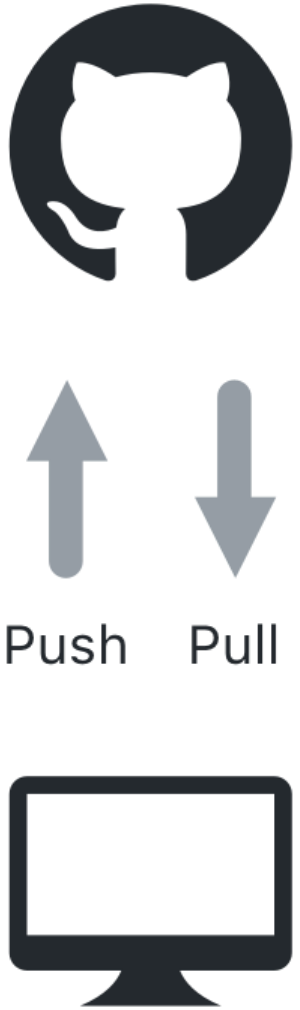
GIT

Introduction

- git: an open source, distributed version-control system
  - Repository for storing code changes
  - Track History / Centralize / Distribute
- GitHub: a platform for hosting and collaborating on Git repositories



Why?

Working by yourself:

Gives you a “time machine” for going back to earlier versions

Gives you great support for different versions of the same basic project

Working with others:

Greatly simplifies concurrent work, merging changes

Centralize

Distribute

CMD / TERMINAL

- cd <path>
- dir
- mkdir <dir>
- echo <text> > <filename.ext>
- explorer .
- pycharm .
- code .

create a new repository

create a new directory, open it and perform a

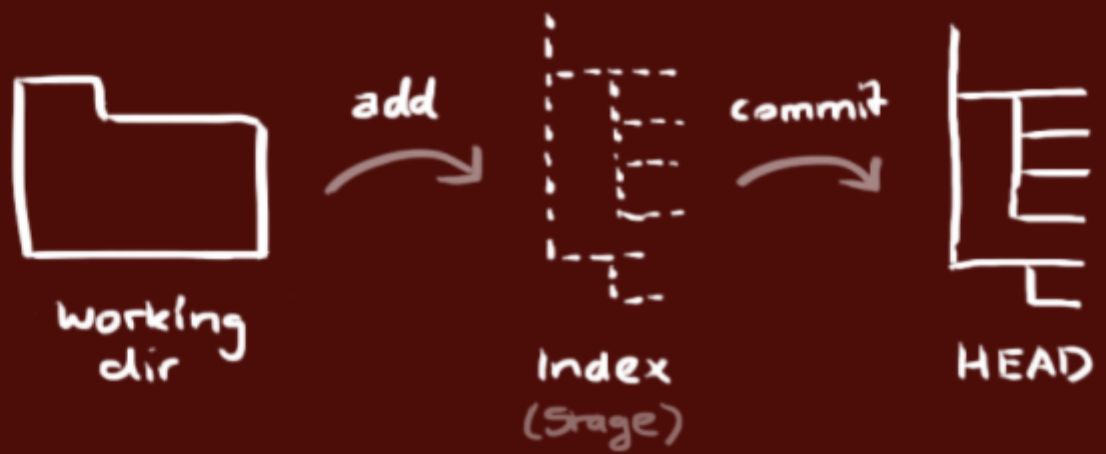
```
git init
```

to create a new git repository.

A repo Is a directory or storage space where your projects can live

workflow

your local repository consists of three "trees" maintained by git. the first one is your `Working Directory` which holds the actual files. the second one is the `Index` which acts as a staging area and finally the `HEAD` which points to the last commit you've made.



add & commit

You can propose changes (add it to the **Index**) using

```
git add <filename>
```

```
git add *
```

This is the first step in the basic git workflow. To actually commit these changes use

```
git commit -m "Commit message"
```

Now the file is committed to the **HEAD**, but not in your remote repository yet.

pushing changes

Your changes are now in the **HEAD** of your local working copy. To send those changes to your remote repository, execute

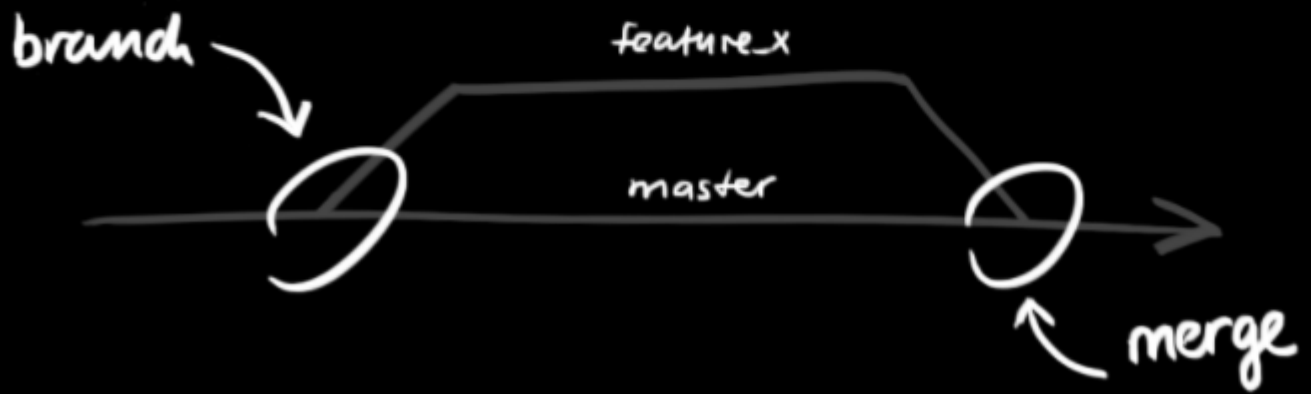
```
git push origin master
```

Change *master* to whatever branch you want to push your changes to.

If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with git remote add origin <server>

branching

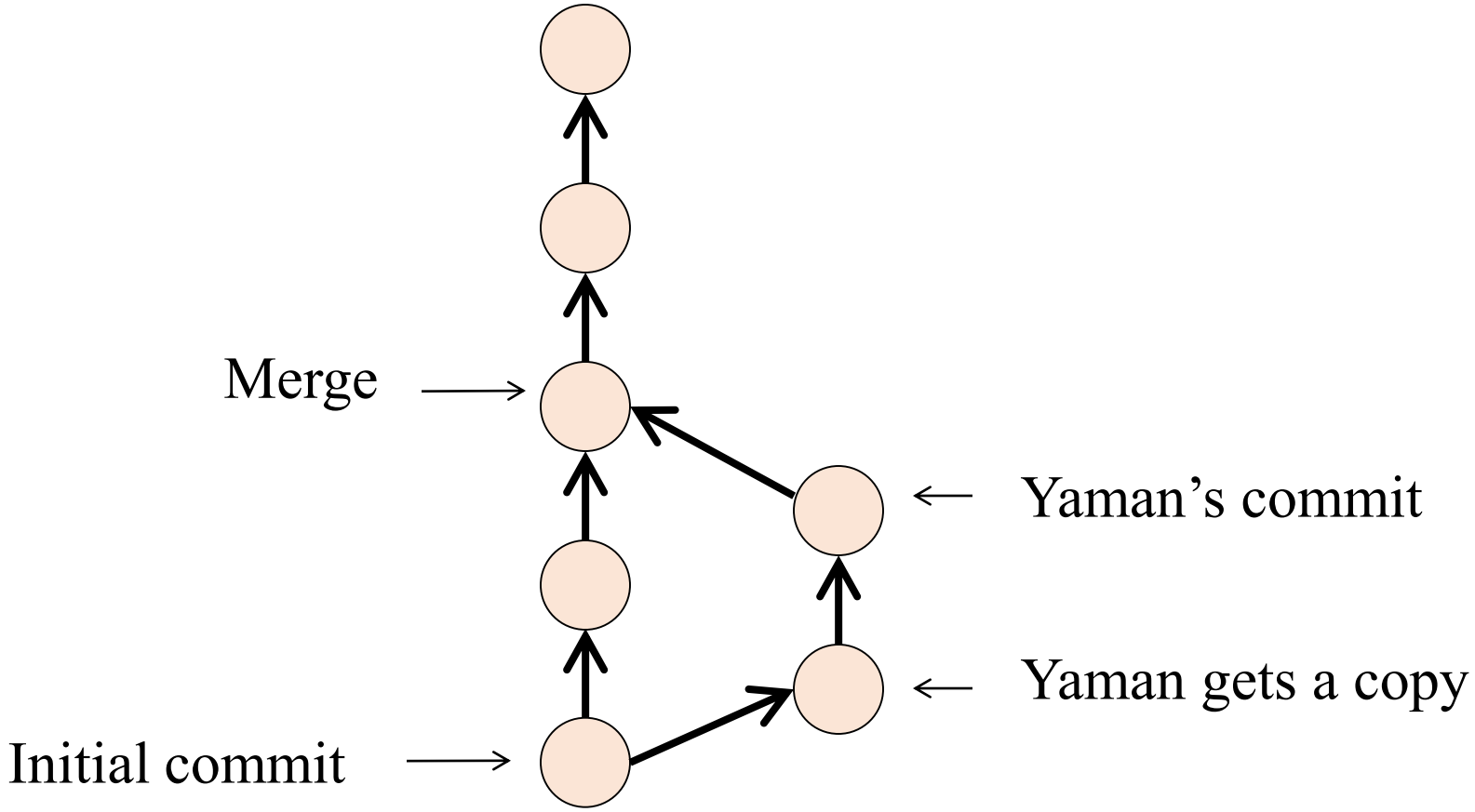
Branches are used to develop features isolated from each other. The *master* branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.



Create a new branch

- Say you want to make a new feature but are worried about making changes to the main project while developing the feature. This is where git branches come in.

- Branches allow you to move back and forth between 'states' of a project.
- Official git docs describe branches this way: ‘A branch in Git is simply a lightweight movable pointer to one of these commits.’
  - For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project.
  - Once you're done with the page, you can merge your changes from your branch into the primary branch.
  - When you create a new branch, Git keeps track of which commit your branch 'branched' off of, so it knows the history behind all the files.


Merge
Yaman's commit
Yaman gets a copy
Initial commit

- Let's say you are on the primary branch and want to create a new branch to develop your app.
- Here's what you'll do:
  - Run git checkout -b <my branch name>.
  - This command will automatically create a new branch
  - and then 'check you out' on it, meaning git will move you to that branch.

- Now we'll push the commit in your branch to your new GitHub repo.
  - This allows other people to see the changes you've made. If they're approved by the repository's owner, the changes can then be merged into the primary branch.

- To push changes onto a new branch on GitHub, you'll want to run git push origin yourbranchname.
  - GitHub will automatically create the branch for you on the remote repository

create a new branch named "feature_x" and switch to it using

```
git checkout -b feature_x
```

switch back to master

```
git checkout master
```

and delete the branch again

```
git branch -d feature_x
```

a branch is *not available to others* unless you push the branch to your remote repository

```
git push origin <branch>
```

update & merge

to update your local repository to the newest commit, execute

```
git pull
```

in your working directory to *fetch* and *merge* remote changes.

to merge another branch into your active branch (e.g. master), use

```
git merge <branch>
```

in both cases git tries to auto-merge changes. Unfortunately, this is not always possible and results in *conflicts*. You are responsible to merge those *conflicts* manually by editing the files shown by git. After changing, you need to mark them as merged with

```
git add <filename>
```

before merging changes, you can also preview them by using

```
git diff <source_branch> <target_branch>
```


GIT MERGE

log

in its simplest form, you can study repository history using.. `git log`

You can add a lot of parameters to make the log look like what you want.

To see only the commits of a certain author:

```
git log --author=yaman
```

GIT Exercise

- git clone <path>
- git status
- git add <file>
- git commit –m “msg”
- git push
- git config --global user.email
- git config --global user.name
- git merge -
- git reset --hard

Revision

- git: an open source, distributed version-control system
- GitHub: a platform for hosting and collaborating on Git repositories
- commit: a Git object, a snapshot of your entire repository compressed into a SHA
- branch: a lightweight movable pointer to a commit
- clone: a local version of a repository, including all commits and branches
- remote: a common repository on GitHub that all team members use to exchange their changes
- fork: a copy of a repository on GitHub owned by a different user
- HEAD: representing your current working directory, the HEAD pointer can be moved to different branches, tags, or commits when using git checkout

- Pull Requests are used to propose changes to the project files. A pull request introduces an action that addresses an Issue. A Pull Request is considered a "work in progress" until it is merged into the project

- Forking projects on GitHub
  - The process where we create our copy of someone else's project is called forking.

Resources

- https://gitexplorer.com/
- https://learngitbranching.js.org/
- https://www.w3schools.com/git/exercise.asp?filename=exercise_getstarted1