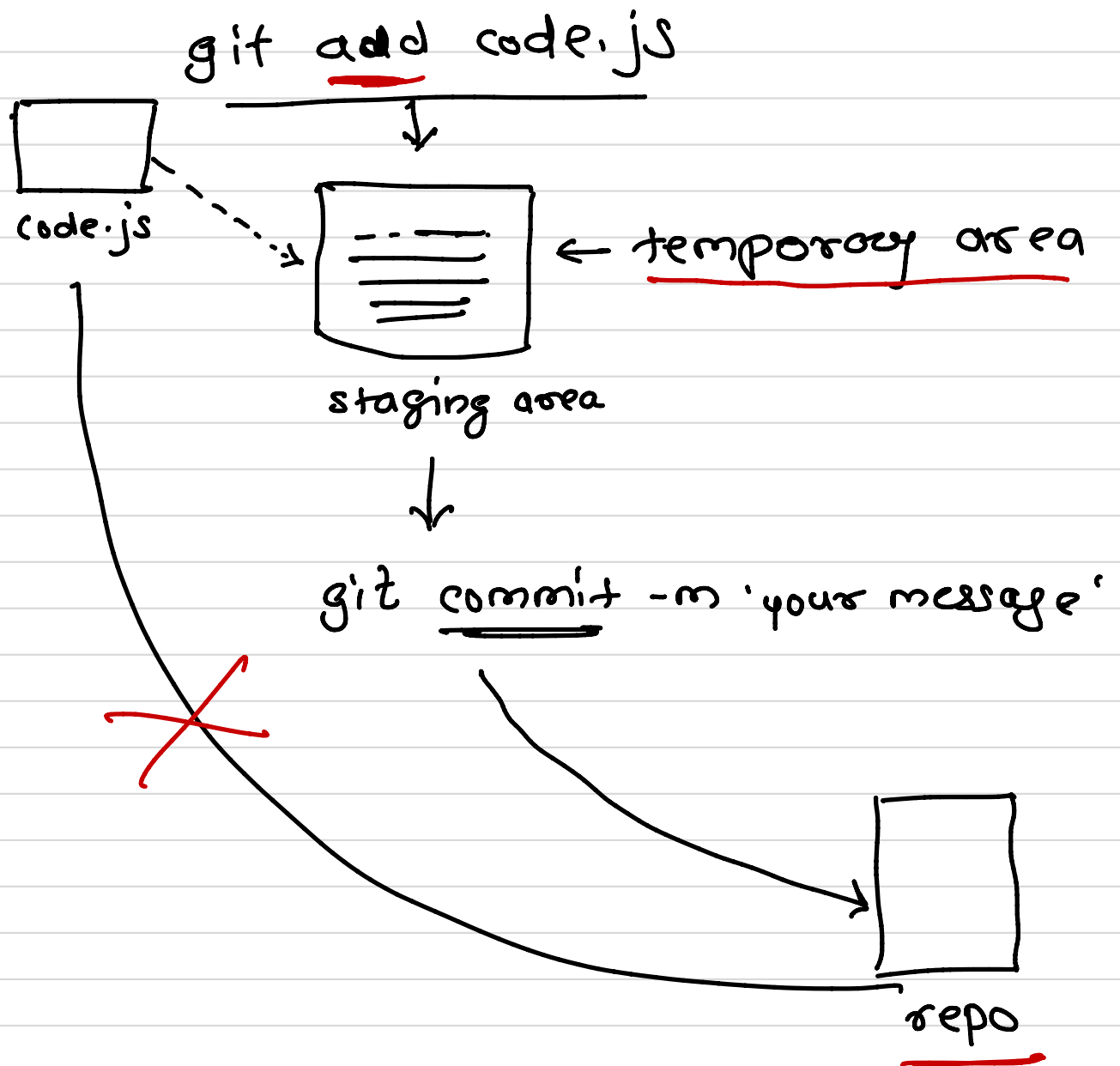
- Initialize a repository  
> git init
- Checking status  
> git status
- Adding files to commit  
> git add .
- Committing the changes  
> git commit -m '<log>'

git status - S

→ ?? → untracked  
(yet to be added to the repo)

- A → will be added

m - modified



# Basic Commands

- Checking logs

> git log

- Checking difference

> git diff

- Moving item

> git mv <source> <destination>



# Basic Commands

- Rename item

> git mv <old> <new>

- Delete Item

> git rm <item>

- Remove unwanted changes

> git checkout file