# INTRODUCTION TO GIT

Introduction to Git

**Dr. Igor Steinmacher**
e-mail: igorfs@utfpr.edu.br
Twitter: @igorsteinmacher

## Code Management/Versioning

- Team development
  - **Code sharing and versioning…**

## Code Management/Versioning

git

SUBVERSION

mercurial

CVS

Bazaar

…

## We will focus on:

git

# WHO OFFERS THIS SERVICE

- Your machine! (?)

GitHub

Atlassian Bitbucket

GitLab

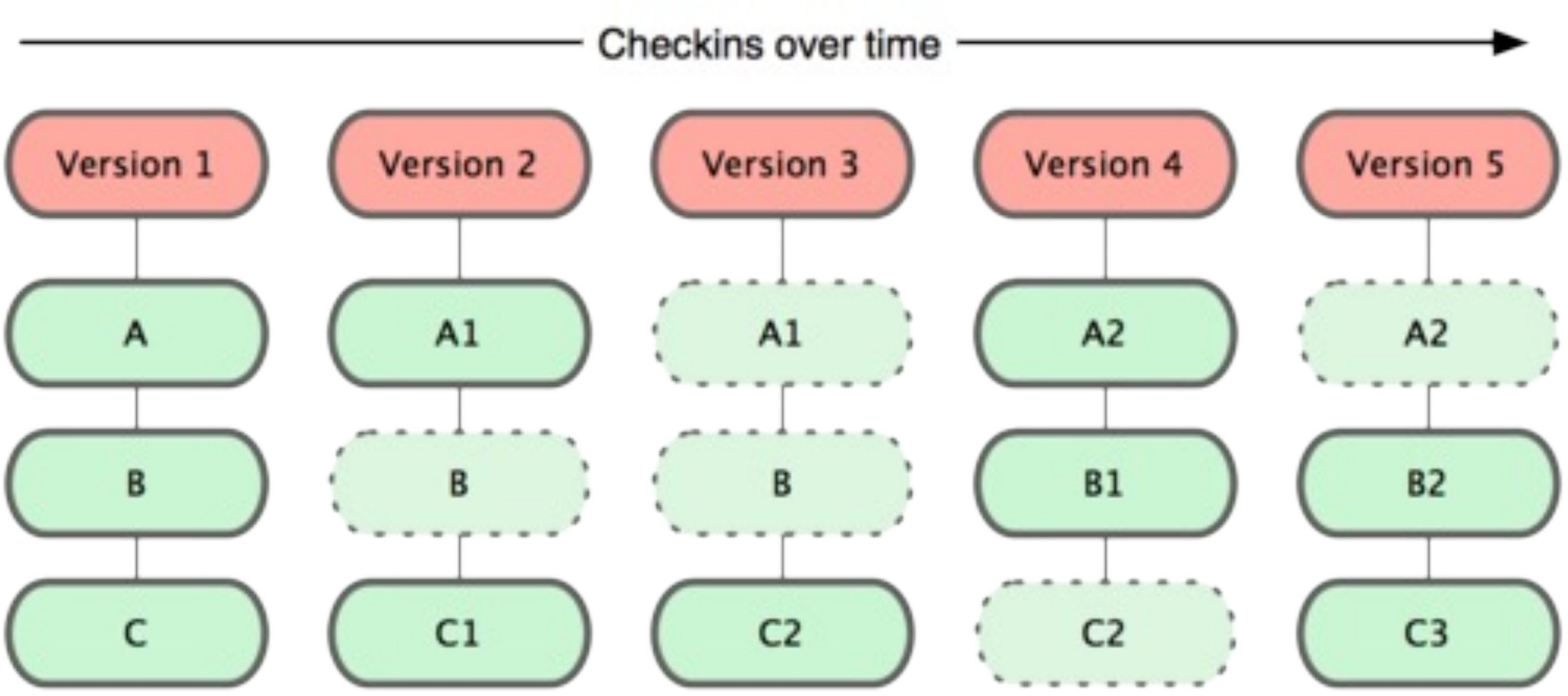
beanstalk

SOURCEFORGE®

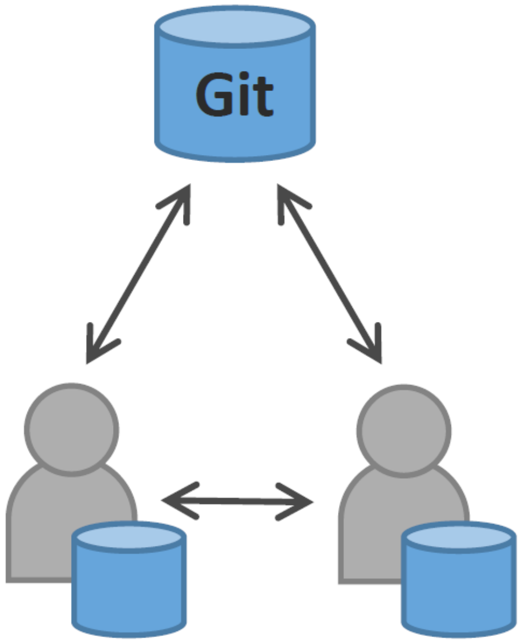
HelixTeamHub

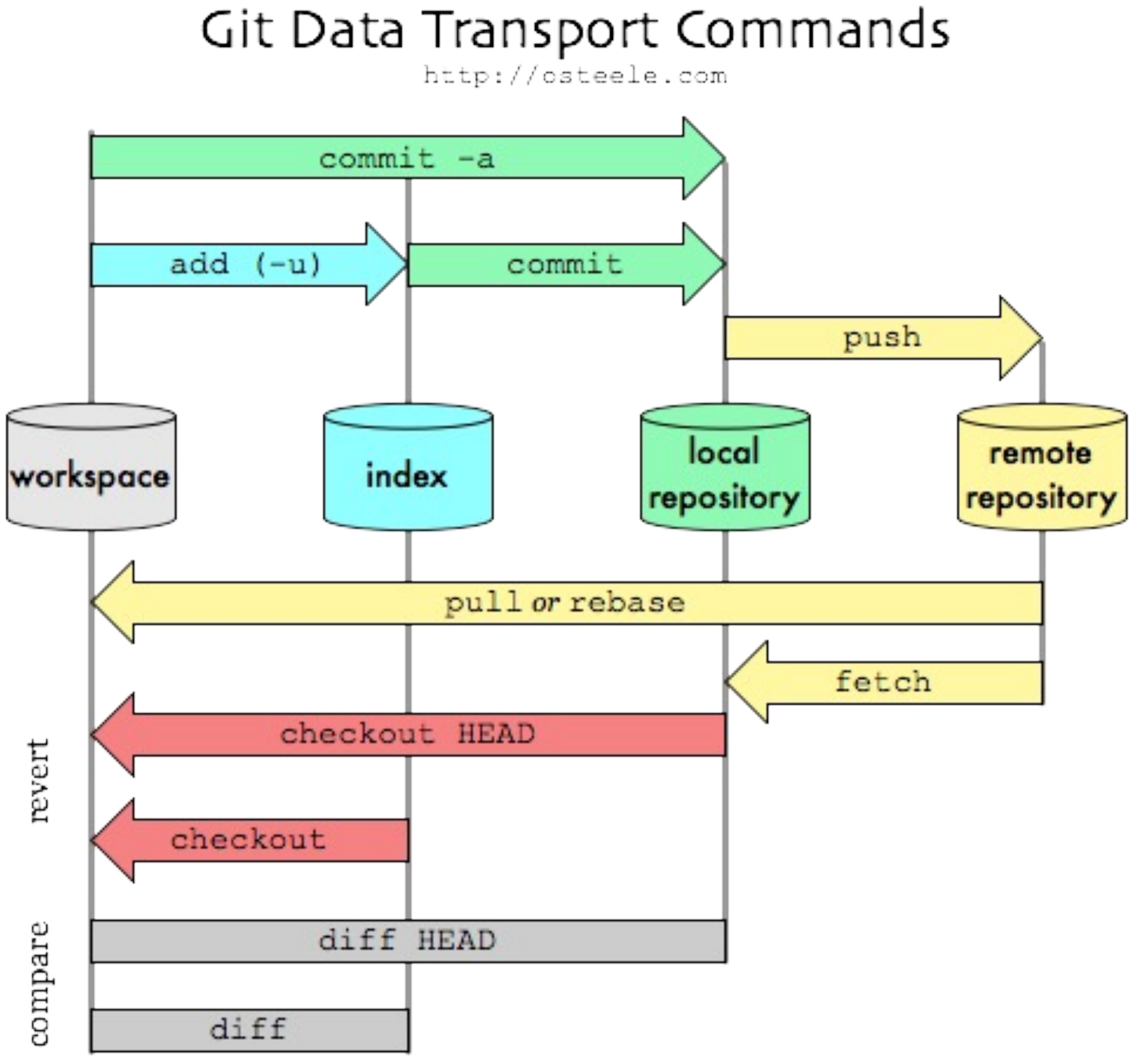
codebase

## How Git Manages Files Over Time

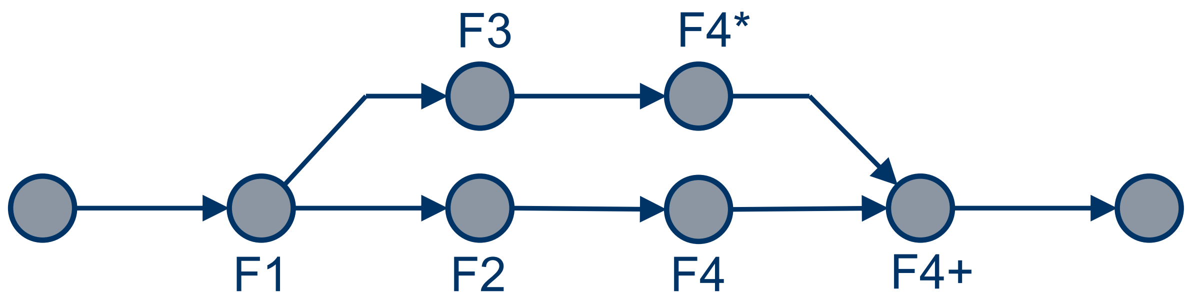


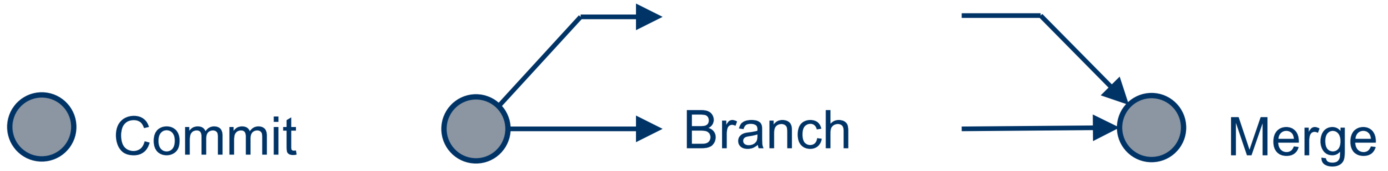
# GIT - OVERALL


Git

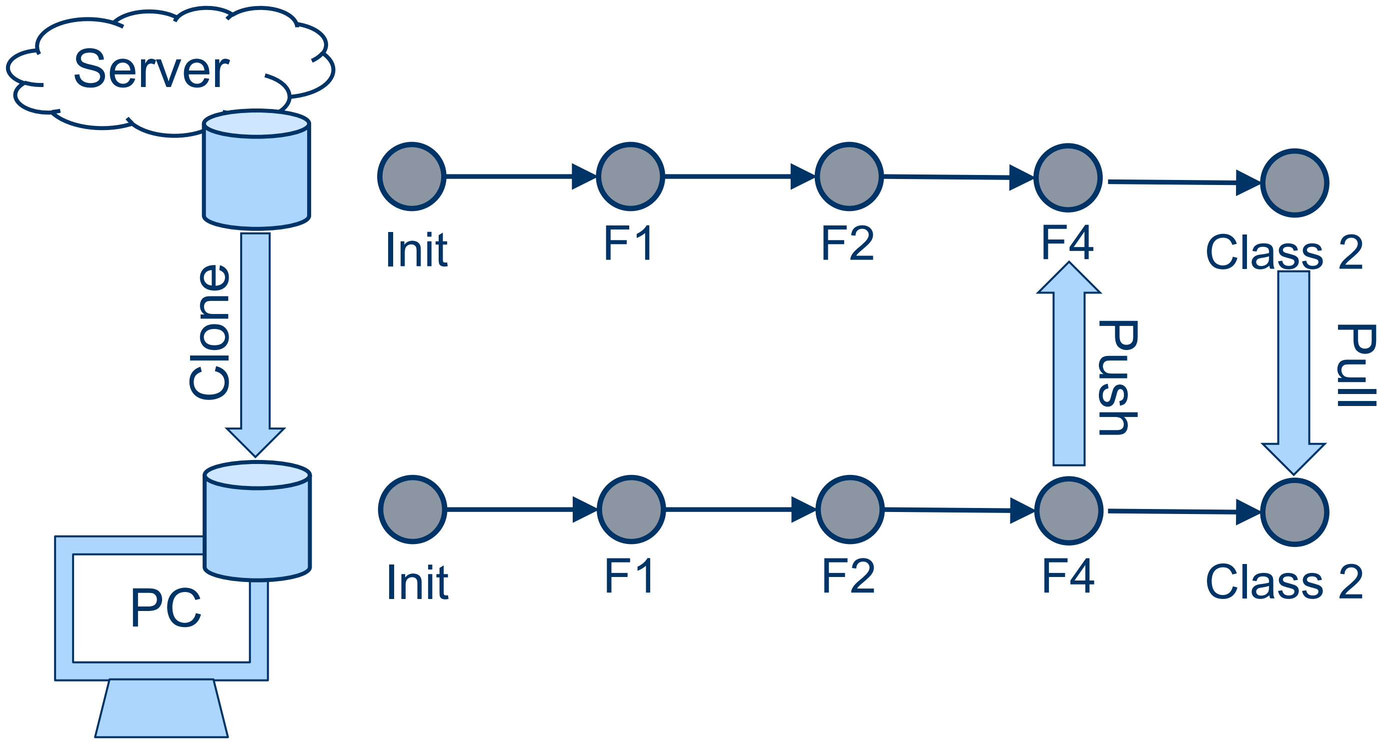

Git Data Transport Commands
http://osteele.com
commit -a
add (-u)
commit
push
workspace
index
local repository
remote repository
pull or rebase
fetch
checkout HEAD
checkout
revert
diff HEAD
diff
compare

# Git Local Flow - Example





# Git Local Flow - Example



## It's Hands On Time

- 2 Moments

- Moment 1:
  - **Local commands: add, commit, branch, merge, conflicts..**

- Moment 2
  - **Interaction with the remote repo: Push, pull**

## KICKING OFF

- git config --global user.name “Igor Steinmacher”

For all repos

- git config --global user.email “igor.Steinmacher@nau.edu”

- Create a folder / access this folder
- git init
  - **This folder is a repo**

## Hands On

- Create a file
- Check the status of the repo
  - **git status**
- Add the file to the index
  - **git add <filename>**
- Check the status

## Hands On

- Our first commit
  - **git -a -m “Our first commit!!!”**
    - -a → all files
    - -m → will include a commit message
- Check the last commits
  - **git log**
- Check what has been done in the last commit
  - **git show**

## Hands On

- Let’s!
  - Change the file
  - git status
  - git commit …
  - git status
  - git log

- And this is the basic flow to put your contributions back to the repo
  - add
  - commit
  - status
  - log
  - show

## Let's Branch it out!

- Listing your branches
  - **git branch**

- Create a branch
  - **git branch <NEW_BRANCH>**

- Use another branch
  - **git checkout <BRANCH_NAME>**

- Latest two in one command
  - **git checkout –b <NEW_BRANCH>**

# Working in a new Branch

- git checkout -b branchNew
- <Change one existing file here>
  - **“Hi, my name is Hugh.”**
- git commit -a -m “Introducing myself”
- <Check the content of the file>

- git checkout master
- <Check the content of the file>
  - **What?!?!**

## Updating the Master Branch

- Usually we branch out for versions, features, bug fixes…
  - **Later on merging back to master**
- How to merge our recently changed file, then?
  - **In the master branch:**
  - **git merge <other_branch>**
- If everything goes smooth… sweet

# Dealing With Small Conflicts

- Imagine if you change a file in your branch, and someone else changed the same file
  - **CONFLICT!!!!**

- Can we still merge it?!?!?
  - **Let's see:**
    - Change branch
    - Change file
    - Commit
    - Back to master
    - Change the same file
    - Commit
    - MERGE!

```
Auto-merging <file>
CONFLICT (content): Merge conflict in <file>
Automatic merge failed; fix conflicts and then
commit the result.
```

## Usually… For the Easy ones

```
Here comes common text before
the area where the conflict happens
and bla bla bla
<<<<<<< HEAD
This is what was in the
master branch
=======
And this…
was in the other branch
>>>>>>> other branch
More text that was common,
and no conflict happened here
```

# Exercising it out

- Make a new branch called bugfix
- Checkout the bugfix branch with git checkout bugfix
- Create/change a file and commit
- Go back to master with git checkout
- Change/create a file (different from the previous one) and commit again
- Merge the branch bugfix into master with git merge

## Rebasing it all!!!

- Another way of combining branches
- We can take a set of commits, and copy them in another branch

- Master and our branch are not sync'ed
  - **Commit made in master**
  - **Commit made in the branch**

- From the branch, we can
  - **git rebase master**
  - **Now the index of the master is "outdated" → HEAD is pointing to bugfix last commit**
    - git log --graph --all

## Moving From Here to There

- HEAD is the pointer name for the last checked out commit

- We can "detach" the head by "checking out" a specific commit
  - **git checkout <commit SHA>**
- This is not "safe"
  - **git checkout <branch>**
    - To get back
- Moving in the commit tree
  - **Moving one commit at a time with ^**
    - git checkout HEAD^
    - git checkout master^
  - **Moving a number of times with ~<num>**
    - git checkout master~3

## Moving From Here to There

- We can move a branch!
  - **git branch -f <BRANCH> HEAD~3**

- We can also revert changes
  - **git reset HEAD~2**
    - Move the current branch to HEAD - 2 commits position
    - Works LOCAL
    - git reflog → see previous commits
    - git reset <SHA> → SHA of the commit before the reset for "unresetting"

  - **git revert HEAD → To reverse changes to send upstream**

## Cherry-Picking

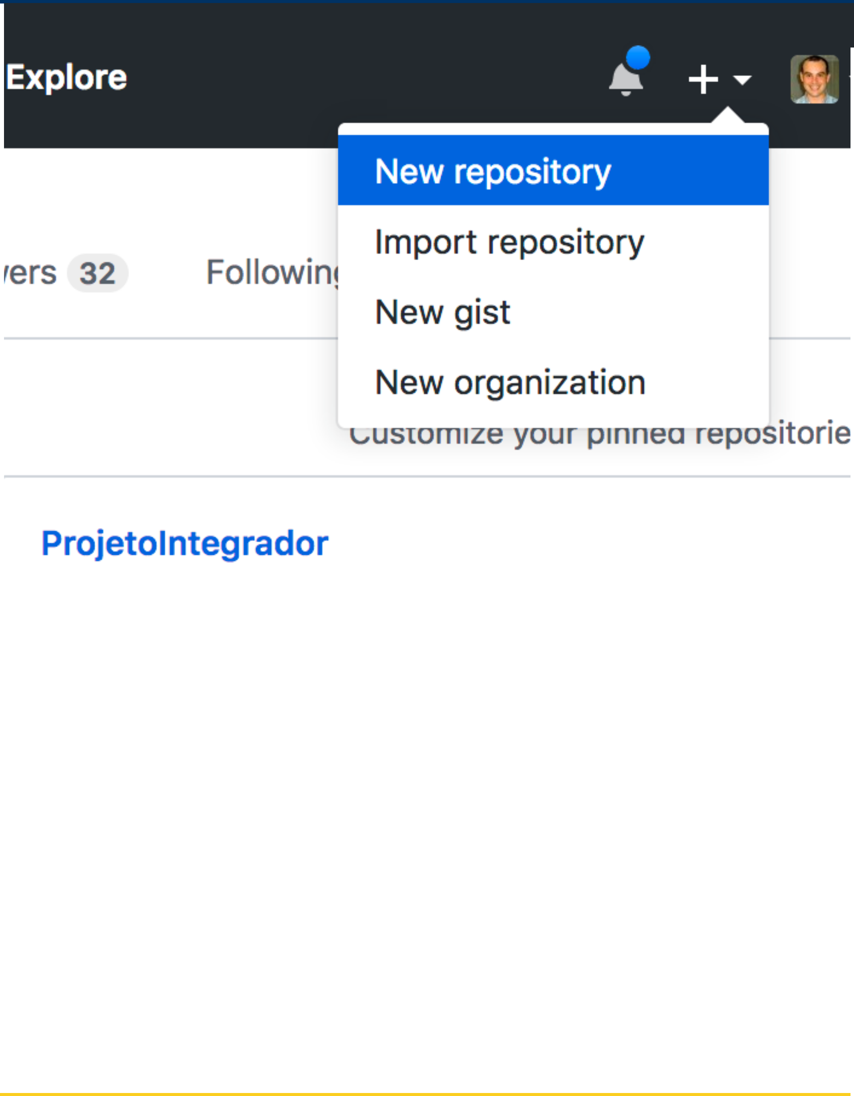
- Use when you don’t want to copy ALL commits from a branch to another
  - **We can cherry pick those that are of interest**

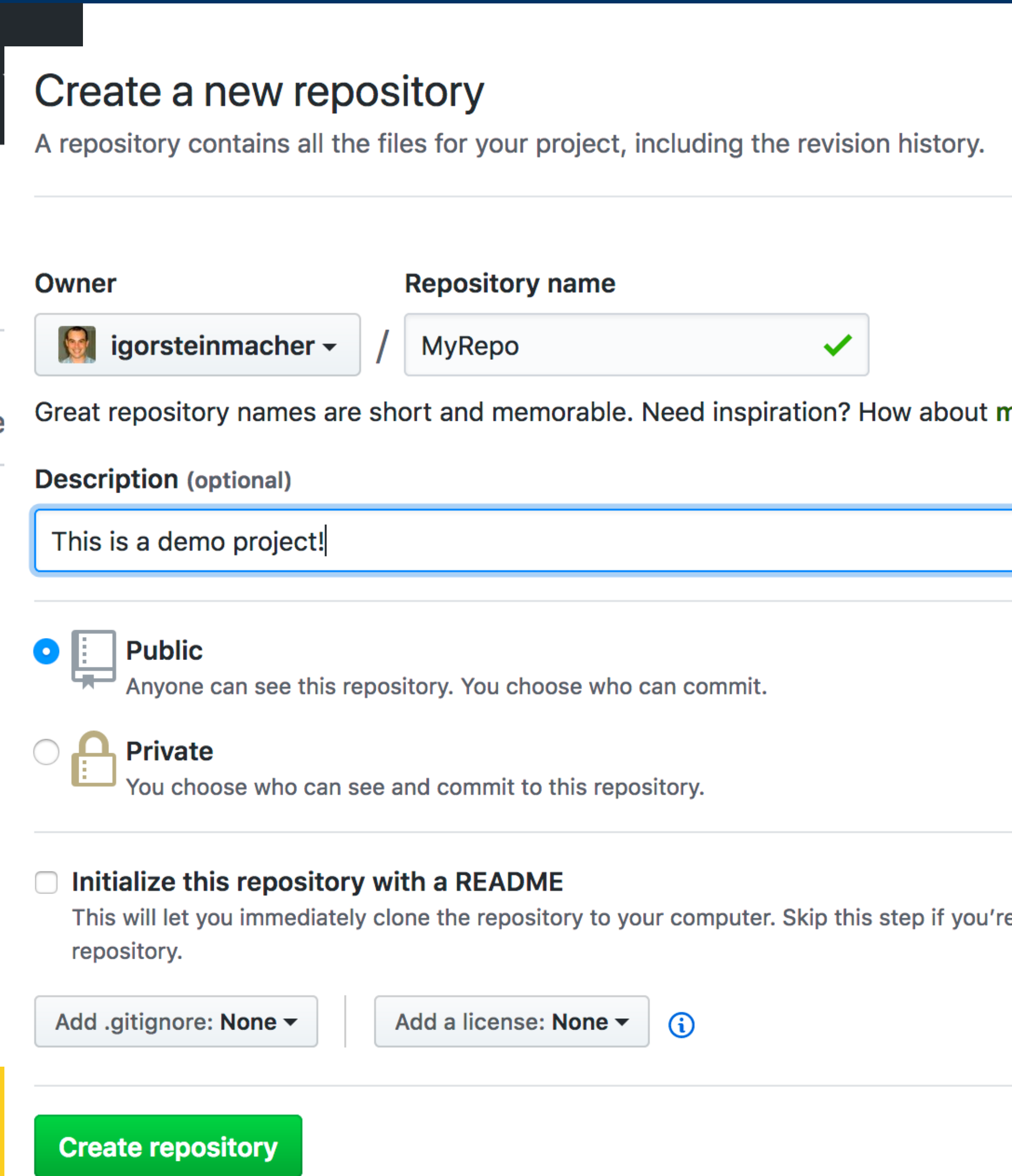
- git cherry-pick <SHA1> <SHA2> <SHA3>

## Dealing with the Remote Repo

github
SOCIAL CODING

# Create a Repo in GitHub





Create a new repository

A repository contains all the files for your project, including the revision history.

**Owner** igorsteinmacher / **Repository name** MyRepo ✓

Great repository names are short and memorable. Need inspiration? How about m

**Description** (optional)

This is a demo project!

● **Public**
Anyone can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're
repository.

Add .gitignore: **None** | Add a license: **None**

**Create repository**

## Cloning To a Local Repo

- That's simple! Cloning means bringing all the history to a local repo
  - **git clone https://github.com/<owner>/<repo>**

- Testing it out
  - **git clone https://github.com/NAU-OSS/githandson.git**

```
Cloning into 'githandson'...
warning: You appear to have cloned an empty repository.
```

## Working In This Repo

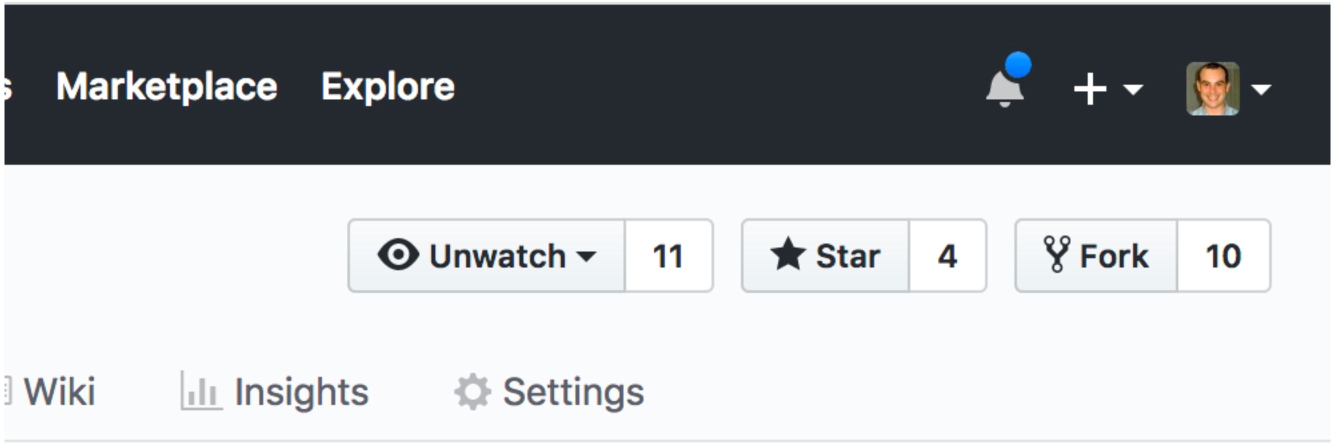
- Some different commands to deal with remote repo
  - **git branch –r**
  - **git pull** //pulls everything from the remote repo and updates the local repo
  - **git fetch** //pulls changes from remote repos, but it doesn't integrate any of this new data into your working files
  - **git push**
    - git push <remoteName> <branchName> //push your local changes to an online repository (git push **origin master**)
    - git push *<remoteName>* *<localBranchName>*:*<remoteBranchName>* *//* This pushes the LOCALBRANCHNAME to your REMOTENAME, but it is renamed to REMOTEBRANCHNAME.

## Usual Workflow

- git clone
  - **branch out to add your changes locally**
  - **your adds/commits**
  - **pull changes to your local repo**
  - **merge your branch back (LOCALLY)**
    - Resolve any conflict
  - **push changes back to the remote repo**

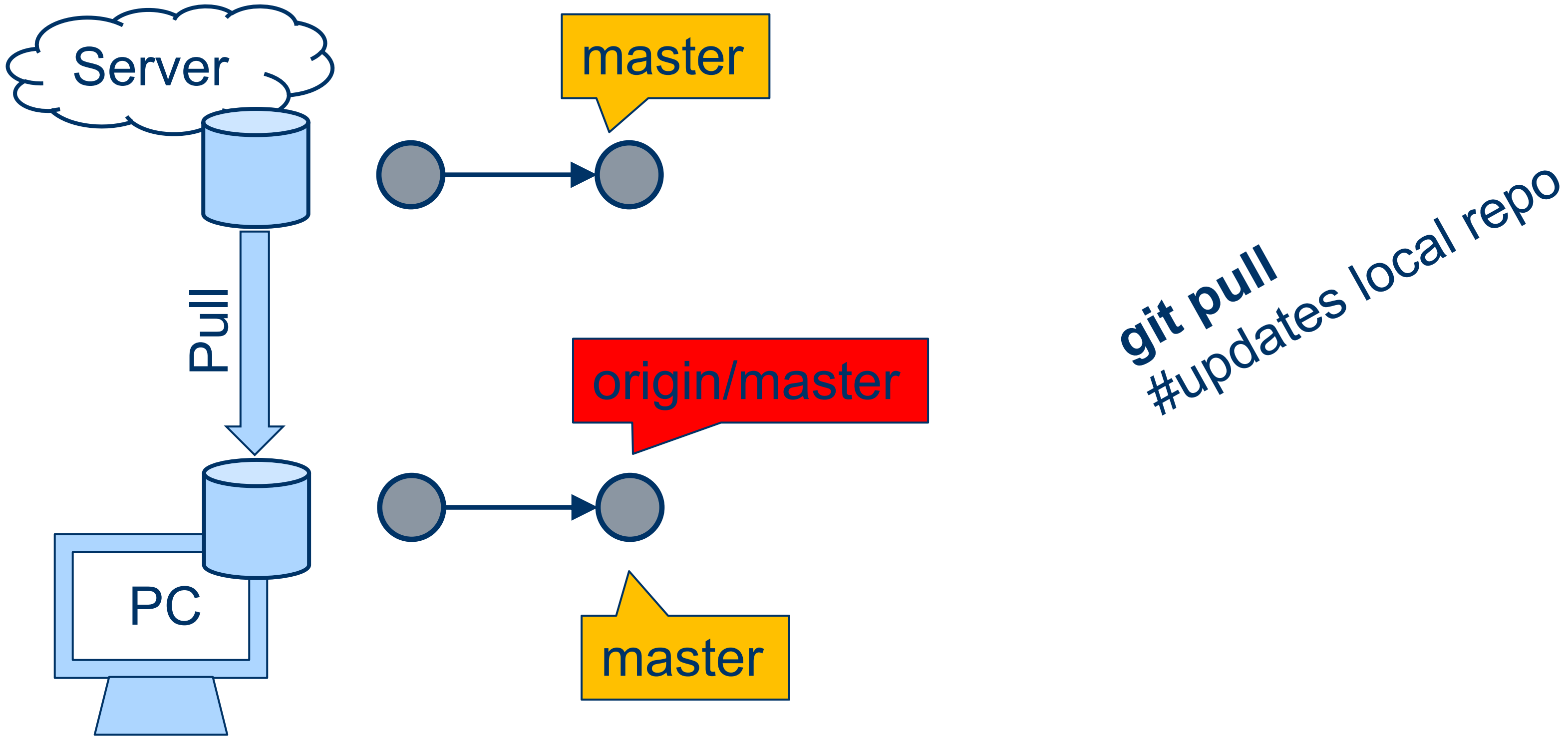
## GitHub Workflow

- Fork + pull-request
  - **You usually creates a fork for your repo**
    - “A **fork** is a copy of a repository. **Forking** a repository allows you to freely experiment with changes without affecting the original project”
  - **Creating a fork**



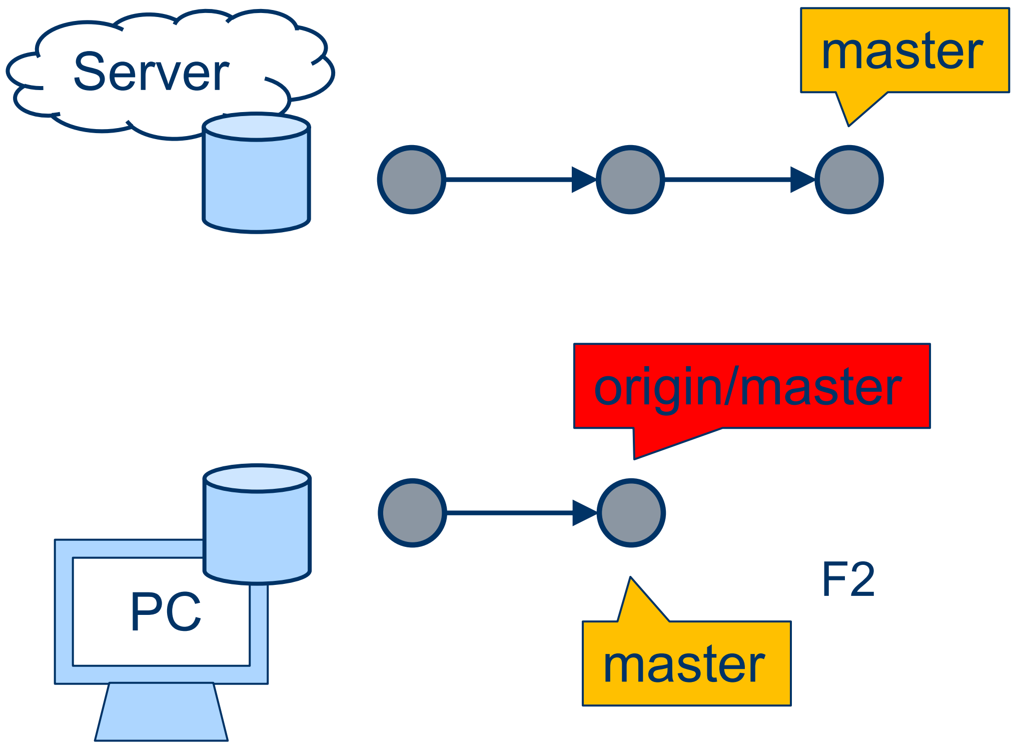
  - **Then, you usually clone your fork… work, and send a pull request against the main repo**

# EXAMPLE



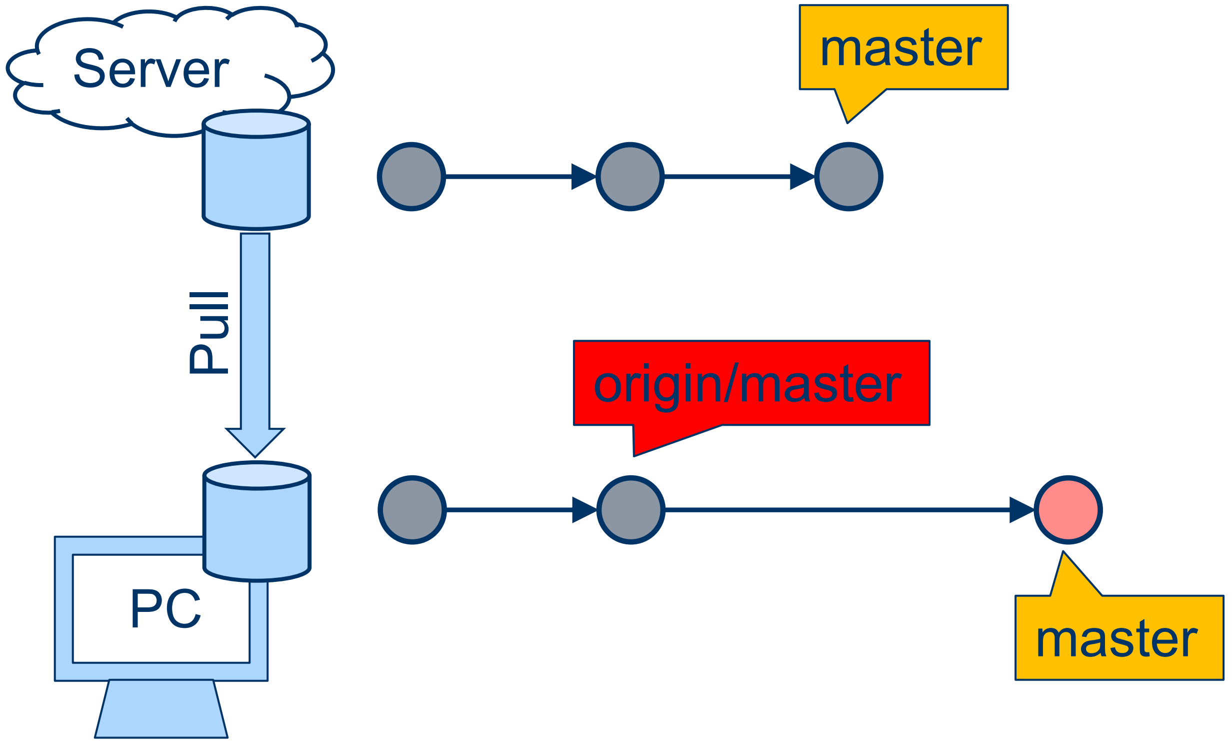
**master** is a local branch
**origin/master** is a remote branch (a *local copy* of the branch named "master" on the remote named "origin")
**origin** is a remote

# EXAMPLE



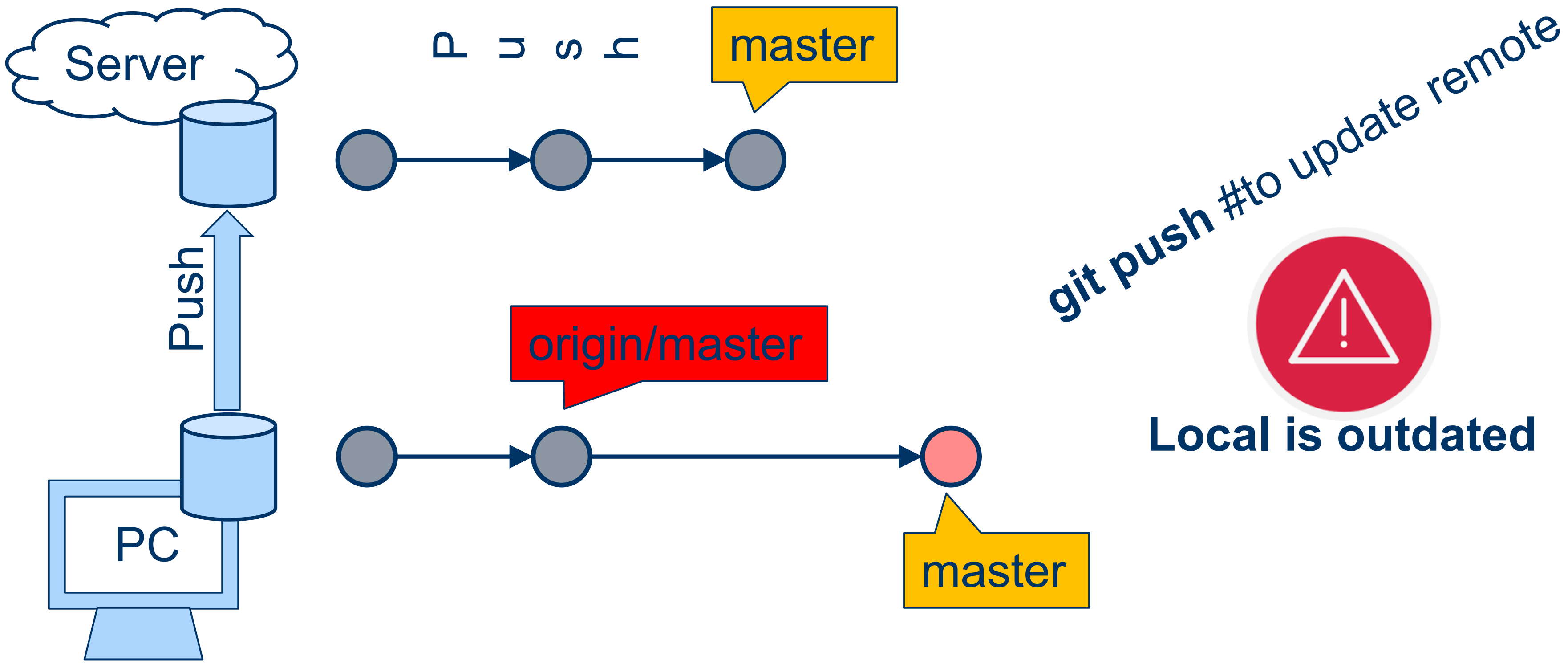
**Someone changed the remote**

# Example

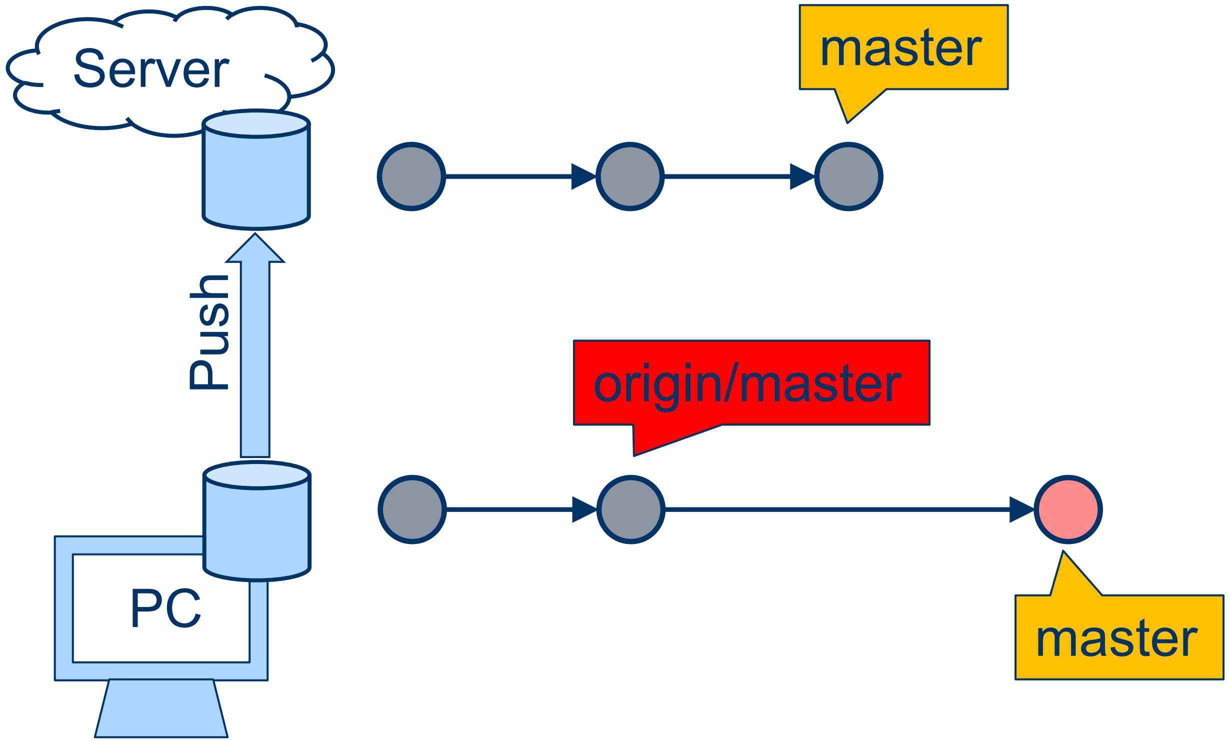


Me too
**git commit -a**

# EXAMPLE

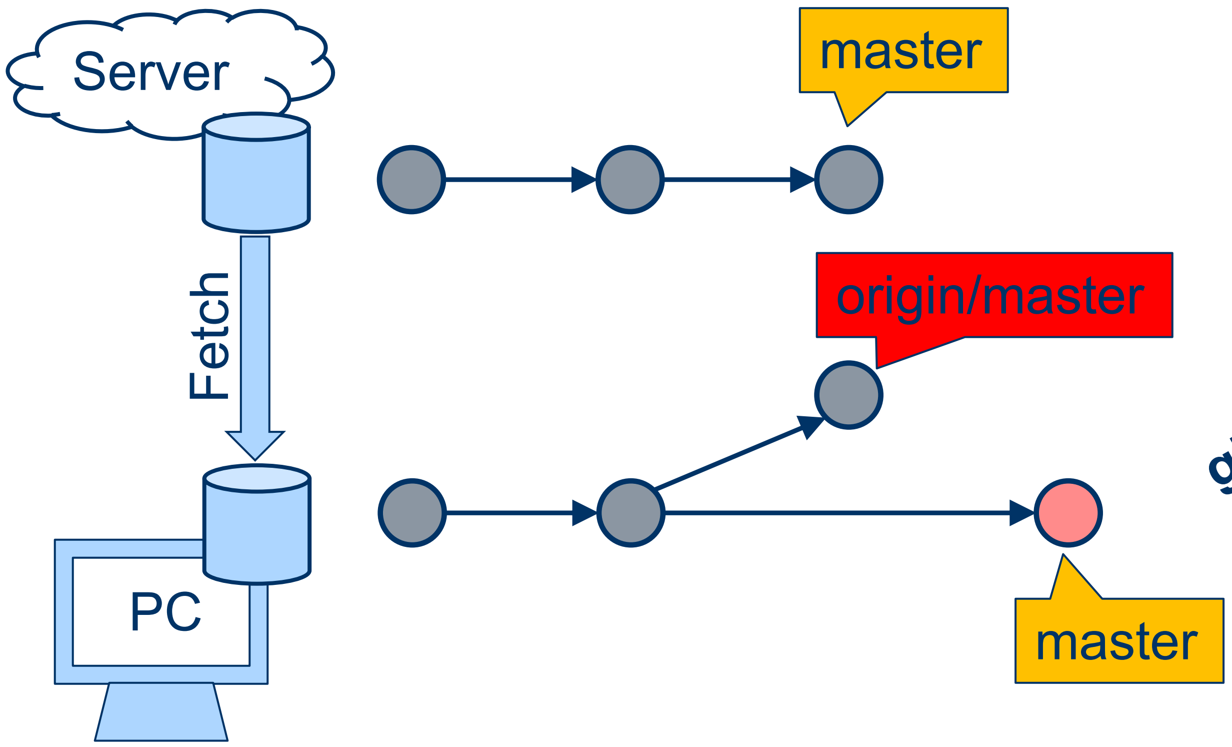

Server
Push
master
Push
origin/master
master
PC
git push #to update remote
Local is outdated

## Example



**Solution 1:**
Fetch +
Rebase +
Push

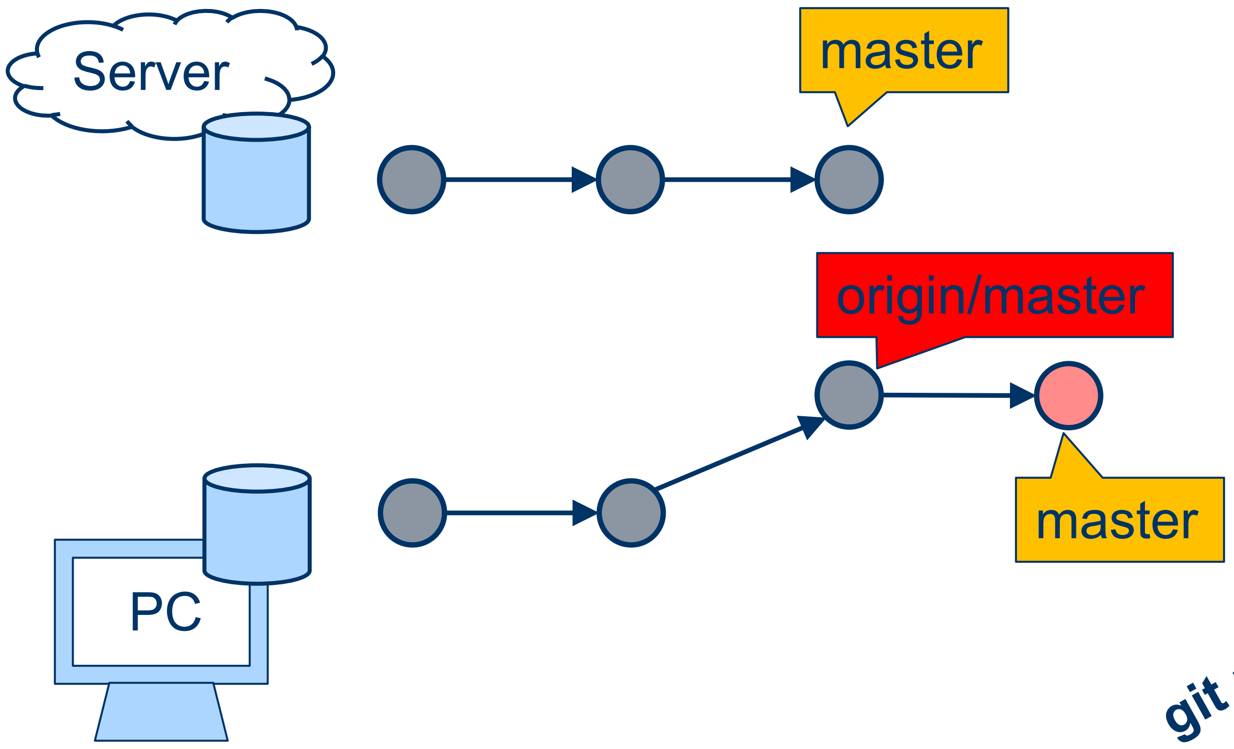
# EXAMPLE



**Solution 1:**
Fetch +
Rebase +
Push

**git fetch**
#download remote changes

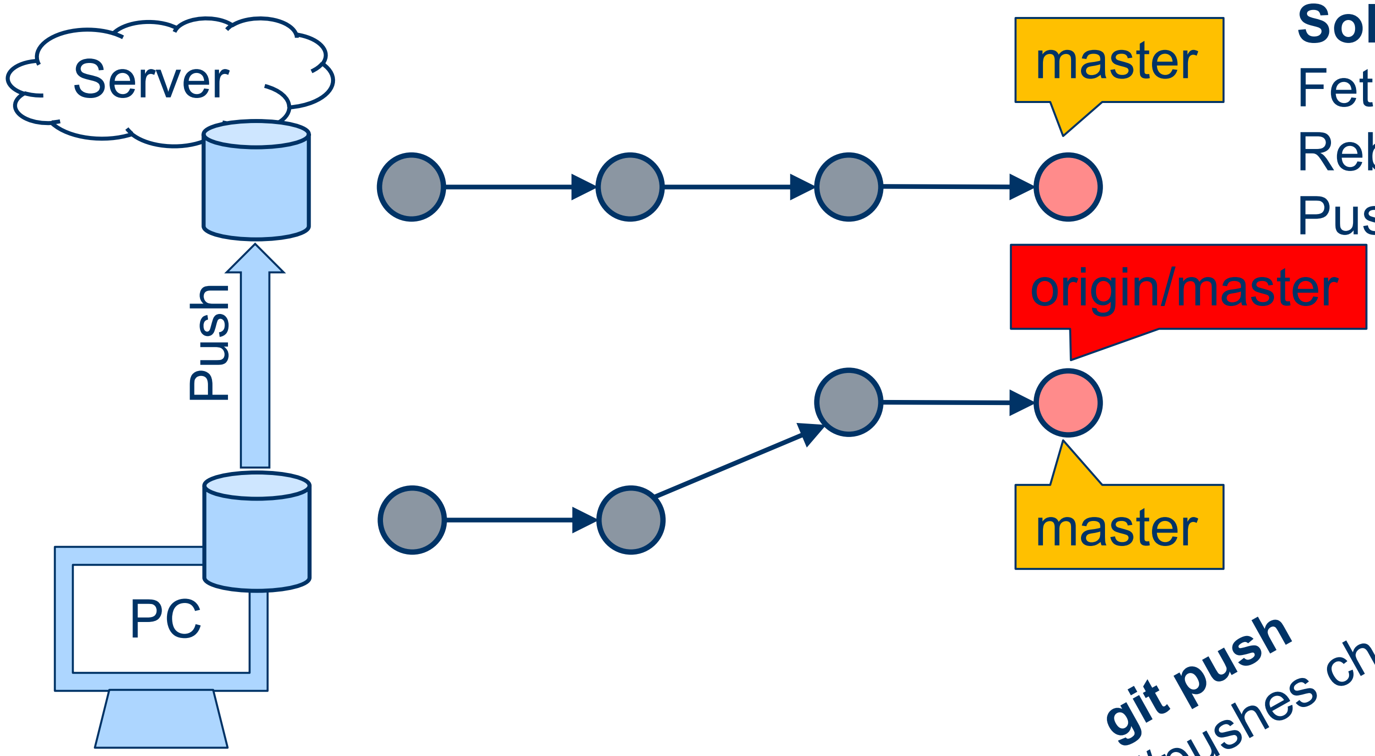
# EXAMPLE



**Solution 1:**
Fetch +
Rebase +
Push

**git rebase origin/master**

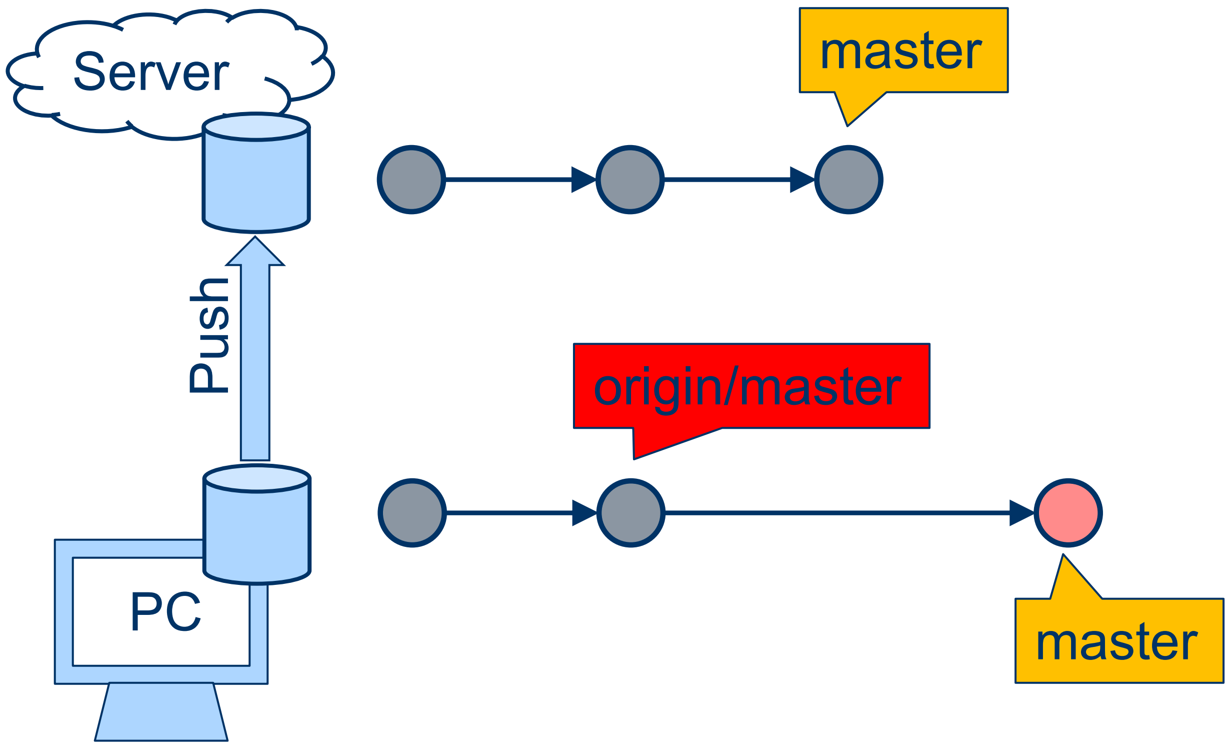
# EXAMPLE



**Solution 1:**
Fetch +
Rebase +
Push

**git push**
#pushes changes to remote

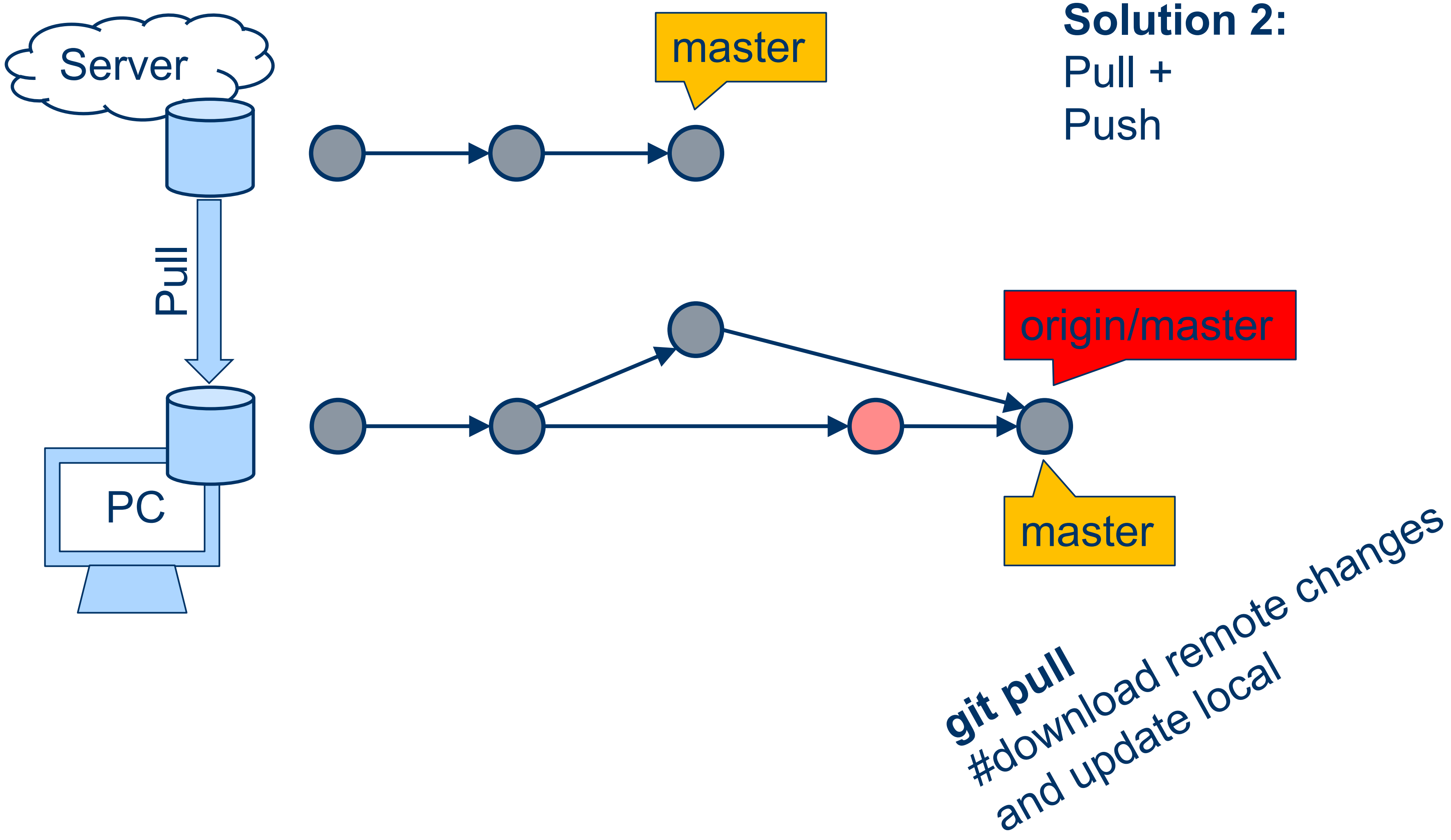
# EXAMPLE



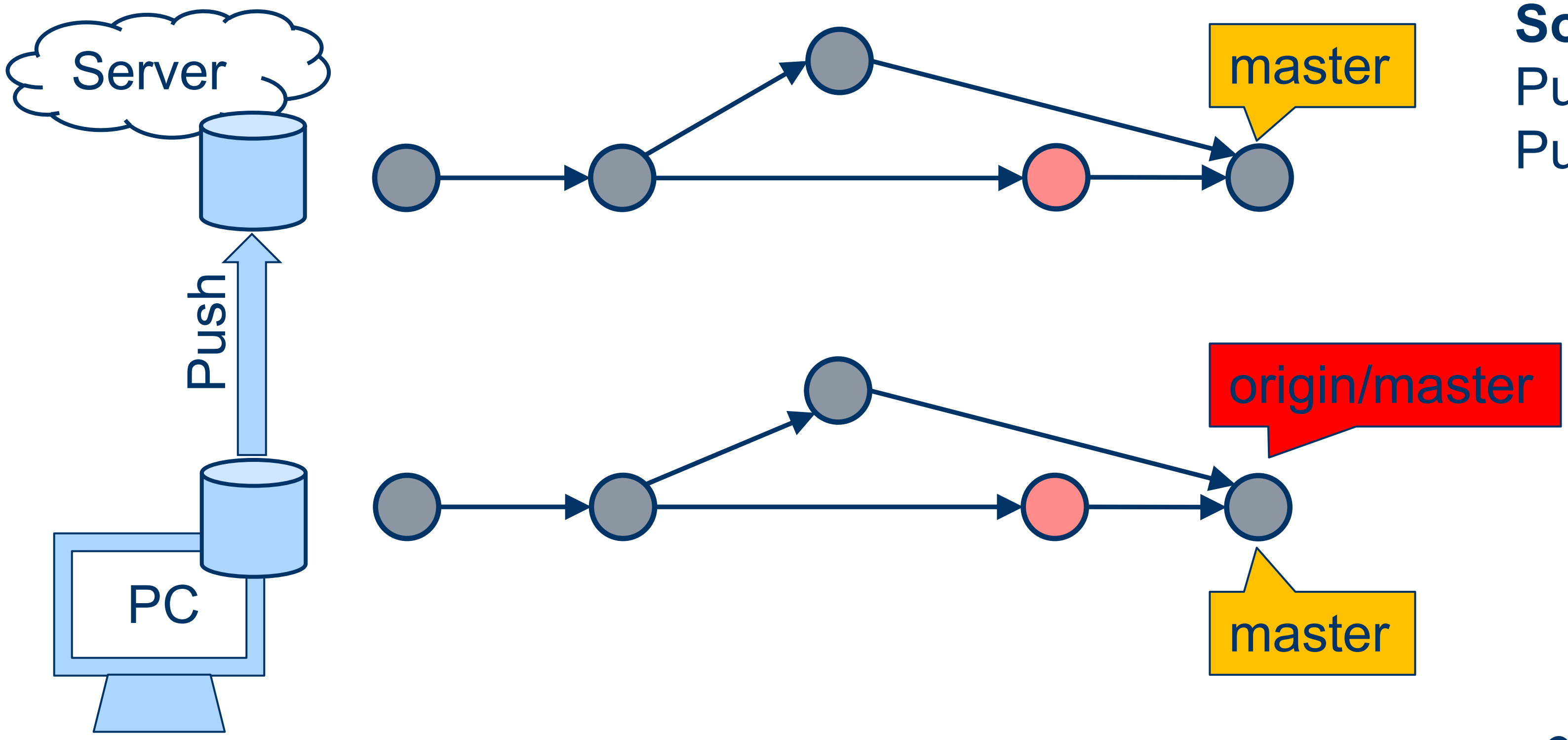
**Solution 2:**
Pull +
Push

# Example

**Solution 2:**
Pull +
Push



# EXAMPLE



**Solution 2:**
Pull +
Push

**git push**
#pushes changes back

## A Pull Request Example

- (GitHub) Fork
- (CLI) Clone (the fork)
- (CLI) Commits
- (CLI) Push
- (GitHub) Send Pull Request
  - **Follow it**
  - **Revise it**
  - **Update it**

- Keep your fork up-to-date
  - **https://help.github.com/articles/syncing-a-fork/**