

INFO 30005

WEB INFORMATION TECHNOLOGIES



**LECTURER:
GREG WADLEY**

Git & **VERSION CONTROL**

You have probably encountered versioning ...





VCS ALTERNATIVES



WHY GIT?

Free and open-source

Fast, good for teams

Branching, staging

Very widely-used

Lots of help (<http://git-scm.com/doc>)



CLOUD-BASED HOSTS FOR TEAMS



GIT CLIENTS

```
$ git log
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date: Mon Mar 17 21:52:11 2008 -0700

    Change version number

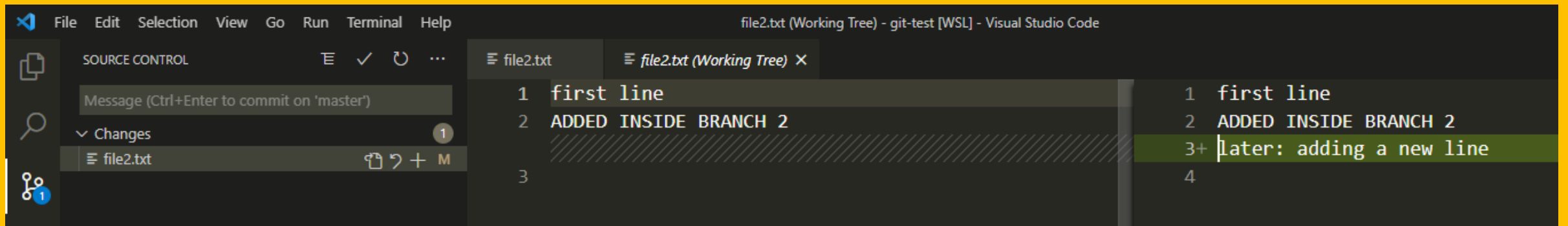
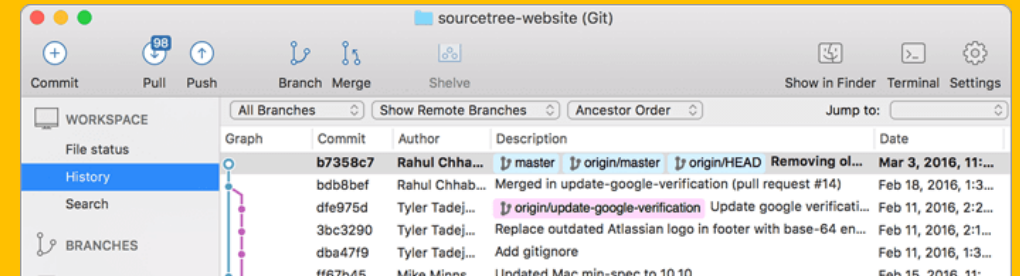
commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date: Sat Mar 15 16:40:33 2008 -0700

    Remove unnecessary test

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gee-mail.com>
Date: Sat Mar 15 10:31:28 2008 -0700

    Initial commit
```

(WSL has no X-Windows yet ☹)



PRE-REQS



*On your laptop,
install GitHub Desktop, or command-line Git.*

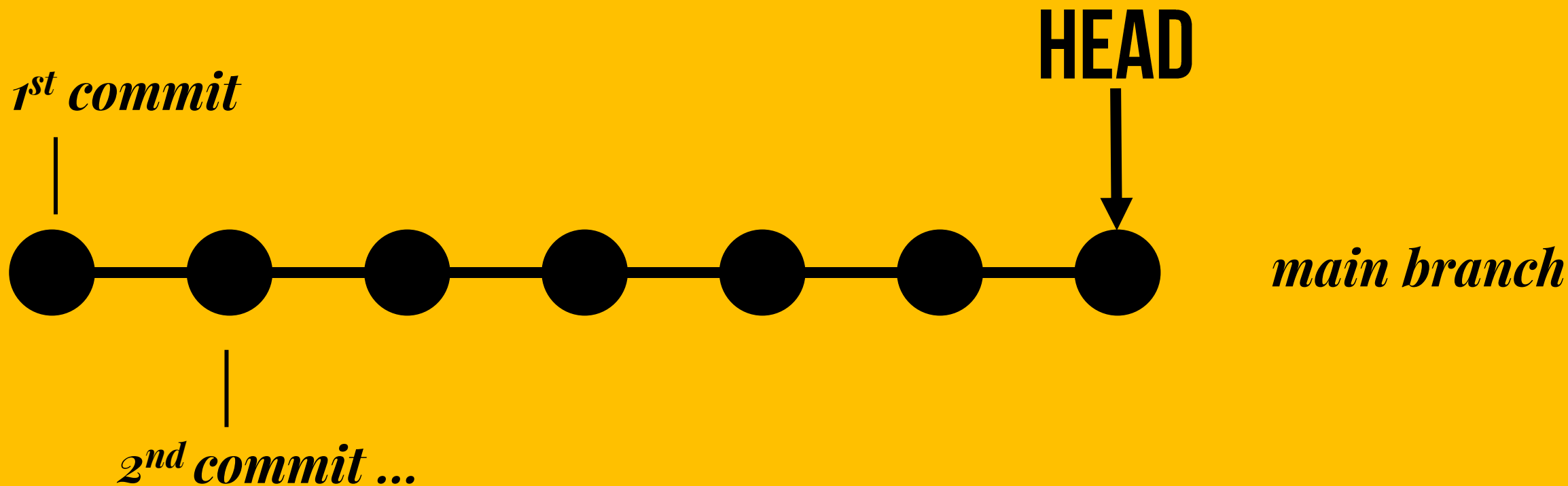
Create a personal GitHub account.



git

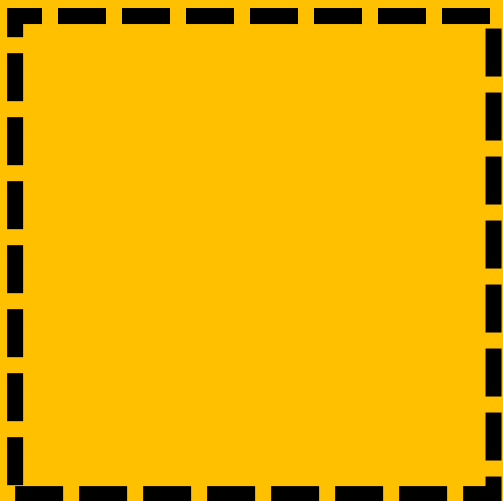
SNAPSHOTS
“COMMITTS”



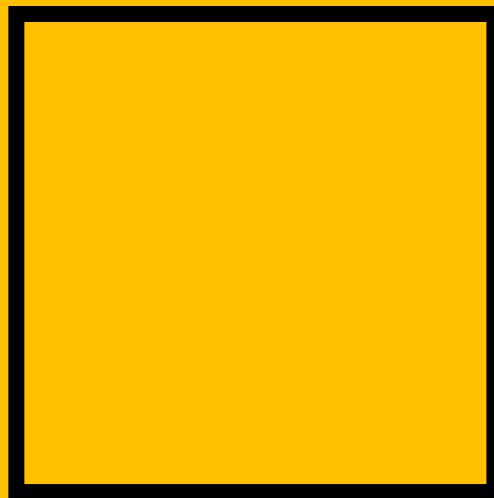




WD



STAGING



LOCAL



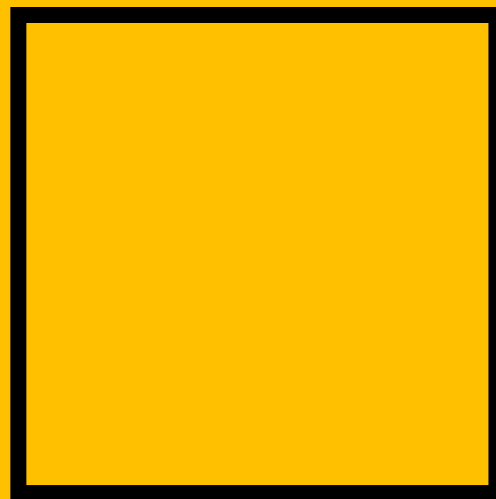
REMOTE

add



WD

STAGING

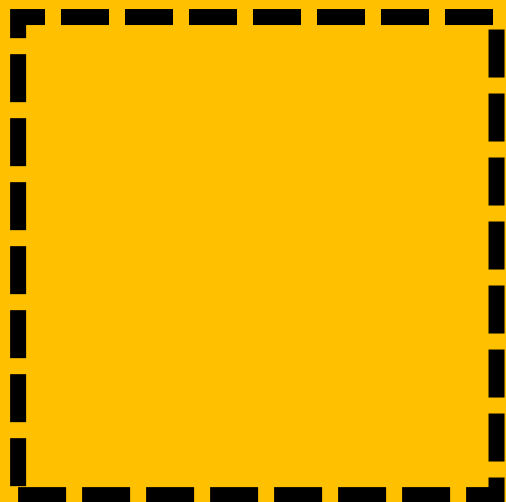


LOCAL



REMOTE

`commit`



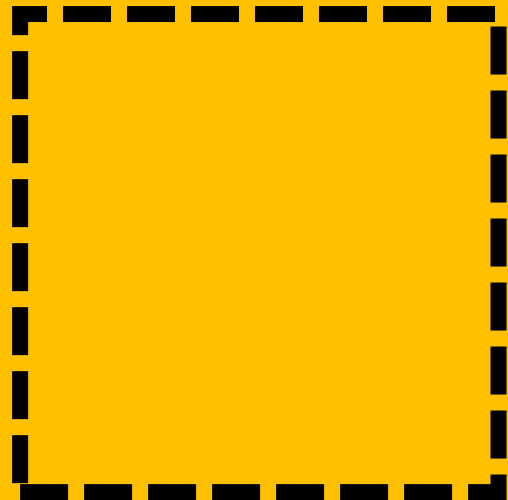
WD

STAGING

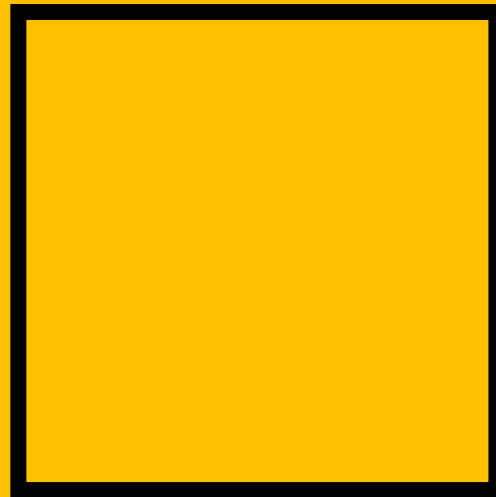
LOCAL

REMOTE

WD



STAGING



LOCAL



REMOTE

push

git init

creates the repo

git clone

clones a remote repo

git status

status of the repo

git diff

compares with last commit

git add <file>

add file to staging area

git add -A

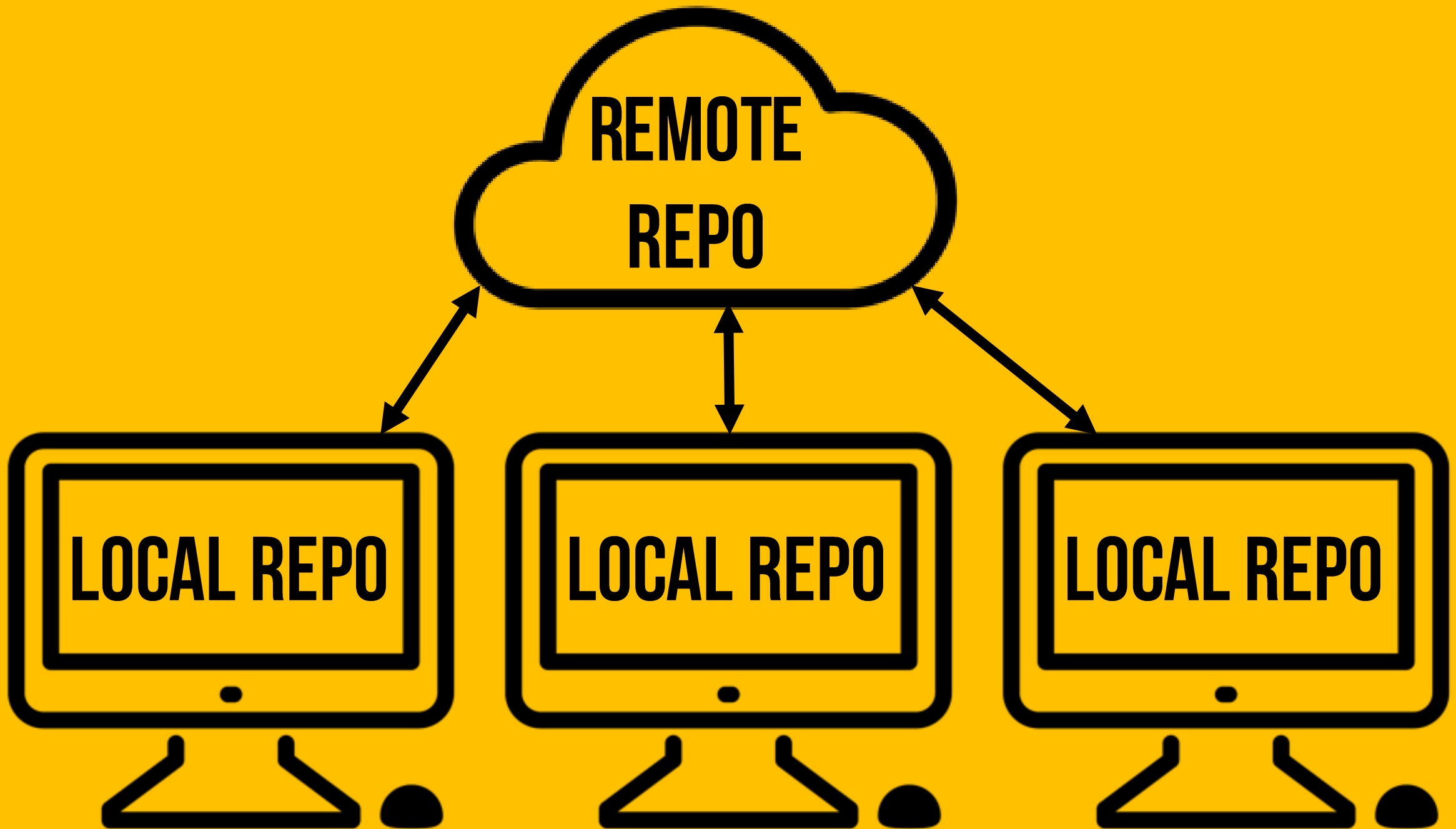
add all to staging area

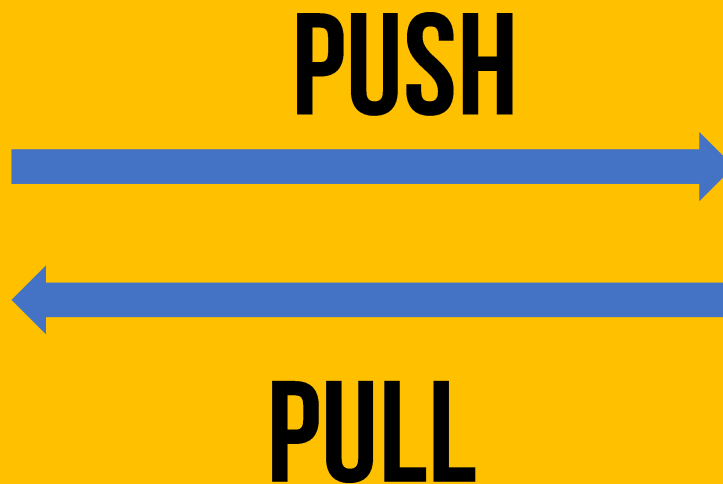
git commit -m "<message>"

commit changes to repo

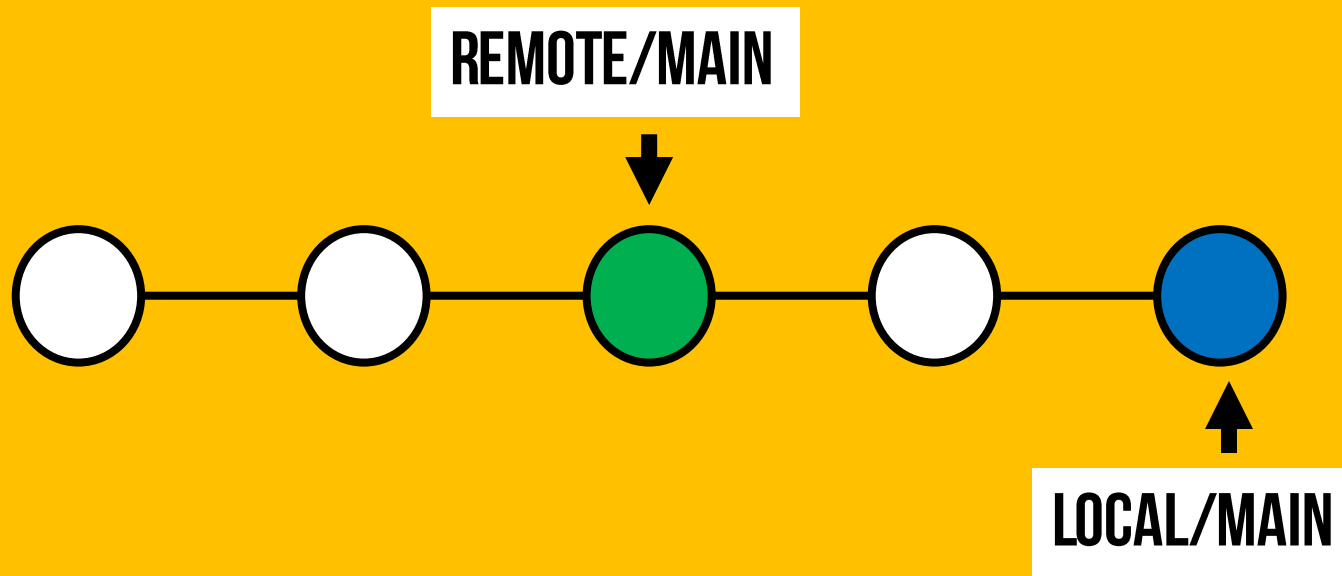
DEMO

set up local and remote repos
make some commits, push to remote

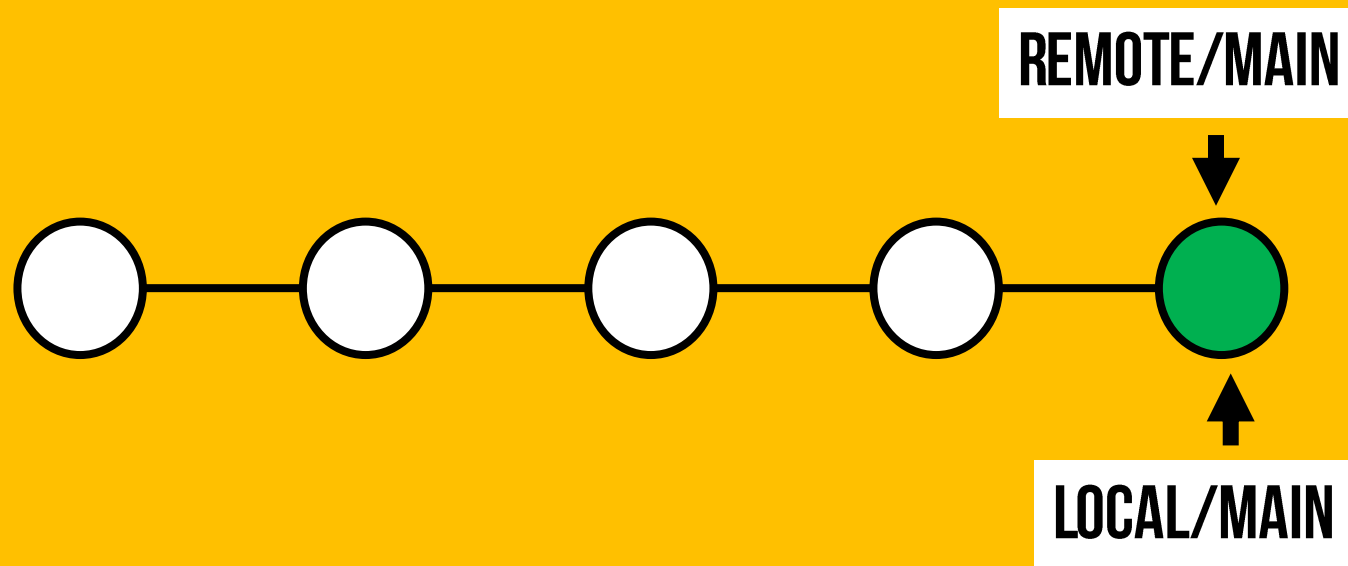





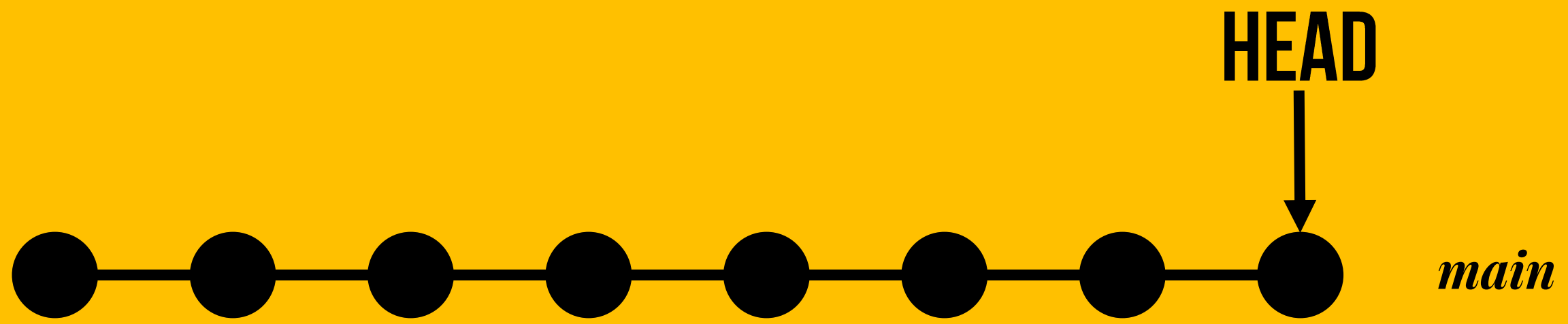
Before push:



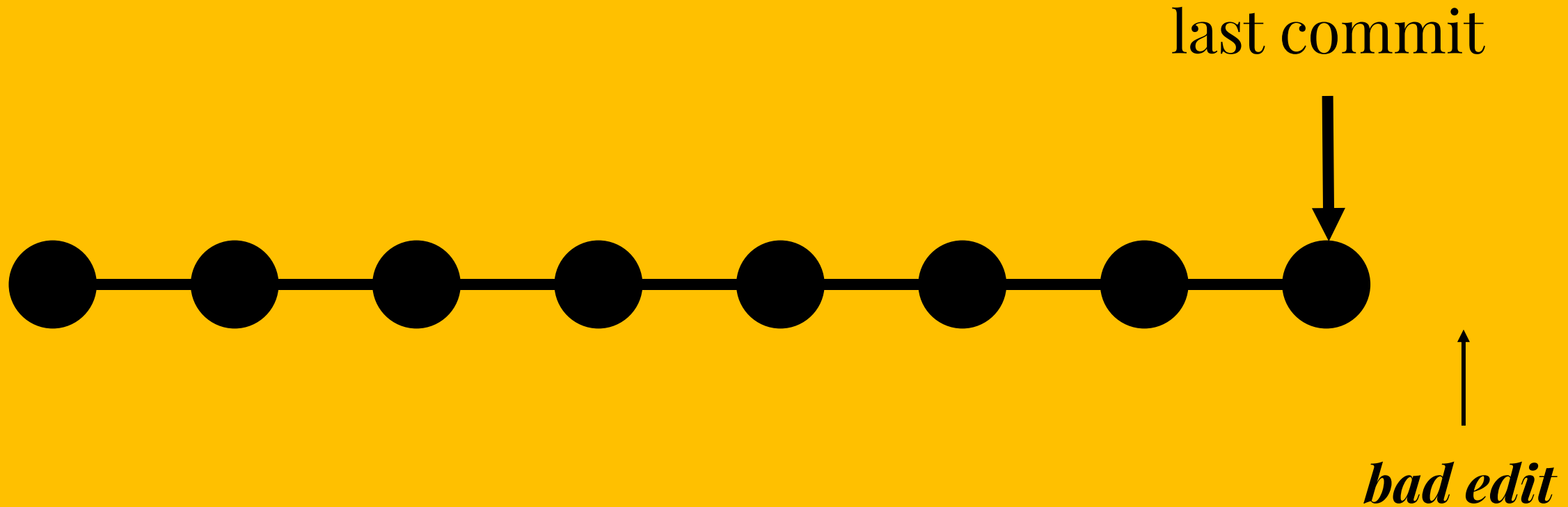
After push:



CHECKOUT  **git**



checkout: to retrieve a lost/broken file



```
rm file1
```

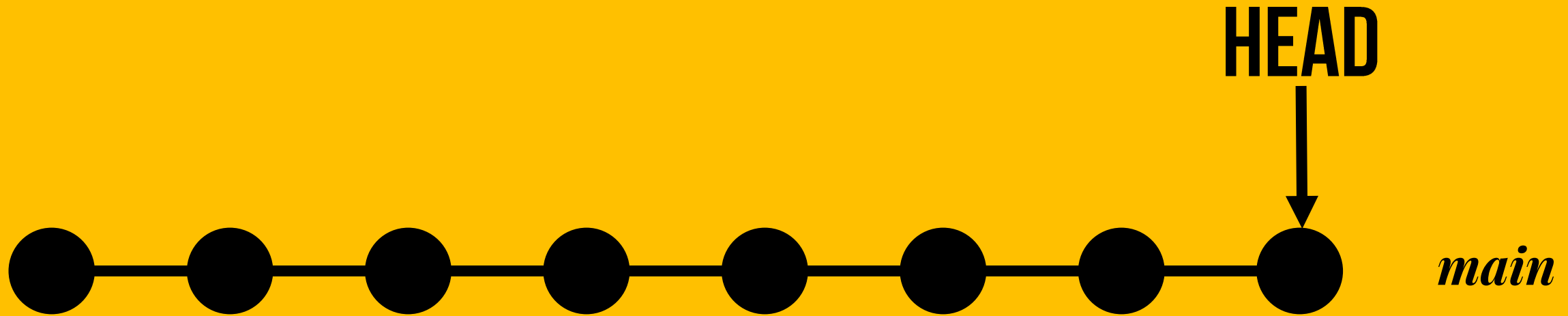
oops!

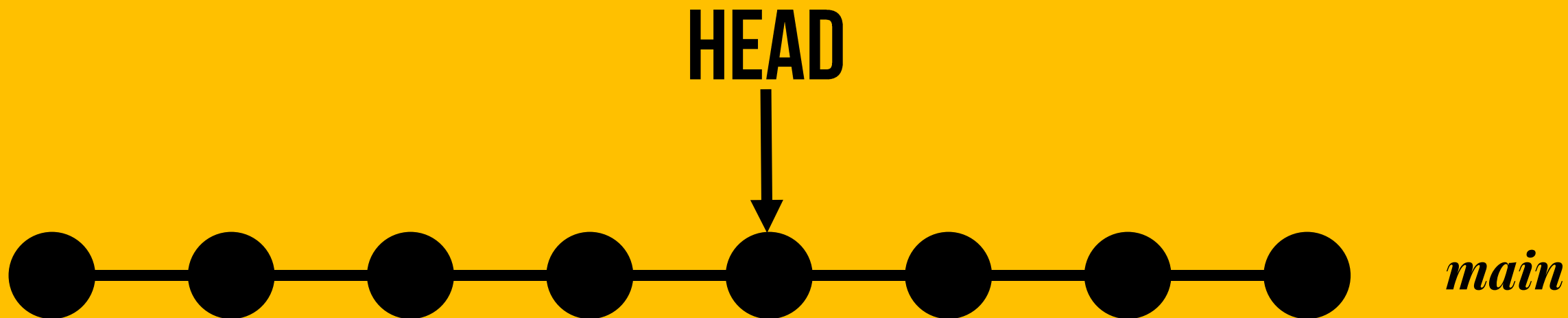
```
git checkout file1
```

retrieves from last commit

demo

checkout: to look at an earlier snapshot





(detached head)


```
git log
```

shows the commit history

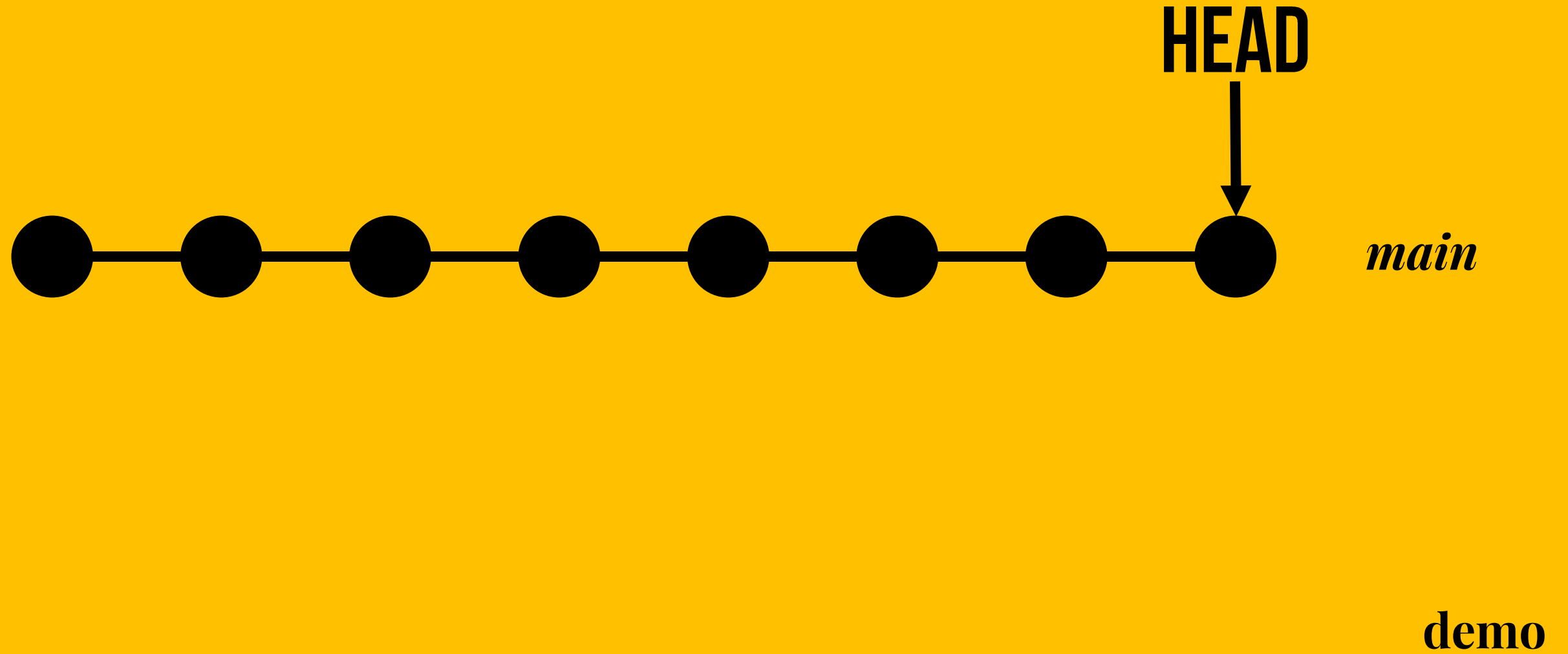
```
git checkout <commit_id>
```

moves head to that commit

temporarily look at that earlier snapshot ...

```
git checkout <branch>
```

back to 'current' commit

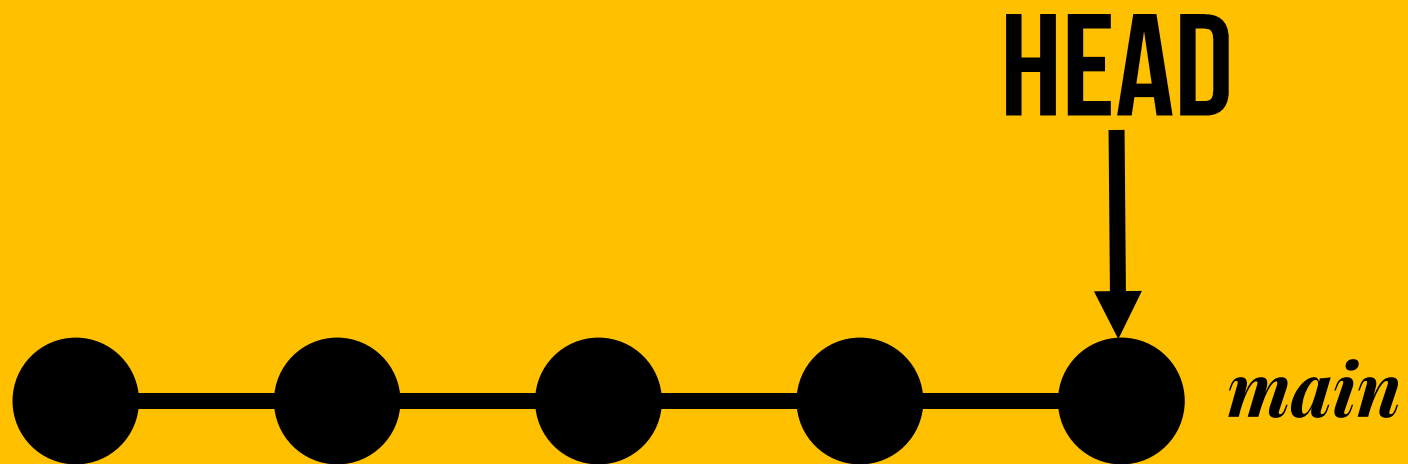




git

BRANCHES



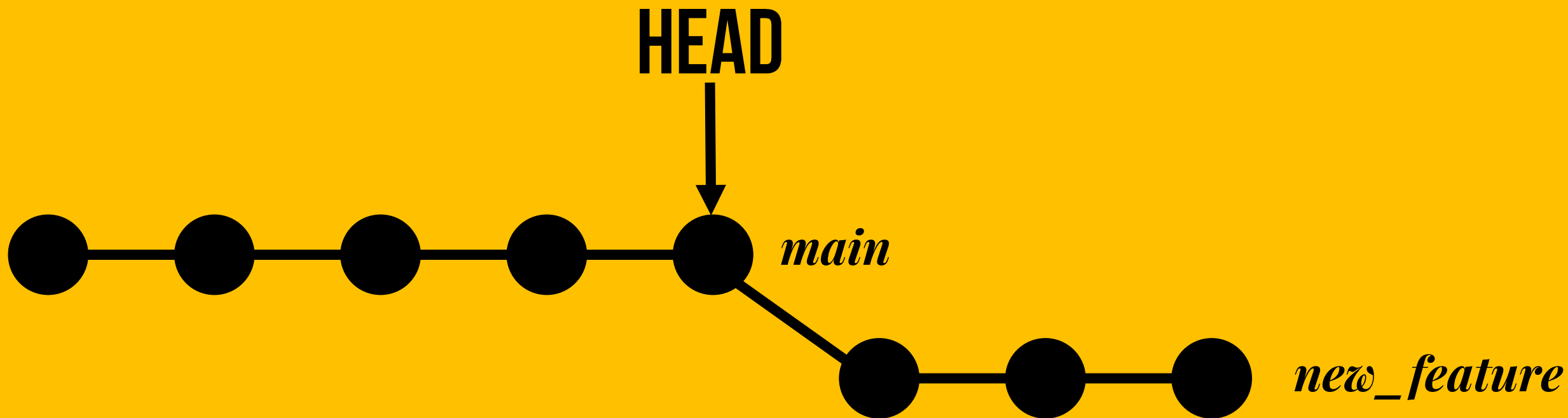


git branch

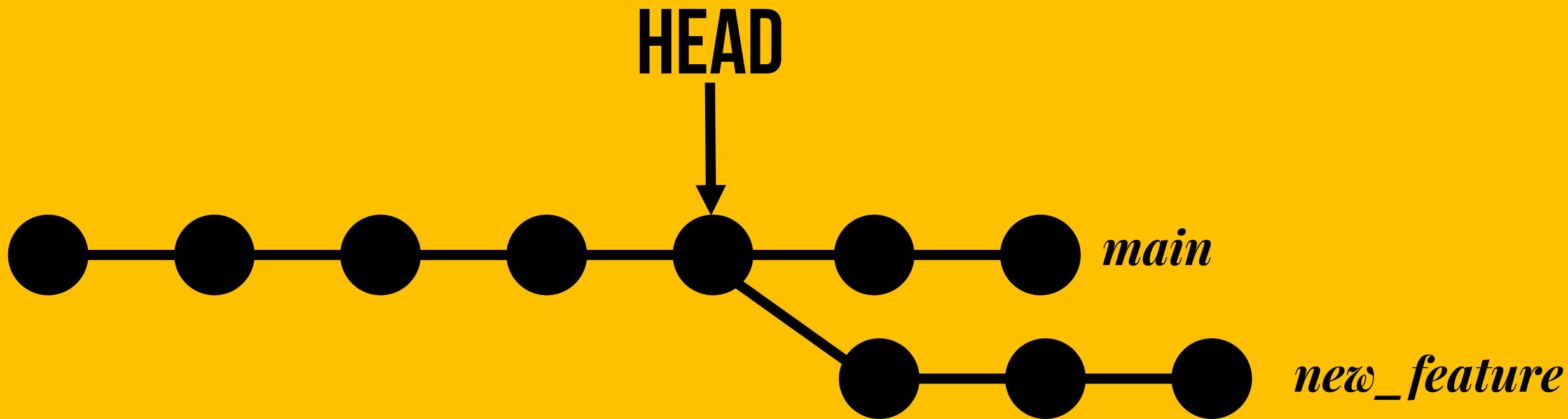
shows branches

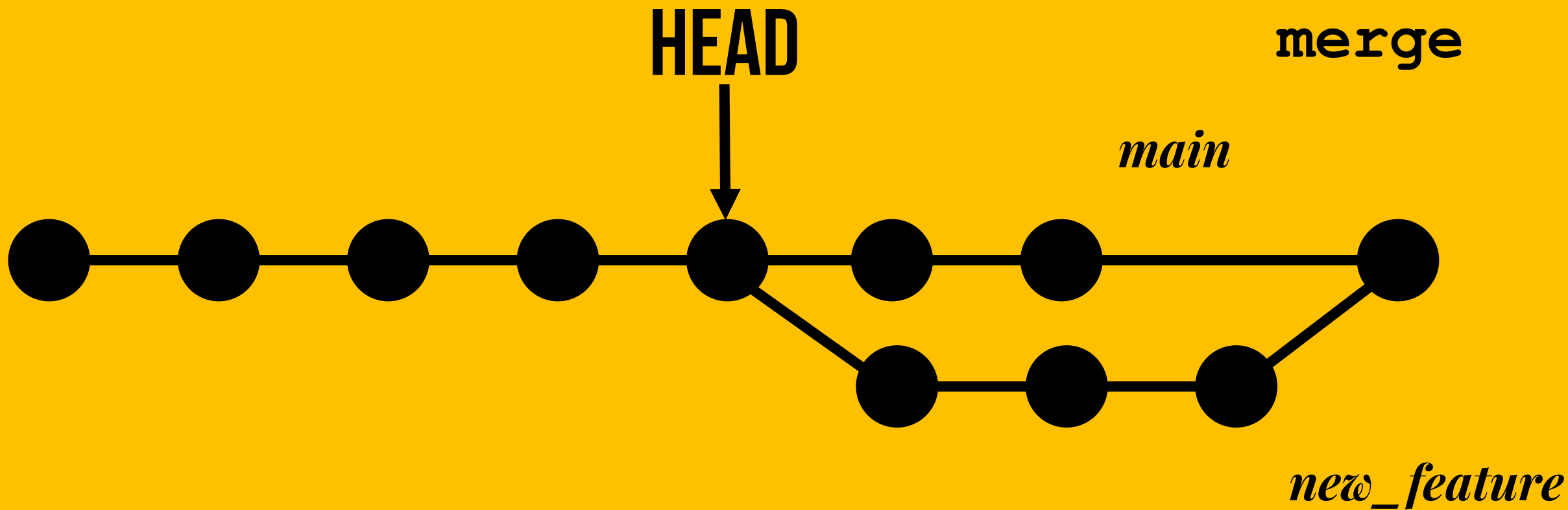
git checkout -b <branch>

create a new branch



demo





git branch

shows branches

git checkout -b <branch>

create a new branch

git merge <branch>

merges with branch

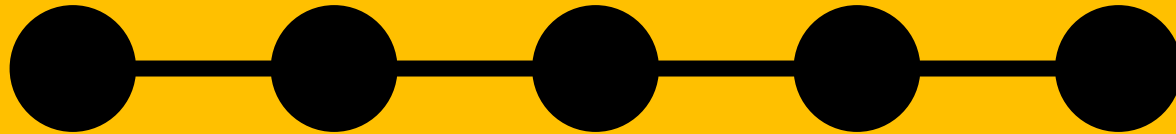
git branch -d <branch>

deletes branch

demo

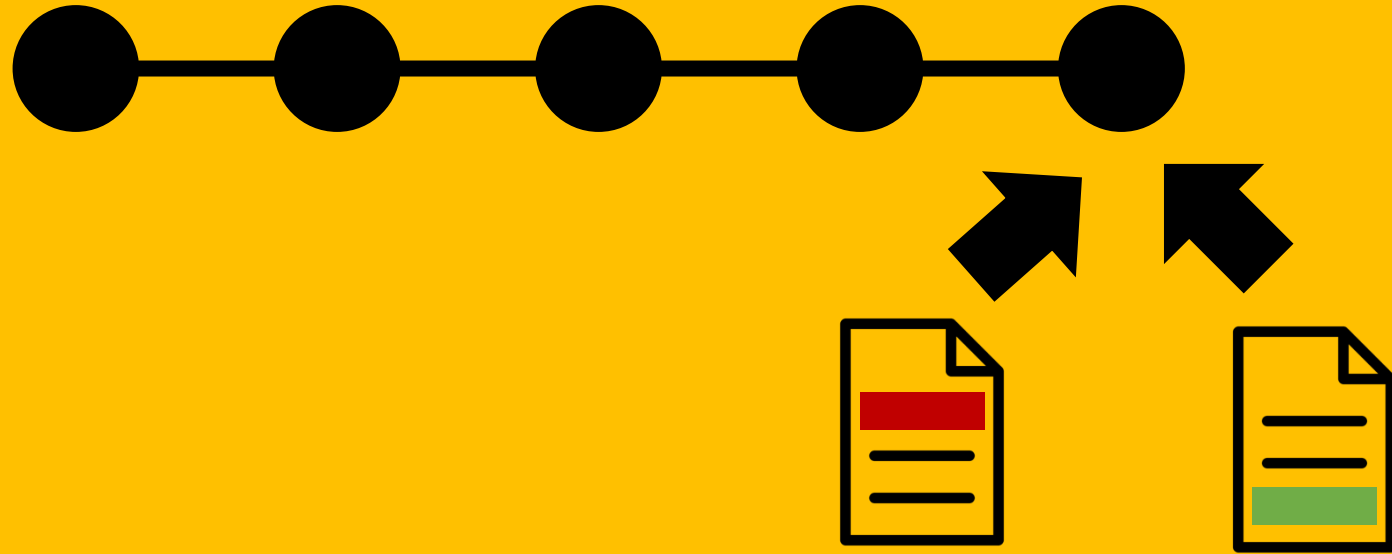
BEST PRACTICES

1) Commit often



but ...

2) Commit related changes together



3) Commit completed work



4) Commit with meaningful messages



4.1) *Commit with meaningful messages*

- Use the imperative mood in the subject line
A properly formed Git commit subject line should always be able to complete the following sentence:
If applied, this commit will *your subject line here*
For example:
If applied, this commit will *refactor subsystem X for readability*
If applied, this commit will *update getting started documentation*
If applied, this commit will *remove deprecated methods*

5) Branch before you build



5.1) *Naming Branches*



- Naming
 - The naming convention simply adds prefixes to branch names, so that branches of the same type get the same prefix
 - feature/username/CRnnnnn (JIRA issue key)
 - release/username/CRnnnnn (JIRA issue key, if applicable)
 - bugfix/username/CRnnnnn (JIRA issue key)
 - test/username/CRnnnnn (JIRA issue key)

This lets you search for branches in many git commands, like this:

```
git branch --list "feature/*"
```

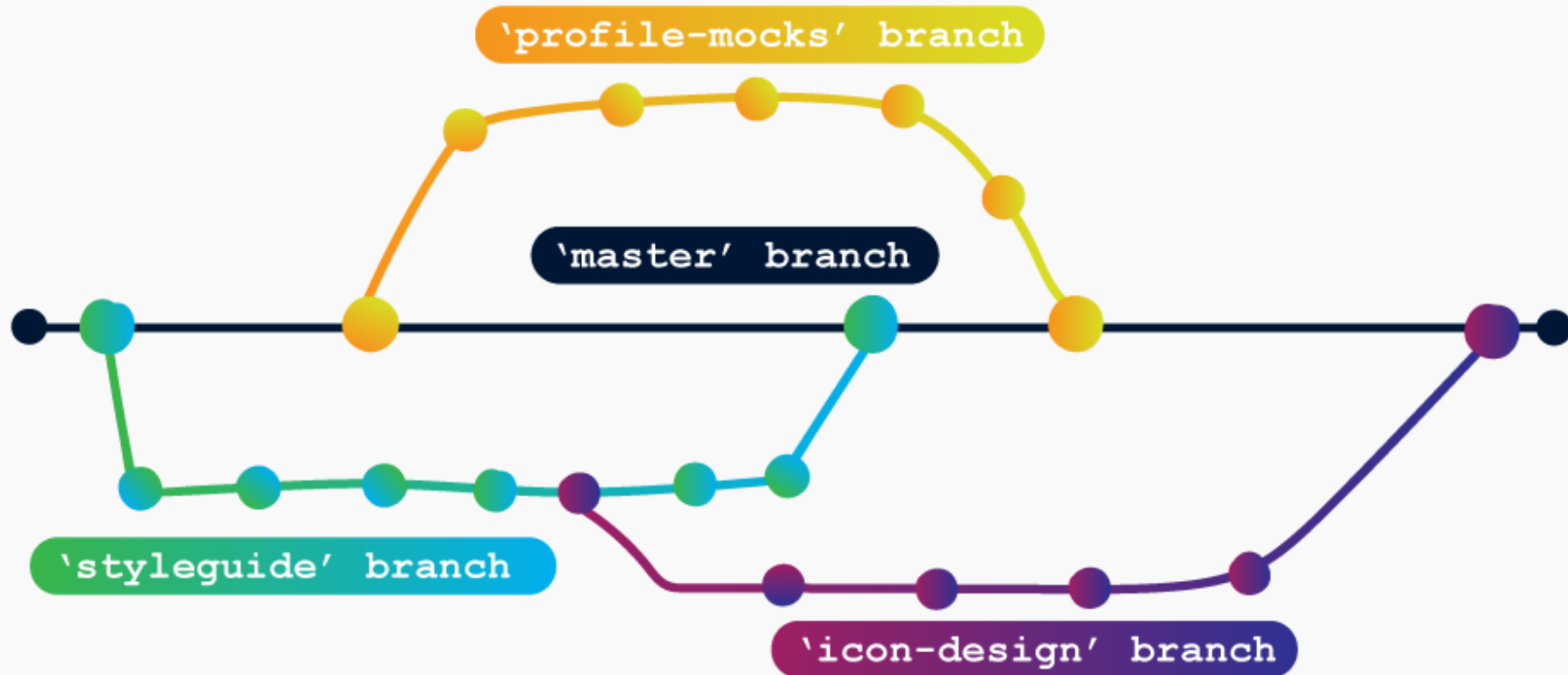
```
git log --graph --oneline --decorate --branches="feature/*"
```

```
git --branches="feature/*"
```

6) Take extra backups

- Local Git is not the same as a backup.
(What if your computer crashes?)
- Remote Github kind-of is. But other might change the repo.
- Recommend: make occasional backups outside of Git.

7) *Agree on a workflow*



8) Learn Git as a team

- Get together to learn git as a team
- Arrange your computers so everyone can see what everyone is doing
- Set up the team's remote repository
- Each team members clones it locally
- Practice changing files, commit, push, pull, branch, merge etc
- This will help you learn Git and agree on your team's workflow