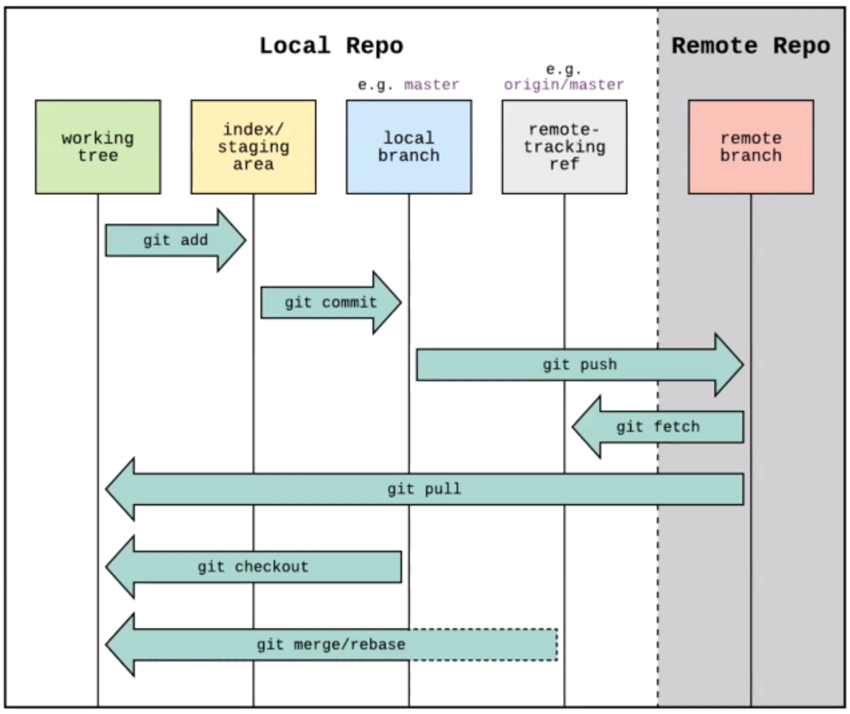
גיט עם Eli Sorotzkin מאיביי

יום חמישי 18 נובמבר 2021 09:04

Git allows us to work on code and create many save points, so we can restore to previous versions.
Git is not the only tool for that..

Whats the workflow?
Create file > add > commit > push and again.

The local is where you edit your files.
Staging: telling github we want to add to the remote repo



Commit > התחייבות > its still local.
Only after push it went to remote and acceible from other pcs/

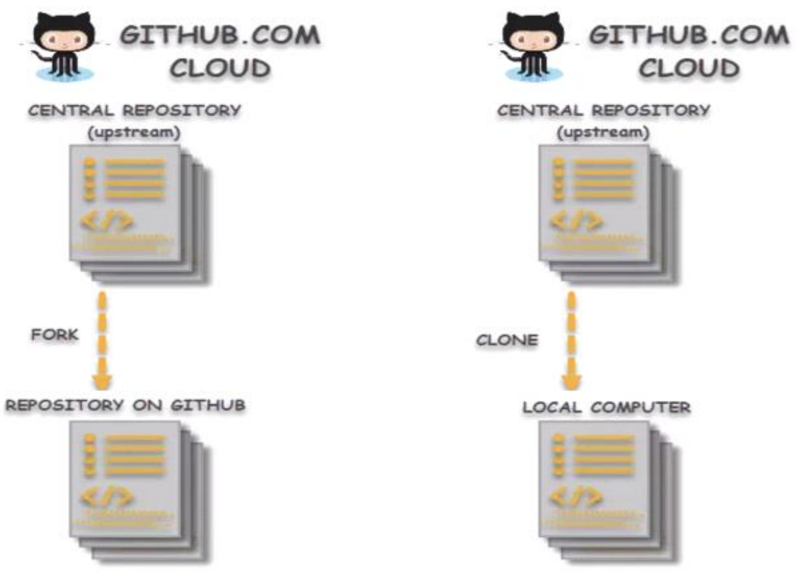
Fetch > get branch to you, check whats different wihtout changing the local code..
Fork >
Create another repo which is a copy. A friend for example. > you copy their code without having their password and access.
Clone is the first time you put the entire repo in your local.

How to pull?
If we want to know which branches > fetch
If we want to move between local branchs > check out
Pull
Take all the snapshots of te remote to your local, inc. brances, and merge them automatically ??
After your'e done working > push.
If you want to continue working, and the code has been updated, you pull again and start working.

Pull is instead of two things (fetch + merge ?)

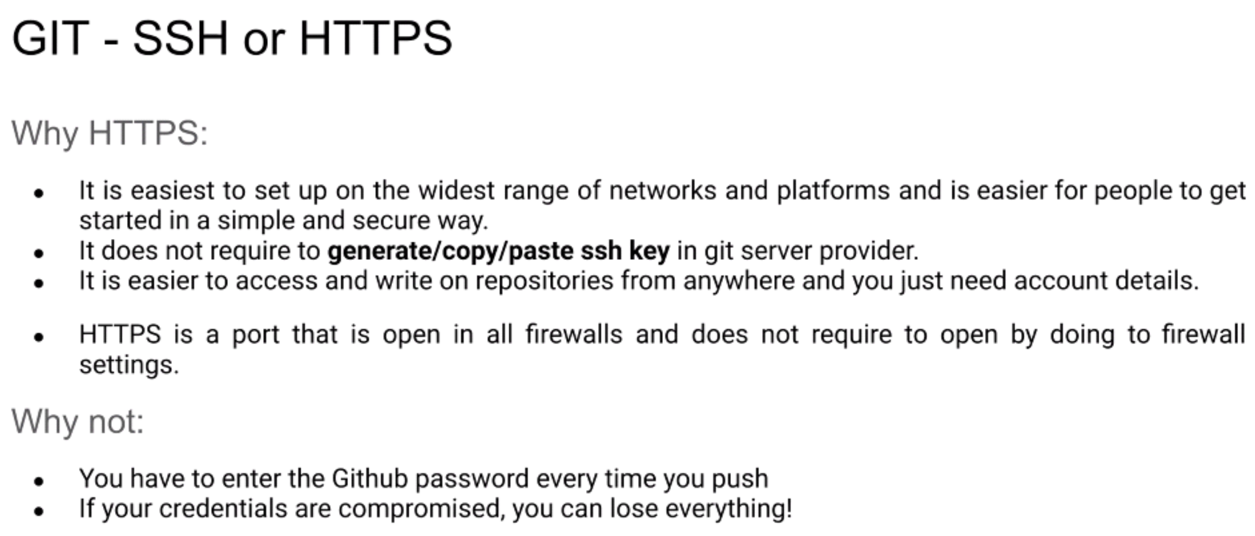
Clone / fork

GIT clone/fork



גזירת מסך נלכדה ב: 18/11/2021 09:08

Repo of other pple we want to work on > fork. Complete copy but still on remote. Has all files.
Clone > take remote repo to local.
If we want to work on other ppl code > fork > then clone



GIT - SSH or HTTPS

Why HTTPS:

- It is easiest to set up on the widest range of networks and platforms and is easier for people to get started in a simple and secure way.
- It does not require to **generate/copy/paste ssh key** in git server provider.
- It is easier to access and write on repositories from anywhere and you just need account details.
- HTTPS is a port that is open in all firewalls and does not require to open by doing to firewall settings.

Why not:

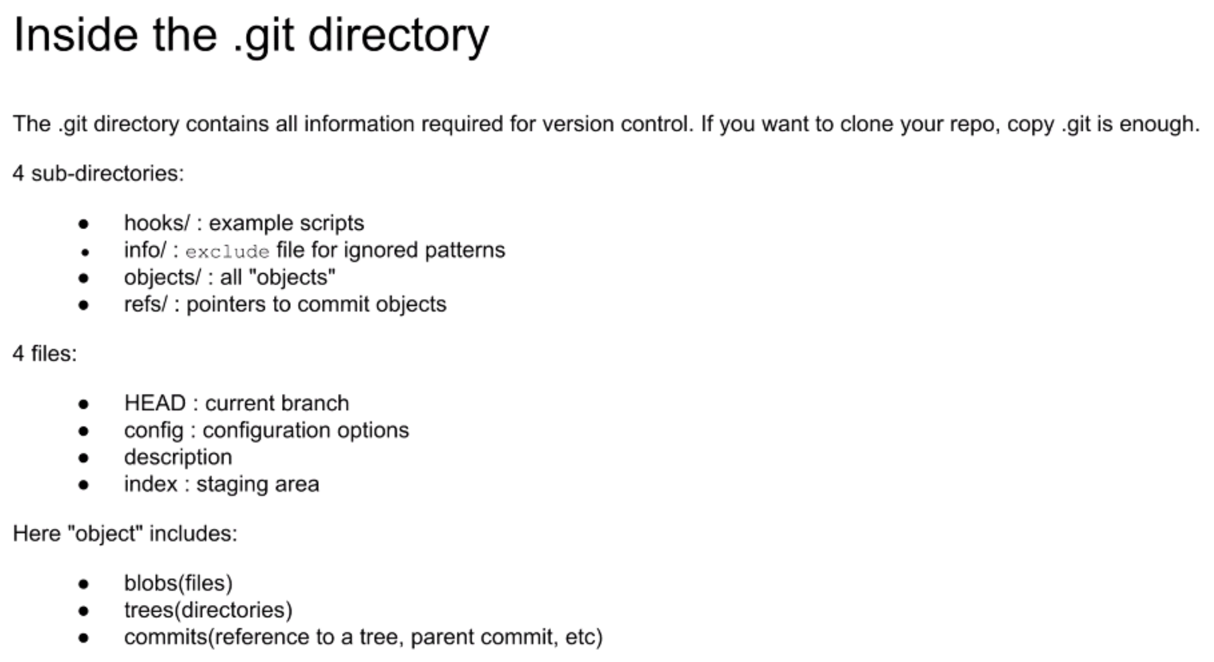
- You have to enter the Github password every time you push
- If your credentials are compromised, you can lose everything!

גזירת מסך נלכדה ב: 18/11/2021 09:09

Https/ SSH are protocols to access stuff with the browser. Http is unsecure, https is secure. SSH is different, allows more secure but requires setup/ definitions.

Why use http?
You just need usename and passwor. Easy to work from all pcs. Open across firewalls. From everywhere.
Why not?
Need to enter usr+pass each time.
If we lose them, we can lose all the code.
Why SSH?
Much more secure.
Can share private key and other ppl can use it to push to your repo but cant change your account or delete stuff.

Why not SSH?
You need key for each use.



Inside the .git directory

The .git directory contains all information required for version control. If you want to clone your repo, copy .git is enough.

4 sub-directories:

- hooks/ : example scripts
- info/ : `exclude` file for ignored patterns
- objects/ : all "objects"
- refs/ : pointers to commit objects

4 files:

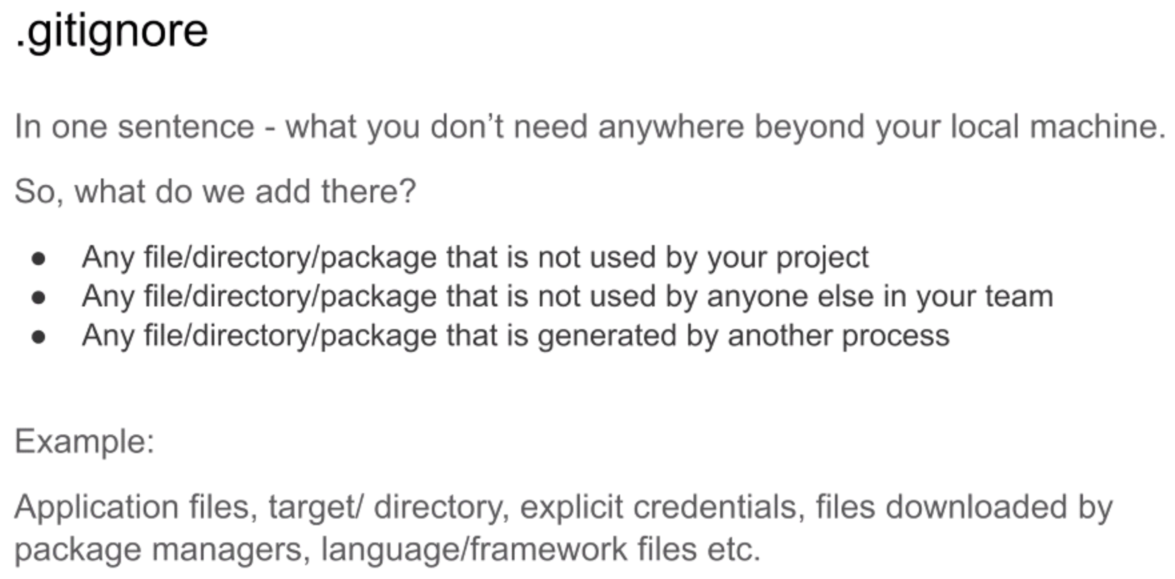
- HEAD : current branch
- config : configuration options
- description
- index : staging area

Here "object" includes:

- blobs(files)
- trees(directories)
- commits(reference to a tree, parent commit, etc)

גזירת מסך נלכדה ב: 18/11/2021 09:12

^ this is the structure inside the local git repo, just FIY.
Head : the branch were working on
Index: staging area



.gitignore

In one sentence - what you don't need anywhere beyond your local machine.

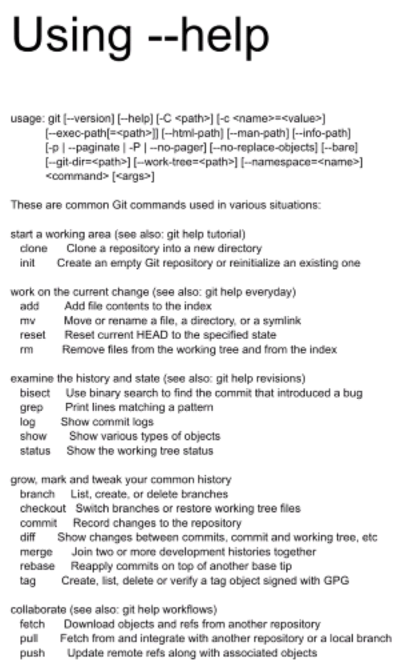
So, what do we add there?

- Any file/directory/package that is not used by your project
- Any file/directory/package that is not used by anyone else in your team
- Any file/directory/package that is generated by another process

Example:

Application files, target/ directory, explicit credentials, files downloaded by package managers, language/framework files etc.

Not every code we want to share.
This is for files we only want locally like passwords/ credentials, config for local pc, stuff only relevant for me, or files needed to run locally (i.e files created by intelliJ).
Files that are created as output and no need to track.
Fukes that are needed just to run locally.

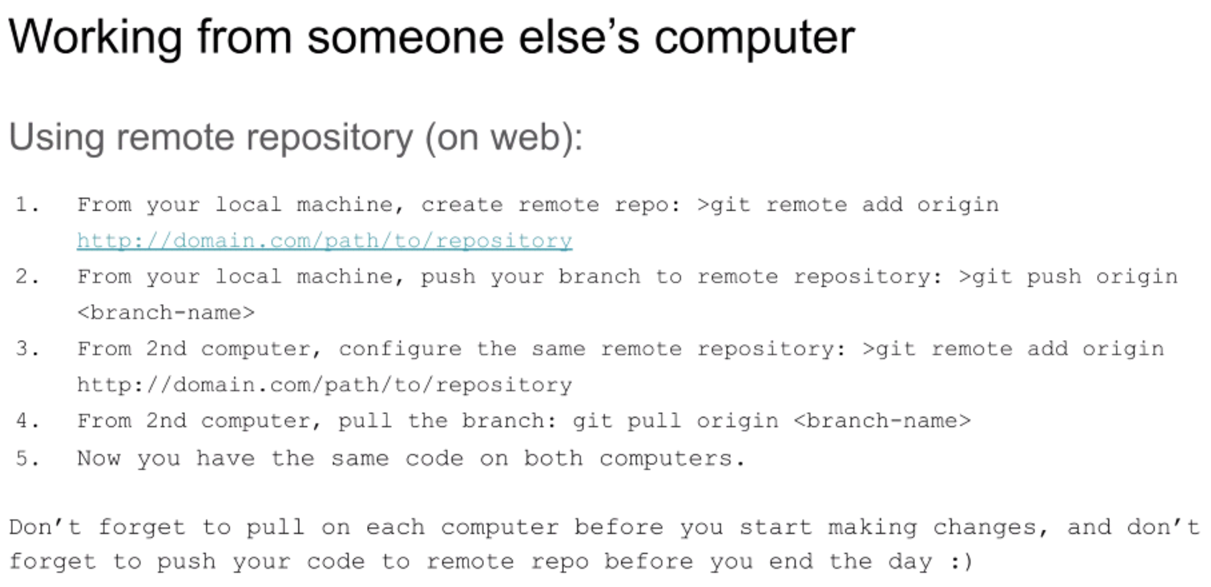


Using --help

```
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   branch     List, create, or delete branches
   checkout   Switch branches or restore working tree files
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   merge      Join two or more development histories together
   rebase     Reapply commits on top of another base tip
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects
```

גזירת מסך נלכדה ב: 18/11/2021 09:22

Can always use git --help or action name --help > gets you all the info.
All the different flags you can use etc.



Working from someone else's computer

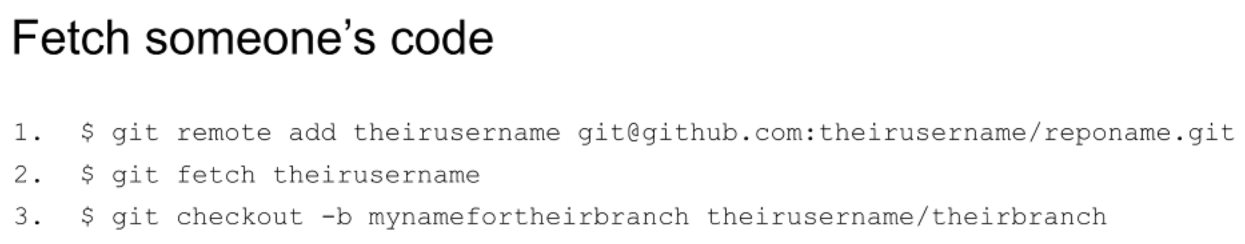
Using remote repository (on web):

1. From your local machine, create remote repo: >git remote add origin http://domain.com/path/to/repository
2. From your local machine, push your branch to remote repository: >git push origin <branch-name>
3. From 2nd computer, configure the same remote repository: >git remote add origin http://domain.com/path/to/repository
4. From 2nd computer, pull the branch: git pull origin <branch-name>
5. Now you have the same code on both computers.

Don't forget to pull on each computer before you start making changes, and don't forget to push your code to remote repo before you end the day :)

גזירת מסך נלכדה ב: 18/11/2021 09:23

Working with remote repo - from other pc.
From 2nd pc, git remote add and git pull > now we have 3 copies: cloud, 1pc, 2pc.
Before switching pcs, esp. when working with other ppl, PULL!
Finished working on pc? Push! Will help you save your code.
When working on solo code, you can do it a lot of times.

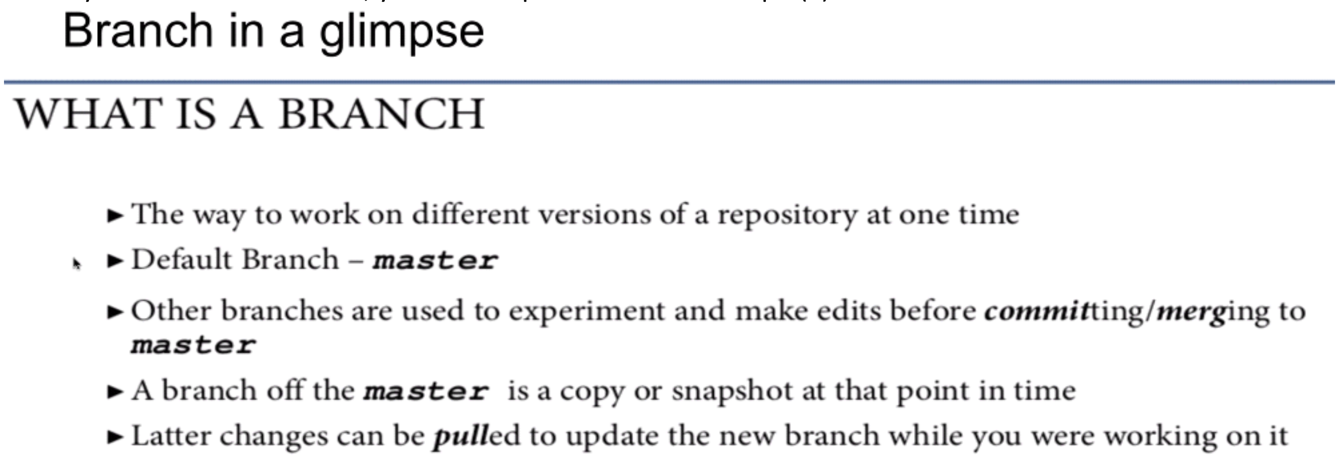


Fetch someone's code

1. $ git remote add theirusername git@github.com:theirusername/reponame.git
2. $ git fetch theirusername
3. $ git checkout -b mynamefortheirbranch theirusername/theirbranch

גזירת מסך נלכדה ב: 18/11/2021 09:26

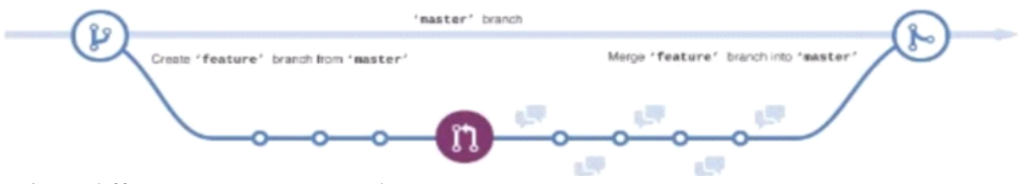
Create an access in your local to remote repo using git remote
When you use their username, you wil be updated with their repo (?)



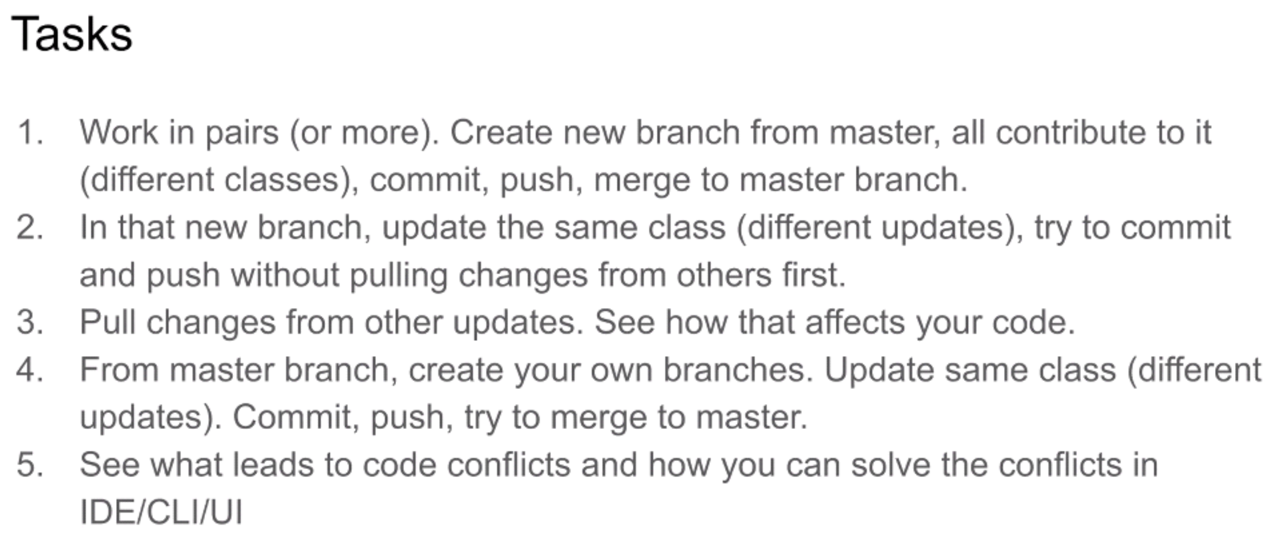
Branch in a glimpse

WHAT IS A BRANCH

- ► The way to work on different versions of a repository at one time
- ► Default Branch – ***master***
- ► Other branches are used to experiment and make edits before ***commit***ting/***merg***ing to ***master***
- ► A branch off the ***master*** is a copy or snapshot at that point in time
- ► Latter changes can be ***pull***ed to update the new branch while you were working on it


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'

Branch helps work on different versions on code.
Branch is another תצורה of the code, the branch is a copy that has changes (?)
The default branch is master/main. There the branch has to go when finished working on the version.
The branch allows us to work on the version before and while its not ready to join the master.
As long as the branch isnt deleted you can access its different stanges("saves")



Tasks

1. Work in pairs (or more). Create new branch from master, all contribute to it (different classes), commit, push, merge to master branch.
2. In that new branch, update the same class (different updates), try to commit and push without pulling changes from others first.
3. Pull changes from other updates. See how that affects your code.
4. From master branch, create your own branches. Update same class (different updates). Commit, push, try to merge to master.
5. See what leads to code conflicts and how you can solve the conflicts in IDE/CLI/UI

Try in couples to work on the same branch but on different files. Does it create problems?
What happens when 2 ppl update the same class?
Try commit + push, what does git tell you?
What happens when you try to revert to old code?
What happens when you update the same file on different branches - can you merge it all back?

Solving conflicts on github takes a lot of time!! And developers hate it.
***
Q&A
****
Connecting to a DB is not by usr+pass, needs to be more secure! you can only save the hash.
Another way is for the system to generate OTP inside your code. For secure connection.

Main > can be called also release / develop / whatever.

is it possible to create brance to a brach which is not the main? Is it a good practice? Can be done, depends on the org. if its pair programming, you can go to feature branch and each one of the couple merges their changes to the branch.
Otherwise, stay as close as possible to main.

can more than one user check out/update the same file at the same time? Yes, no prob.

Can we push at the same time? Mostly, git will stop the second one and let them know they are not on the most recent code. Then you need to pull the most recent version and push back after updaing.
The solution is not git, but teamwork! And always working on the latest version.
If you develop a feature for a month and havent updated, there are probably many many changes..
There are tools that help manage conflict, show the differences so that you decide what to keep etc.
Sometimes you get a conflict and the code breaks. Make sure after pushing, that your code doesn't break everything!
How? Pull a fresh version, implement your changes, and run it to make sure all is well.

Git does not know if you override someone else's code!!

does git have different security levels? ex: team lead can only delete to avoid accidental loss by less experienced staff? Yes, there are admins. There are ppl that can push to main. And contributors can edit but not necessarily to main. That's good practice that makes sure a merge is done correctly.
Another way to protect is to use fork. You cant push - have to make pull request to the original coder, he has to review your code. That is recommended but takes longer.

If you want to protect your branch from changes, you can create branch protection rules.
You can require pull reqs to be reviewd beforehand. You can require status checks to make sure the new code was based on updated version.
And many other definitons.
You can use webhooks and connect other systems like jenkins.

How many times you pushed code > sometimes is accounted as your **contribution** to the company.
Don't save your code to yourself! End of day > git push

Commit msg > descibe what you did!! Not which classes you added (ppl can see that) but WHY.
Whats the task you accomplished. Why you commited it? Couple of lines.

Corp. vs startup
Crop: work correctly, review. Strong code and processes. Tech updates slow. Decisions are made mostly outside the team.
Startup: newest techs, work fast not always best processes. More room for decisions.

Getting a job
- Write clean code with good var names, if it explains itself, no need for comments
- Buzzwords on CVwill get you past HR but if you cant explain it, tech interview will fail you
- Research the company (glassdoor tec.)
- Commit a lot
- Show how you approach a task: plan ahead, which funcs will u need? Don't be afraid to ask if you don't understand anything.
- Talk about langs you know.
- Contribute to open source