CS: Pod of Delight

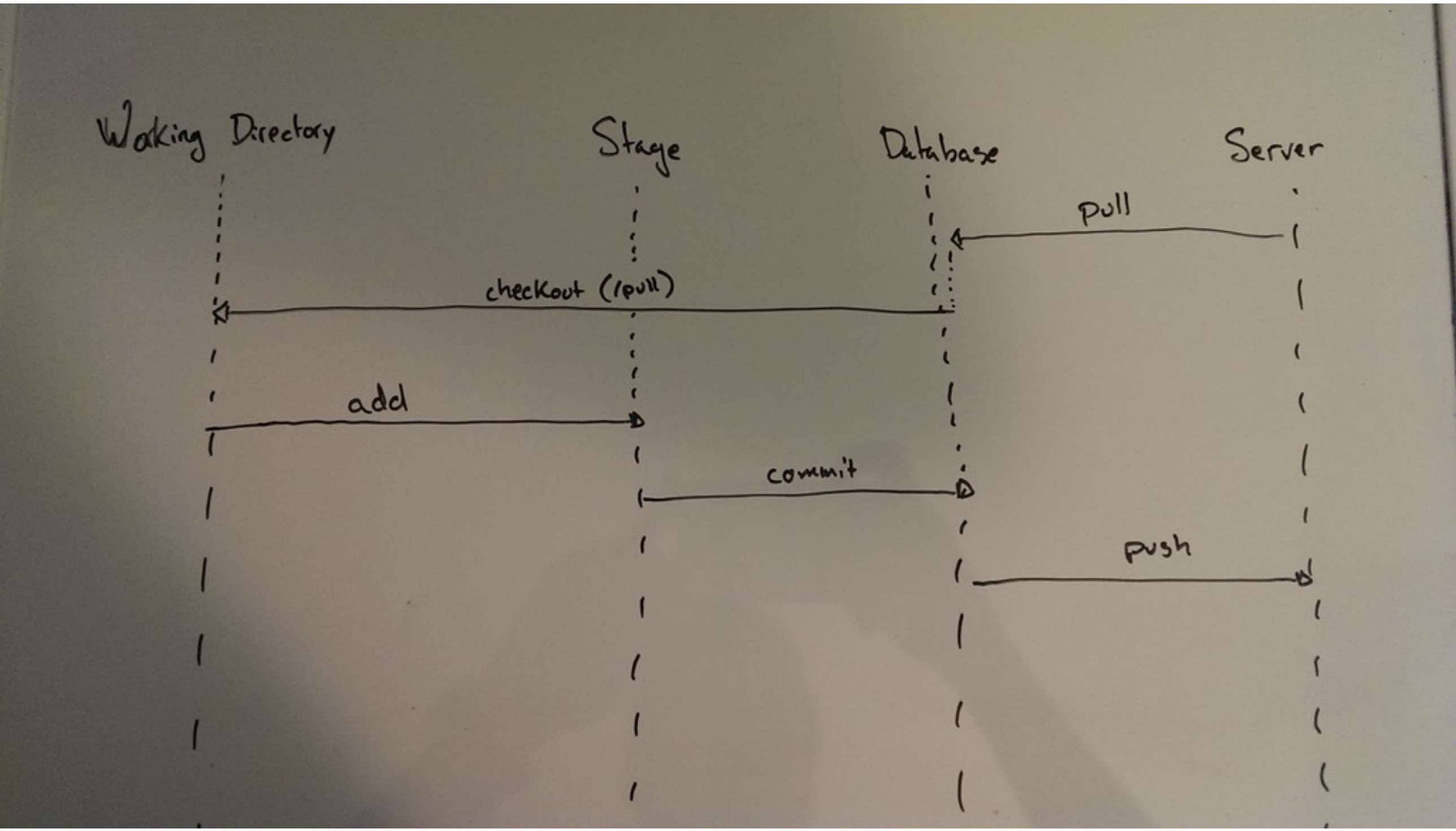
Week 11: Git

Git

What is Git?

- Distributed version control tool
- Keep track of changes
- Synchronize changes across machines/server
- Allows for collaboration
- Git is not github

Basic Git


Working Directory
Stage
Database
Server
pull
checkout (pull)
add
commit
push

Basic Git Commands

- add
  - stage a file/changes (or start tracking if wasn't tracked)
  - git add test.txt
- commit
  - store (some or all) staged changes in the .git database
  - commits track: datetime, author, files changed, changes (diff), and a commit message
  - git commit test.tx -m "commit a single change"
  - git commit -am "commit all changes"

Basic Git Commands

- push
    - Push all your commits to a remote location
- pull
    - Pull commits from a remote location

Basic Git Commands

- status
  - Tell you the status of stage, working dir, server, files changed, etc…
- diff
  - Use to see the changes between any two commits
  - git diff
  - git diff file.txt

Basic Git Commands

- log
    - See the history of all commits
- checkout
    - switch to a different snapshot

Branches

Basic Branches

- “alternate realities”
- Allow you independently work on different things
- You brach out from a common point, changes aren’t propagated
- Used for collaboration, working on features, backup, trying things
- Easy to create, switch, and delete

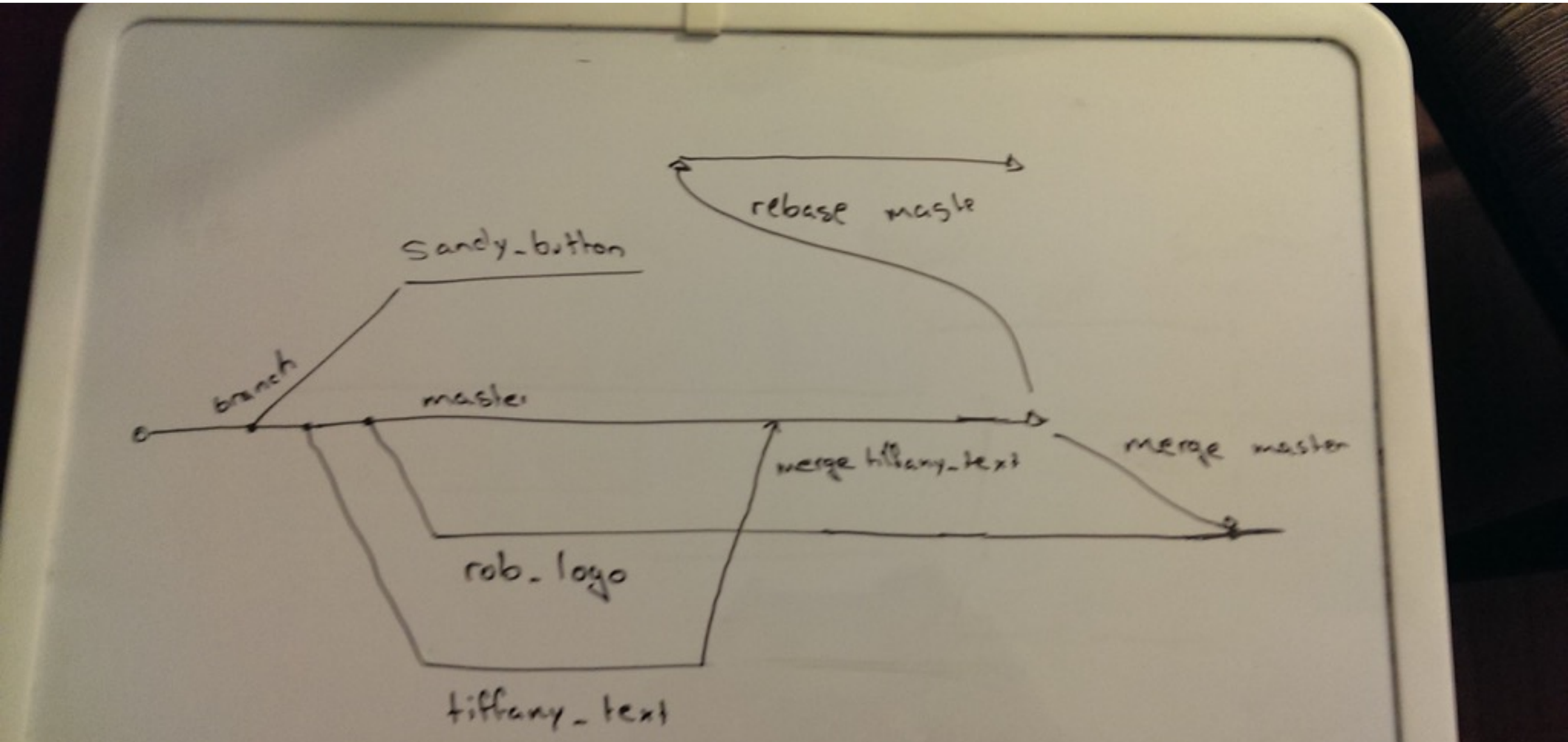
Branching

- Can branch and checkout branch with
  - git checkout -b mybranch
- Or independently create branch
  - git branch my branch
- Can list all branches
  - git branch

Merging

- Merging is taking all changes in one branch and applying them to another
- git merge branch
    - Will merge all (disjoint) changes from branch to your current branch
    - Will create a merge commit, so histories are left untouched
- Can also use rebase, which will “replay” all commits on your local branch without creating a merge commit
    - This “rewrites” history by moving the branching point to the head of branch you are rebasing from
    - git rebase branch

Branching Illustrated



Git Ideology: Branches

- Master: pristine, always works, ie: production
- Dev: practice for master, should still attempt to work, but no heads roll if it doesn't
- Feature branches: each developer has a branch for each feature

Git Ideology: Workflow

- Developer Sandy wants to add a button
- git checkout sandy@mystartup:~codebase // get the codebase
- git checkout dev // work off the development branch
- git checkout -b sandy_button // make her own branch
- <work on code, add the button>
- git checkout dev // go back to dev branch
- git pull // pull any changes from the server
- git checkout sandy_button // go back to her branch
- git merge dev (or git rebase dev) //brings her feature branch back up to date
- At this point she can submit it to QA, merge into dev, etc…

Git Remotes

- Remote git databases
- Allow multiple people to work on codebase
- Github, gitlab, bitbucket, are all common examples
- But you only need a filesystem!
  - VPN, RPI, cs machines, dropbox, gdrive

Starting a Git repo

- Starting from an existing
    - git clone
- Starting a new one
    - git init

Contributing to OSS

- “Fork” their repo (not actually git feature)
- Clone it
- Make your changes
- Create a pull request
    - Essentially asking the original owner to merge your branch
    - Code review, comments, back and forth
- Pull request accepted, your code gets merged back upstream

Some other pros

- Everyone has their own copy, no stepping on each other's shoes
- Open source
- Works without internet connection, everything is local
- Simple to use

Cons of Git

- Decentralized means no easy way to keep track of progress unless pushed
- Requires everyone to have a full copy of codebase
- If things go bad, they can go real bad