Git workshop II

TYH Lab git workshop II
by Paul Z Cheng

Quick recap

- Study get for reproducibility
- Basic Git workflow

Porcelain command

git add

git commit

git push

git pull

git branch

git checkout

git merge

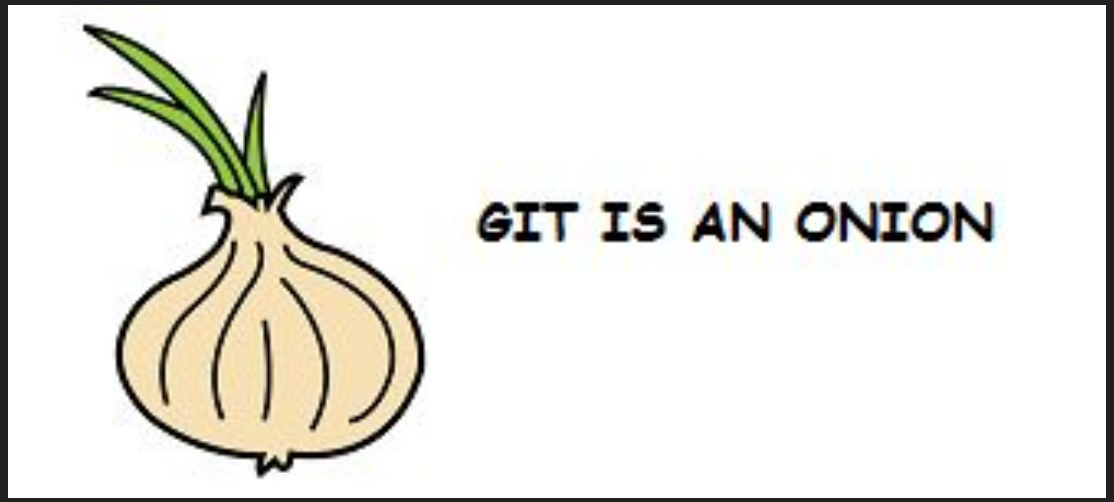
git rebase

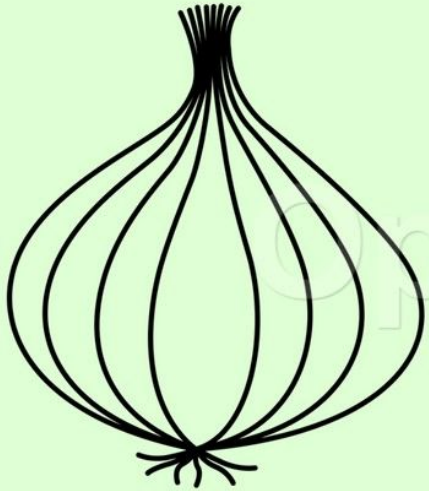
Plumbing command

git cat-file

git hash-object

git count-objects

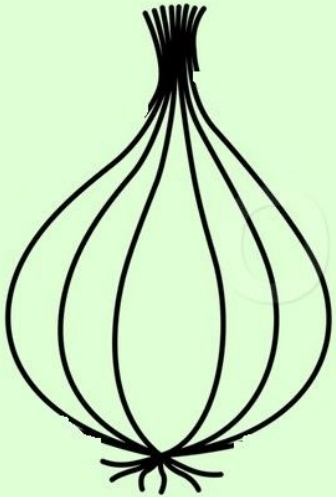

GIT IS AN ONION



Git is a
Distributed Revision Control System

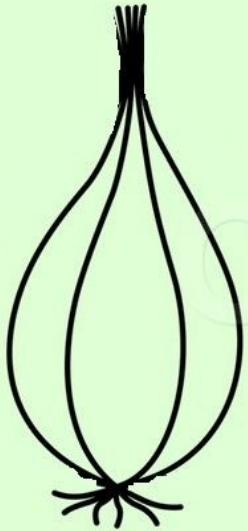
OpenGenus

https://iq.opengenus.org/git-is-an-onion/



Git is a
Revision Control System

OpenGenus



Git is a Simple Content Tracker

OpenGenus

Git is a Persistent Map

OpenGenus

SHA1

Coca-Cola

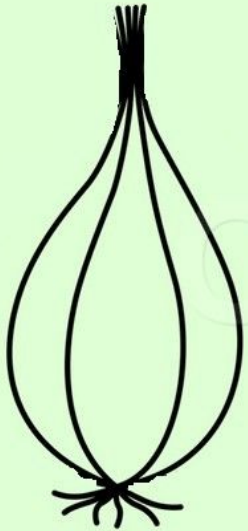
6ee3af491710caada52faf3b89b16d85786249a2

This is how Git: Storing things

```
# Demo how git is a pessistant map
git hash-object "Coca-Cola"
# can't works

# git started
git init
ls -a
echo "Coca-Cola" | git hash-object --stdin -w
# Look into store at 23
# and shia object tell it how to show up.
# this is what git call "blob" , can't just open it
git cat-file "sha-1" -t
# -t == type
# -p == pretty print
```

blob



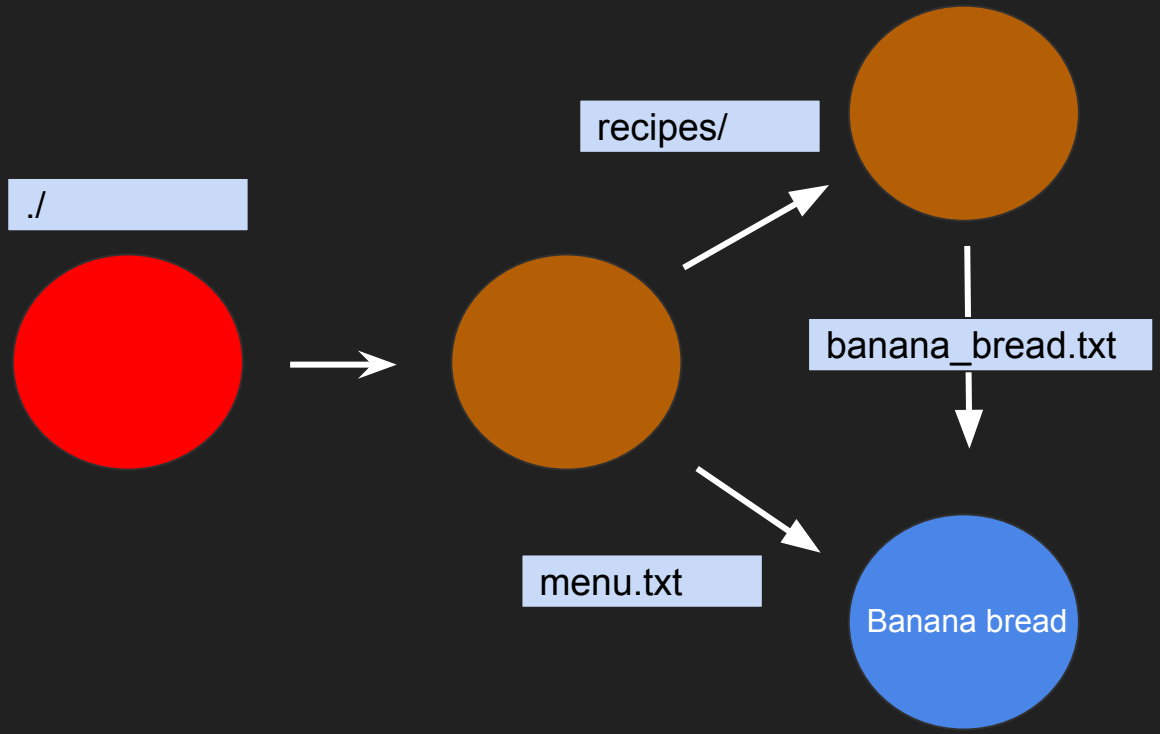
Git is a Simple Content Tracker

OpenGenus

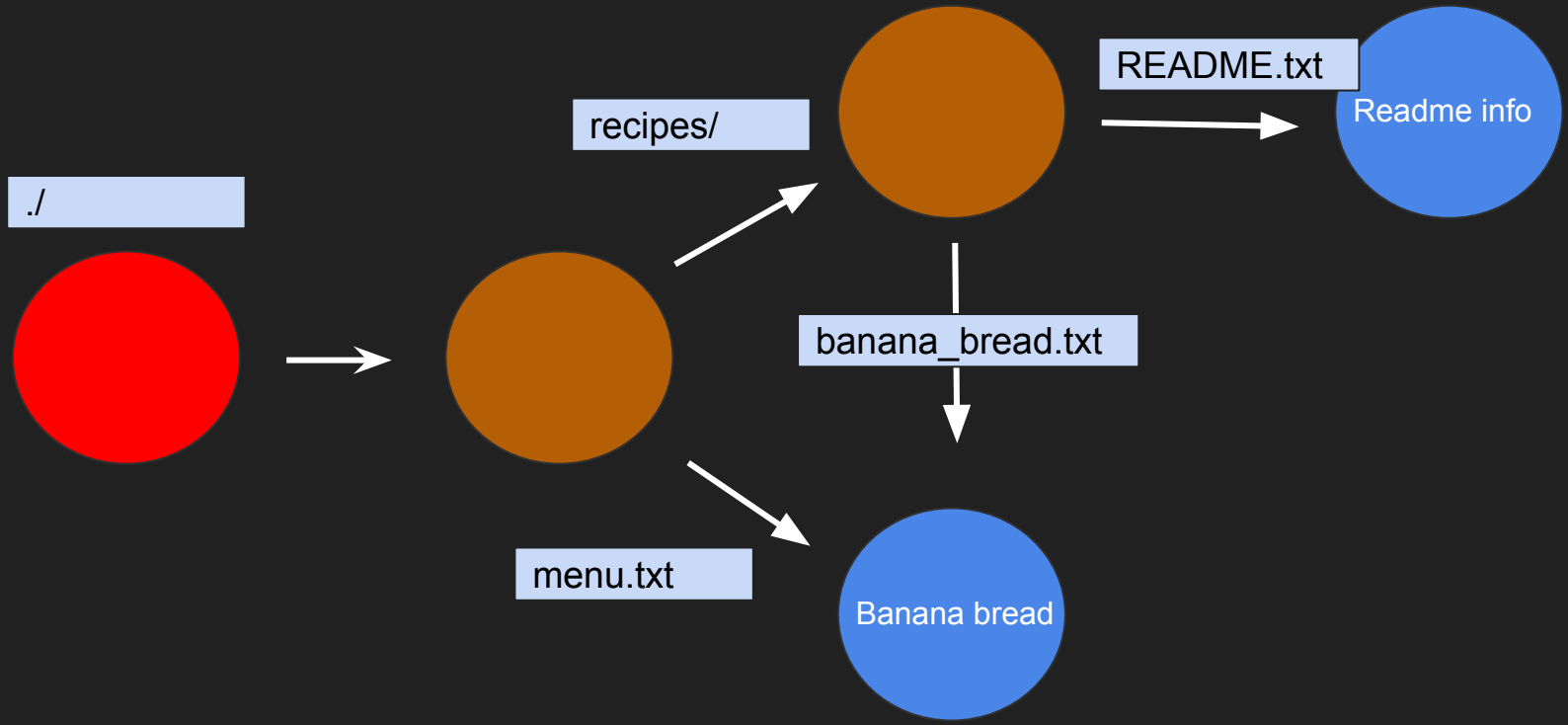
Disssecting .git

- Look into object folder of .git
- Blob of content
- How tree reference blob
- Blob of efficiency

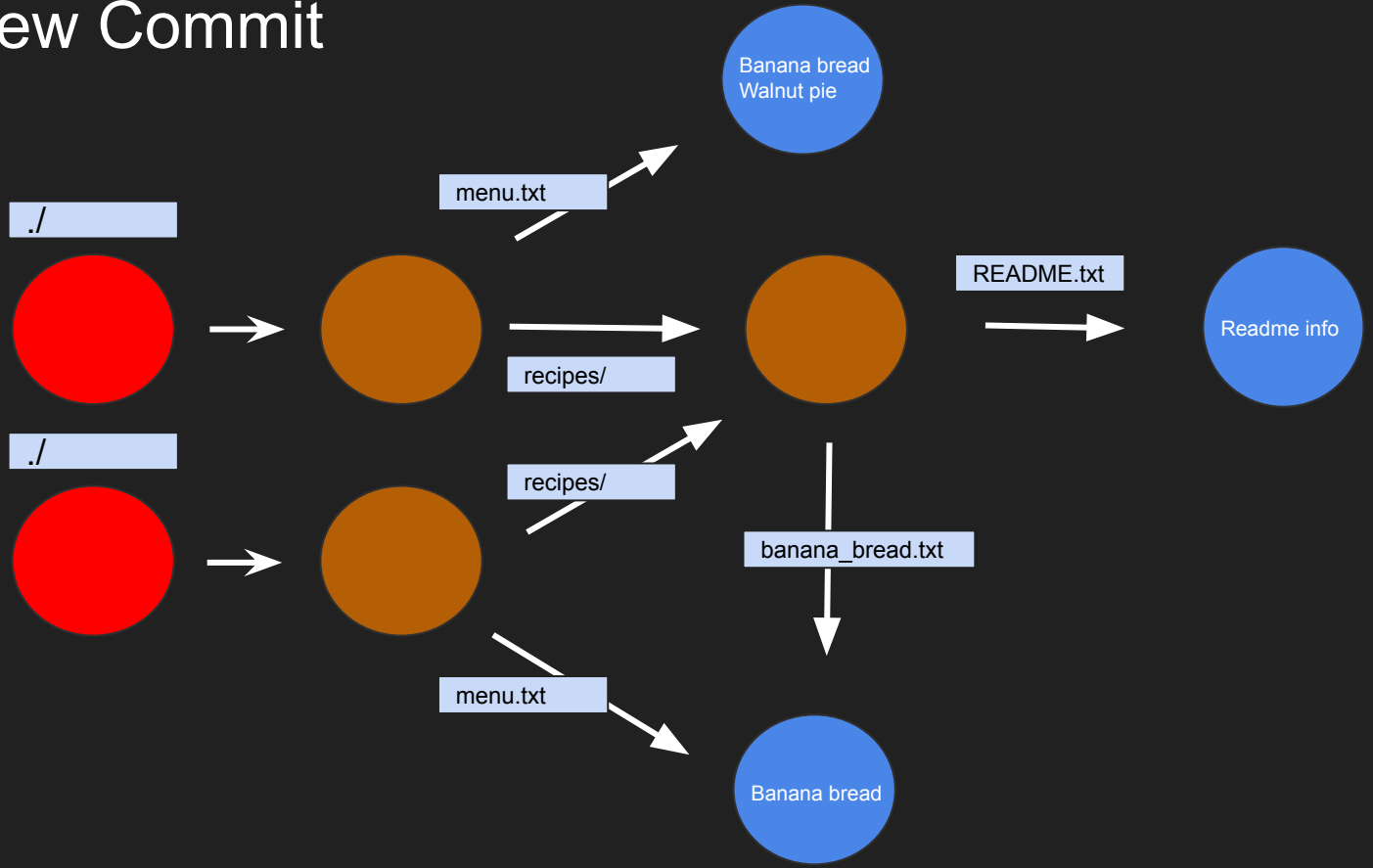
Tree



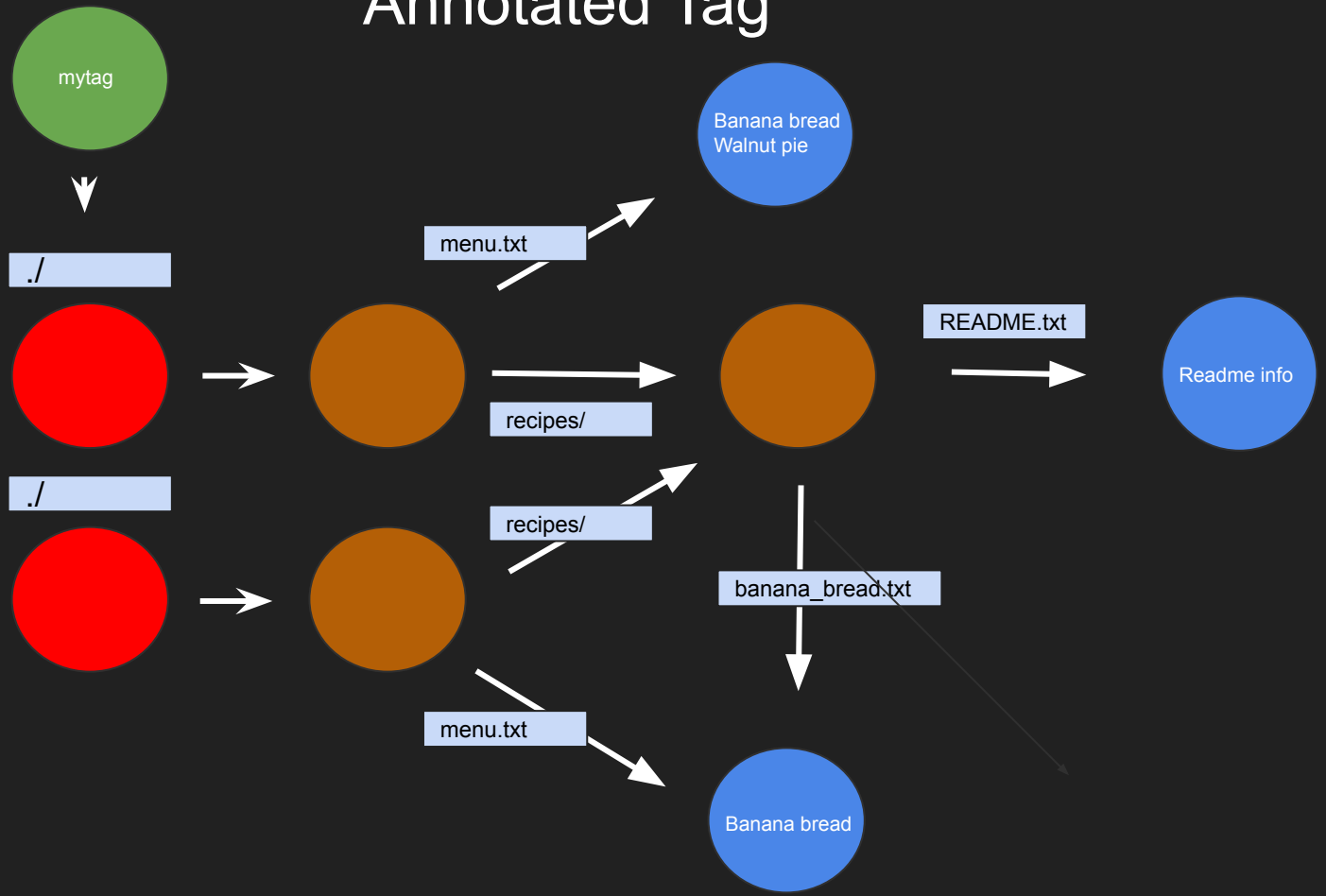
add ReadMe.txt



Add new Commit



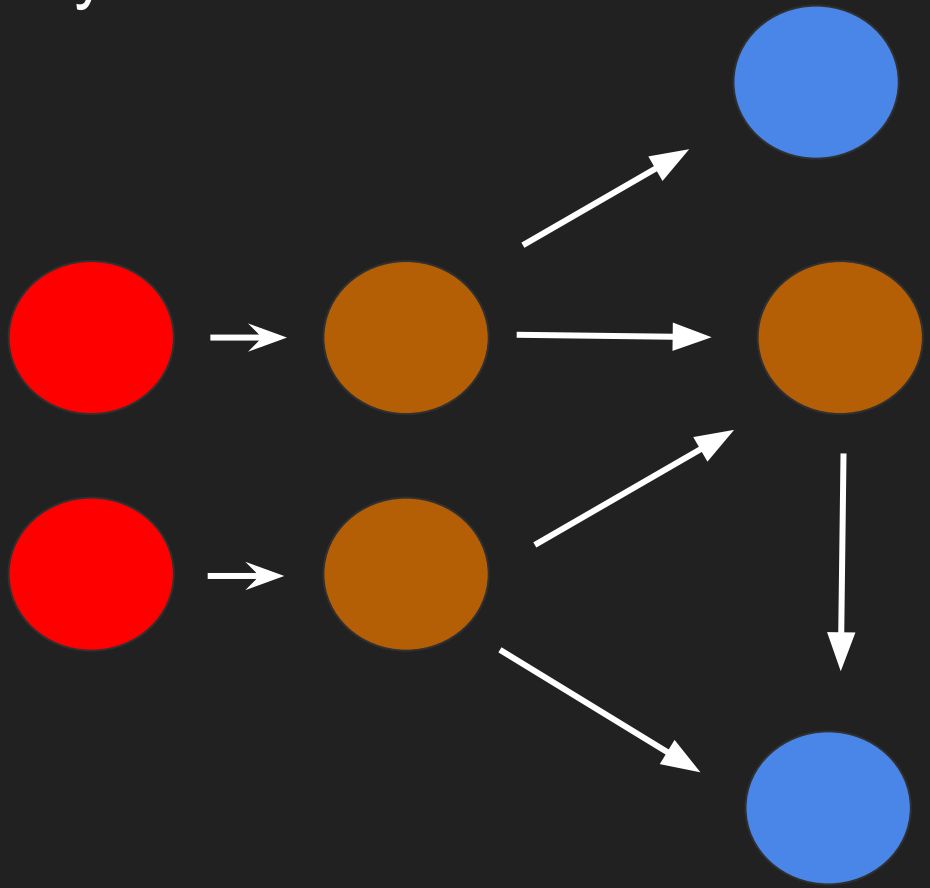
Annotated Tag

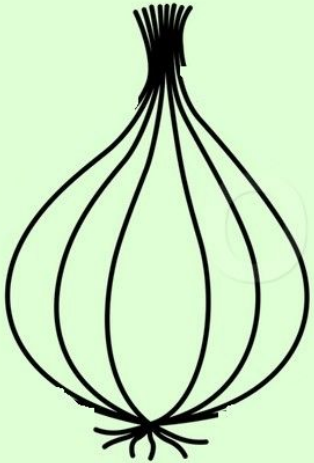


Basic componends of git

1. blob
2. tree
3. Commits
4. Annotated Tags

Git Really is



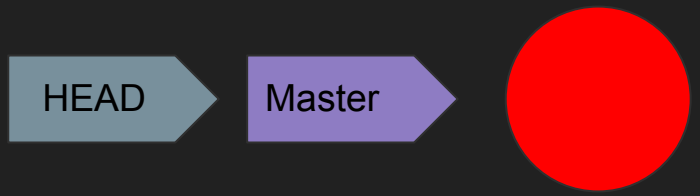


Git is a
Revision Control System

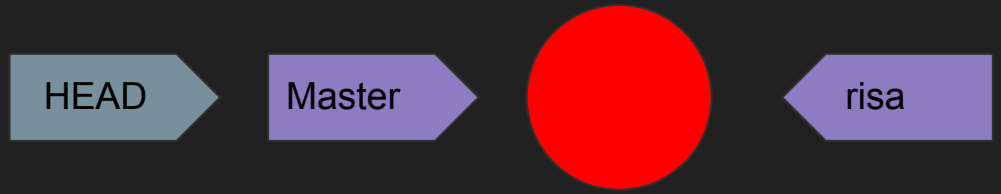
OpenGenus

Git is a version control system

- A branch is just a reference to a commit



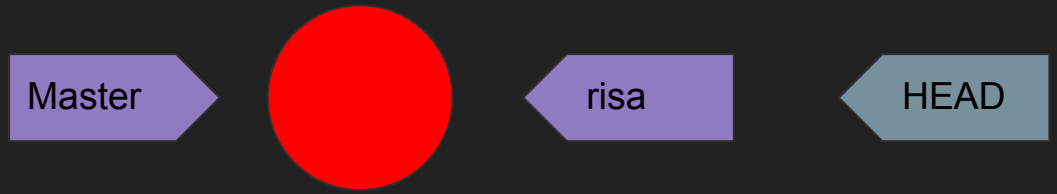
Branching



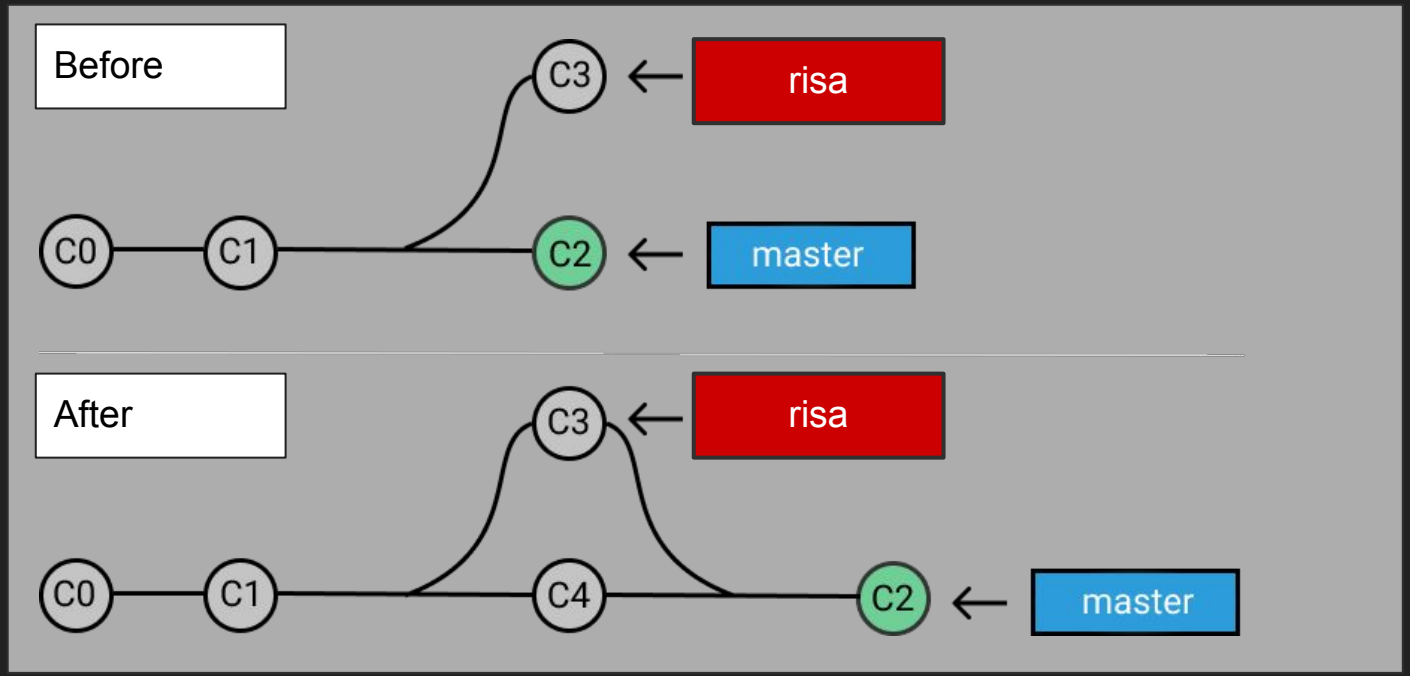
HEAD

Checkout

Moving Head to referencing to a branch or commit

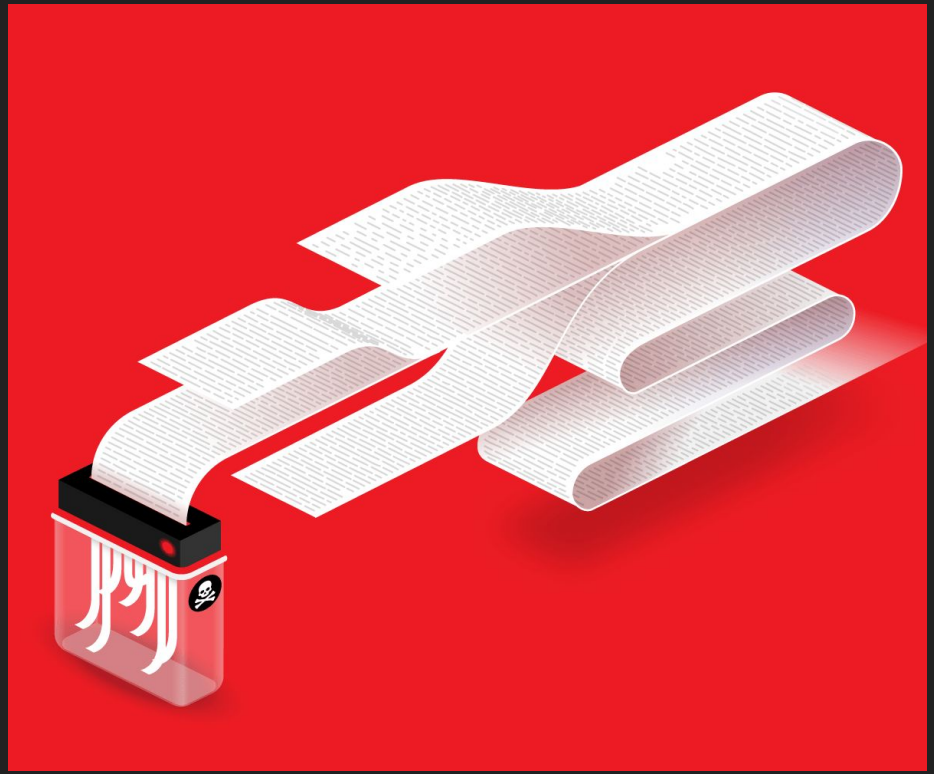


Merging

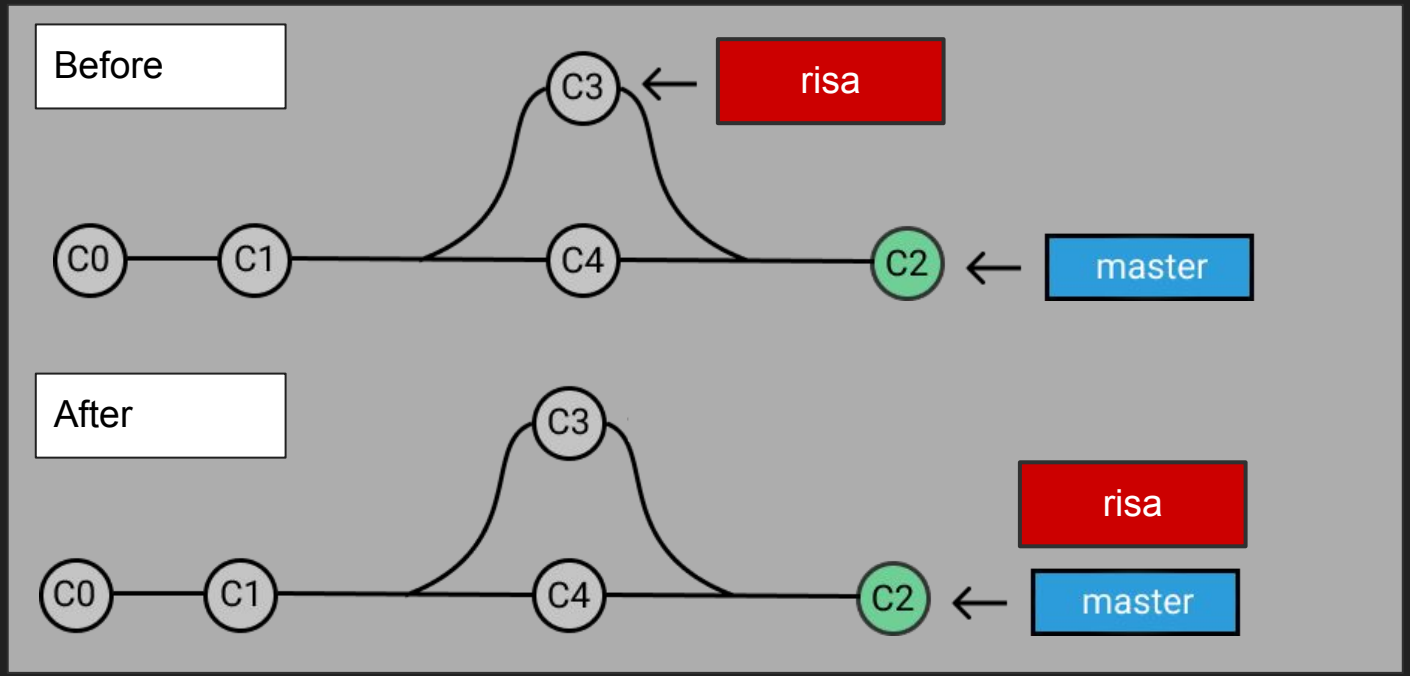


Git Doesn't care about your work directory

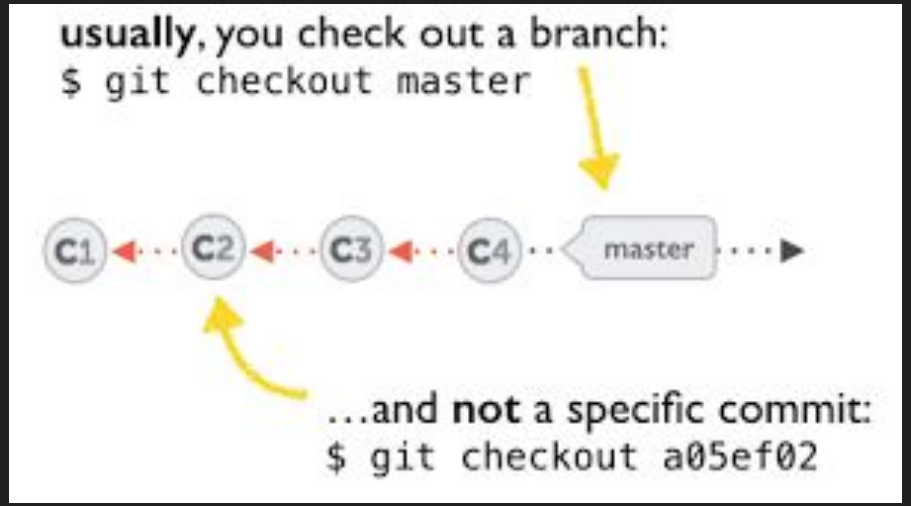
- git will remind but always be careful

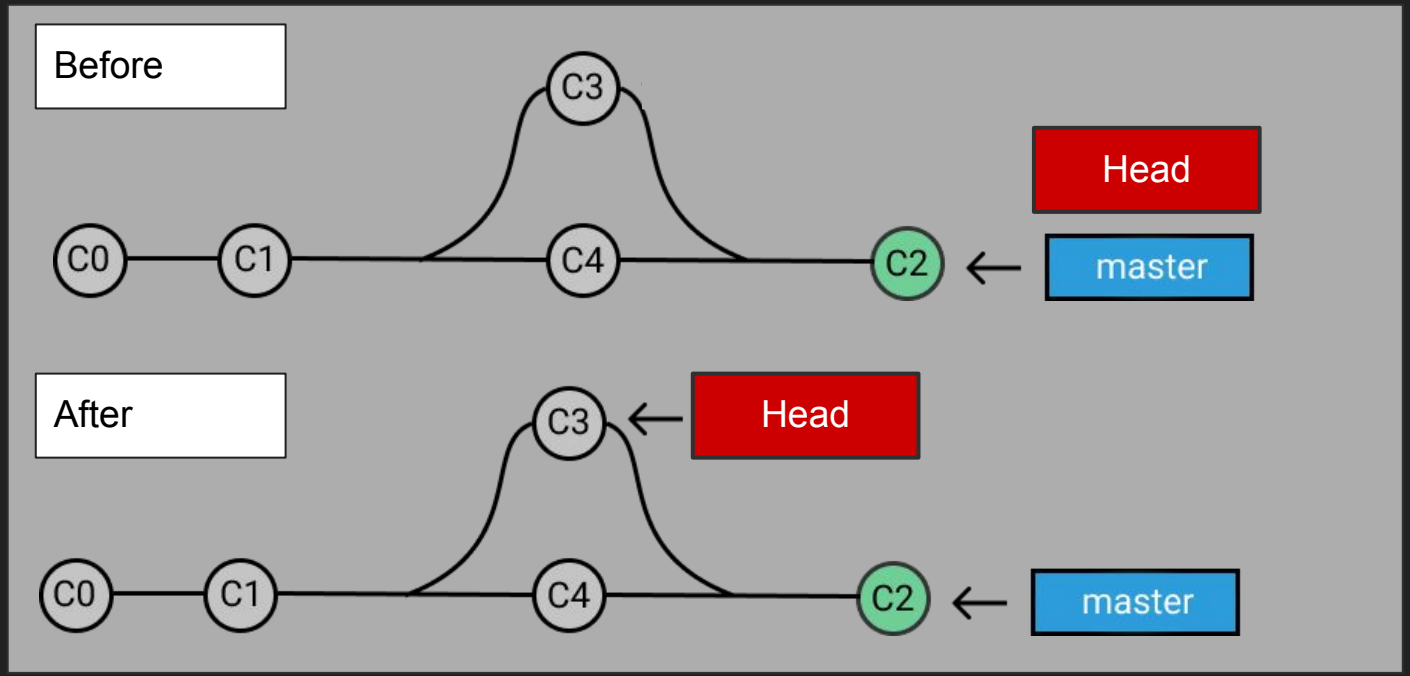


Fast-forward : Efficiency of git



Checkout a commit




Before
C3
Head
C0
C1
C4
C2
master
After
C3
Head
C0
C1
C4
C2
master

Detached Head

At this level

- Don't worry about tree and blob (git got your back)
- As long your history is correct you will be fine

The Git object Model rules

1. The current branch tracks new commits
2. When you move to another commit, Git updates your working directory
3. Unreachable objects are recycle

Break and try

Revert time exercise.

Step 1: setting up your repository with a python "MissNum.py" file.

(1) write a python function that find a missing number in a list

ex:

```
iter =  [0,1,2,4,5,6]
miss = miss_num_func(iter)
print(miss) # 3
```

(2) add and commit the changes

Step 2: Make some changes to "MissNum.py" file.

(1) write a python function that find missing numbers in a list

ex:

```
iter =  [0,1,2,4,5,6,8,9]]
miss = miss_num_func(iter)
for i in miss:
        print(i) # 3 7
```

(2) add and commit the changes after editing MissNum.py

Step 3: Reverting to step one and change file name.

(1) Check out Step 1

(2) Change the file name of MissNum.py to MissNum_2.py or cp.

Step 4: Create a new branch with reverted time step three.

(1) Make a new branch with name: revertime

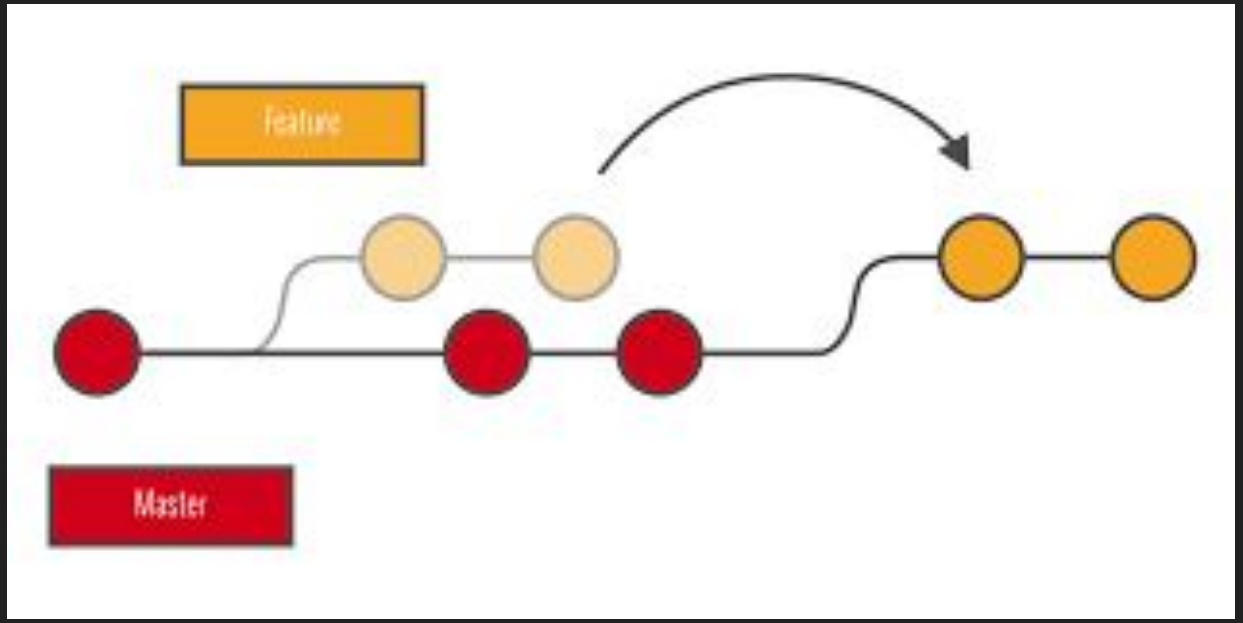
Step 5: Merge branches

(1) Merge revertime with master

Results:

Master with MissNum.py and MissNum_2

Rebasing



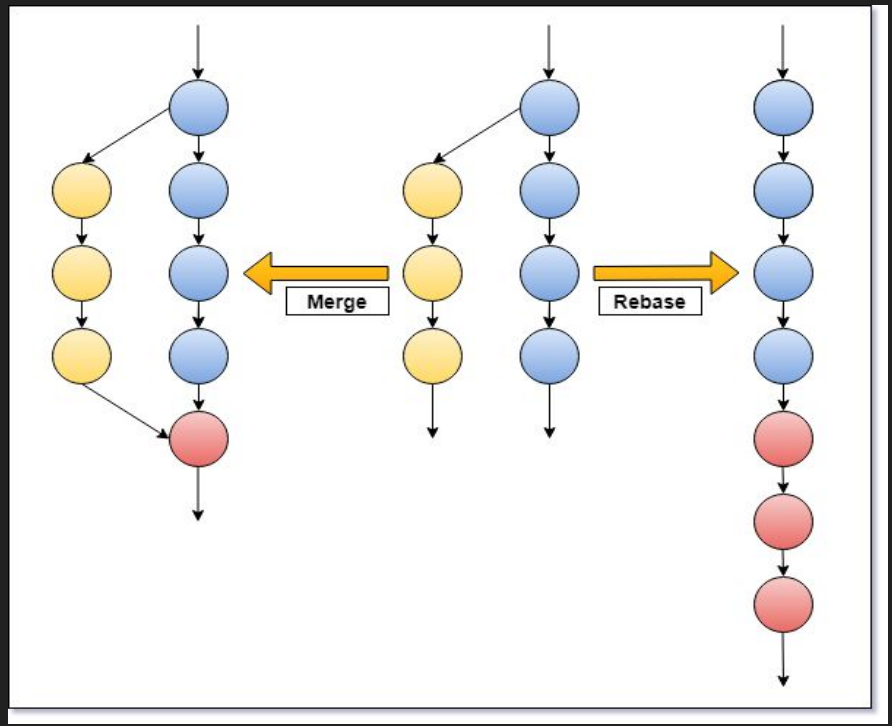
An illusion of movement

- New data with new commit,except the parent..
- Leaving the old commit behind.
- No worries taking up space, it will be recycled

Difference between merge and Rebase

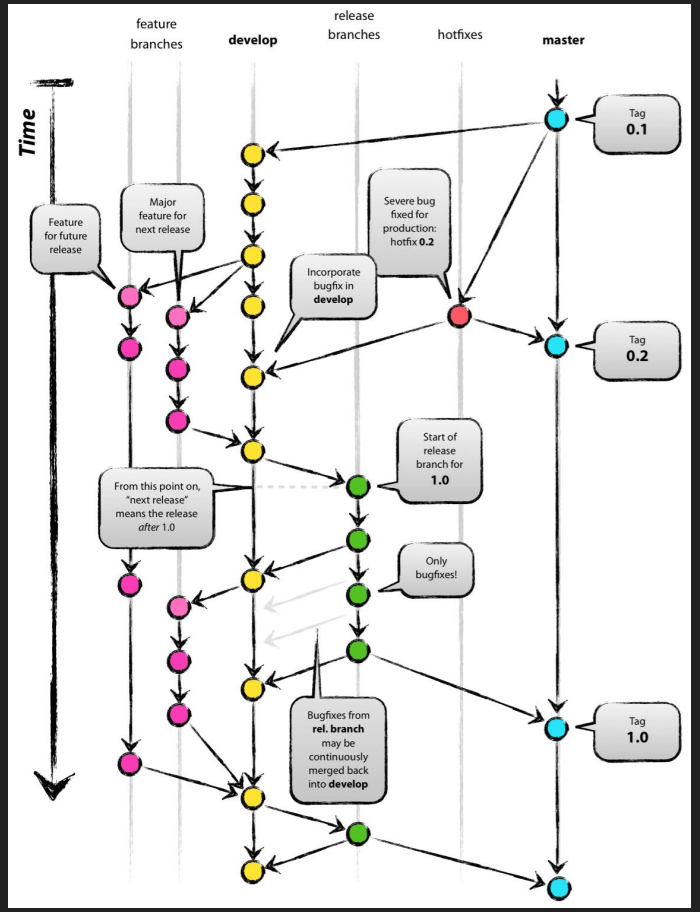
Rebase:

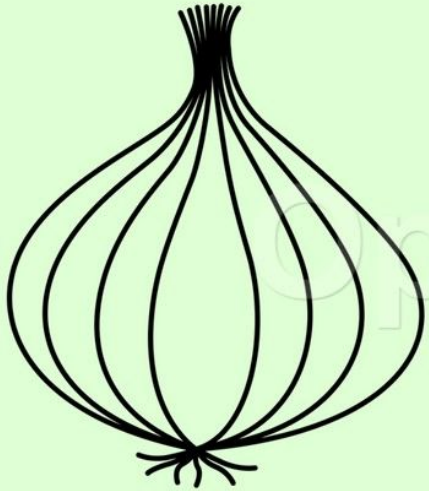
- Keep the history clean and linear
- but can be misleading



When in doubt, Just Merge

Git is a revision control system



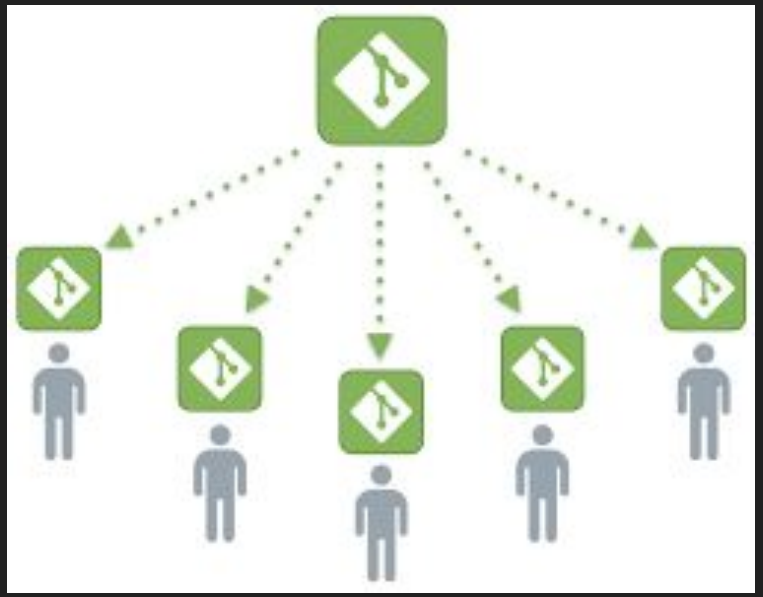


Git is a
Distributed Revision Control System

OpenGenus

https://iq.opengenus.org/git-is-an-onion/

Relation among network of repository



Between repository

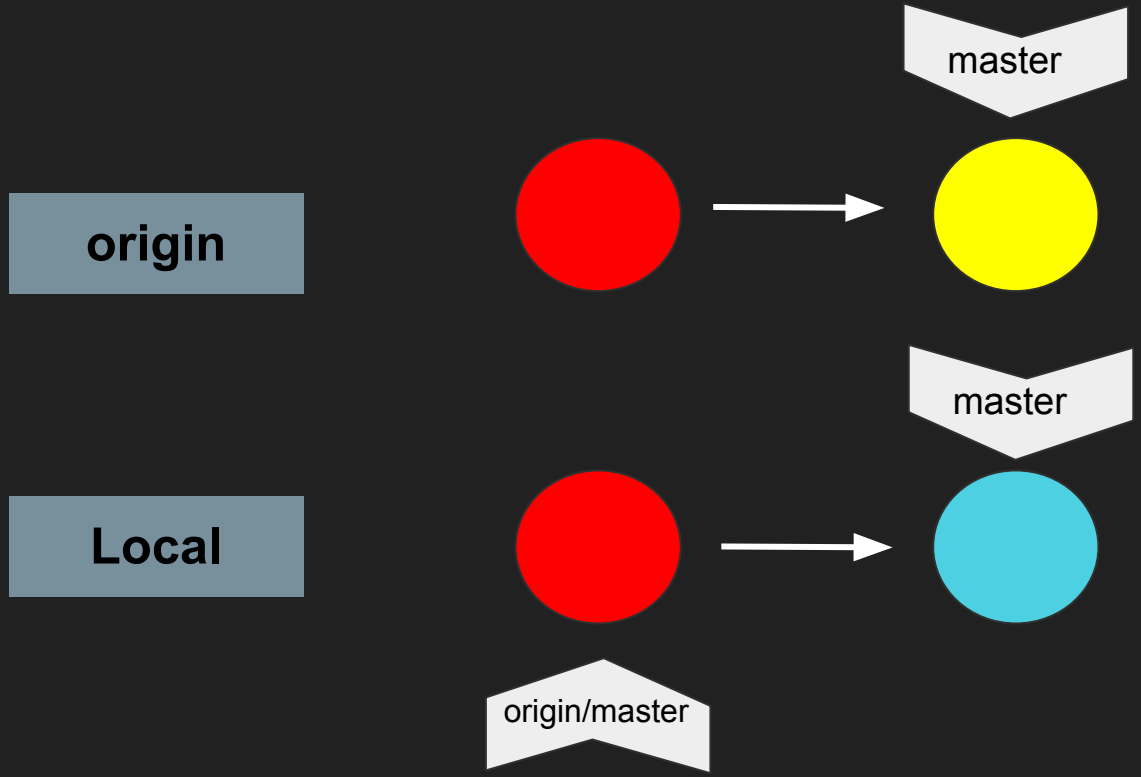
- A remote branch is just a reference to commit, just like a local branch
- The missing "remote branches" are compacted in packed-refs

synchronizing between remote repositories

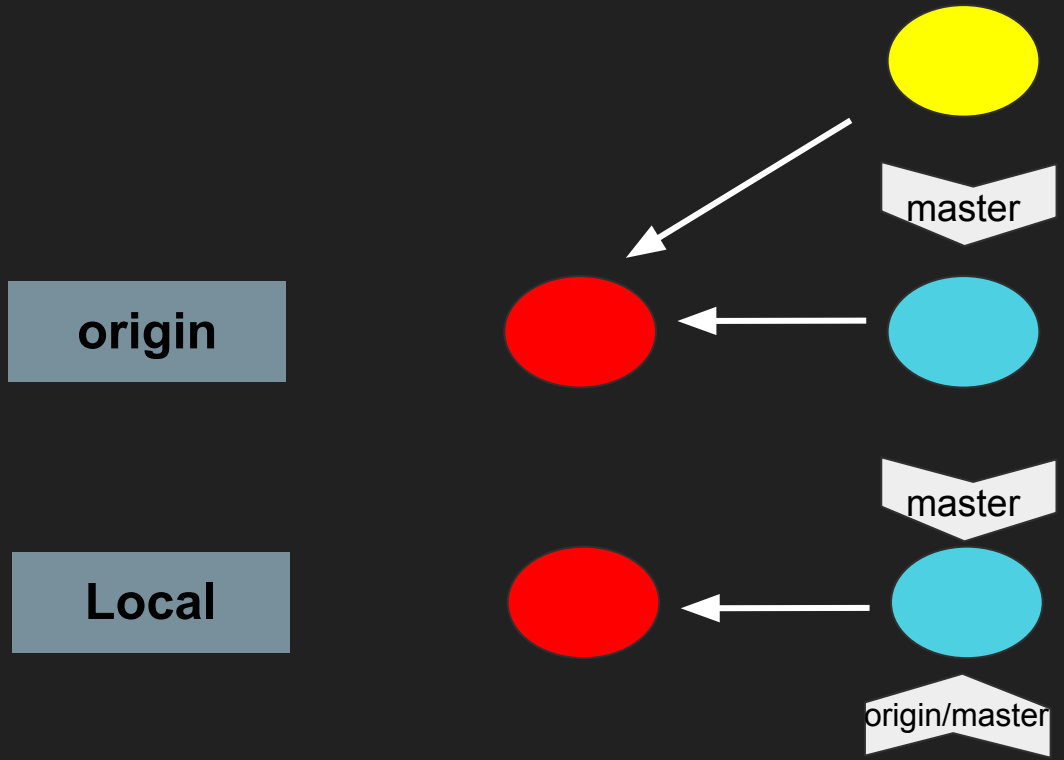
- force method
- fetch or pull method

Force method

What Happen



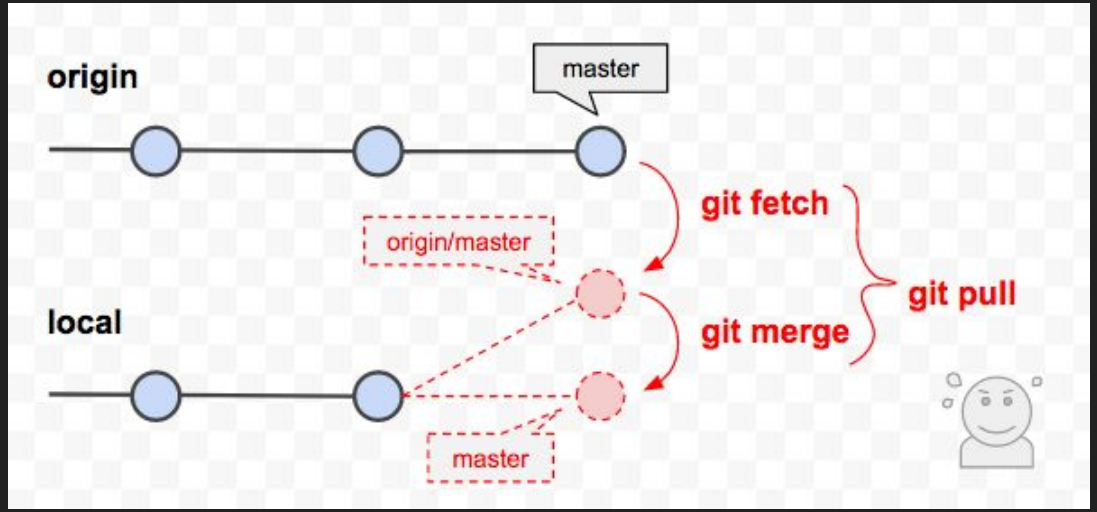
git push -f



Trouble!!

git fetch / pull method then push

- fetch or pull first
- solve conflict
- push

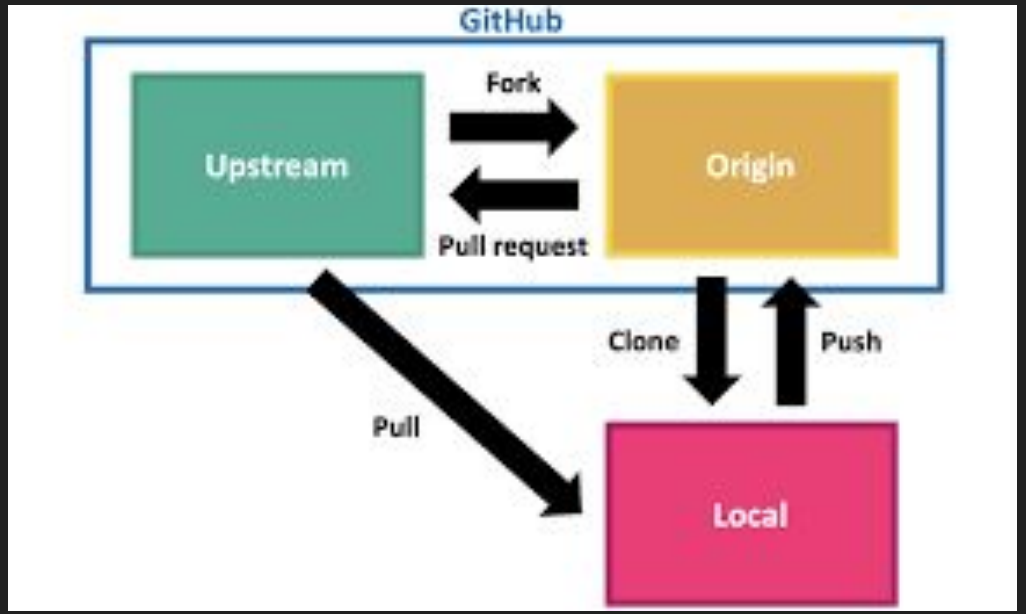


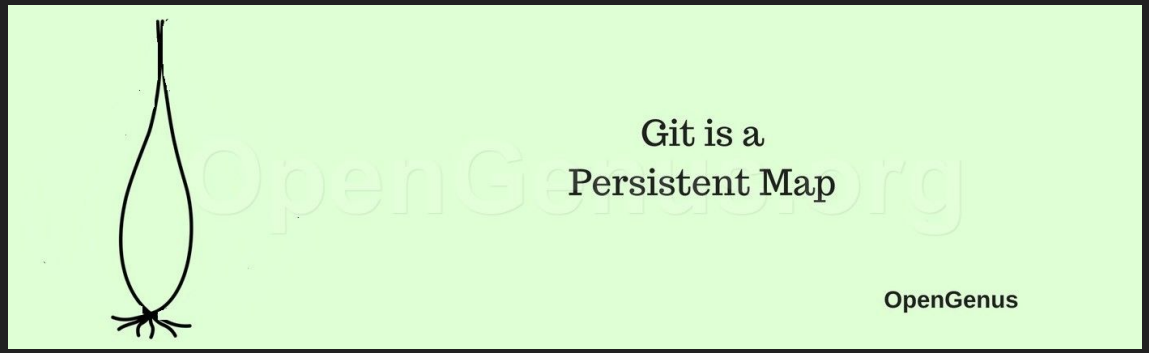

GIT PULL UPSTREAM
MASTER
GIT PUSH ORIGIN MASTER

Github convenitions

- Forking

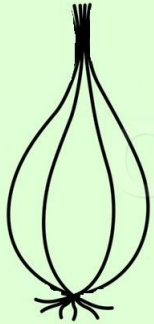
- synchronize with upstream





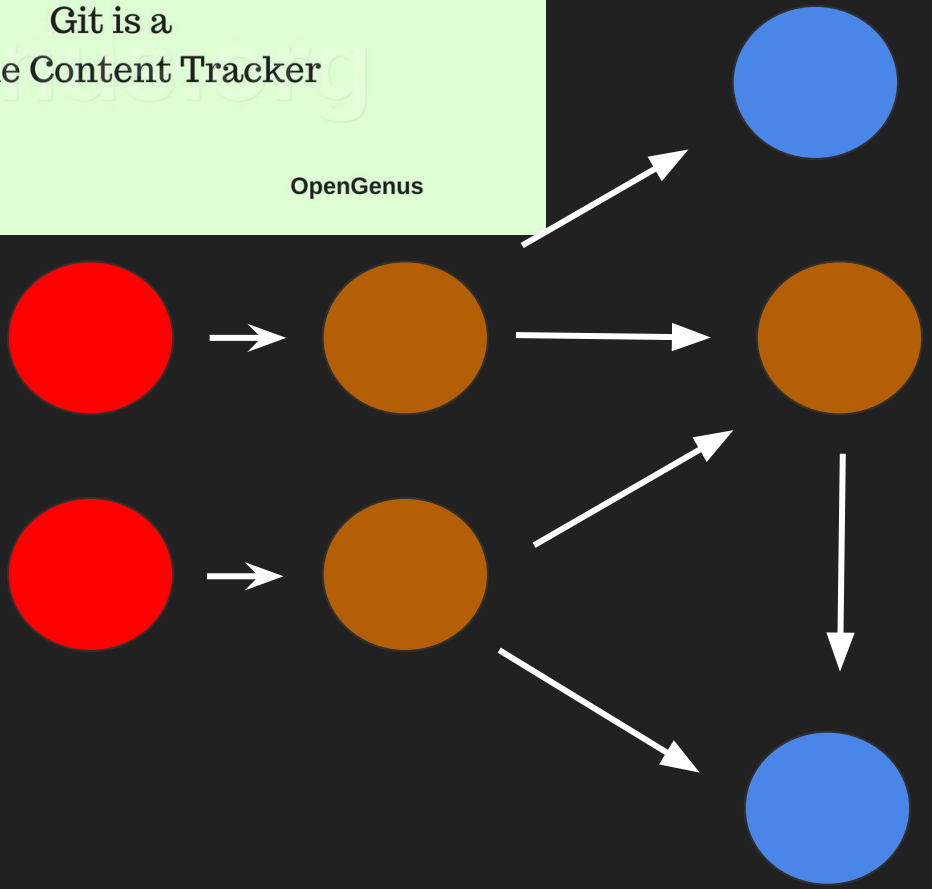
Coca-Cola

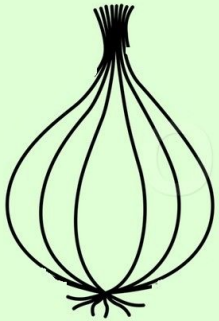
6ee3af491710caada52faf3b89b16d85786249a2



Git is a
Simple Content Tracker

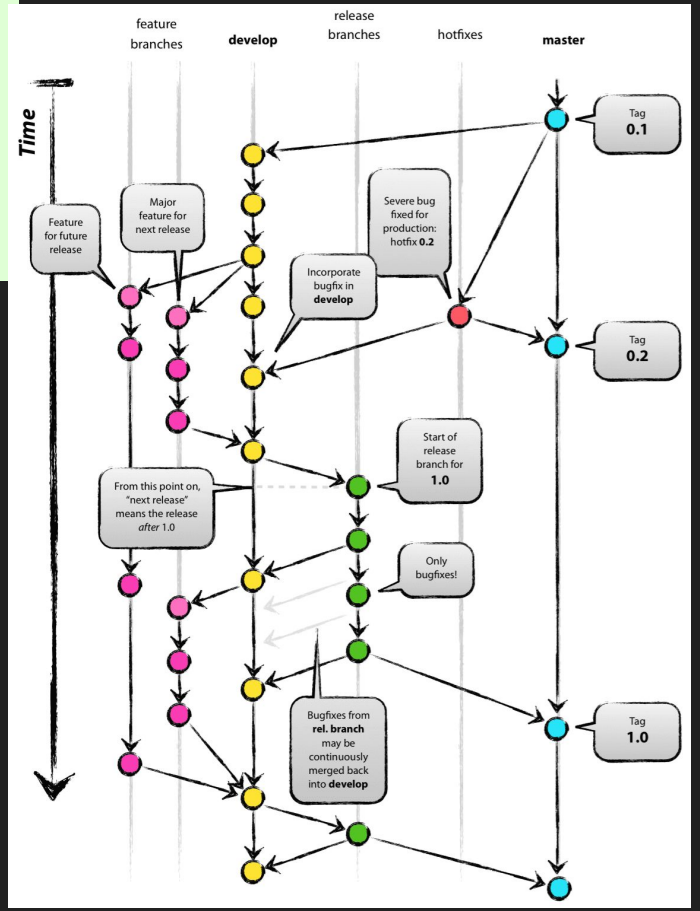
OpenGenus

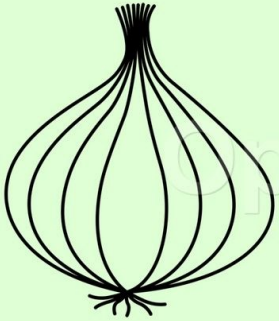




Git is a Revision Control System

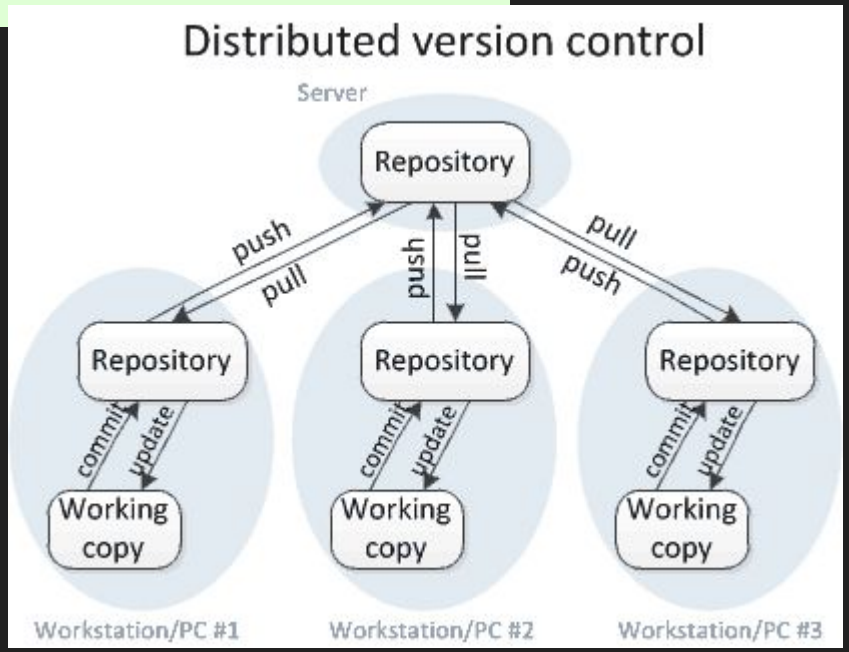
OpenGenus





Git is a
Distributed Revision Control System

OpenGenus



Thanks!!

References

This course is highly inspired by Paolo Perrotta's "How git works" course

Free online reference:

- git internal:
  - https://git-scm.com/book/en/v2/Git-Internals-Git-Objects
  - https://realpython.com/python-git-github-intro/#aside-what-is-a-sha
  - https://www.youtube.com/watch?v=P6jD966jzlk
- Onion model:
  - https://iq.opengenus.org/git-is-an-onion/