Basic-Commits

-\*git status\*: näyttää missä branchissä ollaan. Tämän hetkinen branch on master.

-Viimeksi tehdyt kommentoinnit (=commits), käyttäjän, päivämäärän ja kommentoinnin sisällön.

-Tiedoston luonti komennolla \*touch <file\_name>\*.

-Tiedoston luonnin jälkeen \*git status\* näyttää jäljittämättömät tiedostot, jossa näkyy juuri luotu tiedosto.

-\*git add <file\_name>\* lisää tiedoston “staging” -alueelle, jonka jälkeen git status näyttää “muutokset joita kommentoida” ja sen alla juuri luomani tiedoston.

-\*git commit -m\* kommentoi tehtyjä muutoksia, jonka jälkeen git status näyttää, että mitään ei ole kommentoitavissa, mutta yksi kommentointi branchissani on “pushattavana”.

-\*code <file\_name>\* avaa tiedoston jossain editorissa (esim. Visual Studio Code), jossa tiedoston sisältöä voi muokata, jonka jälkeen tiedosto tallennetaan.

-\*git status\* näyttää yhden tiedoston “changes not staged for commit” alla.

-\*git add <file\_name>\* jälkeen \*git status\* näyttää yhden tiedoston “muutokset joita kommentoida” alla.

-\*git status\* näyttää ettei muutoksia ole kommentoitu (koska \*git add <file\_name>\* ei tehty välissä), log näyttää kaikki sisältöön tehdyt kommentoinnit.

Basic-Staging

-\*code <file\_name>\* näkee tiedoston sisällön, tällä hetkellä ei mitään.

-\*git diff\* kertoo muutokset “Working Directoryn” tiedostoissa, kun taas \*git diff --staged\* näyttää muutokset “Staged”:n tiedostoissa.

-\*git status\* näyttää file.txt :n kaksi kertaa.

-\*git reset HEAD file.txt\* poistaa kaikki tiedostot, joita ei ole kommentoitu “Staged” alueella. Tämän jälkeen \*git status\* näyttää vain tiedostot, joita ei ole \*git add <file\_name>\* “Staged” alueelle.

-Kun tiedosto on \*git add <file\_name >\* “Staged” alueelle ja kommentoitu, \*git log\* näyttää tehdyt muutokset kyseiseen tiedostoon.

-Tiedoston sisältönä on “4”.

-\*git status\* kertoo, että yksi kommentointi on “pushattavana” \*git push\* ja muutokset joita ei ole \*git add <file\_name>\* “Satged” aluelle.

-\*git checkout <file\_name>\* palauttaa tiedoston edelliseen kommentointiin.

-Tiedoston sisältö näyttää “3”, eli toisiksi viimeiseksi tehdyn muutoksen.

-\*git status\* näyttää, että uusia muutoksia ei ole kommentoitavana.

Basic-Branching

-\*git branch\* näyttää olemassa olevat branchit, nykyisenä “master”.

-\*git branch <branch\_name >\* luo uuden branchin ja näkyy seuraavaksi komennolla \*git branch\*.

-\*git checkout <branch\_name>\* vaihtaa toiseen branchiin.

-Komennolla \*git status\* näkee myös missä branchissa on. Vaihtamalla “master” ja “mybranch”:n välillä “git statuksen” branch nimi myös muuttuu.

-\*git log --oneline --graph\* näyttää branchissa tehdyn kommentoinnin.

-Vaihdettaessa takaisin “master” branchiin, tehdyt kommentoinnit toisessa branchissa eivät näy.

-Käyttämällä \*git log --oneline --graph --all\* nähdään molempien branchien erilaiset kommentoinnit.

-Vaihtamalla takaisin “mybranch”:in ja komennolla \*git status\* nähdään “Working Directory” ja huomataan, ettei “master”:ssä tehtyä tiedostoa näy.

-\*git diff mybranch master\* nähdään kyseisten branchien erot ja huomataan, että “Working Directoryeissa” on eroja.

Reverting:

* Tässä katassa harjoiteltiin git revert komennon käyttöä. Periaatteessa revert tarkoittaa samaa kuin undo. Se ei kuitenkaan ole ns. normaali undo komento. se ei poista committia projektin historiasta, vaan luo uuden commitin jossa se etsii tiedot jota edellisessä commitissa käytettiin ennen revertin tekemistä. Tämä estää gitiä hukkaamasta historiaansa, joka on tarpeellista kun tehdään collabina.
  + teimme uuden branchin (git branch)
  + siirryimme branchiin (git checkout <branch-name)
  + tutkimme branchissa lokiaa. (git log –oneline –graph –all)
  + lisäsimme branchin (git add)
  + commitoimme branchin (git commit -m)
  + sitten reverttasimme kyseisen branchin. ( git revert <sha1> )
  + katsoimme mitä muutoksia kommiteisa oli (git diff)
  + tutkimme loki puuta ( git log –oneline –graph –all)

Cleaning:

* tässä katassa harjoiteltiin git clean komennon käyttöä. git clean siis toimii untracked tiedostoille, eli tiedostoille jotka ovat luotu working directoryyn mutta ei ole vielä addatty repositoryyn. Git clean poistaa siis untracked filut gitistä mutta ei oikeasti poista niitä.
  + listasimme directoryn filut (ls -R)
  + dry run clean komento ( git clean -n)
  + huomasimme että löytyi tiedostoja muodossa .c jota ei poistettu.
  + lisäsimme src/mylib.c stagin areaan. ( git add src/mylib.c)
  + ajoimme git clean -n uudelleen, .c ei poistunut eikä objectin directorysta tullut listausta filuista.
  + ajoimme git clean uudelleen -n ja -d optionilla. (git clean -n -d)
  + filut poistui.
  + putsasimme koko directoryn (git clean -f -d)

Stashing:

* tässä katassa harjoiteltiin git stash komentoa. komento piilottaa / hyllyttää tehdyt muutokset jotka teit tiedostoosi. Voit siis tehdä jotain muuta välillä ja lisätä piilotetut muutokset myöhemmin työhösi. Stash kometo on hyvä apu jos pitää nopeasti vaihtaa kontekstia ja työskennellä jonkun muun asian parissa. mutta vanha työ ei kuitenkaan ole vielä valmis kommitoitavaksi.
  + selailtiin repoa, ja katsottiin työtä miedän working directoryssa ( git status)
  + mikä työn staged (git status -s)
  + tutkittiin kommiteja ( git diff ja git diff master)
  + Piilottime/ hyllytimme tämän hetkisen työn ( git stash)
  + katsoimme mitä working directorysta löytyyn nyt ( git status)
  + miltä kommit näyttää ( git diff / master)
  + miltä stash list näyttää ( git stash list)
  + korjasimme virheet bug.txt tiedostossa masterissa ja kommitoimme ne ( code bug .txt, git add bug.txt, git commit -m , git push -u origin master)
  + siiryimme takasin vanhaan työhömme ja lisäsimme stashin muutokset masteriin. (git stash apply)
  + git status
  + kaikki tiedostot ovat unstaged.
  + perutaan kaikki tehdyt muutokset. ( git reset – hard HEAD)
    - tämä on unsafe sillä se removaa kaikki filut indexista ja poistaa koko working directoryn. Onneksi tiedot ovat edelleen hyllyllä piilossa joton on ok resettaa.
  + lisätään stashin tiedot masteriin ( git stash apply –index)
  + git status
  + nyt on tiedot staged jotka olivat alussakin staged.
  + voimme tehdä nyt git add ja git commit töille jotka saimme valmiiksi.
  + git push -u origin master.
  + emme tarvitse enää hyllytettyjä tietoja joten voimme pudottaa/ unohtaa nuo tiedot komennolla ( git stash drop)
  + git stash list =empty
  + git log –oneline –all –graph on muuttunut viimeisien committien takia.