git

An introduction (Solo)

Ali Farnudi, November 2022

Introduction

The problem

- my_code_1.py
- my_code_2.py
- my_code_2_1.py
- my_code_2_2.py
- my_code_2_2_new_function.py
- my_code_2_3_rewrote.py
- my_code_3_with_Alis_suggestion.py
- my_code_4.py
- my_code_5_buggy.py
- my_code_6_bug_fixed.py

I want to try something in the version before the bug was introduced

The problem

- my_code_1.py
- my_code_2.py
- my_code_2_1.py
- my_code_2_2.py
- my_code_2_2_new_function.py
- my_code_2_3_rewrote.py
- my_code_3_with_Alis_suggestion.py
- my_code_4.py
- my_code_5_buggy.py
- my_code_6_bug_fixed.py

+

=

Loads of emails
confusion

Bug reports + fixes

Confusion

Anger

I want to try something in the version before the bug was introduced

The problem

- Report_1.tex
- Report_2.tex
- Report_3.tex
- Report_3b.tex
- Report_4.tex
- Report_5.tex
- Report_6_showed_supervisor.tex
- Report_7_implamented_comments.tex
- Report_8_semi_final.tex
- Report_9_final.tex
- Report_10_final_2.tex
- Report_11_final_final.tex
- ....

+

=

Loads of emails
confusion

Bug reports + fixes

Confusion

Anger

I want to try something in the version before the bug was introduced

The solution:
Version Control System (VCS)

git

I ♥ Free Stuff

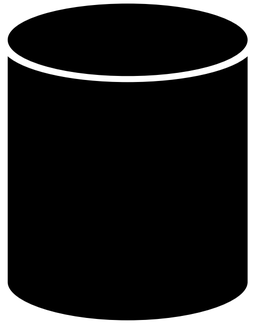
open source

Fast!!

How git works

Records snapshots

Store



Repository

Local/remote

```
def getColourIndex(r):
    if r==100:
        return 0
    elif r == 1000:
        return 1
    elif r == 10_000:
        return 2
    else:
        return 3

def getPlotStat(tempStat, Teff,tempThreshold):
    if tempStat=='all':
        plotStat=True
    elif tempStat=='high':
        if Teff>tempThreshold:
            plotStat=True
        else:
            plotStat=False
    elif tempStat=='low':
        if Teff<tempThreshold:
            plotStat=True
        else:
            plotStat=False
    return plotStat
def main():

    df = loadData()
    df = df.sort_values(by=['R'])

    from matplotlib.pyplot import cm
    color = cm.rainbow(np.linspace(0,1,len(df.index)))
    radiuslist = [500,1000,5000,10000,50000,100000]
    color = cm.rainbow(np.linspace(0,1,len(radiuslist)))
    alpha= 1
    for c in color:
```

History

What?

When?

Why?

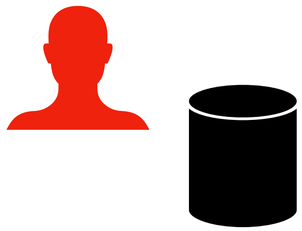
Revert back to any point in history

```
    if tempStat=='all':
        plotStat=True
    elif tempStat=='high':
        if Teff>tempThreshold:
            plotStat=True
        else:
            plotStat=False
    elif tempStat=='low':
        if Teff<tempThreshold:
            plotStat=True
        else:
            plotStat=False
    return plotStat
def main():

    df = loadData()
    df = df.sort_values(by=['R'])

    from matplotlib.pyplot import cm
    color = cm.rainbow(np.linspace(0,1,
```

git

Solo

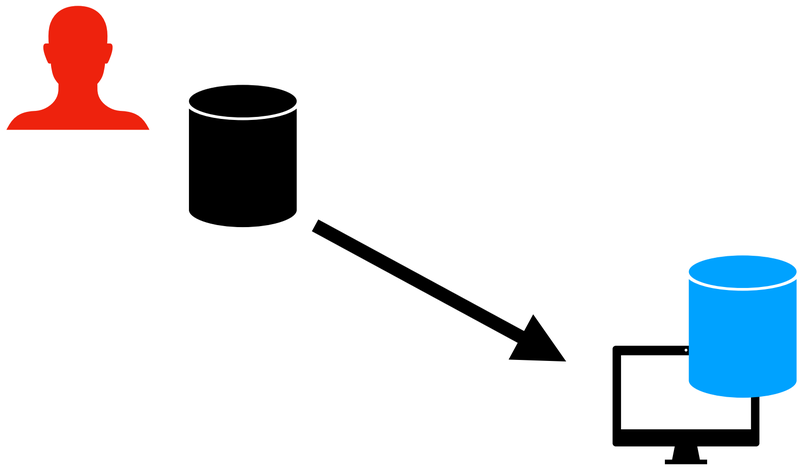
- Code in a repository
- Track all past versions + rollback
- Compare past versions
- Branch development



git

Solo

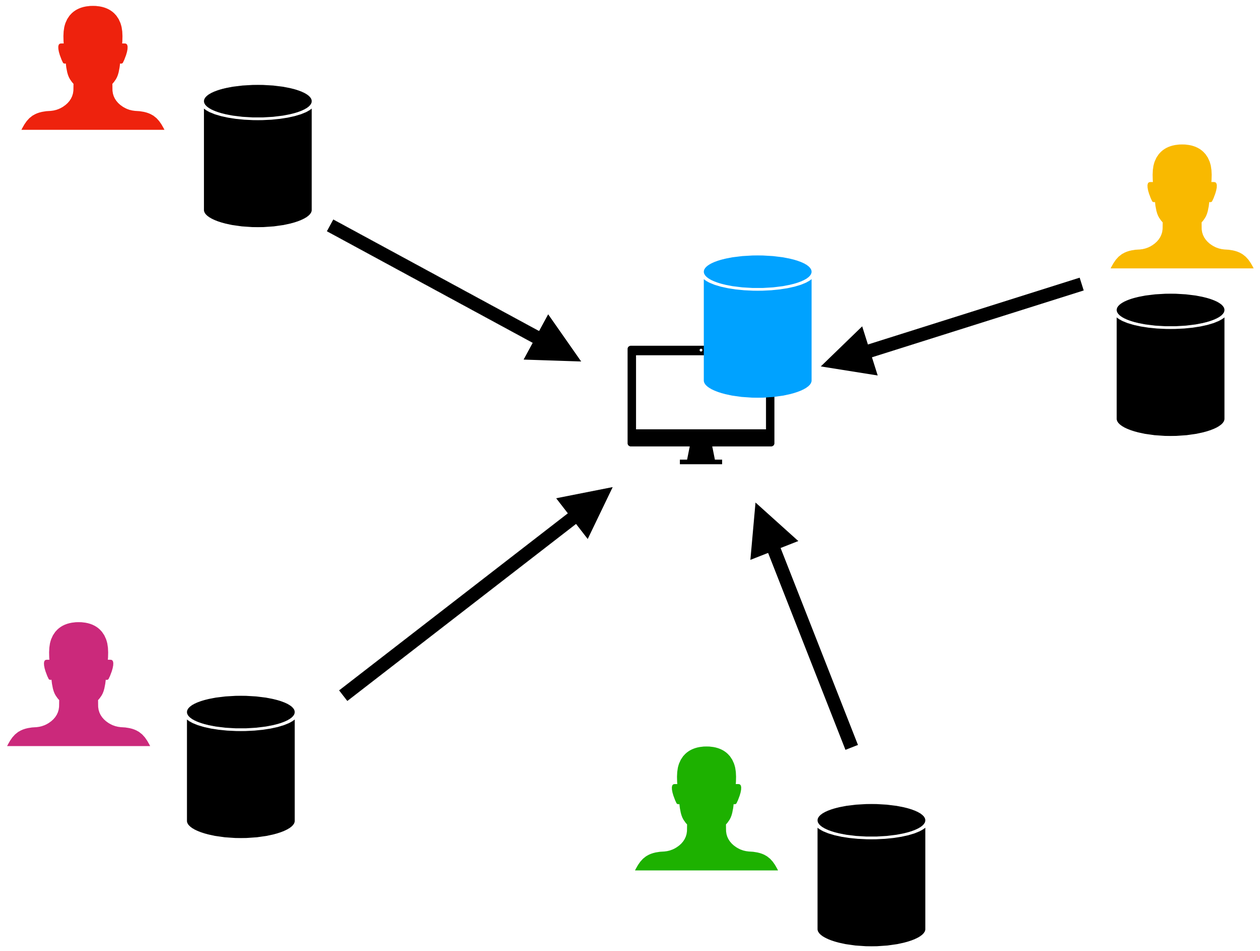
- Code in a repository
- Track all past versions + rollback
- Compare past versions
- Branch development



git

Solo

- Code in a repository
- Track all past versions + rollback
- Compare past versions
- Branch development



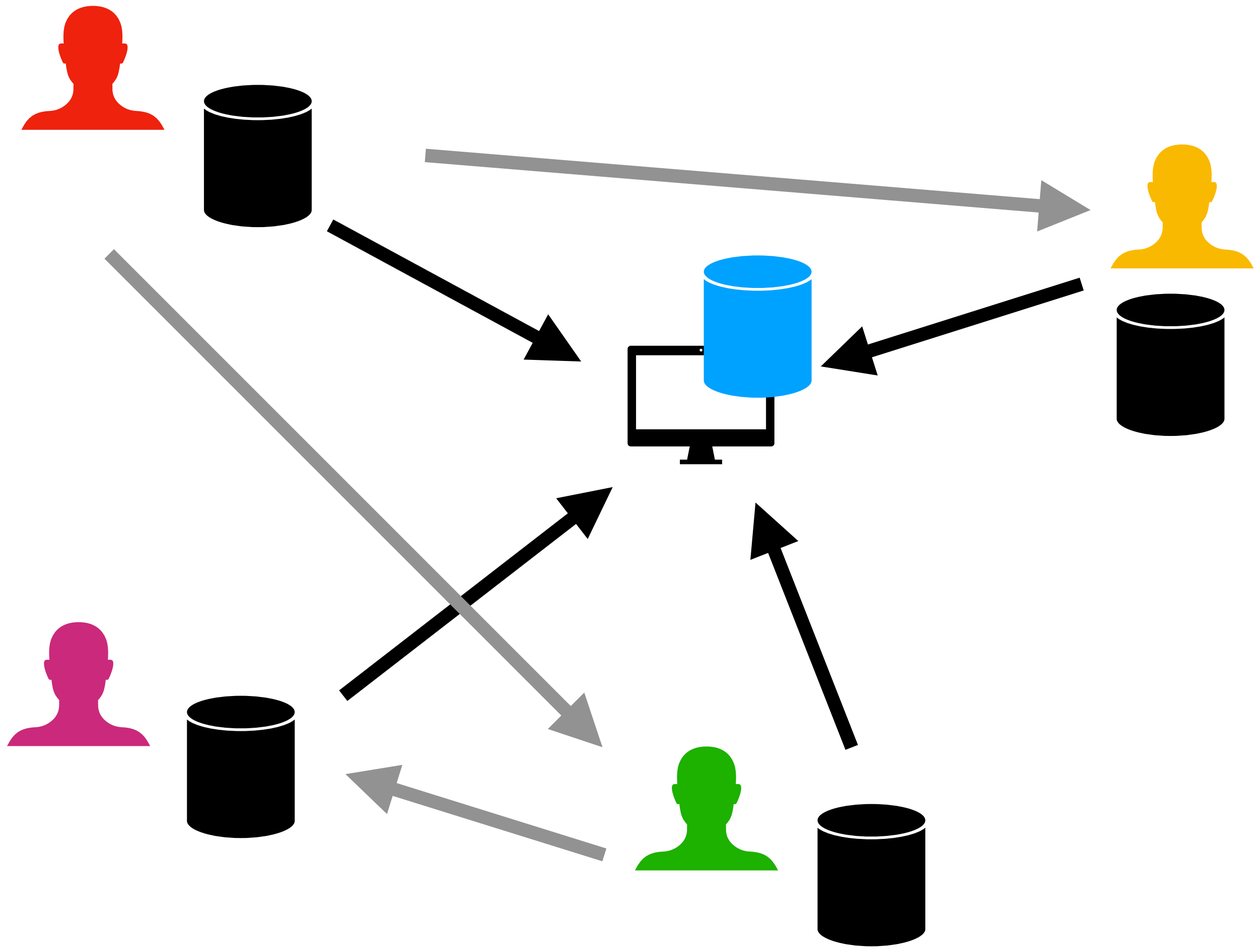
Collaboration

- Common remote repository
- Merge contributions of different developers
- See **who**, **when** wrote **what**, and **why**
- Remote server optional.

git

Solo

- Code in a repository
- Track all past versions + rollback
- Compare past versions
- Branch development



Collaboration

- Common remote repository
- Merge contributions of different developers
- See **who**, **when** wrote **what**, and **why**
- Remote server optional.

Install Git

Git website
https://git-scm.com

Installation:

- Mac
- Linux
- Windows
  - Git BASH

How to use

- **Command line**
- IDEs + code editors
  - Xcode (MacOS)
  - Visual Studio (MS Windows)
  - Code blocks (Linux)
- GUIs
  - Tower (free for students)
  - Git kraken
  - Sourcetree (Mac and Windows)
  - More on the git website

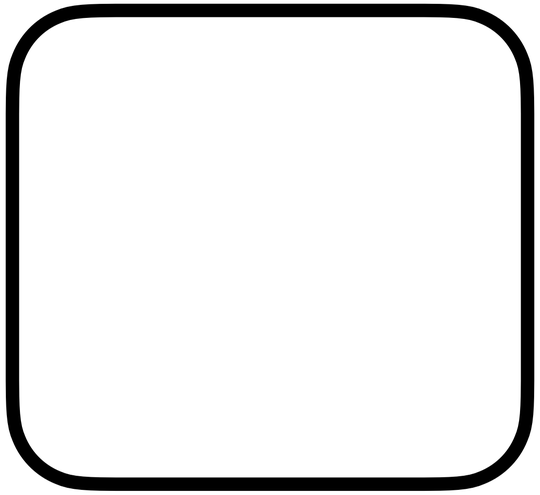
Configuring and initiating

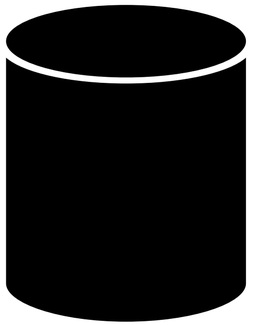
Hands-on config

- Configure your git settings:
  - $ git config --global user.name "[name]"
  - $ git config --global user.email "[email address]"
  - $ git config --global color.ui auto
- More options:
  - $ git config --global core.editor “editor name”
  - $ git config --global -e
  - $ git config -h
  - $ git config --help

Initiated directory

Staging area (index)

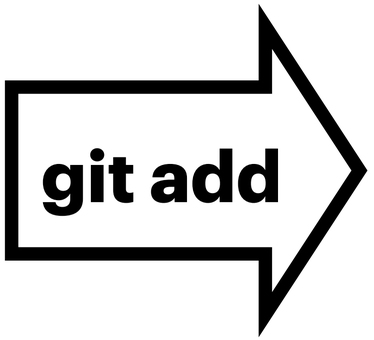




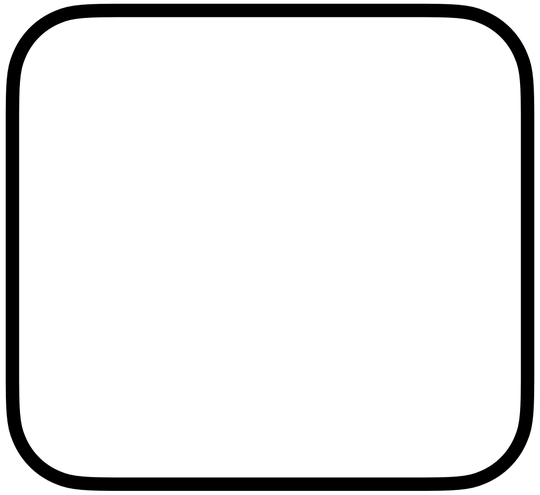
Change to file 1
Change to file 2
Change to file 3
Delete file 4
Rename file 5

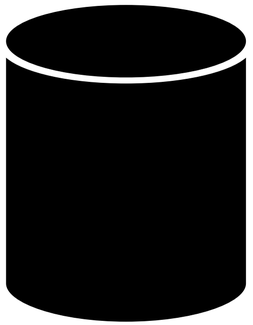
Initiated directory

Change to file 1
Change to file 2
Change to file 3
Delete file 4
Rename file 5



Staging area (index)





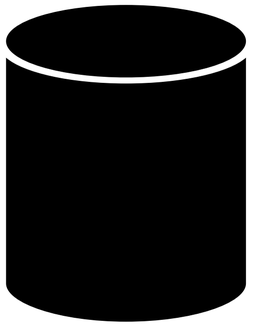
Initiated directory

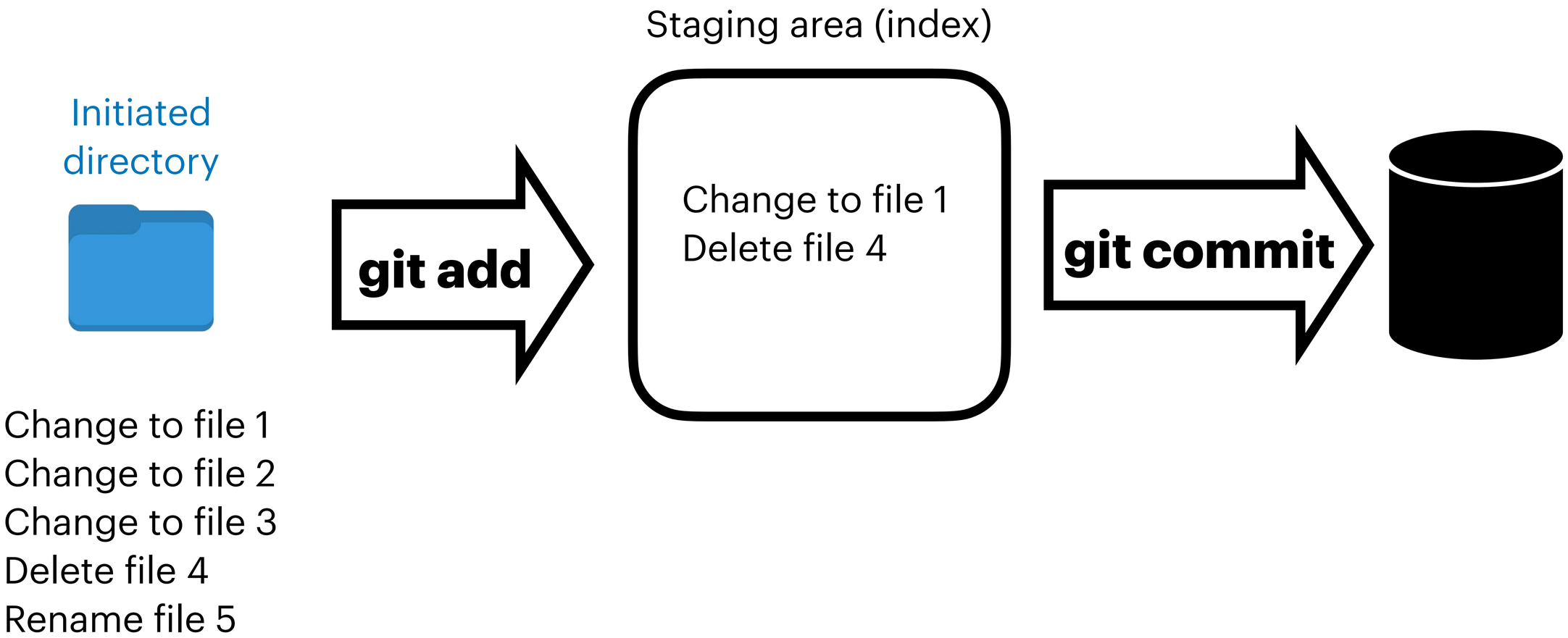
Change to file 1
Change to file 2
Change to file 3
Delete file 4
Rename file 5

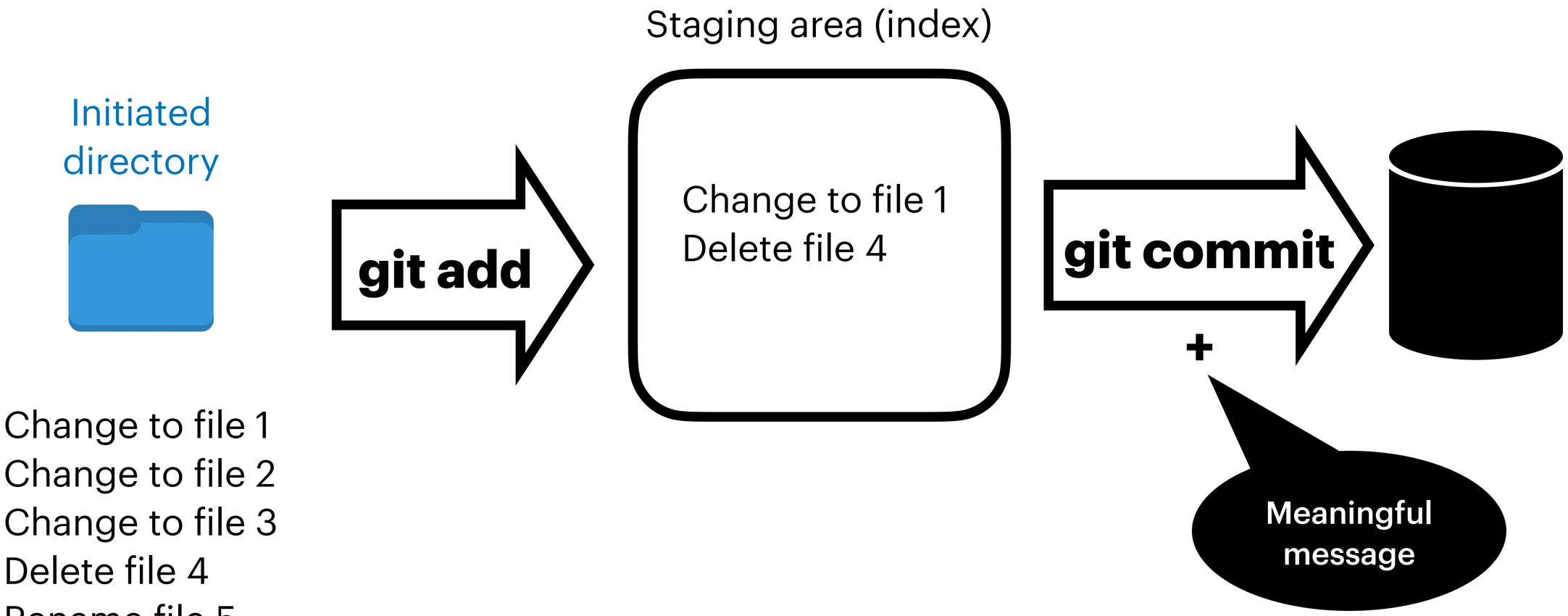
git add

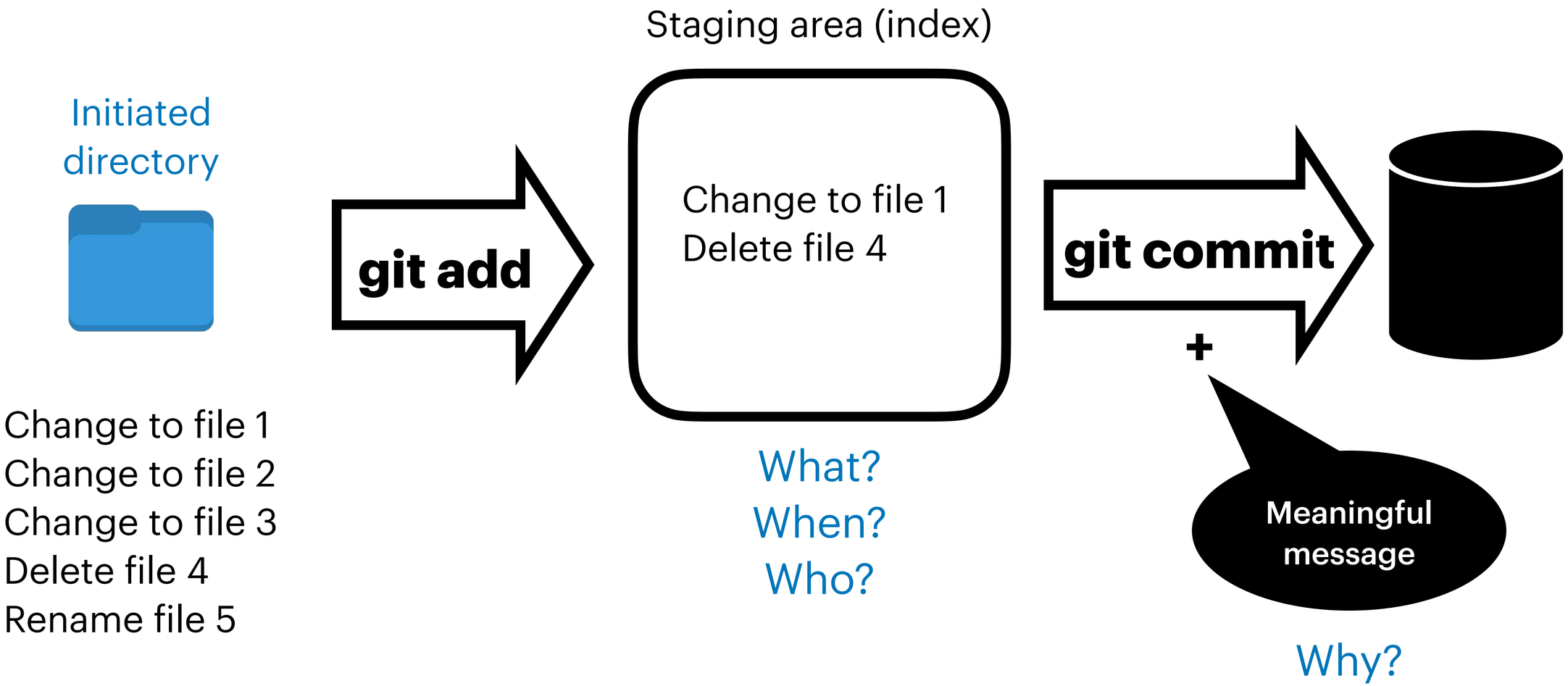
Staging area (index)

Change to file 1
Delete file 4




Initiated directory
Change to file 1
Change to file 2
Change to file 3
Delete file 4
Rename file 5
git add
Staging area (index)
Change to file 1
Delete file 4
git commit


Initiated directory
Change to file 1
Change to file 2
Change to file 3
Delete file 4
Rename file 5
git add
Staging area (index)
Change to file 1
Delete file 4
git commit
+
Meaningful message


Initiated directory
Change to file 1
Change to file 2
Change to file 3
Delete file 4
Rename file 5
git add
Staging area (index)
Change to file 1
Delete file 4
What?
When?
Who?
git commit
+
Meaningful message
Why?

Hands-on init

- Initiate git in a directory:
  - $ git init
- Make some files:
  - $ git status
  - $ git add
- Committing to changes:
  - $ git commit

Hands-on commit

- Notes: git commit
  - Message size
  - Title and and details
  - Don’t cram multiple tasks into one commit: typo, bugfix, new function

Commit message example 1:

Fixed potential bug. Could have resulted in division by zero.

If user inputs included negative numbers, the sum of inputs may have resulted in zero. Added the “input_parser” function that prevents negative inputs.

Commit message example 2:

Renamed “input_parser” to “filter_negative_numbers”

Diff and log

Hands-on status diff

- Try:
  - $ git status -s
- Add a file to the staging environment
  - $ git diff
- Try:
    - $ git diff -h
    - $ git diff --stat

Hands-on log

- Looking at the changes
  - $ git log
  - $ git log -3
  - $ git log -p
  - $ git log --stat --summary
  - $ git log --follow [file]
  - $ git log --oneline
  - $ git log --after 2017-07-04
  - $ git log —author=“ali”
  - $ git log —grep=" word of phrase to search”
- $ git show 1b2e1d63ff (some identifier)
- $ git show HEAD
- $ git show HEAD~1
- $ git show HEAD~2:file1.txt

Removing and renaming files

Hands-on rm

- Delete files
  - $ git status
  - $ git ls-files
  - $ rm file2.txt
  - $ git add file2.txt

} $ git rm file2.txt

Hands-on mv

- Rename files
  - $ mv file3.txt main.cpp
  - $ git add file1.txt
  - $ git add main.cpp

} $ git mv file3.txt main.cpp

Restore/recover

Hands-on restore

- Made changes to file1.txt
  - $ git restore file1.txt
- Added changes to file1.txt to the staging area
  - $ git restore --staged file1.txt
  - $ git restore file1.txt
- Restoring deleted files
  - $git rm file1.cpp; $ git commit
  - $ git restore --source=HEAD~1 file1.cpp
  - [more options] $ git restore -h

ignore

Hands-on ignore

What if you don’t want to track some files?

- Create bin/app.out
- Ignore the files
  - Create a .gitignore file
  - $ git add .gitignore
    - : *.DS_Store *.log *.aux
  - Modify bin/app.out
  - $ git status

Hands-on ignore

Stop tracking files

- Create bin/app2.out
- Add and commit
- Add bin/ to .gitignore
- Modify bin/app2.out
  - $ git status
  - $ git ls-files
  - $ git rm -h
  - $ git rm --cached bin/
  - $ git rm --cached -r bin/

https://github.com/github/gitignore

In case of fire

1. git commit
2. git push
3. leave building