*** is there a flaw? Do I get a lot of history on a new clone?

If you start at a new company and join a team, that is already working on a projekt from some time, you want a kickstart. But there is a lot of information to gether before you can have a real impact. Besides the usual on binding meetings, where you learn how they work and what they do. You will soon facing the cloneing of a fresh and tasty git repository.

This might be the moment when you want to read the git log. Not the whole log but maybe from the last month or two. The logs are a powerfull feature of git, because you can see what has been done.

Checkout the following command:

```
git log --all --numstat --date=short --pretty=format:'--%h--%ad--%aN' --
no-renames --after=YYYY-MM-DD > logfile.log
```

It will print the log since the given Date `-after=YYYY-MM-DD` in a pretty format into a `logfile.log` file. e.G:

```
--4e9b57208--2019-10-02--foo
21  0   bar.java

--245e3631c--2019-10-01--kaa
42  6   mug.java

etc.
```

And this is the structure:

```
--Revision hash--date--developer
n-added n-deleted entity
```

So the information ranges from when has a commit been done over who did it to what changed.

That is very interesting for questions like: Which part in our codebase is changed frequently? or Which files are changed together often?

But if you are new to a team you may not be able to interpret the information.

Anyhow you may want to know what has recently happend in the repository. Compared with the hashes you can investigate into features recently developed and where to expect upcomming bugs. This may be a part of the code you want to spent some time to be able to follow the talks in your team if an issue is discussed.

You may even find a flaw which you might want to report or fix after communication because of uncercentry if it is a bug or a feature.

Whats next?

You checked the recent git history. Maybe followed some hashes and found the Jira (or whatever) ticket and read the story.

We started analyzing the git log, of the last months. The next step ist to get it for maybe a year into the past and add some magic to know whom to ask if an issue occurs in a specific module.

Reading a git log of a year is a lot of work. You need to assitance of a tool. Either you write your own which might be a nice kata.

Ot you can use code-maat a tool made to analyze git logs. It will reduce and agregates the informations into some sweet `*.csv` files which you can than funnel in your favourite table app (e.G. Exel or Numbers)

To get code-maat follow the installation explanations in its readme.md

You will notice code-maat was developed with the intention to do data driven decisions on refactoring(For more watch this talk) although it is possible to interpret the information differently.

For example:

The following command aggragates a table of files, number of authors and number of revisions or changes to this file.

```
java -jar code-maat-1.1-SNAPSHOT-standalone.jar -l logfile.log -c git2 >
first.csv
```

The n-authors are not in our interesst but the number of changes to the specific file. Finding the most changed files may hint to information about the hot spots of that repository. These are the files that may be important to know, because you may change them a lot too.

Another interesting aggregation can be gethered with:

```
java -jar code-maat-1.1-SNAPSHOT-standalone.jar -l logfile.log -c git2 -a
entity-effort > entity-effort.csv
```

This will create an table of entities (files), and the number of commits in total and for every author. By dividing the number of commits per author by the total amout you get a percentage which shows you who might have the most knowledge about that code entity. So if you have to work in a specific part and got stuck. You know who most likly will help you out the best.