Minu GIT (hõlmab eesti ja inglisekeelseid materjale)

Põhiraamat - https://git-scm.com/book/en/v2

PÕHIMÕTTED:

`git diff` in more detail later, but you'll probably use it most often to answer these two questions: What have you changed but not yet staged? And what have you staged that you are about to commit?

Kokkuvõtvalt delad käsud:

`git init`**:** Initializes a new Git repository. Until you run this command inside a repository or directory, it's just a regular folder. Only after you input this does it accept further Git commands.

`git config`**:** Short for "configure," this is most useful when you're setting up Git for the first time.

`git help`**:** Forgot a command? Type this into the command line to bring up the 21 most common git commands. You can also be more specific and type "git help init" or another term to figure out how to use and configure a specific git command.

`git status`**:** Check the status of your repository. See which files are inside it, which changes still need to be committed, and which branch of the repository you're currently working on.

`git add`**:** This does *not* add new files to your repository. Instead, it brings new files to Git's attention. After you add files, they're included in Git's "snapshots" of the repository.

`git commit`**:** Git's most important command. After you make any sort of change, you input this in order to take a "snapshot" of the repository. Usually it goes `git commit -m "Message here."` The `-m`indicates that the following section of the command should be read as a message.

`git branch`**:** Working with multiple collaborators and want to make changes on your own? This command will let you build a new branch, or timeline of commits, of changes and file additions that are completely your own. Your title goes after the command. If you wanted a new branch called "cats," you'd type `git branch cats`.

`git checkout`: Literally allows you to "check out" a repository that you are not currently inside. This is a navigational command that lets you move to the repository you want to check. You can use this command as `git checkout master` to look at the master branch, or `git checkout cats` to look at another branch.

`git merge`: When you're done working on a branch, you can merge your changes back to the master branch, which is visible to all collaborators. `git merge cats` would take all the changes you made to the "cats" branch and add them to the master.

`git push`: If you're working on your local computer, and want your commits to be visible online on GitHub as well, you "push" the changes up to GitHub with this command.

`git pull`: If you're working on your local computer and want the most up-to-date version of your repository to work with, you "pull" the changes down from GitHub with this command.

First time setup

https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

```
$ git config --global user.name "AndresNamm"
$ git config --global user.email Andres.namm.001@gmail.com
```

Mul veits teine käsk

```
$ git config --global core.editor "'C:/Program
Files/Notepad++/notepad++.exe' -multiInst -nosession"


$ git config --list
```

https://git-scm.com/book/en/v2/Getting-Started-Getting-Help

Abi – laeb html faili, seega päris okei lugeda

```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
```

https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

Git unstage part

```
git rm --cached README
```

Git untrack ...

git reset HEAD "Minu GIT.docx"

Git commit without doing the staging part beforehand

```
$ git commit -a -m 'added new benchmarks'
```

Kaevame nüüd natukene ajaloos

Commit history

```
$ git log
```

Commit history show diffs (-p) for only 2 (-2)

```
$ git log -p -2
$ git log --since=2.weeks
```

Pretty käsk on loodud formaatimiseks.

```
$ git log --pretty=format:"%h - %an, %ar : %s"
```

Otsime historyst

https://git-scm.com/book/en/v2/Git-Basics-Viewing-the-Commit-History

```
$ git log --pretty=format:"%h - %an, %ar : %s" -Spenis -p
```

For example, if you want to see which commits modifying test files in the Git source code history are merged and were committed by Junio Hamano in the month of October 2008, you can run something like this:

```
$ git log --pretty="%h - %s" --author=gitster --since="2008-10-01" \
--before="2008-11-01" --no-merges -- t/
```

Probleem1 Ma tahan nüüd leida vai failed, kus olen rohkem muutnud kui <n> rida koodi

Probleem2 Ma tahaks samuti olla võimeline saama spetsiifilist logi mingi kausta või mingi faili kohta. V.T

https://www.atlassian.com/git/tutorials/git-log/formatting-log-output

https://www.atlassian.com/git/tutorials/viewing-old-commits

https://www.atlassian.com/git/tutorials/inspecting-a-repository

```
$ git log --pretty=format:"%h - %an, %ar : %s" - -p
```

Undoing part 1

https://git-scm.com/book/en/v2/Git-Basics-Undoing-Things

Kui ma sooritan commity ja vahetult pärast SEDA tahan midagi väikest selles commitis parandada

```
$ git commit --amend
```

Kui ma tahan lihtsalt oma working directorys midagi asendada commiti failiga

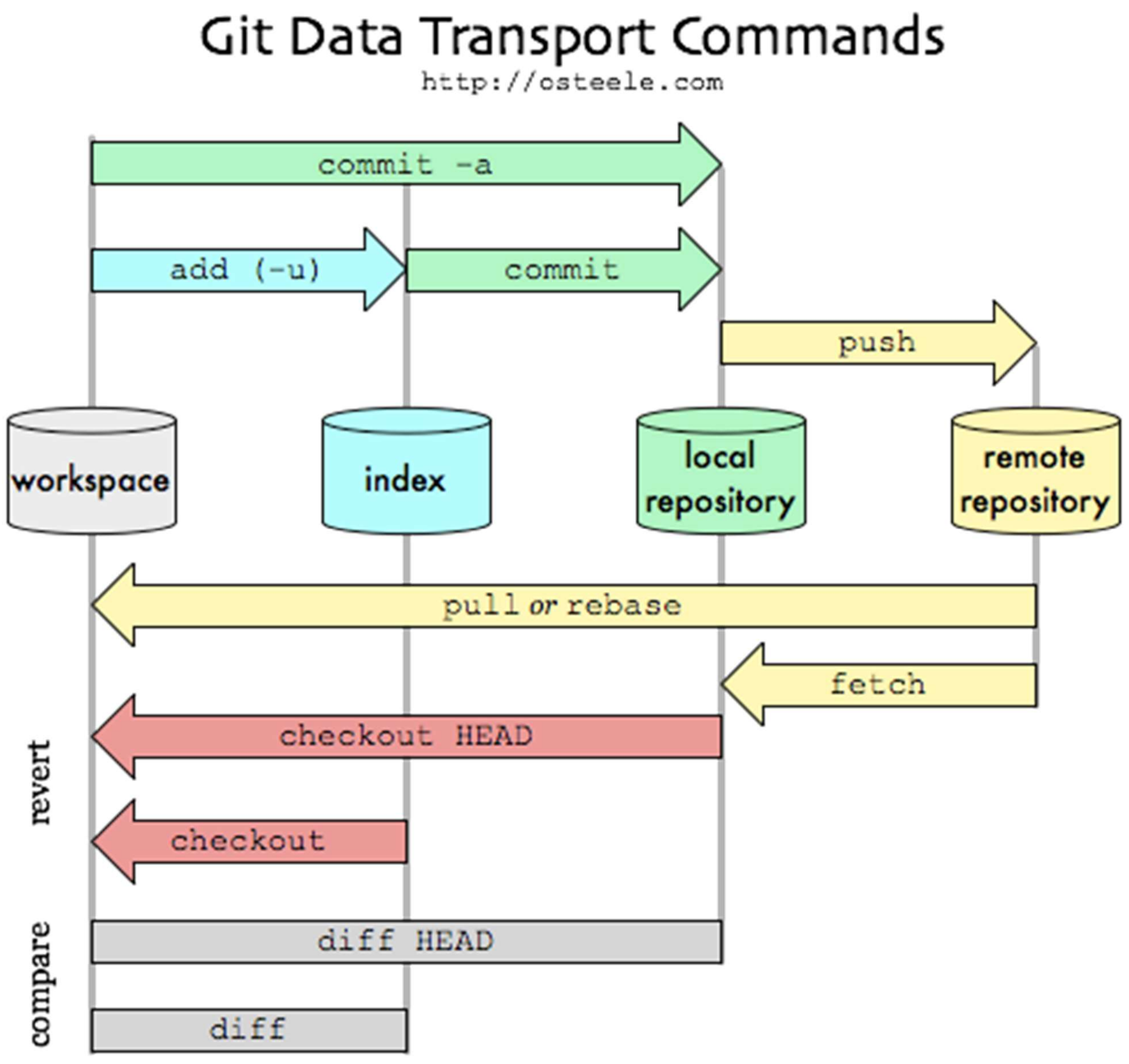
```
$ git checkout --<filename>
```

https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes

Teeme seda internetivärki

Esiteks üldine skeem:

g



Undoing things

1) HARJUTUS – UNDOING THINGS ::Tahan tõmmata alla mingi vanema commiti ja siis taandadada selle uuesk, ja siis uuesti kõige uuem käima panna.
    a. Korrasta praegue olukord -> commii see
    b. Leia mingi vana commit, tõmba see endale working directorysse
    c. Vaata, mis juba siin saab.
        i. Tahaks proovida siiin mitut erinevat asja. Kõigepealt esimese directory viimase asjana commitida
        ii. Ma olen mingis detached HEAD states
        iii. Loen selleg kohta https://www.atlassian.com/git/tutorials/viewing-old-commits/
            1. Siin mainitakse midagi sellise asja kohta nagu HEAD- kinda a pointe
            2. No longer points to the branch, instead points to the specific commit – called detached head state.
            3. https://www.atlassian.com/git/tutorials/undoing-changes/git-clean Olen siin, proovin revertida, aga mingid konfliktid on ☺
2) tahan mergida selle mis guthubis on enda jamaga. Kuldas selling merge shti käia võiks ?- Branching
3) Tahan kutsuga logi üles nii, et mu prompt kohe pärast seda hulluks ei läheks. Käsk:

```
git --no-pager log
```

Branching and merging

1. HEAD state ??
    a. https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes- uurisin seda, ei saanud midagi
    b. https://www.atlassian.com/git/tutorials/syncing/git-remote – fetchiga luuakse sisuliselt uus branch local repositorys

	Tree Roles
The HEAD	last commit snapshot, next parent
The Index	proposed next commit snapshot
The Working Directory	sandbox

    c.
2. Conflicts:
    a. List of tutorial sources
        i. https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
        ii. http://www.gitguys.com/topics/merging-with-a-gui/
    b. Oluline on leida endale mingi hea tööriist mugavaks mergimiseks k3diff
    c.
3. Käsud:

   a. Git checkout <branch name> # Kui on remote branch, siis ei ole mingit erilist vahet tuleb lissalt lisada <remote name>/<branch name> kui tahad checkouti teha. Sama kehtib ka kõikidele teistele käskudele.
   b. Git merge <branch name> #Merges the named branch into current branch
   c. Git mergetool –t kdiff3
   d. Git branch # get names of branches
   e. Git log –oneline –graph # makes it all so visual and beautiful

4.

17:01 Currently looking for how to clear the working directory -> This wasn't the hardest task.

I just had to use the command:

git clean --interactive

Git online shit

1) Git fetch <remote name, tõmbab kohaliku remote branchi onlinest kõik branched, mis seal on.
2) https://git-scm.com/book/en/v2/Git-Branching-Remote-Branches Hea tutorial selle kõige kohta, kuidas onlainiga toimida.

Github File structure

Üleüldine ja kindlasti ka erandidega Põhimõte on järgida siiski oma ONEDRIVE .**mõisltlikuse piirides.**

Igas folderis mida kohalikust folderis lisama hakkan takkajärgi peab olema oma eraldi separate kaust VCS jaoks .

MÕTLE IGAST REPOST KUI ERALDI PROJEKIST MILLEGA TEGELEMA HAKKAD. PROJEKTIL VÕIKS OLLA MÕISLTIKUD PIIRID et liiga suurt ampsu korraga ei võtaks.

Plaanin alustada sellise mudeliga

1) HKU-Image Processing
   a. LABS
   b. SIFT PROJEKT
2) HKU- UML (IBM rational)
   a. HW1
   b. HW2…
3) HKU WEB ENGINEERING
   a. LAB1
4) MOBILE APP DEVELOPMENT
   a. HW1
   b. HW1

JNE..,

5) MACHINE LEARNING
6) INTERVIEWBIT KOODI

Eraldi projektide jaoks:

https://github.com/kriasoft/Folder-Structure-Conventions

Gitignore

https://www.atlassian.com/git/tutorials/gitignore

ESIMENE COMMIT JA REMOTE SIDUMINE

1) Kui paned takkajärgi projekti üles, siis tee folderisse eraldi REPO vcs-I jaoks.
2) .gitignore fail
3) Kui alustad oma projektiga, siis uuri kuidas jätta mingi sitt lisamata ja salvestada vaid vajalik.
4) LOO githubi uus rühi repository mingi mõistliku nimega
5) Edasi mine oma kohaliku repose
6) Ava command prompt ja sisesta Käsud

Git init

Git status

Git add [kõik mis vajalik eelmisest käsust]

Git commit

Git remote add origin [siia copy aadress mis punktist 4 said]

Git push origin master https://git-scm.com/docs/git-push LÕPUS ON LIST NÄIDETES

ESIMENE ALLA TÕMBAMINE

Täiesti esimene

Git clone <web address>

Kui toimub mergemisega

git remote add origin <web address>

remotes s master branch is now accessible locally as `origin/master`

Git remote -v

Git remote fetch origin

It’s important to note that the `git fetch` command only downloads the data to your local repository – it doesn’t automatically merge it with any of your work or modify what you’re currently working on

Git remote merge origin/master

At point that you want to share, you have to push it upstream. The command for this is simple: `git push [remote-name] [branch-name]`

Git push origin master

Stsenaarium kui jub githubis on miskit

1 Lood git repo githubis

1.5 paned sinna juba midagi sisse

2 git init

3 git remote add origin https://<githubs..>

4 git commit -m “mingi lamp “

5 git fetch origin

6 git merge origin/master

7 git push origin master