HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Versionhallinta

Matti Paksula ja Esko Luontola

Syksy 2009

Tietojenkäsittelytieteen laitos
Kuje Research Group

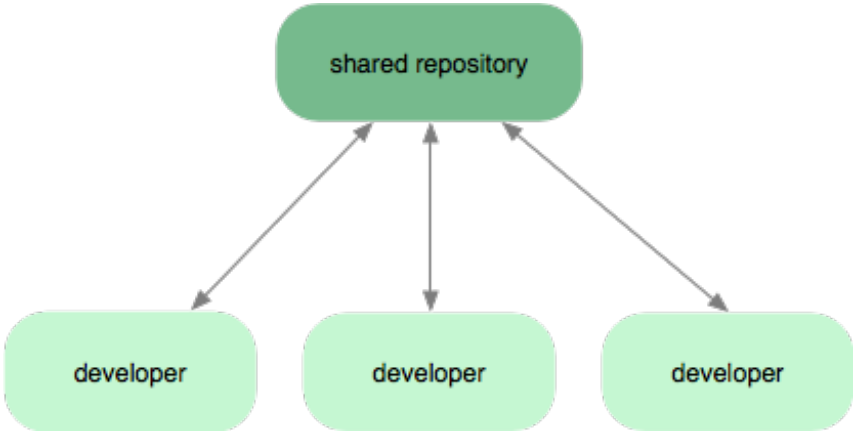
http://www.youtube.com/watch?v=4XpnKHJAok8#t=2m20s (2:20-4:00)

Keskitetty vs hajautettu versionhallinta

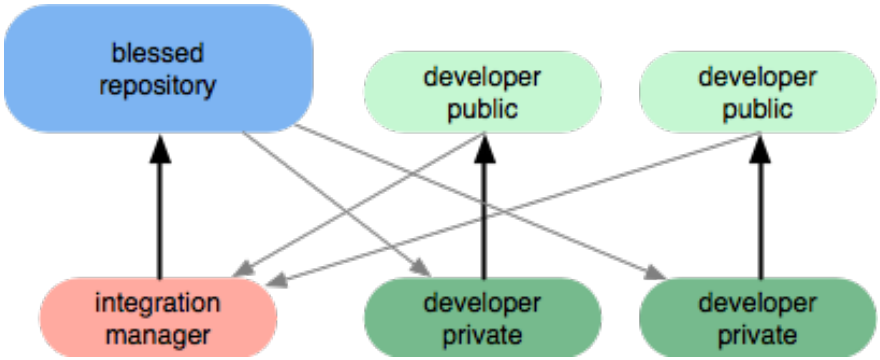
- Yksi repo, jossa on kaikki tieto. Jos ei ole yhteyttä sinne tai se tuhoutuu, niin voi voi.
- Kullekin käyttäjälle pitää antaa kirjoitusoikeus repoon, tai sitten toisten pitää lähetellä patcheja (toimii yrityksissä, mutta mättää open source -projekteissa).
- Lähes kaikki operaatiot ovat hitaita, koska tarvitaan yhteys internetin yli.
- Jokaisella repon käyttäjällä on oma kopio koko reposta. Voi commitoida ilman nettiyhteyttä, eikä ensisijaisen repon tuhoutuminen haittaa.
- Kukin käyttäjä voi kommitoida omaan repo-kopioonsa, eikä kirjoitusoikeutta tarvita. Joku, jolla on kirjoitusoikeudet, voi vetää muutokset ensisijaiseen repoon (patchien käyttö on silti myös mahdollisia).
- Lähes kaikki operaatiot ovat nopeita, koska ne ovat paikallisia.

Hajautettu versionhallinta mahdollistaa useat erilaiset työnkulut

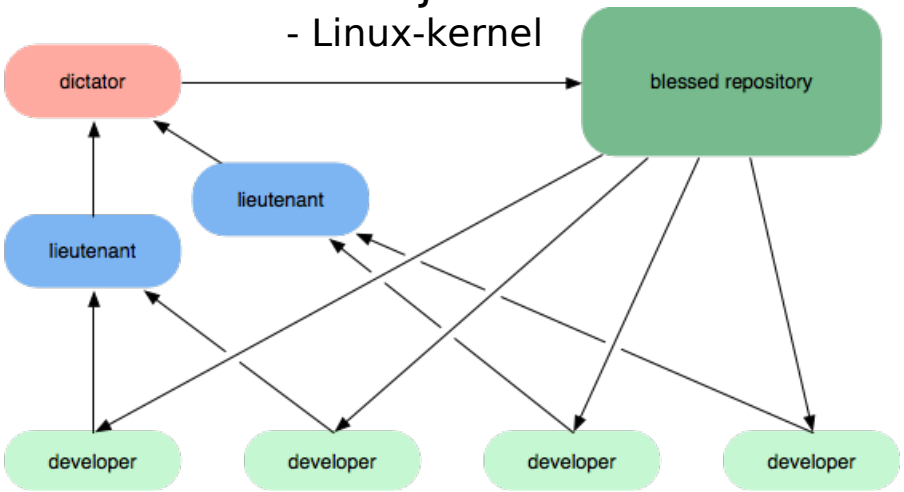
"Keskitetty"
- firmat



"Integrointimestari"
- monet open source -projektit, GitHub



"Diktaattori ja luutantit"
- Linux-kernel



http://whygitisbetterthanx.com/

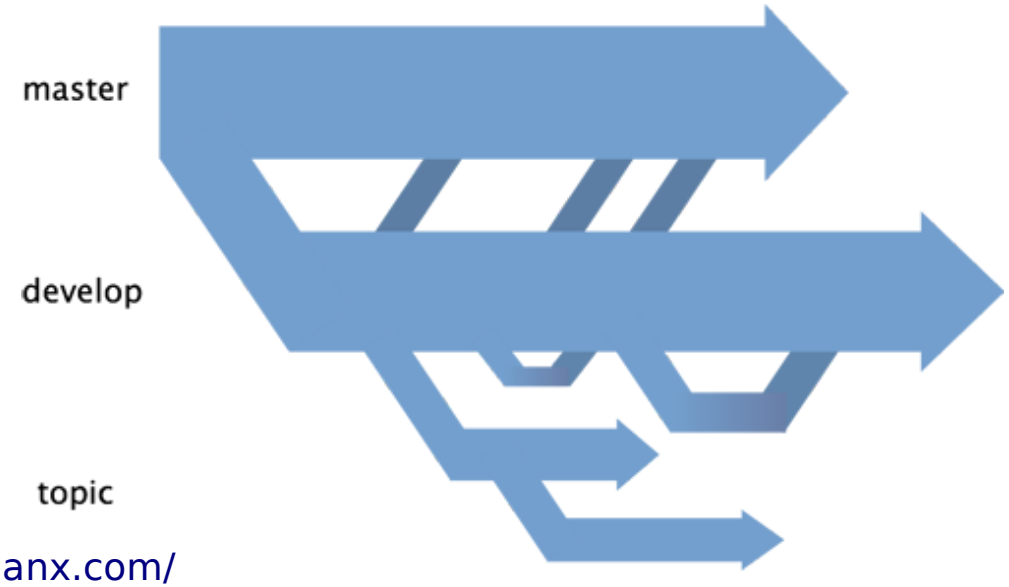
Gitin historia

- Linux-kernelin kehityksessä käytetyn BitKeeper-versionhallinnan lisenssiehtojen muutos esti sen käytön jatkamisen.
- Linus Torvalds ei löytänyt hyvää korviketta, joten hän teki oman.
- Tavoitteet:
  - CVS on esimerkki siitä, miten asioita ei pidä tehdä.
  - Tue hajautettua, BitKeeper-tyylistä työnkulkua.
  - Vahva suojaus korruptiota vastaan – sekä vahingossa että tarkoituksella tapahtunutta.
  - Hyvä suorituskyky.

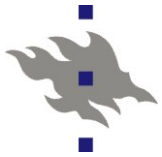
http://en.wikipedia.org/wiki/Git_(software)

Miksi Git?

- Halvat paikalliset branchit. Helppo merge.
  - On järkevää tehdä jokaiselle toiminnolle oma “*topic branch*” ja poistaa se sitten, kun se on mergetty.
  - Omissa brancheissa voi tehdä kokeiluja, eikä branchin poistamisen jälkeen niistä jää merkkiä kenellekään.
  - Voi julkaista vain haluamansa branchit päärepoon.



http://whygitisbetterthanx.com/



Miksi Git?

- Kaikki hajautetut ovat nopeampi kuin keskitetyt, mutta Git on jopa nopeampi kuin muut hajautetut.

http://whygitisbetterthanx.com/

Miksi Git?

- Jokainen commit on varmistettu SHA-1:llä niin, ettei kukaan/mikään voi muuttaa/korruptoida repoa huomaamatta.
- “Staging area” eli ”index” tekee Gitistä joustavan; mm. on helppoa commitoida vain osa tiedostoon tehdyistä muutoksista.
- Voimakkaita työkaluja, kuten *git rebase*.
- Ei ärsyttäviä *.svn* tai *CVS* -kansioita joka paikassa.
- Repot ovat tiiviiksi pakattuja.
  - 21 vuotta Perlin historiaa (1987-2009): repo 87 MB, checkout 67 MB
    http://github.com/blog/276-perl-mirror-on-github
- Gitiä voi käyttää omalla koneellaan, vaikka keskitetty repo olisi SVN.
- SVN:ssä ei ole brancheja eikä tageja (vain kansioita).

Miksi *ei* Git?

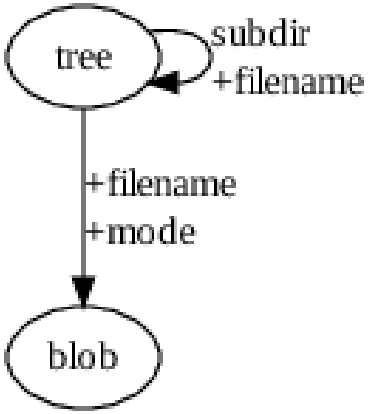
- Yhteensopivuus muiden kuin Linuxin ja UTF-8:n kanssa on välttävä.
  - Windowsissa Cygwin on buginen (on myös msysGit, mutta sekin on ollut buginen) ja hyviä graafisia ohjelmia ei juurikaan ole.
  - Ääkköset tiedostonimissä eivät toimi luotettavasti, jos käytät mitä tahansa muuta kuin Linuxia ja UTF-8:aa.
    - Windowseissa tiedostonimien enkoodaus voi olla melkein mitä vain.
    - Maceissa tiedostonimien normalisointi aiheuttaa harmaita hiuksia.
    - Git on tyhmä ja autuaan tietämätön tiedostonimien enkoodauksesta – sille kaikki on tavujonoja.

Mentaalimalli Gitin rakenteesta

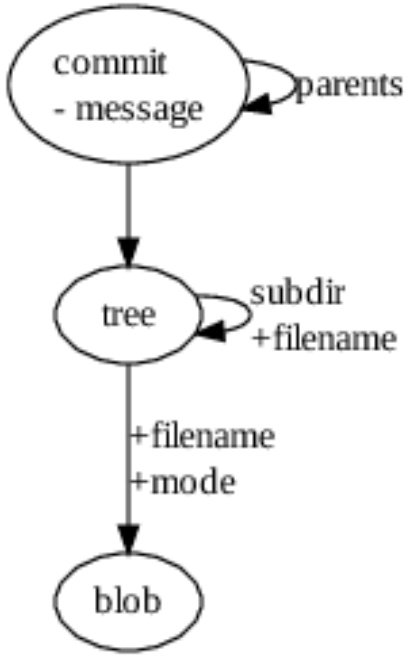
- Git-repo on rakenteeltaan suunnattu asyklinen verkko.
- Jokainen repossa oleva olio on yksilöity SHA-1:llä.
- Oliot eivät voi muuttua sen jälkeen, kun ne on luotu.

Tavujono, joka voi olla tiedosto, symbolinen linkki tai melkein mitä vaan.

Hakemistot sisältävät tiedostoja ja toisia hakemistoja.



Commit sisältää senhetkisen hakemistopuun ja kaikki aikaisemmat commitit (0..N kpl).



Huom! Unohda kaikki mitä luulet tietäväsi brancheista SVN:n käyttämisen perusteella.

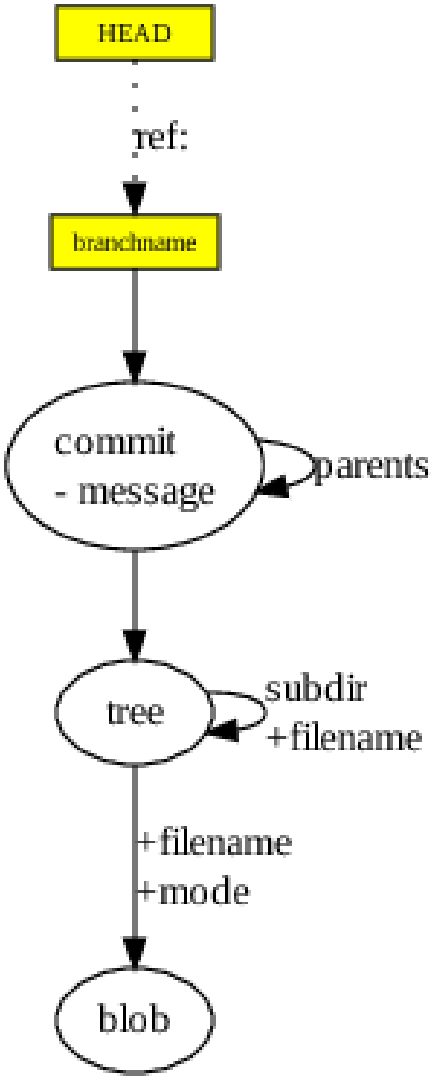
http://eagain.net/articles/git-for-computer-scientists/

Mentaalimalli Gitin rakenteesta

Branchit ovat nimilappuja, jotka osoittavat johonkin committiin.

Commitit eivät voi muuttua, vaan niitä voi vain lisätä (ja roskienkerääjä poistaa ne kun mikään ei viittaa niihin), mutta branchit ja tagit voivat muuttua vapaasti. Ne eivät ole osa historiaa, eivätkä ne siirry itsestään repojen välillä.

HEAD viittaa tällä hetkellä aktiivisena olevaan branchiin/committiin, eli siihen joka on sillä hetkellä checkoutattuna.

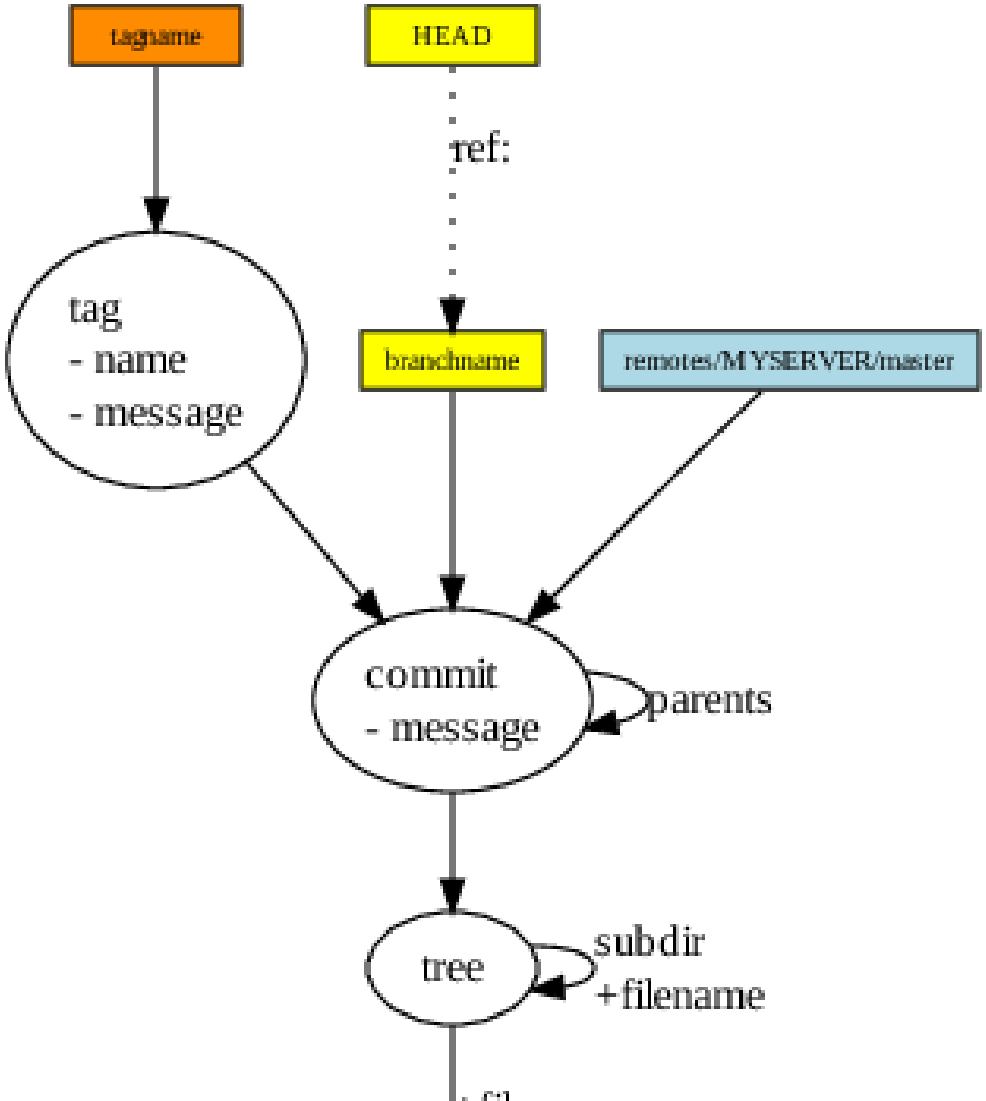


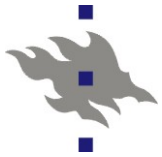
Mentaalimalli Gitin rakenteesta

On olemassa remote brancheja, jotka kertovat mihin jossain toisessa repossa ollut branch viittasi, kun siihen repoon viimeksi oltiin yhteydessä

Tagit osoittavat johonkin committiin ja voivat lisäksi sisältää vapaaehtoisen viestin ja GPG-allekirjoituksen.

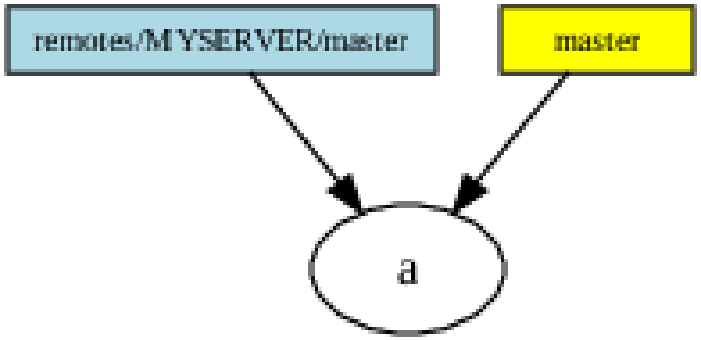
Tagit eivät siirry itsestään, toisin kuin branchit siirtyvät commitoidessa. Muuten tagit ovat samanlaisia nimilappuja kuin branchit.





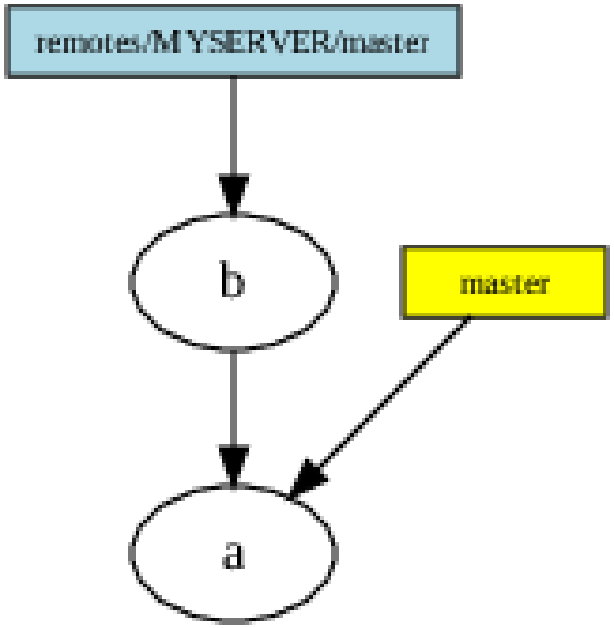
Historian käpistely

- `git clone git://github.com/plim/plom.git`
  - On kloonattu repo, jossa on yksi commit.



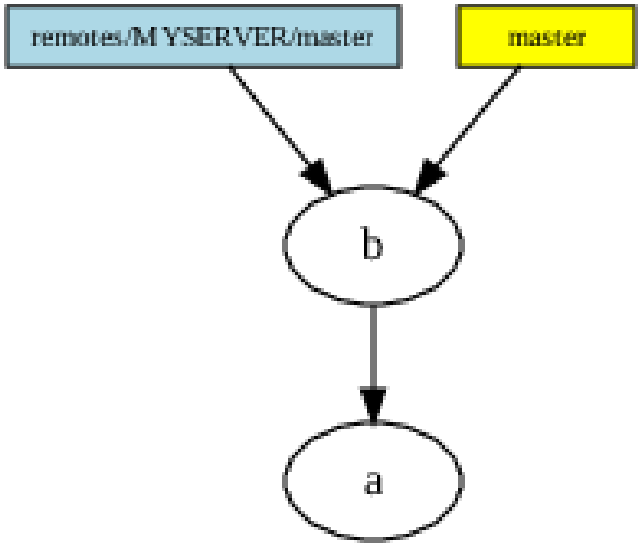
Historian käpistely

- `git fetch`
  - Joku on commitoinut b:n alkuperäiseen repoon. Fetchataan se meidän paikalliseen repoon.

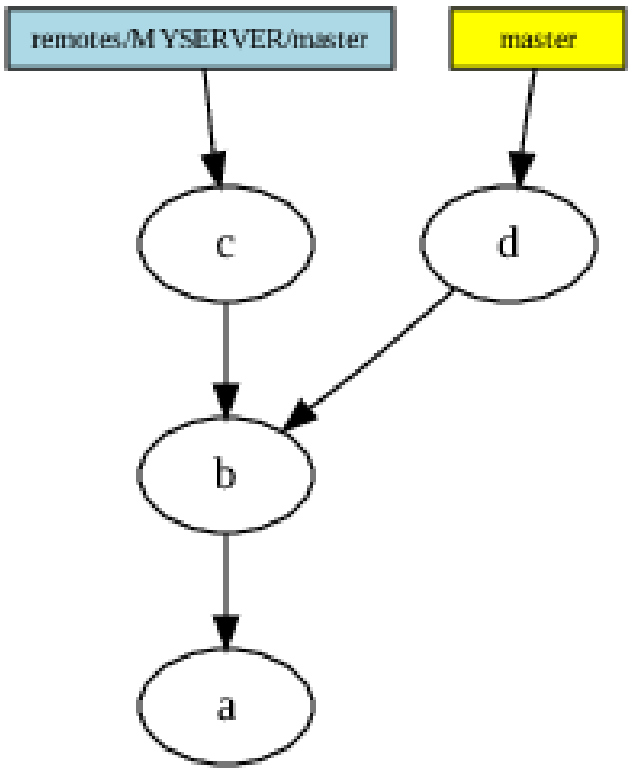


Historian käpistely

- `git merge MYSERVER/master`
  - Mergetään remote branch meidän omaan paikalliseen master-branchiin. Koska meillä ei ole omia committeja (merge on fast-forward), niin ainoastaan nimilapun paikka vaihtuu.

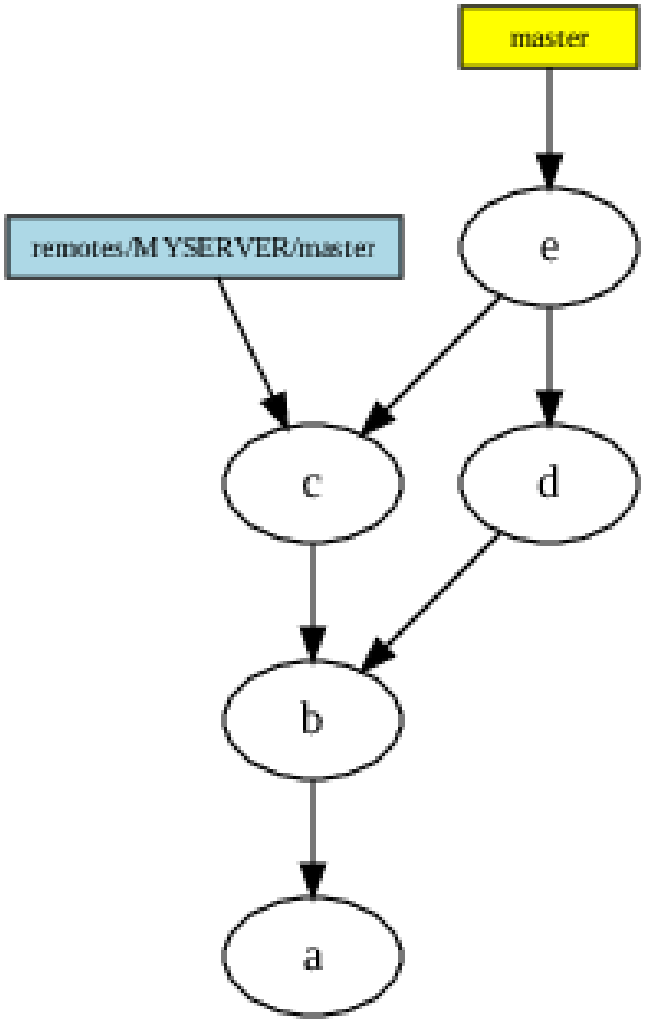


Historian käpistely



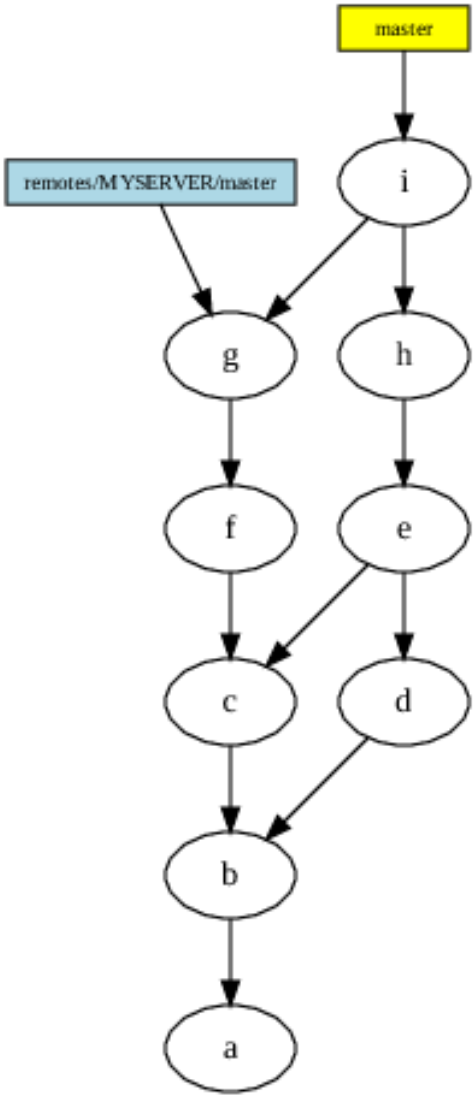
- Tilanne yksi pari committia ja fetch myöhemmin. Tarvitaan mergeä.

Historian käpistely



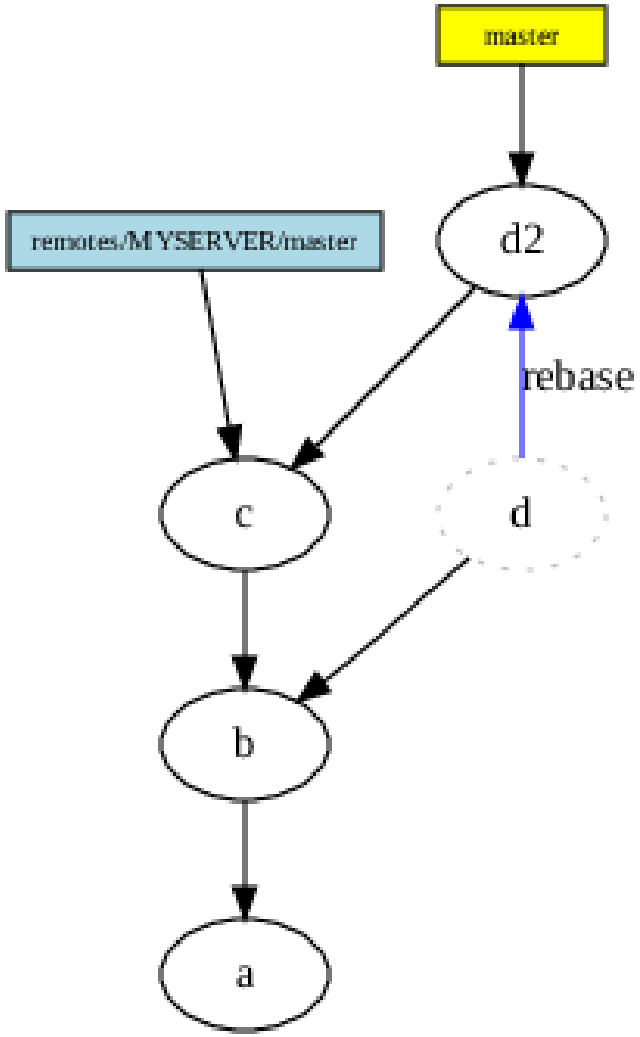
- `git merge MYSERVER/master`
  - Mergetään remote branch meidän omaan paikalliseen master-branchiin. Koska meillä *oli* omia committeja (merge ei ollut fast-forward), niin historiaan luotiin uusi commit, joka on kahden aiemman branchin viimeisten committien lapsi.

Historian käpistely



- Vielä useampi commit ja merge myöhemmin. Muodostuu tikapuut.
- Tikapuita on vaikea lukea. Jos omaa branchia ei vielä ole pushattu julkiseksi, niin on suositeltavaa käytää rebasea...

Historian käpistely

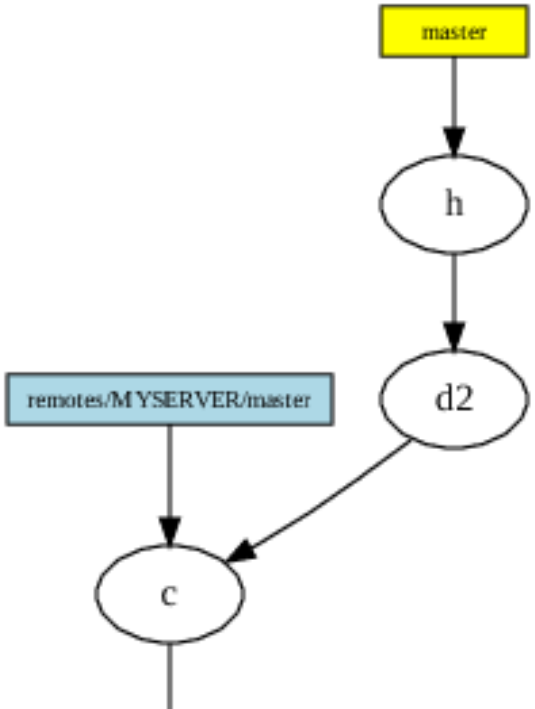


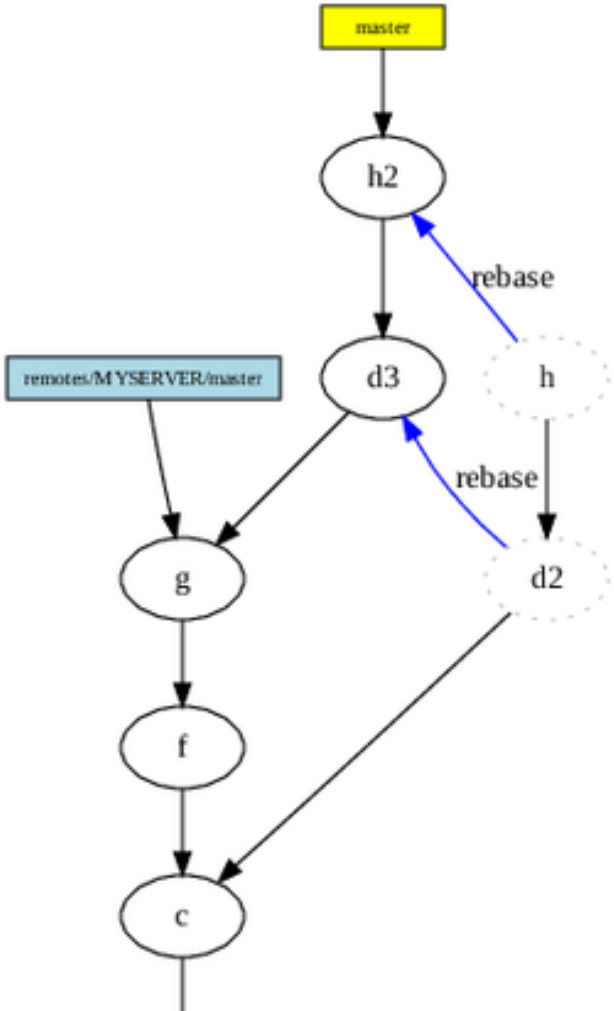
- `git rebase MYSERVER/master`
  - Luodaan uusi commit *d2*, joka on muuten sama kuin *d*, mutta sen parent commit on eri.
  - Vanha commit pysyy repossa, kunnes aikanaan Gitin roskienkerääjä (`git gc`) poistaa sen. Gitin työkalut (esim. `gitk`) eivät kuitenkaan näytä orpoja committeja, mutta halutessaan niiden hashin saa kaivettua esille (`git reflog` tai `git fsck`) ja sen voi sitten checkoutata ja branchata.

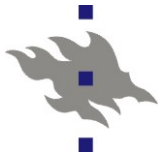
http://gitfu.wordpress.com/2008/04/06/git-reflog-no-commits-left-behind/

Historian käpistely

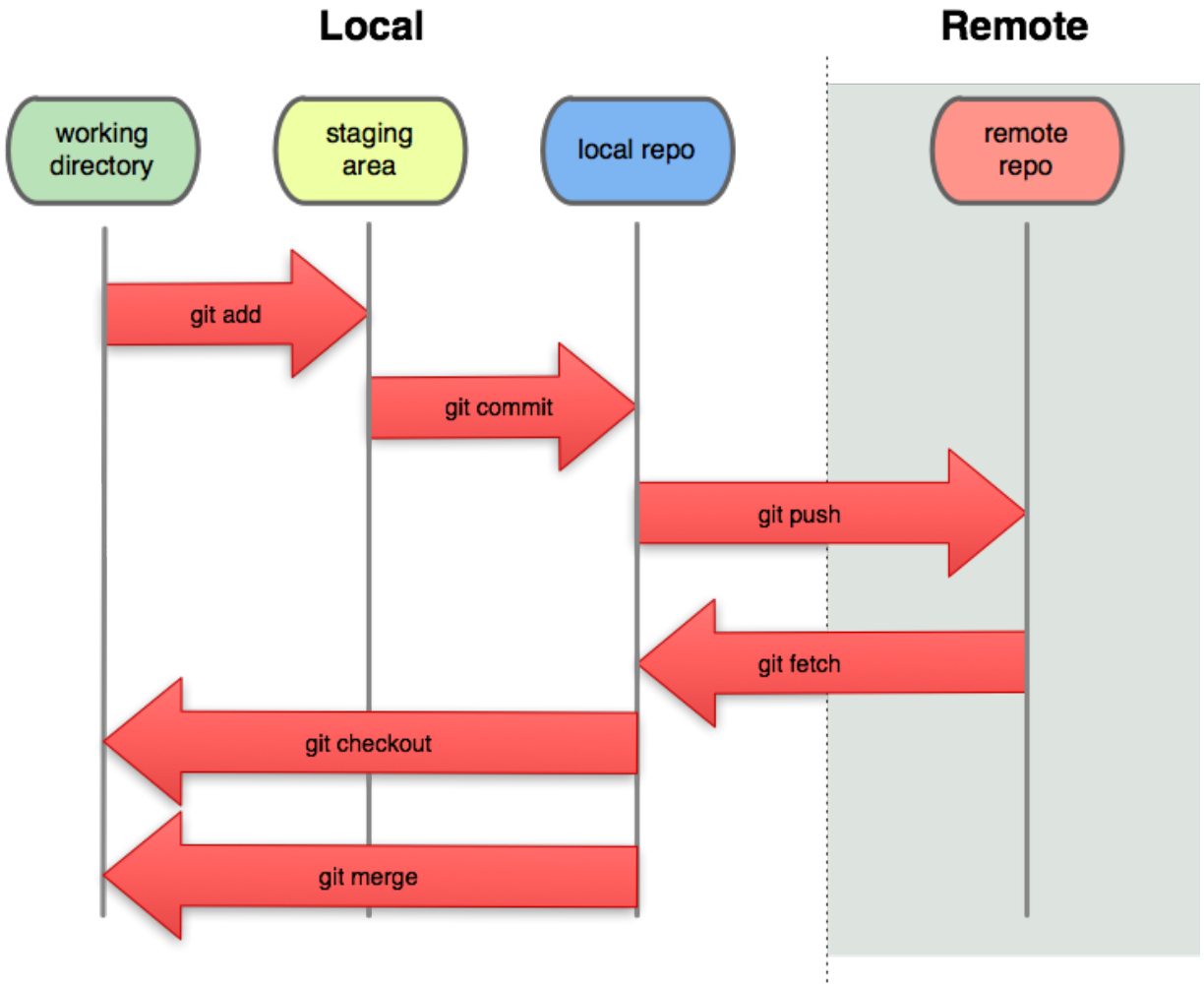
- Rebase toimii myös useille commiteille kerralla.







Index eli *staging area*



- Paikka, jonne muutokset työkansiosta kopioidaan yksitellen, jotta ne voi commitoida kerralla.
- Helpottaa ja lisää ilmaisuvoimaa.
- Vaikka muut VCS:t sisältävät sisäisesti vastaavan, niin ne eivät paljasta sitä käyttäjälle.

Käsitteitä Gitin tapaan

- Nyrkkisääntö: Yhdessä repossa pidetään tiedostot, jotka tyypillisesti halutaan checkoutata aina kaikki samalla kertaa.
  - Gitin submodule-toiminnolla voi hieman joustaa.
- Nyrkkisääntö: Kaikki operaatiot käpistelevät nykyistä checkoutattua branchia.
  - Jos checkoutattu commit/tag, niin ei olla branchissa ja commitit menevät branchien ulkopuolelle.

    Luo uusi branch headista: `git checkout -b <nimi>`
- Gitissä ei ole rename/copy -komentoa. Ne havaitaan automaattisesti tiedoston sisällön perusteella.
- Git ei seuraa kansioita, vaan pelkästään tiedostoja.

Käsitteitä Gitin tapaan

- `git init` tai `git init --bare`
  - Luo uusi repo nykyiseen kansioon, jossa “master” branch eikä yhtään committia.
- `git clone`
  - Ota itsellesi kopio palvelimella olevasta reposta. Kloonattava remote repo saa nimen “origin”.
    - Vertaa: `svn checkout`
- `git push`
  - Työnnä muutokset paikallisesta reposta palvelimelle.
    - Vertaa: `svn commit`
- `git fetch` tai `git pull`
  - Vedä muutokset palvelimelta paikalliseen repoon.
    - Vertaa: `svn update`

http://git-scm.com/documentation
http://www.kernel.org/pub/software/scm/git/docs/everyday.html
http://git-scm.com/course/svn.html
http://zrusin.blogspot.com/2007/09/git-cheat-sheet.html

Käsitteitä Gitin tapaan

- `git add`
  - Kopioi tiedoston muutokset *indeksiin*, eli *staging areaan*, eli valmistaudutaan commitoimaan muutokset repoon.
    - Hyödyllinen parametri: `-u` tai `-A`
- `git rm`
  - Poista tiedosto ja lisää poisto myös indeksiin.
    - Myös `git add -A` merkitsee poistot.
- `git reset`

  `git reset --hard`
  - Muutosten poistaminen indeksistä ja/tai työkansiosta.
    - Muista myös `git checkout <file>`
- `git commit`
  - Tee uusi commit indexin sisällöstä.
    - Hyödyllinen parametri: `-a` tai `--amend`

Käsitteitä Gitin tapaan

- `git checkout`
  - Ottaa työkansioon jonkin branchin/tagin/commitin tiedostot muokattavaksi. (Suunnilleen vastaava kuin `svn switch`)
- `git branch`
  - Luo ja poistaa brancheja.
    - Myös git checkout -b <newbranch>
- `git tag`
  - Luo ja poistaa tageja.
- `git merge`
  - Yhdistää brancheja.
- `git rebase`
  - Muokkaa aiemmin tehtyjä committeja.

Käsitteitä Gitin tapaan

- `git status`
  - Työkansion ja indeksin tilan tarkastelu.
- `git diff` tai `git diff --cached`
  - Työkansion ja indeksin muutosten sisällön tarkastelu.
- `git log`
  - Historian tarkastelu.
- `gitk`
  `gitk --all`
  `git gui`
  - Graafisia apuvälineitä historian tarkasteluun ja committien tekemiseen.

Käsitteitä Gitin tapaan

- Edellisiin committeihin viittaaminen:
  - `HEAD, HEAD^, HEAD^^`
  - `HEAD, HEAD~1, HEAD~2`
    - Toimii myös hashilla tai branhilla: `master~5`
- Asetusten muuttaminen: `git config`
  - `git config --global user.name "Mikko Mallikas"`

    `git config --global user.email "mallikas@gmail.com"`
- Ignoraus .gitignore ja .git/info/exclude -tiedostoilla.

http://help.github.com/git-email-settings/

Käsitteitä Gitin tapaan

- Repositoryyn yhdistäminen, joitakin url-muotoja
  - Paikallinen kansio: /home/tunnus/repo.git
  - Git-protokollalla: git://github.com/tunnus/repo.git
  - SSH:n kautta: git@github.com:tunnus/repo.git
- Bare vs. working copy -tyyppinen repo
  - `mkdir projekti.git`
    `cd projekti.git`
    `git init --bare`
    - Repon osoite: /foo/bar/projekti.git
  - `mkdir projekti`
    `cd projekti`
    `git init`
    - Repon osoite: /foo/bar/projekti/.git

http://www.kernel.org/pub/software/scm/git/docs/git-fetch.html#URLS

Merge conflict

- Kun merge ei onnistu automaattisesti, niin Git lisää konfliktoivaan tiedostoon väkäsiä:

```
Here are lines that are either unchanged from the common
ancestor, or cleanly resolved because only one side changed.
<<<<<<< yours:sample.txt
Conflict resolution is hard;
let's go shopping.
=======
Git makes conflict resolution easy.
>>>>>>> theirs:sample.txt
And here is another line that is cleanly resolved or unmodified.
```

- Muokkaa tiedosto sellaiseksi kuin haluat sen tulevan, lisää se indeksiin `git add`:lla ja kommitoi `git commit`:lla.

http://www.kernel.org/pub/software/scm/git/docs/git-merge.html

Edistyneitä toimintoja

- `git rebase -i`
  - Rebase steroideilla. Mahdollistaa committien uudelleenjärjestelyn, yhdistämisen, pilkkomisen ja yms. historiankirjojen muuttamisen mieleisekseen.
- `git svn`
  - Vaikka firman/projektin virallinen repo on SVN-muodossa, niin Gitiä voi silti käyttää omalla koneellaan. (SVN:n “brancheja” ei kannata yrittää käyttää Gitin kautta.)
  - CVS:lle löytyy vastaava toiminto.
  - Yhteistoiminta muiden hajautettujen versionhallintojen kanssa onnistuu myös.

Edistyneitä toimintoja

- `git format-patch`
  `git am`
  `git apply`
  - Patchien luominen ja käyttäminen. Patchit sisältävät myös commitin kommentin, authorin ja ajan.
- `git stash`
  - Laita työhakemiston muutokset tilapäisesti syrjään.
- `svn2git`
  - SVN-repon muuntaminen pysyvästi Git-repoksi. Käsittelee SVN:n tagit ja branchit paremmin kuin pelkkä `git svn clone`.

http://github.com/nirvdrum/svn2git/tree/master
http://blog.orfjackal.net/2009/05/converting-confused-svn-repositories-to.html

Hooks

- Skriptien ajaminen, kun esim. commitoidaan tai pushataan muutokset palvelimelle.
  - Huom! Repoa muuttavat toimenpiteet pitää tehdä työkansiossa (eli repo ei saa olla `git init –bare`).
  - esim. webbisivujen päivittäminen post-receive -koukulla:

```
if [ -n $GIT_DIR ]; then
  # Current dir is "<reporoot>/.git/", but
  # we need to run reset at "<reporoot>/".
  # Clearing GIT_DIR is needed, or reset
  # will fail with "fatal: Not a git repository: '.'"
  unset GIT_DIR
  cd ..
fi
git reset --hard
```

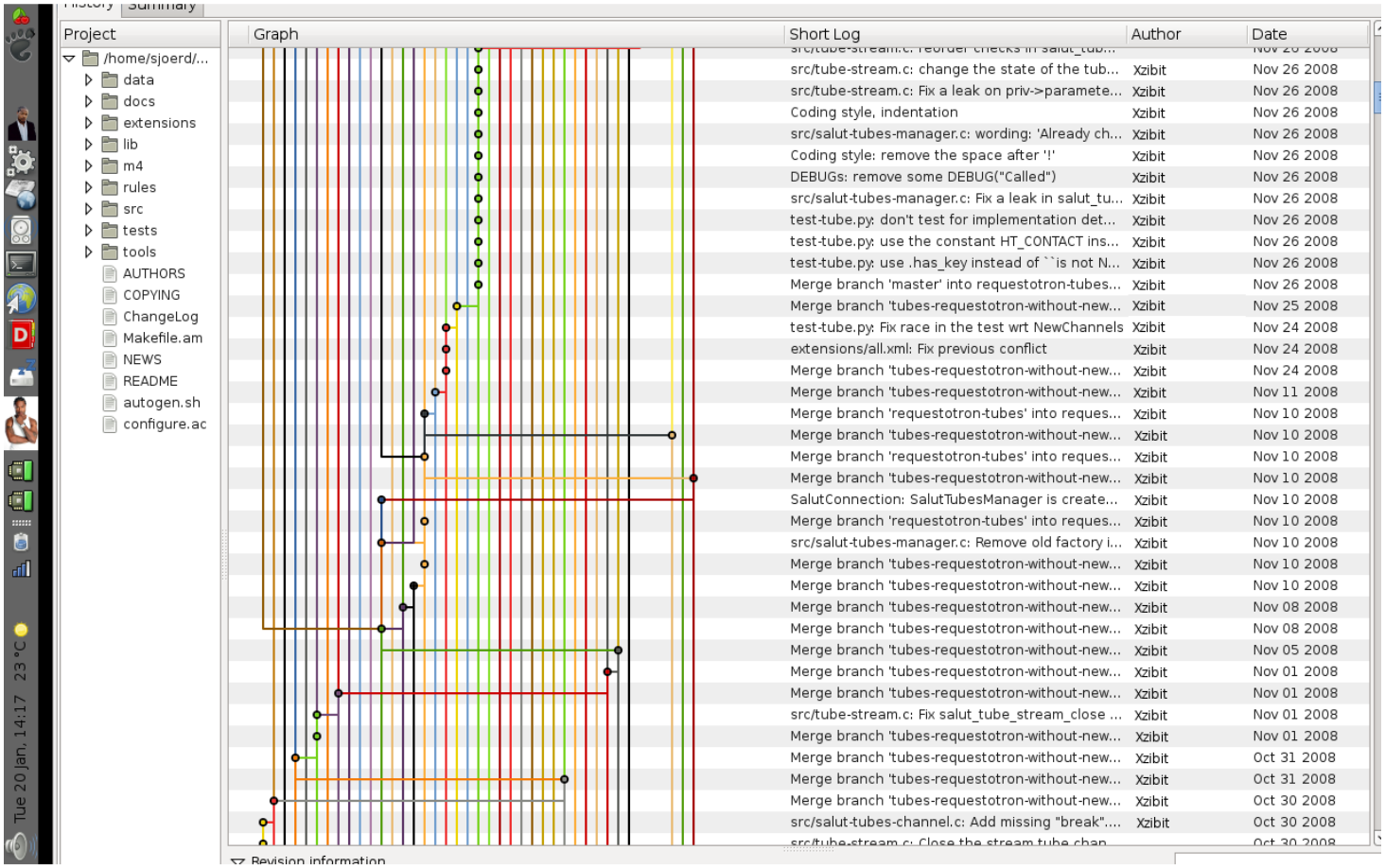
http://www.kernel.org/pub/software/scm/git/docs/githooks.html

http://debuggable.com/posts/git-tip-auto-update-working-tree-via-post-receive-hook:49551efe-6414-4e86-aec6-544f4834cda3

Lisämateriaalia

- http://www.kernel.org/pub/software/scm/git/docs/user-manual.html
- http://git-scm.com/documentation
- http://www.gitready.com/
- http://github.com/

YO DAWG, I HEARD YOU LIKE DVCS



SO I PUT A BRANCH IN YOUR BRANCH SO YOU CAN MERGE WHILE YOU MERGE