# Gitlab-järjestelmän versionhallintatyökalut tietojenkäsittelyn opetuksessa

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Hämeenlinna, Tietojenkäsittelyn koulutusohjelma

Kevät 2017

Antti Kaukoharju

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Visamäki, Hämeenlinna

**Tekijä** Antti Kaukoharju **Vuosi** 2017

**Työn nimi** Gitlab-järjestelmän versionhallintatyökalut tietojenkäsittelyn opetuksessa

**Työn ohjaaja/t** Erkki Laine

TIIVISTELMÄ

Opinnäytetyön toimeksiantajana oli Hämeen ammattikorkeakoulun tietojenkäsittelyn koulutusohjelma. Tavoitteena oli tutkia Gitlab-versionhallintapalvelua ja selvittää, miten tämän palvelun työkaluja voitaisiin soveltaa tietojenkäsittelyn opetuksessa projektien seurantaan. Opinnäytetyössä tutkittiin selaimella toimivaa Gitlabin Online -versiota sekä paikallisesti asennettavaa Community Edition -versiota.

Opinnäytetyössä käsitellään aluksi versionhallintaa yleisesti. Tämän jälkeen selitetään versionhallintajärjestelmien toimintaa yleisesti ja käsitellään Git-versionhallintajärjestelmän perustoimintoja lyhyesti. Viimeisenä käsitellään eri versionhallintapalveluja ja käydään läpi Gitlabin toimintoja pääasiassa siltä osalta, millaisia yhteenvetoja ja kaavioita Gitlab tekee projektien etenemisestä ja ryhmänjäsenten aktiivisuudesta.

Tutkimuksessa selvisi, että ainakin osa Gitlabin työkaluista soveltuu hyvin tietojenkäsittelyn koulutusohjelmassa tehtyjen projektien seurantaan. Gitlab-versionhallintapalvelu voidaan myös nähdä parhaimpana vaihtoehtona eri versionhallintapalveluista.

**Avainsanat** versionhallinta, versionhallintajärjestelmä, versionhallintapalvelu

**Sivut** 28 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Visamäki, Hämeenlinna

**Author** Antti Kaukoharju **Year** 2017

**Subject** Gitlab version control tools in teaching for Business information technology

**Supervisors** Erkki Laine

ABSTRACT

The client for this thesis was the Degree Programme in Business Information Technology of Häme University of Applied Sciences. The goal was to research Gitlab version control service and to find out if the tools of this service could be used to follow projects in the Degree Programme in Business Information Technology. Browser based Gitlab Online version and locally working Gitlab Community Edition version were researched for this thesis.

At first this thesis goes through version control in general. After this the basics of version control systems is explained in general and some basic functions of the Git version control system are explained briefly. Last this thesis explains different version control services and goes through graphs and summaries that Gitlab does in terms of team activity and progress of the project.

The conclusion of this research was that at least some of the tools that Gitlab has can be used to follow the progress of the projects in Degree Programme in Business Information Technology. Gitlab version control service can also been seen as the best choice when compared to other version control services.

**Keywords** version control, version control system, version control service

**Pages** 28 pages

SISÄLLYS

# 1 JOHDANTO

Opinnäytetyön tarkoituksena on tutustua Gitlab-järjestelmän versionhallintatoimintoon sekä selvittää, millaisia yhteenvetoja ja raportteja järjestelmä tekee projekteista versionhallinnan avulla ja kuinka näitä toimintoja voitaisiin soveltaa Hämeen ammattikorkeakoulussa tietojenkäsittelyn koulutusohjelman opetuksessa ja esimerkiksi eri moduulien projektitöiden edistymisen seurannassa. Ominaisuuksiltaan testattavia versioita Gitlabista ovat selaimella toimiva Online-versio, sekä koneelle ladattava Community Edition-versio, jotka molemmat ovat järjestelmän ilmaisia versioita.

Työn toimeksiantajana toimii Hämeen ammattikorkeakoulun tietojenkäsittelyn koulutusohjelma. Tarkoituksena on selvittää voitaisiinko Gitlab-järjestelmän versionhallinnan työkalujen tekemiä yhteenvetoja hyödyntää Hämeen ammattikorkeakoulussa tietojenkäsittelyn opetuksessa.

Ammattikorkeakoulun projektitöissä on usein hankalaa se, että esimerkiksi ohjelmoinnin ryhmäprojektissa koodit saattavat olla jakautuneet useammalle projektin henkilölle ja näitä voi olla useammasta syystä vaikea yhdistää yhdeksi toimivaksi kokonaisuudeksi, varsinkin jälkeenpäin. Gitlab saattaisi mahdollistaa helpomman projektityöskentelyn, kun kaikkien projektin jäsenten koodit ovat samassa jakopalvelussa kaikkien projektin jäsenten käytettävissä. Tämän pitäisi helpottaa projektin yhdessä pitämistä, palautusta ja opettajan näkökulmasta työn tarkastamista.

Opetuksen kannalta Gitlab-järjestelmän tärkein ominaisuus on sen projekteista luomat yhteenvedot ja raportit, jotka ovat erilaisia graafisia diagrammeja projektin edistymisestä. Näistä voisi nähdä esimerkiksi missä vaiheessa projekti on ylipäätänsä, mitä vaiheita projektissa on viimeksi tehty, ketkä projektin jäsenet ovat tehneet mitäkin vaiheita ja kuinka paljon tai mihin aikaan. Näiden avulla voitaisiin mahdollisesti selvittää, kuinka projekti edistyy. Mahdolliset ongelmatilanteet voidaan myös selvittää, esimerkiksi jos joidenkuidenkin projektien jäsenten aktiivisuus on alhainen.

Lyhyesti tavoitteena oli selvittää millaista tietoa Gitlabin versioiden järjestelmänhallinnan yhteenvedot kertovat, miten Gitlabin Online- ja Community -versio eroavat toisistaan ja voiko järjestelmää soveltaa tietojenkäsittelyn koulutusohjelmassa.

# 2 VERSIONHALLINTA

## 2.1 Yleistä versionhallinnasta

Versionhallinnalla tarkoitetaan yleisesti erilaisia tekniikoita ja käytäntöjä, joilla voidaan pitää kirjaa dokumentteihin ja tiedostoihin tehdyistä muutoksista ja säilöä samalla dokumenttien ja tiedostojen vanhemmat versiot. Esimerkiksi ohjelmoinnissa tällä tarkoitetaan projektin lähdekoodiin tehtyjen muutosten dokumentointia ja vanhempien versioiden säilyttämistä, jolloin tarvittaessa voidaan siirtyä takaisin käyttämään projektin vanhempaa versiota (Chason & Straub 2014).

Versionhallinta on nykypäivän ohjelmistokehityksessä ja muunlaisessa projektien dokumentoinnissa tärkeää. Ohjelmistokehityksen versionhallinta alettiin nähdä tarpeelliseksi 1960-luvulla, kun ohjelmistoprojektien koko alkoi kasvaa huomattavan suureksi. Versionhallintaa ryhdyttiin kehittämään tällöin, koska nähtiin projektien hallinnan kannalta tarpeelliseksi ohjelmistoprojektien sekä niiden sisältämien dokumenttien ja lähdekoodin hallinnan helpottaminen ja projektin koko kehityskaaren seurannan parempi dokumentointi. Versionhallintaa tehtiin aluksi käsin. Manuaalisesti tehty versionhallinta on kuitenkin varsin työlästä ja erittäin riskialtis virheille. Ensimmäiset varsinaiset palvelinkeskeiset versionhallintajärjestelmät kehitettiin 1970-luvulla. Nämä automatisoivat monia ennen käsin tehtyjä versionhallintamenetelmiä (Chason & Straub 2014).

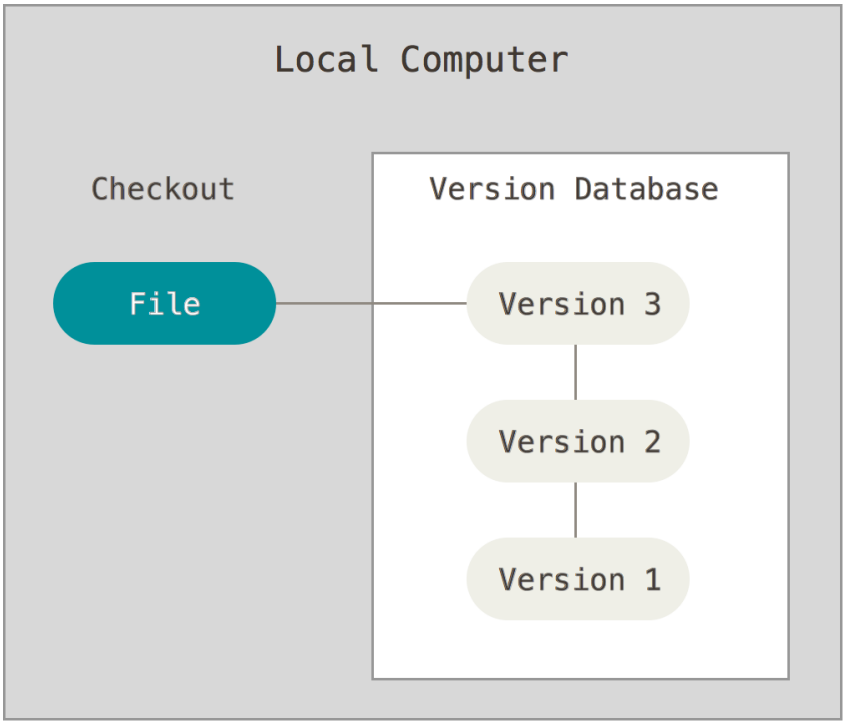
Yksi tärkeimmistä tavoitteista versionhallinnassa on se, että ohjelmistoprojekteja voidaan kehittää useamman projektinjäsenen kanssa yhtäaikaisesti ilman ristiriitoja. Ohjelmistoprojektia voidaan myös eri versioilla viedä eri suuntiin esimerkiksi jos ohjelmaa pitää räätälöidä eri alustoilla toimivaksi tai eri asiakkaiden tarpeisiin sopivaksi, mikäli eri versioihin on esimerkiksi lisätty jotain uusia ominaisuuksia (Chason & Straub 2014).

Yleisin versionhallintatapa, jota kaikki ovat varmaan joskus käyttäneet, on paikallisesti omalla koneella tehty versionhallinta. Tässä käyttäjä manuaalisesti kopioi ohjelmistoprojektin tai dokumentin eri kansioihin tai eri nimellä ja dokumentoi tiedoston version sen nimellä. Esimerkiksi käyttäjä tekee muutoksia tekstitiedostoon ja nimeää sen *projekti10.2016.doc* ja uudemman version *projekti3.2017.doc* ja niin edelleen. Vastaavanlaisesti ohjelmistokehityksessä projektin eri tiedostot voivat olla esimerkiksi nimetty tyyliin *projekti1.php*, *projekti2.php* ja *projektiFinal.php*. Tällöin projektinvaiheet ja tiedostot on dokumentoitu (Lenkkeri 2013).

Projektia on myös viety eteenpäin tallentamalla uusia projektin vaiheita ilman, että vanhoja projektin vaiheita olisi ylikirjoitettu uusilla tiedostoilla. Tämä on versionhallintatavoista kaikkein yksinkertaisin. Se on myös erittäin altis virheille, jos esimerkiksi käyttäjä on vahingossa nimennyt tiedos-

ton virheellisesti, tallentanut sen väärään kansioon, jatkanut projektia väärään versioon, tallentanut väärän tiedoston päälle tai vahingossa poistanut väärän tiedoston. Versionhallintajärjestelmät kehitettiin pääasiassa tällaisten virheiden välttämiseksi (Chason & Straub 2014).

Ensimmäiset versionhallintaohjelmat olivat paikallisesti toimivia yksinkertaisia tietokantoja. Nämä ohjelmistot säilyttävät tiedoston eri versiot kiintolevyllä erikoisformaatissa, jolloin vanhempiin versioihin voidaan tarvittaessa palata. Suosituin näistä on *RCS* (Revision Control System). Paikallisen versionhallintajärjestelmän toimintaperiaate on hahmotettuna kuvassa 1 (Chason & Straub 2014).




Kuva 1. Paikallinen versionhallinta (Chason & Straub 2014, 28).

## 2.2 Versionhallinnan edut

Versionhallintajärjestelmän avulla kaikki projektin jäsenet voivat työstää mitä tahansa tiedostoa milloin tahansa, ja projektinjäsenten tiedostoihin tekemät muutokset voidaan yhdistää yhteiseen versioon. Kun koko projekti sijaitsee yhteisessä paikassa versionhallintajärjestelmässä, ei tarvitse huolehtia, missä projekti sijaitsee tai mikä sen uusin versio on (Chason & Straub 2014).

Ilman versionhallintajärjestelmää projektinjäsenet joutuisivat sopimaan keskenään projektin tiedostojen tallentamisesta. Esimerkiksi tallennetaanko vain muutetut tiedostot vai valmiit projektit. Pelkkien muutettuja

tiedostoja tallentaessa koko projektin seuraaminen on hankalaa. Valmiiden projektien tallentaminen taas vie paljon tilaa. Tämän lisäksi projektin jäsenten pitää sopia eri versioiden nimeämisestä. Eri versioiden nimeäminen manuaalisesti on kuitenkin työlästä ja altis virheille. Sekaannuksia voi tulla esimerkiksi, jos projektin samasta versiosta on useampi variantti. Ilman versionhallintajärjestelmää on myös vaikeaa seurata projektin eri versioihin tehtyjä muutoksia. Projektin jäsenet eivät usein dokumentoi kaikkia tekemiään muutoksia esimerkiksi projektin kansiossa olevaan Readme-tiedostoon (Chason & Straub 2014).

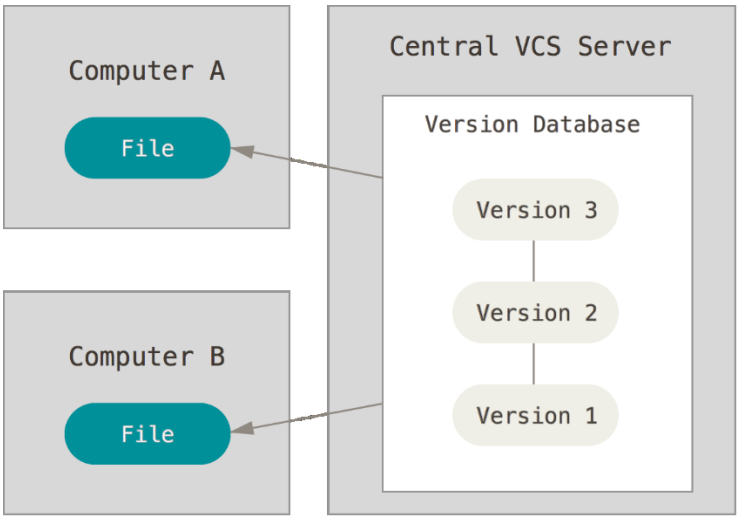
Versionhallintajärjestelmän käyttämällä on mahdollista palata tiedoston tai koko projektin vanhempaan versioon käden käänteessä. Tämä tekee projektityöskentelystä vähemmän riskialtista, koska kaikki tehdyt virheet voidaan korjata helposti. Versionhallintajärjestelmän avulla projektiin tehtyjen muutosten seuraaminen on helpompaa ja esimerkiksi Gitissä projektin jäsenet voivat kirjoittaa lyhyen viestin tehdessään muutoksia projektiin. Hajautettua versionhallintaa, kuten Gitiä käytettäessä jokaisella projektin jäsenellä on myös oma henkilökohtainen kopio koko projektin historiasta. Tällöin, mikäli esimerkiksi keskitetty palvelin tai oma kovalevy hajoaa, voidaan koko projekti vain kopioida jonkun toisen projektinjäsenen paikallisesta Git repositorysta (Chason & Straub 2014).

# 3 KESKITETYT JA HAJAUTETUT VERSIONHALLINTAJÄRJESTELMÄT

## 3.1 Keskitetyt versionhallintajärjestelmät

Keskitetyt versionhallintajärjestelmät (CVCS) kehitettiin ohjelmistokehityksen projektin jäsenten yhteistyön helpottamiseksi. Näissä versionhallintajärjestelmissä on yksi keskitetty tietokantapalvelin, joka sisältää kaikki projektin versiot. Sieltä käyttäjät voivat hakea tarvitsemansa tiedostot. Tällaisia järjestelmiä ovat esimerkiksi Perforce, Subversion ja CVS. Keskitetyt versionhallintajärjestelmät ovat toiminnaltaan keskittyneet tiedostojen varmuuskopioimiseen, synkronisointiin ja jäljitettävyyteen (Chason & Straub 2014).

Keskitetyn versionhallinnan etuna on se, että kaikki projektin jäsenet voivat seurata toistensa toimintaa ja näin tietää, miten projekti etenee. kaikki käyttäjät näkevät saman tietokannan sen sijaan, että kaikki käyttäjät toimisivat paikallisen tietokannan kanssa. Järjestelmänvalvojan näkökulmasta eri käyttäjien käyttöoikeuksien hallinta on helpompaa keskitetyissä järjestelmissä. Keskitettyjen versionhallintajärjestelmien toiminta on hahmotettuna kuvassa 2 (Chason & Straub 2014).



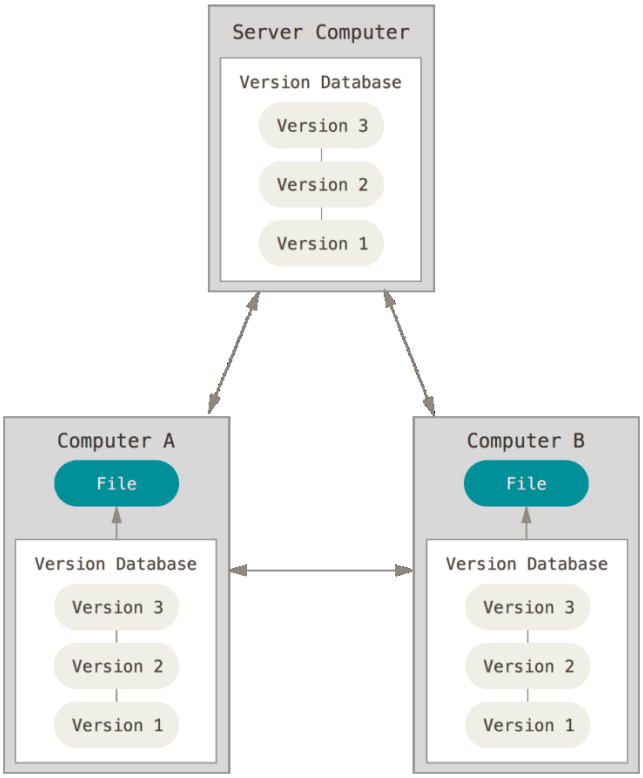

Kuva 2. Keskitetty versionhallinta (Chason & Straub 2014, 29).

Keskitetyn versionhallinnan haittapuolet liittyvät sen etuihin eli palvelimelle keskitettyyn tietokantaan. Jotta käyttäjät voisivat tallentaa muutoksia tietokantaan on heidän saatava yhteys tietokantaan. Mikäli palvelin on alhaalla, kukaan projektin jäsenistä ei voi sillä hetkellä päivittää projektia. Keskitetyssä tietokannassa on myös se riski, että mikäli tietokanta korruptoituu, on riskinä se, että menetetään koko projekti lukuun ottamatta projektin jäsenten omilla kiintolevyillä olevia versioita (Chason & Straub 2014).

### 3.2 Hajautetut versionhallintajärjestelmät

Hajautetussa versionhallinnassa (DVCS) jokaisella käyttäjällä on oma paikallinen kopio tietokannasta, josta muutokset voidaan viedä keskitettyyn tietokantaan ja jakaa muiden käyttäjien tietokantaan. Tämäntyyppinen versionhallinta on viime aikoina yleistynyt. Hajautettua versionhallintaa käyttäviä järjestelmiä ovat esimerkiksi Darcs, Bazaar, Mercurial ja Git (Chason & Straub 2014).

Hajautetun versionhallinnan etuna on se, että jokaisella käyttäjällä on oma henkilökohtainen projektin tietokanta ohjelmiston muokkaamiseen ja haaroittamiseen. Koska muutoksia muokataan paikallisesti, ei palvelimen alhaalla olo ole ongelma. Yhteyttä tarvitaan vain muutosten jakamiseen. Hajautettujen versionhallintajärjestelmien toiminta on hahmotettuna kuvassa 3 (Chason & Straub 2014).




Kuva 3. Hajautettu versionhallinta (Chason & Straub 2014, 30).

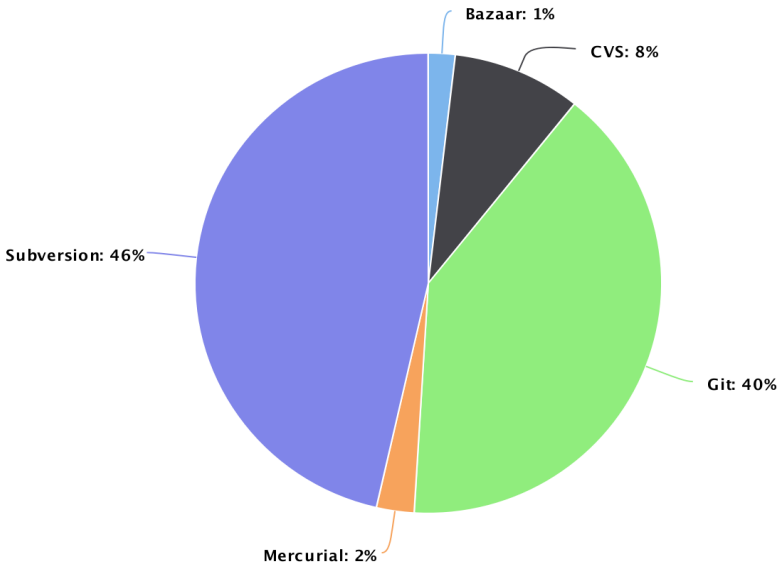
Haittana hajautetussa versionhallinnassa voidaan nähdä se, että sitä käytettäessä tarvitaan todennäköisesti joka tapauksessa keskitetty palvelin. Ensinnäkin sitä voidaan käyttää varmuuskopiointiin, mikäli muilla käyttäjillä ei ole jonkun toisen tekemiä muutoksia. Toiseksi ilman keskitettyä tietokantaa on vaikea tietää kenellä on ohjelmiston uusin versio (Lenkkeri 2013).

# 4 GIT VERSIONHALLINTAJÄRJESTELMÄ

## 4.1 Yleistä

Git on Linus Torvaldsin ja hänen kehitystiiminsä vuonna 2005 kehittämä hajautettu versionhallintajärjestelmä Linux kernelin kehittämisprojektia varten. Vuosina 1991–2002 ohjelman muutokset liikkuivat kehittäjältä toiselle pakattuina tiedostoina. Vuosina 2002–2005 projektissa käytettiin yksityistä Bitkeeper-nimistä hajautettua versionhallintajärjestelmää. Bitkeeperin lisenssiehtojen muutoksen vuoksi järjestelmän käyttöä ei voitu kuitenkaan jatkaa, jolloin Bitkeeperin kehittäneen yrityksen ja Linux kernelin kehitystiimin yhteistyö päättyi. Bitkeeperiä vastaavaa hajautetun versionhallinnan järjestelmää ei kuitenkaan ollut saatavilla, joten Linus Torvalds päätti kehittää samantyyppisen versionhallintajärjestelmän itse käyttäen Bitkeeperistä opittuja kokemuksia esimerkkinä. Tavoitteena oli luoda täysin hajautettu versionhallintajärjestelmä, joka oli nopea, tukee epälineaarista ohjelmistokehitystä ja mahdollistaa useat kehityshaarat ja joka soveltuisi suuriin projekteihin, kuten Linux kernelin kehittämiseen (Paksula & Luontola 2009).

Gitin suosio versionhallintajärjestelmänä on kasvanut viime vuosien aikana merkittävästi, ja se on tällä hetkellä toiseksi suosituin versionhallintajärjestelmä heti Subversion jälkeen 40 %:n osuudella kaikista versionhallintajärjestelmistä. Vuonna 2010 tuo osuus oli vain 11 %. Mikäli Gitin suosion kasvu jatkuu samalla tavalla, tulee siitä lähiaikoina kaikkein suosituin versionhallintajärjestelmä (O'Grady 2015). Black Duck Open Hub-sivusto pitää kirjaa siitä, kuinka paljon eri versionhallintajärjestelmiä käytetään sen mukaan kuinka paljon projekteja kyseisissä versionhallintajärjestelmissä on. Kyseistä kaaviota päivitetään jatkuvasti ja se on näkyvissä kuvassa 4. Se ei ole aivan tarkka, koska on vaikea sanoa, kuinka paljon projekteista on tällä hetkellä aktiivisia. Sitä voi kuitenkin pitää jonkin verran suuntaa antavana tietona.

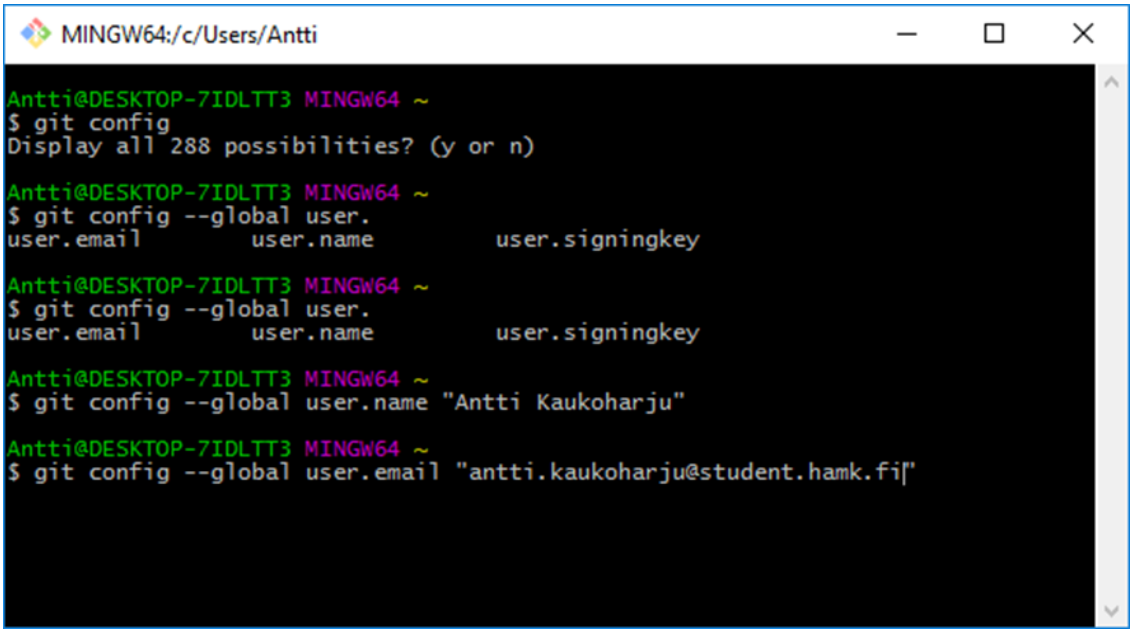



Kuva 4. Versionhallintajärjestelmät (Black Duck Open Hub 2017).

## 4.2 Git versionhallintajärjestelmän asennus

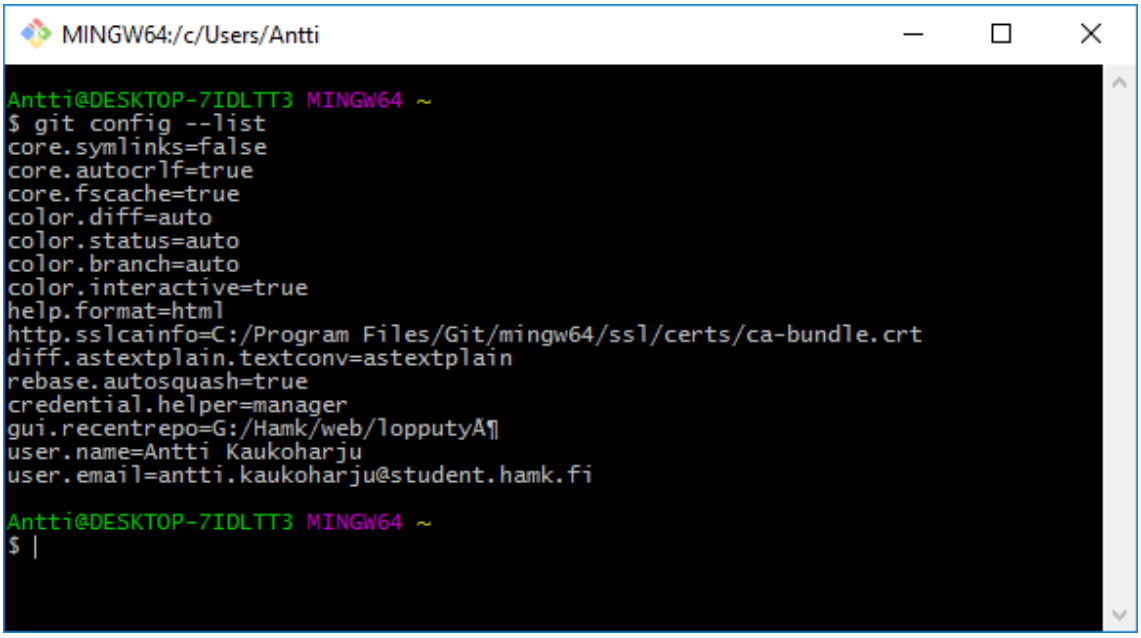
Git-ohjelmiston voi asentaa niin Windowsille, Macille kuin useisiin eri Linuxin versioihin. Linuxille asennettaessa voidaan Git asentaa suoraa lähdekoodista konsolikomennolla, mutta esimerkiksi Windowsissa voidaan Gitin kotisivuilta ladata valmis asennuspaketti. Ohjelmassa voidaan käyttää joko Linuxin komentoja käyttävää komentokehotetta (Git Bash), Windowsia käyttäessä Windowsin omia komentoja käyttävää komentokehotetta (Git CMD) tai vaihtoehtoisesti voidaan käyttää ohjelmiston mukana tulevaa graafista käyttöliittymää (Git GUI).

Ensimmäisenä asiana Gitiin kannattaa lisätä omat tietonsa. Tässä tapauksessa tämä tarkoittaa oman nimen ja sähköpostin lisäämistä. Tällöin käyttäjän Gitillä projektiin tekemät muutokset voidaan tunnistaa tietyn käyttäjän tekemiksi. GitBashin avulla näiden tietojen lisääminen onnistuu komennoilla esimerkiksi kuten kuvassa 5 on näytetty kirjoittamalla **git config --global user.name "Etunimi Sukunimi"** Käyttäjänimeä varten ja **git config --global user.email ”sähköpostiosoite@esimerkki.com”** sähköpostiosoitetta varten.




Kuva 5. Git Bash. omien tietojen lisääminen gitin komentokehotteen kautta, kuvakaappaus.

Tämän jälkeen omat tiedot voi tarkistaa esimerkiksi kuvan 6 tapaan komennolla **git config –list**.

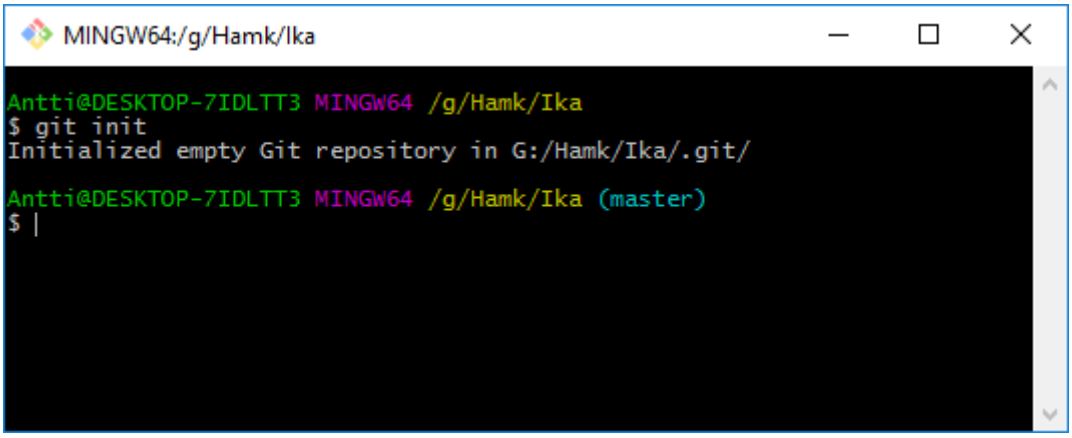



Kuva 6. GitBashissa käytettävä Git config –list-komento näyttää kaikki käyttäjän asetukset, kuvakaappaus.

### 4.3 Projektin lisääminen Gitiin

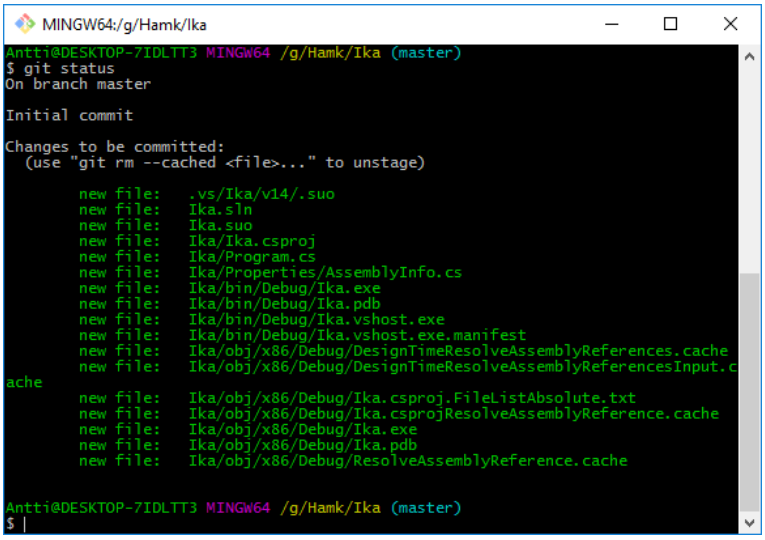
Projekti voidaan tuoda Gitiin kahdella tavalla. Ensimmäinen tapa on valita esimerkiksi kovalevyllä sijaitseva projekti ja tuoda se Gitiin. Toisessa tavassa Gitiin kloonataan projektin tietolähde jostain versionhallintapalvelusta.

Paikallista jo olemassa olevaa projektia tuotaessa GitBashilla aluksi ajetaan kuvan 7 esimerkillä **git init**-komento projektin hakemistossa. Tämä komento luo kansioon piilotetun .git-kansion, johon Git tallentaa tarvittavaa dataa.



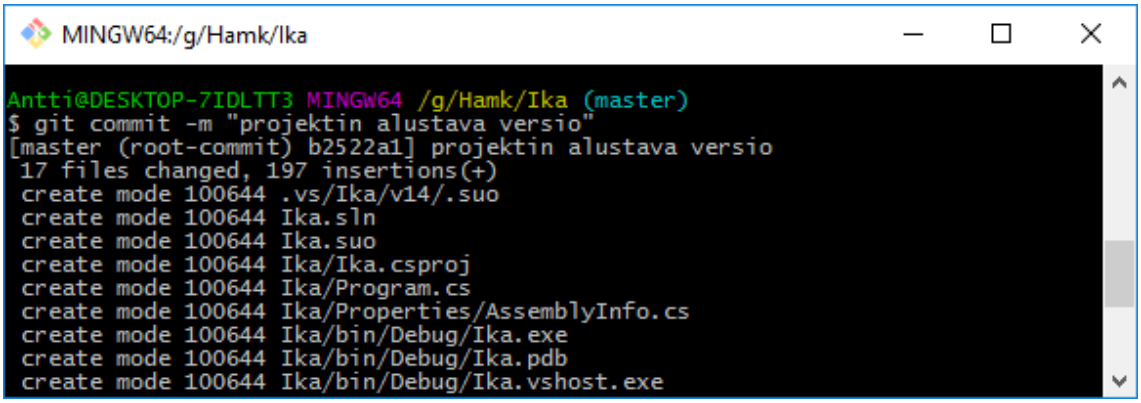

Kuva 7. Projektin alustus GitBashissa, kuvakaappaus.

Tämän jälkeen lisätään projektin tiedostot seurattaviksi **git add**-komennolla, tässä tapauksessa lisätään kaikki projektin tiedostot antamalla **git add .**-komento ja tarkistetaan tiedostojen tila **git status**-komennolla, kuten kuvassa 8.



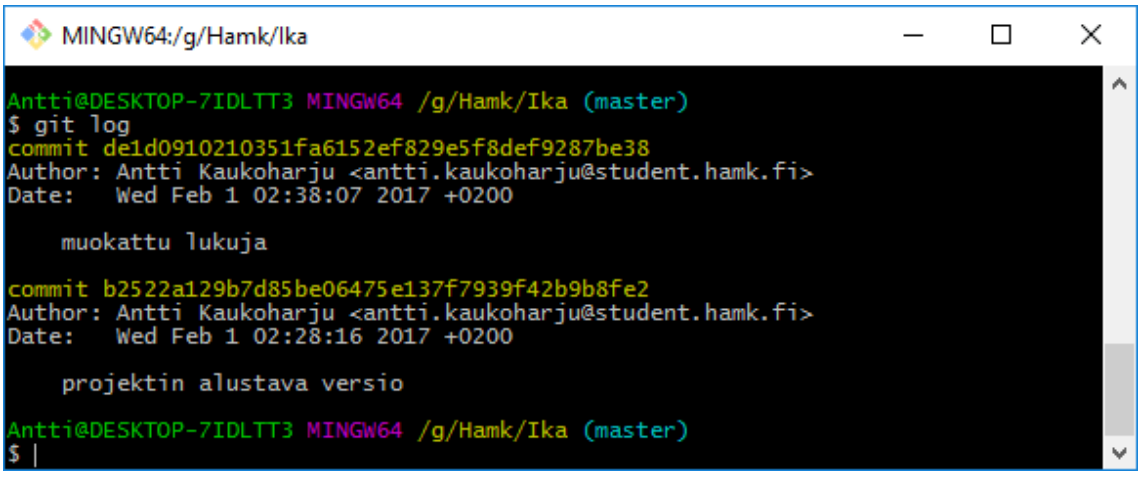

Kuva 8. Seuratut tiedostot näkyvät GitBashissa vihreänä, seuraamattomat tiedostot näkyisivät punaisina, kuvakaappaus.

Tämän jälkeen projekti siirretään versionhallintaan **git commit**-komennolla. Samalla komennolla commitoidaan eli vahvistetaan myös projektiin tehdyt muutokset. Onnistunut committi versionhallintaan on näytetty kuvassa 9.




Kuva 9. Commit –m-komennolla tehtyihin muutoksiin voi kirjoittaa kommentin, kuvakaappaus.

Sen jälkeen, kun projektiin on tehty muutoksia annettaan **git add**- ja **git commit**-komennot uudelleen. Tämän jälkeen käyttäjä saa kuvan 10 mukaisesti kaikki valittuun projektiin tehdyt muutokset, näiden muutosten tehneen nimen, koska muutos on tehty ja käyttäjän committiin liittämän viestin näkyviin listana antamalla komennon **git log**.




Kuva 10. Git log-komennolla saadaan näkyviin projektiin tehdyt muutokset, kuvakaappaus.

# 5 VERSIONHALLINTAPALVELUT

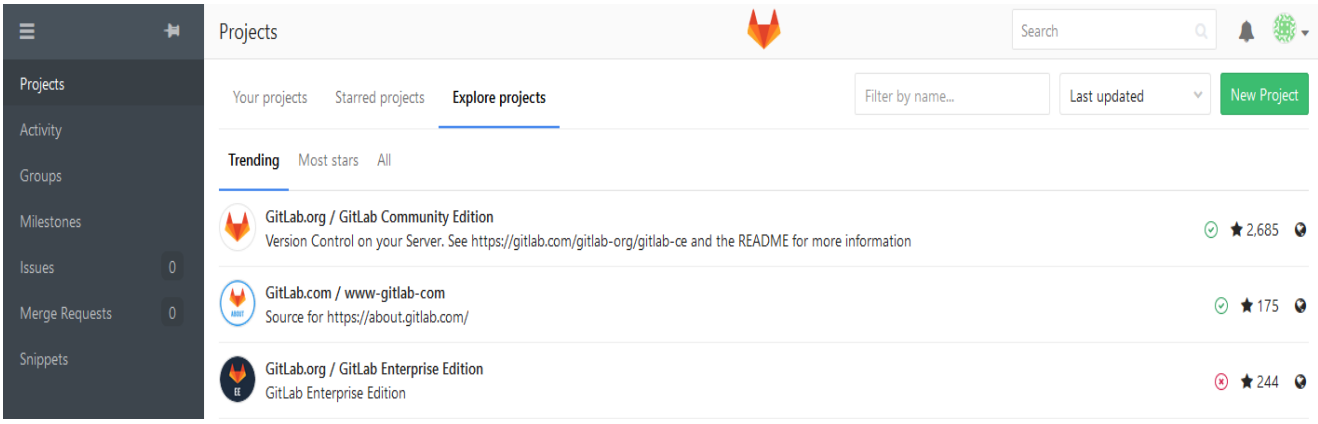
Sen sijaan, että ohjelmistoprojektin kehittäjät pitäisivät yllä omaa palvelinta, ovat monet niistä siirtyneet käyttämään erilaisia web-pohjaisia versionhallintapalveluita. Nämä versionhallintapalvelut muistuttavat toiminnaltaan eräänlaisia sosiaalisen median palveluita ja ne toimivat Git-versionhallintaohjelman ja vastaavien palveluiden eräänlaisina graafisina käyttöliittyminä. Ne myös sisältävät projektihallintaan soveltuvia työkaluja ja antavat graafista tietoa esimerkiksi projektien edistymisestä sekä lähdekoodin tiedostojen sisällöstä ja niihin tehdyistä muutoksista. Tämä helpottaa projektiin tehtyjen muutosten seuraamista, kun on graafista materiaalia siitä, kuka projektinjäsen on tehnyt projektiin muutoksia, minkälaisia muutoksia on tehty ja mihin aikaan. Näistä palveluista kuka tahansa käyttäjä pystyy kopioimaan itsellensä minkä tahansa julkiseksi määritellyn avoimen lähdekoodin projektin tietokannan. Tämän jälkeen käyttäjä voi aloittaa kehittämään omaa versiota projektistaan ja sitten tarjota omaa versiotaan alkuperäiselle projektitiimille, joka voi sitten päättää ottaako se tehdyt muutokset vastaan. Tällaisia palveluita ovat Esimerkiksi Bitbucket, Github ja Gitlab (Viikon valo 2013).

Versionhallintapalveluista vuonna 2007 toimintansa aloittanut ja noin 600 henkilöä työllistävä Github on kaikkein suurin. Vuonna 2017 sillä on 19 miljoonaa käyttäjää ja se ylläpitää yli 52 miljoonaa ohjelmistoprojektia (Github 2017).
Opetuskäytössä Githubin ongelmana on se, että palvelun ilmaisversiossa projektitietokannat ovat aina julkisia eli avointa lähdekoodia eikä niitä saa yksityisiksi eli suljetuksi lähdekoodiksi, kuin palvelun maksullisessa versiossa. Github ei myöskään ole itse avoimen lähdekoodin projekti vaikka se ylläpitää suurimman määrän avoimen lähdekoodin projekteja (Medium 2016).

# 6 GITLAB VERSIONHALLINTAPALVELU

Gitlab on avoimen lähdekoodin versionhallintapalvelu Git-versionhallinnalle. Gitlabia käyttää yli 100 000 organisaatiota ympäri maailman. Sitä käyttävät myös monet tunnetut organisaatiot, kuten NASA, Alibaba ja O'reilly Media (Degeler, 2014).

Palvelusta on olemassa ilmaiseksi selainpohjainen Online-versio osoitteessa Gitlab.com. Toinen ilmainen vaihtoehto on koneelle ladattava Community Edition-versio. Tämä versio tosin toimii ainoastaan muutamalla Linuxin versioista, erimerkiksi Ubuntulla ja Debianilla. Kuvassa 11 on esimerkkinä ruudunkaappaus Gitlabin etusivun projektinäkymästä, jossa käyttäjä voi selata avoimen lähdekoodin projekteja. Ne ovat identtiset Gitlabin Online-versiossa ja Community Edition versiossa




Kuva 11. Gitlabin etusivun näkymä, kuvakaappaus.

Gitlab on ominaisuuksiltaan lähes identtinen Githubin kanssa. Myös Gitlabin käyttöliittymä ja yleinen ulkonäkö ovat Githubin kanssa samankaltaisia. Erona on se, että Gitlab on avointa lähdekoodia, johon kuka tahansa voi päästä käsiksi ja muokata haluamallaan tavalla. Tällöin kaikki Gitlabin käyttäjät voivat olla mukana kehittämässä ohjelmaa. Toisin kuin Githubin ilmaisversiossa voi käyttäjä luoda Gitlabin ilmaisversiossa yksityisiä suljetun lähdekoodin projekteja. Gitlabiin voi tuoda helposti projekteja toisista versionhallintapalveluista suoraan (Technology Conversations 2015).

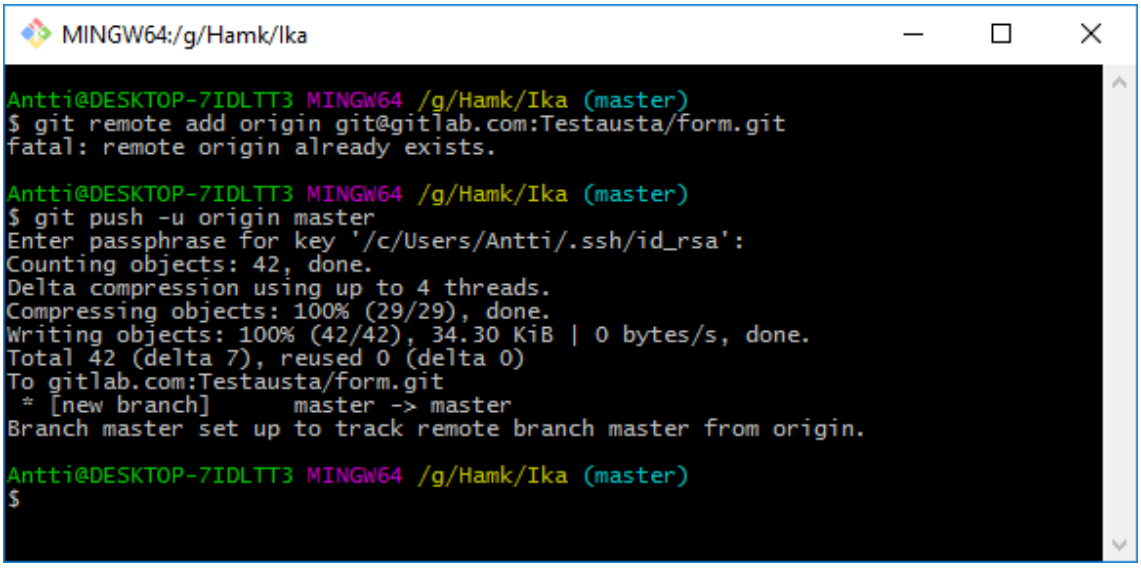
Monissa toisissa versionhallintapalveluiden ilmaisversioissa on mahdolliset projektinjäsenten ja projektien määrä rajattu. Esimerkiksi Bitbucketin ilmaisversiossa projektinjäsenten määrä on rajattu viiteen. Gitlabin ilmaisversiossa taas projekteja ja projektinjäseniä voi olla rajattomasti (van der Voort 2015).

Rajoitteeksi Gitlabissa voidaan nähdä se, että se tukee ainoastaan Git-versionhallintajärjestelmää, kun taas GitHub ja Bitbucket tukevat monenlaisia eri versionhallintajärjestelmiä. Tämän vuoksi Gitlabiin siirtyminen voi olla

ongelmallista, mikäli projektissa on käytetty jotain muuta versionhallintajärjestelmää kuin Gitiä. Toisaalta Git on tällä hetkellä yksi suosituimmista versionhallintajärjestelmistä, joten tämä ei välttämättä ole ongelma (Medium 2016).

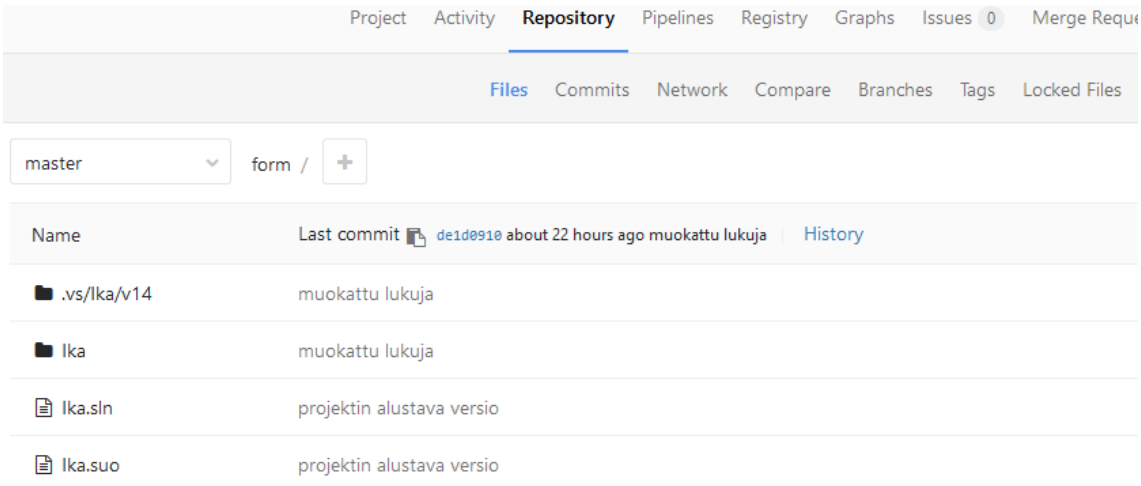
## 6.1 Projektin lisääminen Gitlabiin

Jotta projektin voi lisätä Gitista Gitlabiin täytyy ensin luoda Gitlabiin käyttäjätunnus. Tämän jälkeen annetaan kuvan 12 esimerkin mukaan komento git remote add origin (gitlabin projektikansion osoite) ja sen jälkeen **git push –u master**-komento. Mikäli komento on mennyt läpi, pitäisi onnistuneen Gitlabin palvelimelle työnnön näkyä kuvan 12 mukaan.




Kuva 12. Projektin lisäämiseen tarvittavat komennot, kuvakaappaus.

Onnistuneet projektin lisäämisen jälkeen pitäisi projektin tiedostot ja projektin muut tiedot näkyä Gitlabin projektivalikon alla kuvan 13 mukaisesti.




Kuva 13. Lisätty projekti Gitlabissa, kuvakaappaus.

Projekteja voi myös kloonata itselleen esimerkiksi Gitlabista Gitiin paikallisesti muokattavaksi. Jotta tämä kuitenkin onnistuisi, täytyy kyseisen projektin olla julkinen eli avoimen lähdekoodin projekti. Tämän valitun projektin kloonaus paikalliseen käyttöön onnistuu antamalla GitBashissa komento **git clone https://serveri/nimi/projekti.git**.

## 6.2 Gitlab Community Edition versionhallintapalvelu

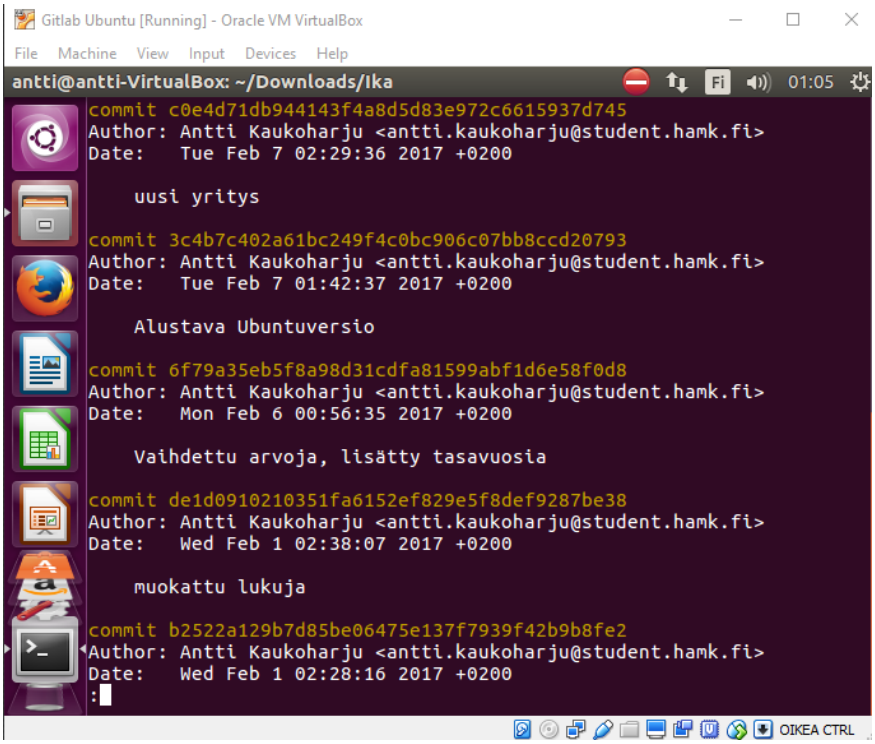
Gitlab Community Edition on Gitlab versionhallintapalvelun koneelle asennettava avoimen lähdekoodin versio. Sen avulla käyttäjä voi luoda itselleen ja projektitiimilleen sisäisen käyttäjänhallintapalvelimen. Se toimii usean eri Linux-käyttöjärjestelmän kanssa, mutta Windows-versiota kyseisetä ohjelmasta ei ole. Tällä hetkellä Gitlabin suosittelema käyttöjärjestelmä on Ubuntu 16.04, mutta ohjelmasta on versiot myös vanhemmille Ubuntun versioille. Muita tuettuja käyttöjärjestelmiä ovat muun muassa Debian, CentOS, OpenSUSE ja Rasberry.

Tätä opinnäytetyötä varten Gitlabin Community Editionin testaamiseen käyttöjärjestelmäksi valittiin Gitlabin suosittelema Ubuntu 16.04. Tämä käyttöjärjestelmä asennettiin Virtualbox-ohjelmistolla luodulle virtuaalikoneelle. Virtuaalikonetta luodessa pitää muistaa, että Gitlab on suhteellisen raskas ohjelma. Tämän vuoksi virtuaalikoneelle kannattaa määritellä muistia vähintään 2 prosessosiydintä, 2Gb muistia ja kovalevytilaa 10 GB.

## 6.3 Gitlab Community Editionin asennus

Gitlabin Online-version käyttöönotto on suhteellisen helppoa. Käyttäjän tarvitsee vain kirjautua Gitlab.comin kautta, lisätä palveluun SSH-avain, luoda uusi projekti ja lähettää oma projektinsa Gitlabiin Git-versionhallintajärjestelmän avulla. Gitlabin Community-version asentaminen on jonkin verran monimutkaisempaa.

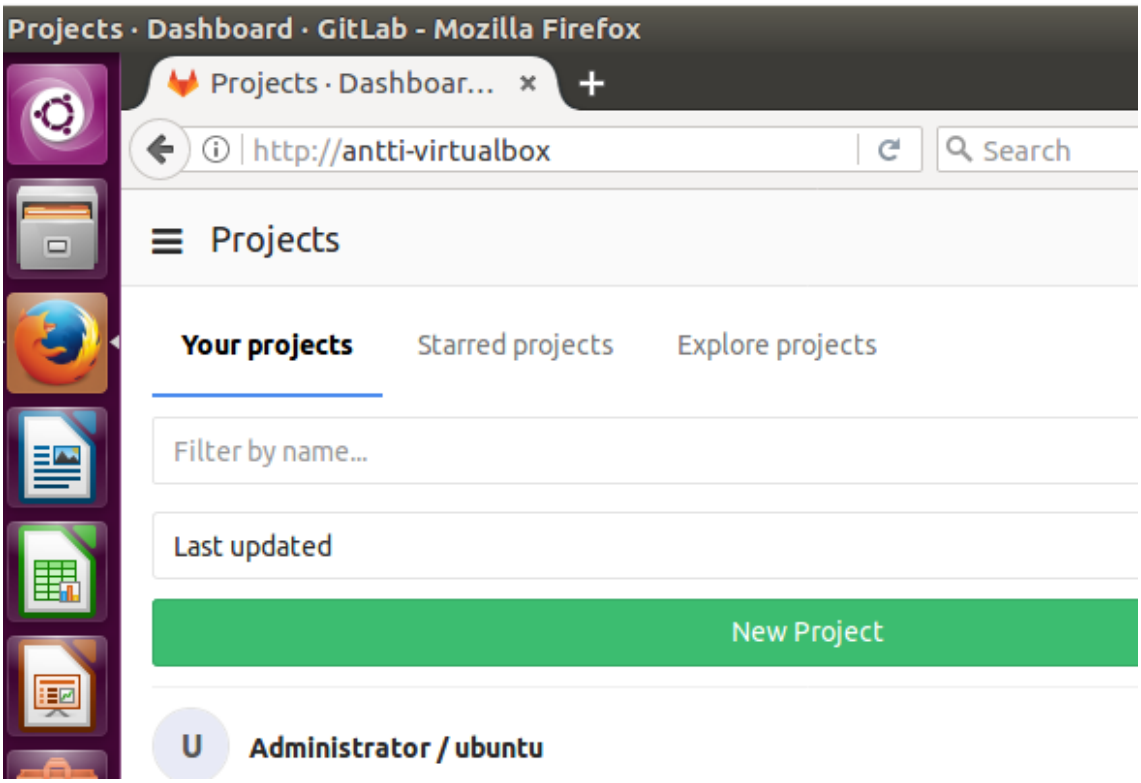
Ensinnäkin käyttöjärjestelmään täytyy asentaa luonnollisesti Git-sisällönhallintajärjestelmä. Tämä onnistuu antamalla Ubuntun terminaaliin komennot **sudo apt-get update** ja **sudo apt-get install git**. Käyttäjänimi ja sähköposti asetetaan samoilla **git config --global user.name "Nimi"** ja **git config --global user.email "sähköposti"**-komennoilla, kuten ennenkin. Vaikka projekti onkin siirretty fyysisesti toiseen käyttöjärjestelmään, **Git log**-komennolla kuvan 14 mukaisesti saa näkyville myös toisella koneella projektiin tehdyt muutokset.




Kuva 14. Testiprojekti siirrettynä virtuaalikoneelle, kuvakaappaus.

Gitlabia asentaessa ensiksi mennään terminaalilla **/tmp**-kansioon. Tässä kansiossa ladataan asennusskripti antamalla komento **curl -LO** https://packages.gitlab.com/install/repositories/gitlab/gitlab-**ce/script.deb.sh** joka ajetaan antamalla komento sudo **bash /tmp/script.deb.sh**. Näillä komennoilla saadaan palvelin käyttämään Gitlabin repositoreja. Tämän jälkeen voidaan varsinaisesti asentaa Gitlab. Tämä onnistuu komennolla **sudo apt-get install gitlab-ce**. Ennen kuin Gitlabia voidaan käyttää pitää se vielä konfiguroida antamalla komento **sudo gitlab-ctl reconfigure**. Seuraavaksi pitää vielä tarkistaa, että Gitlabia voidaan käyttää käyttöjärjestelmän palomuurin läpi. Tämä onnistuu antamalla komennot **sudo ufw allow http**, **sudo ufw allow OpenSSH** ja tämä tarkistetaan komennolla **ufw status**. Virtuaalikoneella tätä vaihetta ei kuitenkaan tarvinnut tehdä.

Tämän jälkeen voidaan mennä omaan Gitlabiin selaimella. Osoite on oman domainin nimi. Virtuaalikoneen tapauksessa domain oli nimeltään **antti-virtualbox**, kuten kuvassa 15 voidaan nähdä. Ensimmäisenä käyttäjän pitää säätää salasana adminkäyttäjälle. Tämän jälkeen käyttäjä kirjautuu palveluun käyttäjänimellä root ja asettamallaan salasanalla.




Kuva 15. Gitlabin Community Editionin aloitusruutu, kuvakaappaus.

## 6.4 Projektin lisääminen Gitlab Community Edition versioon

Samalla tavalla, kuin Gitlabin Online-versiossa, ennen kuin Gitlabiin voidaan tuoda projekteja Git-versionhallintajärjestelmästä, pitää käyttäjän lisätä SSH-avain järjestelmään. Mikäli käyttöjärjestelmässä ei ole valmiina SSH-avainta tämä saadaan luotua käyttöjärjestelmään antamalla terminaalissa komento **ssh-keygen**. Luotu avain saadaan näkyviin komennolla **cat ~/.ssh/id_rsa.pub**. Tällä komennolla näkyviin saatu pitkä merkkirivi kopioidaan kokonaisuudessaan Gitlabin profiiliasetuksissa olevaan SSH Keys kohdan alta löytyvään tekstikenttään. Tämän jälkeen käyttäjän pitäisi pystyä tuomaan projektejaan Gitlabiin. Osoitteeksi projektien työntämiseksi määritellään esimerkiksi **http://domain/root/projekti.git.**

## 6.5 Gitlabin Online- ja Community Editionin eroavaisuudet

Gitlabin Online-versio on Gitlab.comissa sijaitseva selainpohjainen palvelu, joka ylläpitää projekteja omalla serverillään. Käyttäjän ei tarvitse ladata erikseen mitään ohjelmaa sitä käyttääkseen, joten se toimii kaikilla käyttöjärjestelmillä. Online-versiossa käyttäjä voi selata kaikki sivustolle ladattuja avoimen lähdekoodin projekteja ja kloonata niitä itselleen. Käyttäjä voi

myös luoda rajattomasti uusia projekteja ja lisätä niihin haluamansa määrän uusia käyttäjiä.

Gitlab Community Edition versio on koneelle ladattava ohjelmisto, joka toimii käyttäjän omalla palvelimella. Sen voi asentaa ainoastaan joillekin Linuxin käyttöjärjestelmän versioille. Koska Community edition versio järjestelmästä pyörii omalla palvelimellaan, käyttäjä voi selata vain kyseisellä palvelimella sijaitsevia avoimen lähdekoodin projekteja. Community Edition on tosin mahdollista yhdistää Gitlabin Online-versioon. Community Edition voi olla hyvä vaihtoehto, mikäli haluaa rankentaa palvelun omalle palvelimelle. Tässä on etuna parempi tietoturva ja se, että Online –version kaatumiset eivät haittaa projektityöskentelyä.

Ominaisuuksiltaan Gitlabin Online- ja Community Edition versio ovat käytännössä identtiset. Yksi opinnäytteen tavoite oli selvittää, minkälaisia yhteenvetoja ja raportteja nämä järjestelmät tekevät versionhallinnan avulla. Molempien järjestelmien yhteenvetoja ja raportteja tutkiessani tulin siihen johtopäätökseen, että molempien versioiden yhteenvedot esimerkiksi graafisten diagrammien ja käyttäjien aktiivisuuden osalta ovat täysin samanlaisia. Pelkästään ominaisuuksien osalta ei siis voi ainakaan sanoa, kumpi olisi hyödyllisempi eri tilanteissa.

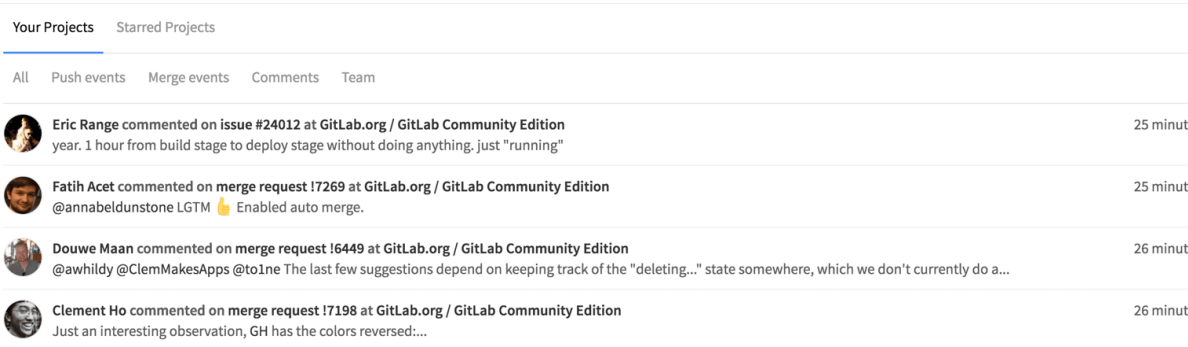
# 7 GITLABIN OMINAISUUKSIA PROJEKTIN SEURANTAAN

## 7.1 Ryhmätasolla

Gitlabiin projekteja tuodessa käyttäjä voi luoda projekteja varten ryhmiä eli Groups. Tähän ryhmän alle lisätään kaikki projektinjäsenet ja sen projektit. Ryhmiä voi luoda useampia ja niihin voi laittaa omat käyttäjät, jotka voivat lisätä ryhmiin omat projektinsa. Ryhmiin voi myös siirtää olemassa olevia projekteja toisista ryhmistä. Ryhmän jäsenille voi ryhmänäkymästä jakaa tavoitteita ja luoda tehtäviä valituille ryhmänjäsenille. Projektit voivat olla luonteeltaan joko julkisia tai yksityisiä.

### 7.1.1 Ryhmän aktiivisuus

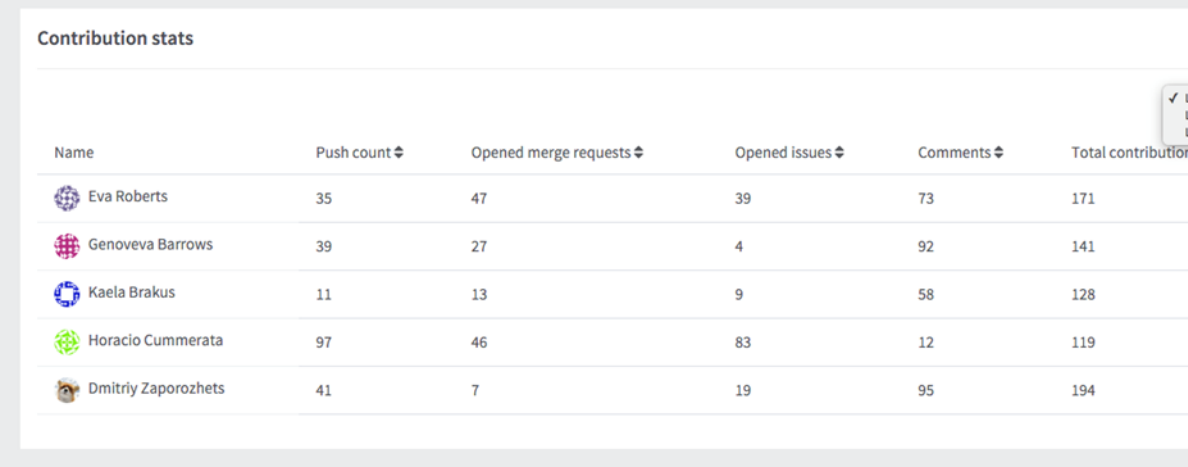
Projektien edistymisen tarkastelu on myös mahdollista projektitasolla. Kuvassa 16 kuvaruutukaappauksessa näkyvän Activity painikkeen alta löytyy koko projektitiimin tämän ryhmän sisällä tekemät lisäykset ja muutokset eri projekteihin. Tämän listauksen alta löytyy kuka on työntänyt muutoksia mihin projektiin ja mihin kehityshaaraan, kaikkiin projekteihin lisätyt kommentit, yhdistämispyynnöt, luodut tavoitteet ja tehtävät sekä mihin aikaan kyseiset aktiviteetit on tehty.




Kuva 16. Projektinjäsenten aktiivisuus ryhmätasolla, kuvakaappaus.

### 7.1.2 Ryhmän työpanosanalyysi

Toinen projektitiimin työskentelyn seurantaan ryhmätasolla soveltuvat yhteenvedot löytyvät Contribution Analytics painikkeen alta. Tämän painikkeen alta löytyy graafisena käyränä kuinka monta työntöä palvelimelle kukakin käyttäjä on projektiin tehnyt, kuinka monta yhdistämispyyntöä kukakin käyttäjä on tehnyt ja kuinka monta tehtävää kukakin käyttäjä on sulkenut. Viimeisenä tämän painikkeen alta löytyvistä tiedoista on kuvassa 17 näkyvä lista kaikista ryhmän jäsenistä ja tiedot heidän tekemien työntöjen, tehtävien avausten, sulkemisten, yhdistämispyyntöjen ja lopussa on listattu kokonaistyöstöjen määrästä.

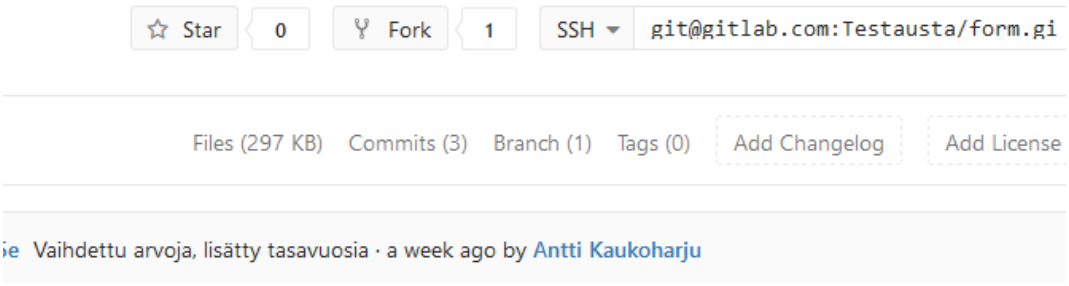



Kuva 17. Kaikkien ryhmäjäsenten aktiivisuus listana, kuvakaappaus.

## 7.2 Projektitasolla

Grouppien eli ryhmien alle käyttäjä voi luoda projekteja. Kuten ryhmät projektit voivat olla luonteeltaan joko julkisia (public), sisäisiä (internal) tai yksityisiä (private). Julkisia projekteja voi kuka tahansa kloonata itselleen ilman palveluun rekisteröitymistä tai muunlaista varmennusta. Sisäisiä projekteja voi kloonata itselleen kuka tahansa palveluun rekisteröitynyt käyttäjä. Yksityisiä projekteja taas voi kloonata vain sellaiset käyttäjät, joille on annettu erikseen oikeudet kyseiseen projektiin.

Projektit ovat versionhallintapalvelimella, kuten Gitlabissa sijaitsevia projektin tietokantoja tai repositoreja. Kaikki projektit kuuluvat joko johonkin ryhmään tai ne kuuluvat jollekin tietylle käyttäjälle. Mikäli projekti kuuluu johonkin tiettyyn ryhmään, eri käyttäjien oikeudet projektiin määritellään ryhmän asetuksista. Mikäli projekti kuuluu jollekin tietylle käyttäjälle, määrittelee tämä käyttäjä muiden käyttäjien oikeudet tähän projektiin.




Kuva 18. Projektin etusivun tietoja, kuvakaappaus.

Projektin tiedoissa Projects-painikkeen alta löytyy kuvan 18 kuvaruutukaappauksessa näkyvä projektin etusivu ja sen yleiset tiedot. Täältä löytyy esimerkiksi kuka on muokannut viimeksi projektia, milloin ja projektinjäsenen mahdollisesti työnnön yhteydessä jättämä viesti. Tämän lisäksi etusivulta löytyy tietoa ja linkkejä esimerkiksi projektin tiedostoista, tehdyistä

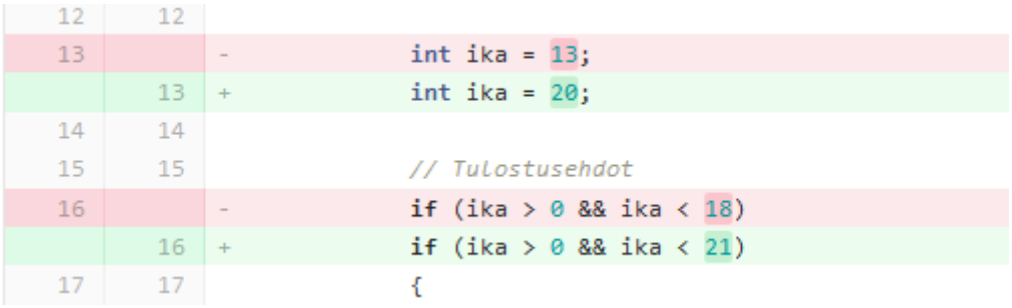
muutoksista ja kehityshaaroista. Projektin etusivulle löytyy usein myös lyhyt kuvaus projektista. Tämän lisäksi Wiki painikkeen alle käyttäjät voivat kirjoittaa pidemmän kuvauksen työstämästään projektista.

### 7.2.1 Projektin aktiivisuus

Projektin Activity-valikon alla olevat tiedot ovat ryhmätasoa vastaavat. Nämä aktiviteettitiedot tosin luonnollisesti koskevat tässä tilanteessa vain kyseiseen projektiin tehtyjä muutoksia.

### 7.2.2 Projektin tiedostojen muutokset

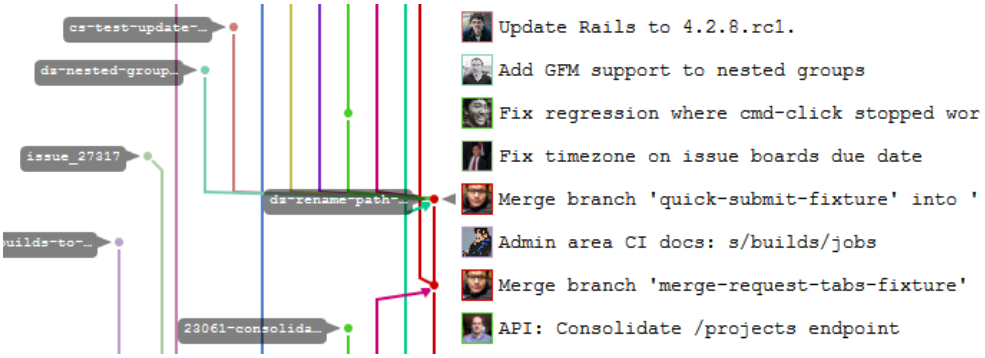
Repositoryvalikon alta löytyvät kaikki projektin tiedostot ja tietoa niihin tehdyistä muutoksista. Commitsin alta löytyy aktiviteetti ainoastaan projektiin tehtyjen muutosten eli committien kohdalta. Näitä klikkaamalla voi käyttäjä nähdä kuvan 19 mukaisesti mitä eroa tällä versiolla on edelliseen versioon verrattuna. Käyttäjät voivat myös lisätä omia kommenttejaan haluamalleen koodirivistölle tarvittaessa. Branches-valikon alta löytyy projektiin tehdyt muutokset projektin haarautumisien ja näiden haarakehitysten yhdistämiseen pääprojektiin osalta.




Kuva 19. Commitin muutosnäkymä. Poistetut tiedot näkyvät punaisena. Lisätyt tiedot näkyvät vihreänä, kuvakaappaus.

### 7.2.3 Projektin haarautumiset

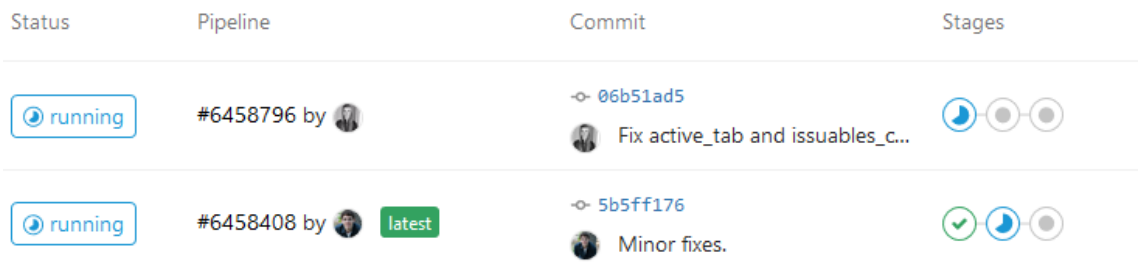
Networks-valikon alta löytyy myös tietoa projektin haarautumisesta. Tämän valikon alta löytyvät tiedot projektin haarautumisista ja näiden haarautumisien yhdistämisestä pääprojektiin, kuten Branches-valikossakin. Erona on se, että Networks-valikon alla nämä tiedot kuvataan graafisina käyrinä, josta näkee kuvan 20 esimerkkikuvan mukaisesti tietoa, siitä kuka projektinjäsen on tehnyt muutoksia, näiden muutosten viestit sekä kuvataan graafisena viivana kaikki haarautumiset pääprojektista sekä näiden haarojen yhdistymisistä. Haarautumiskaavio on ehkä enemmän eduksi suurissa projekteissa, sillä ohjelmointikurssien perusprojekteissa tuskin on kovin montaa kehityshaaraa.




Kuva 20. Projektin haarautumiset ja pääprojektiin yhdistämiset kuvattuina graafisesti, kuvakaappaus.

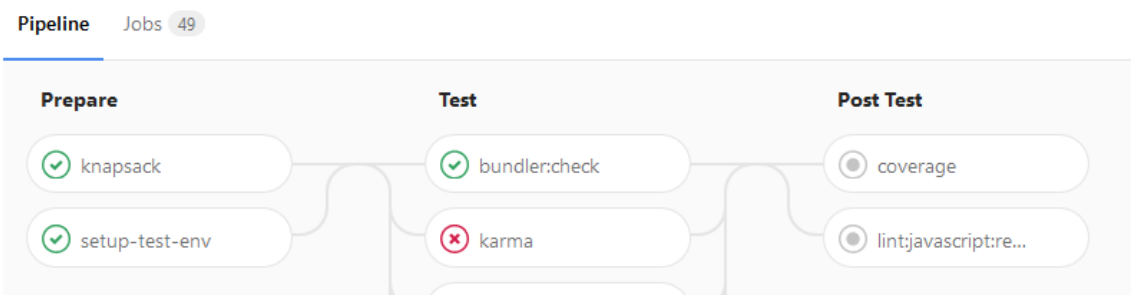
### 7.2.4 Projektin työstövaiheet

Pipelines-valikon alta löytyy kuvassa 21 näkyvä tietoa eri projektin työstövaiheiden tilanteesta. Etusivulla on listauksena projektin uusimmat kehitysvaiheet. Täältä saa tietoa siitä mihin kehitysversioon kyseinen kehitysvaihe liittyy, kuka sitä työstää, mitä vaiheita kyseisessä kehitysvaiheessa on ja mikä niiden tilanne on.




Kuva 21. Lista eri kehitysvaiheista (pipelines) ja niiden tilanteesta, kuvakaappaus.

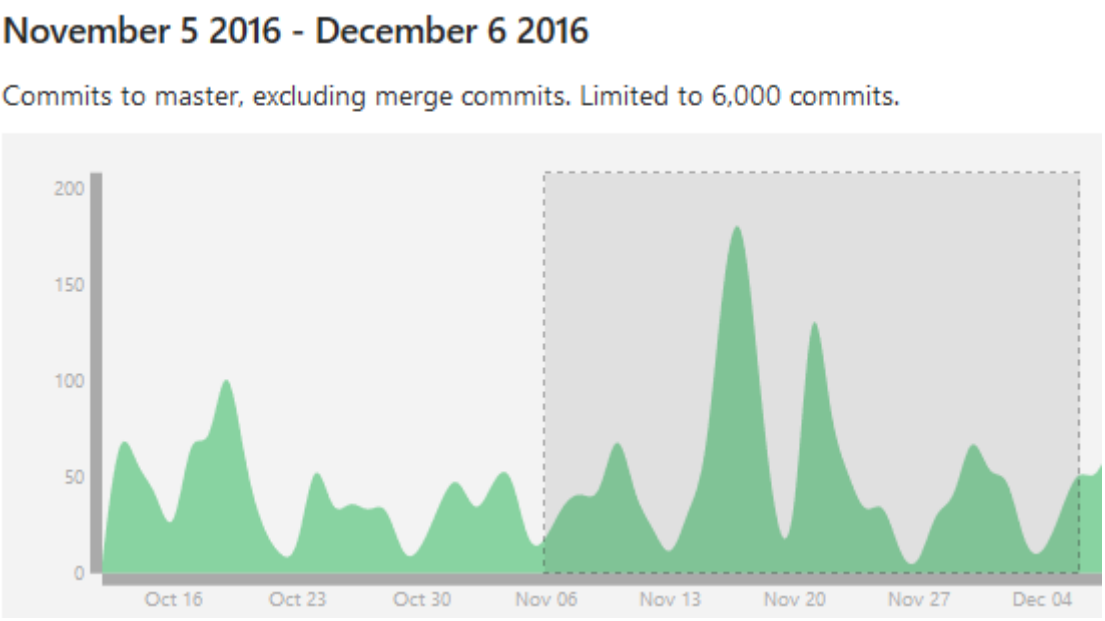
Klikkaamalla jotain kehitysvaihetta saa sen eri vaiheet näkymään kuvassa 22 näkyvänä graafisena kaaviona. Tästä kaaviosta näkyy esimerkiksi minkälaisia alkuvalmisteluja kyseistä kehitysvaihetta varten on tehty, minkälaisia testejä kyseisessä kehitysvaiheessa on tehty ja minkälaisia testien jälkeisiä toimenpiteitä kehitysvaiheessa on tehty. Tästä kaaviosta näkee myös ovatko testit onnistuneet, epäonnistuneet vai onko niille tehty jotain muuta. Klikkaamalla eri vaiheita saa niistä tehdyistä tehtävistä tarkempaa tietoa.




Kuva 22. Esimerkkikaavio kehitysvaiheesta (pipeline), kuvakaappaus.

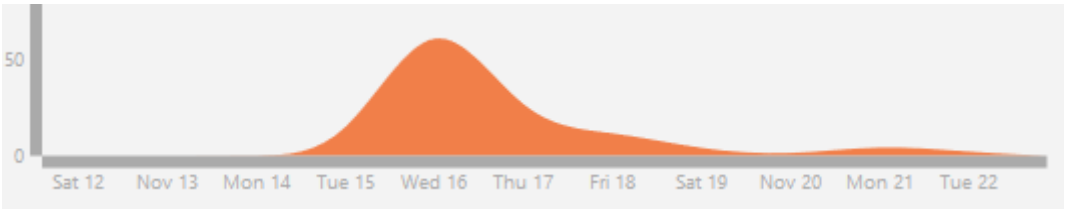
### 7.2.5 Projektin kaaviot

Projektin Graphs-valikon alta löytyy erilaisia projektin edistymistä ja muunlaista yleistä tietoa sen työstämisestä. Ensimmäisenä Contributors-valikon alta löytyy kaavio projektiin koko sen aikana tehtyjen muutosten eli committien määrästä. Käyttäjä voi hiirellä vetämällä valita pääkaaviosta kuvan 23 mukaan jonkun tietyn aikavälin ja tarkastella yksittäisten käyttäjien tällä aikavälillä tekemiä muutoksia. Käyttäjät voivat valita myös haluavatko he tarkkailla tehtyjä muutoksia pääprojektitasolla vai jonkun kehityshaaran mukaan.




Kuva 23. Muutosten kokonaismäärä tietyltä aikaväliltä, kuvakaappaus.

Tämän kaavion alla on lista kaaviosta, jossa näkyvät kaikkien yksittäisten projektinjäsenten tekemät muutokset. Näistä yksi näkyy kuvassa 24. Nämä tiedot näkyvät siltä aikaväliltä, minkä käyttäjä on valinnut pääkaaviosta.




Kuva 24. Yksittäisen käyttäjän tekemät muutokset valitulla aikavälillä, kuvakaappaus.

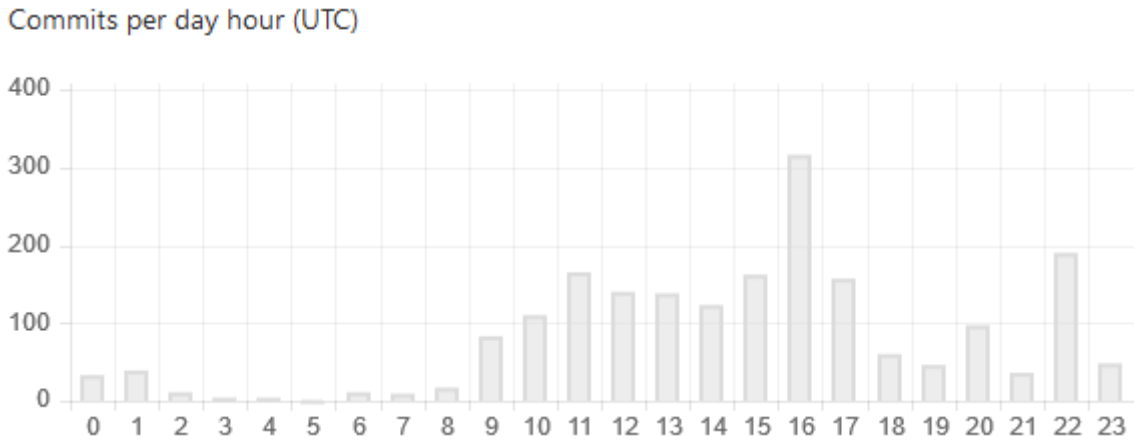
Commits-valikon alta löytyy tietoa projektiin tehdystä muutoksista kuukauden tarkkuudella. Tässäkin tilassa käyttäjä voi valita tiedot joko pääprojektin tai eri kehityshaarojen mukaan. Ensimmäisenä on kuvassa 25 näkyvää yleistä tietoa siitä, kuinka monta muutosta projektiin on tehty tiettyllä aikavälillä, kuinka monta muutosta projektiin on tehty keskimäärin päivässä ja kuinka monta henkilöä on työstänyt projektia kyseisellä aikavälillä.

Commit statistics for **master** Jan 09 - Feb 17

- **2000** commits during **39** days
- Average **50** commits per day
- Contributed by **163** authors

Kuva 25. Yleistiedot projektin muutoksista, kuvakaappaus.

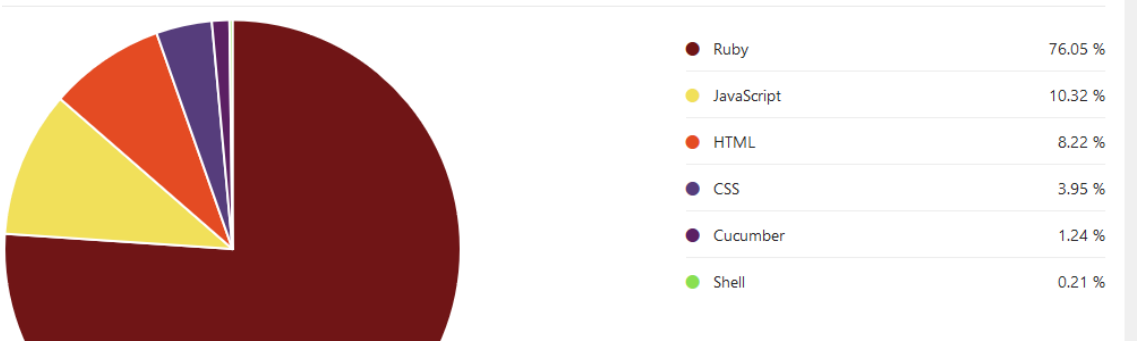
Tämän lisäksi Commits-valikon alta löytyy kaavioita siitä, milloin projektiin on tehty muutoksia. Näistä voi nähdä, minä kuukaudenpäivänä, minä viikonpäivänä tai kuvan 26 mukaan mihin kellonaikaan muutoksia on tehty eniten tai vähiten. Laittamalla hiiren kaavion tiedon päälle saa käyttäjä tarkan luvun, kuinka monta muutosta projektiin on tuona aikana tehty.




Kuva 26. Projektiin tehdyt muutokset kellonajan mukaan, kuvakaappaus.

Languages-valikon alta löytyy tietoa siitä, mitä ohjelmointikieliä projektissa on käytetty. Nämä näkyvät kuvan 27 mukaisesti sekä listana prosenttimäärinä, sekä ympyräkaaviona. Tämä on todennäköisesti myös enemmän

suurten projektien kannalta hyödyllistä tietoa. On epätodennäköistä, että ohjelmointikurssien projekteissa käytettäisiin useampaa ohjelmointikieltä.

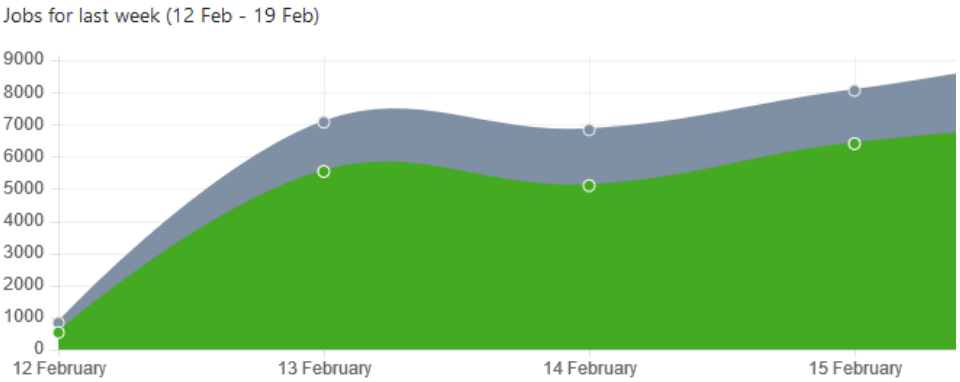



Kuva 27. Projektissa käytetyt kielet, kuvakaappaus.

Continuous Integration-valikon alta löytyy tietoja projektin työstövaiheista. Esimerkiksi kuvan 28 mukaan kuinka monta työstövaihetta projektiin on tehty, kuinka monta niistä on onnistunut, kuinka monta niistä on epäonnistunut ja onnistuneiden työstövaiheiden prosenttiosuus.


Kuva 28. Projektin työstöjen onnistumiset, kuvakaappaus.

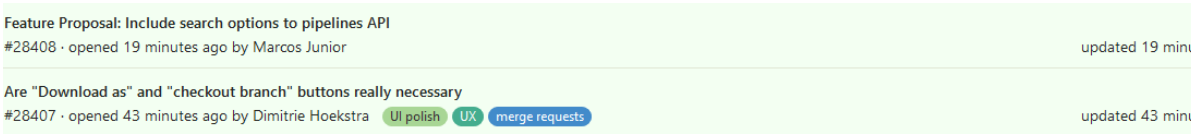
Tämän lisäksi Continuous Integration-valikon alla on kaavioita, joissa näkyy kokonaistyöstöjen ja onnistuneiden työstöjen määrä. Nämä näkyvät vuoden, kuukauden ja kuvan 28 mukaisesti viikon tarkkuudella.




Kuva 29. Projektin työstöt viikon tarkkuudella, kuvakaappaus.

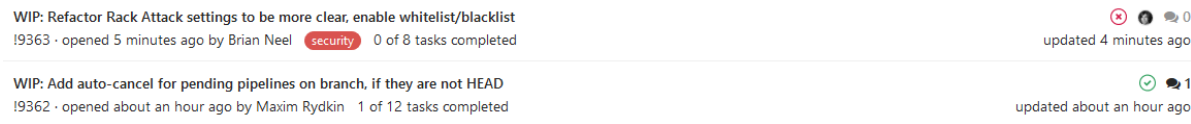
### 7.2.6 Projektin tehtävät ja yhdistämispyynnöt

Viimeisenä projektitasolla nähtäviä tietoja ovat projektin tehtävät, jotka löytyvät Issues-valikon alta ja projektin yhdistämiset, jotka löytyvät Merge request-valikon alta. Nämä molemmat ovat lista projektin tehtävistä ja yhdistämispyynnöistä. Näistä tiedoista näkyy tehtävien osalta esimerkiksi kuvan 30 mukaisesti lista uusimmista avatuista tehtävistä, niiden tiedot ja kuka on avannut kyseisen tehtävän.




Kuva 30. Projektin tehtävät, kuvakaappaus.

Merge request-valikon alta löytyy kuvassa 31 näkyvä listaus projektin kehityshaarojen työstövaiheista. Näistä näkyy esimerkiksi kyseisen työstövaiheen nimi, koska se on luotu, kuinka monta vaiheen tehtävää on valmiita, kenelle tämä työstövaihe on annettu tehtäväksi. Tämän lisäksi tästä listauksesta näkee onko kyseinen työstövaihe käynnissä, onnistunut tai epäonnistunut.




Kuva 31. Projektin työstövaiheita, kuvakaappaus.

# 8 YHTEENVETO

Projektien etenemisen seurannan kannalta Gitlab-versionhallintapalvelu voidaan pitää hyödyllisenä työkaluna. Gitlabissa on monia erilaisia työkaluja, joilla projektien etenemistä voidaan seurata joko ryhmätasolla tai projektitasolla. Näistä tiedoista voidaan nähdä monella eri tavalla esimerkiksi milloin projektia on viimeksi muokattu, kuinka aktiivinen ryhmä on ollut koko projektin tai jonkun tietyn aikavälin aikana, milloin projektia on pääasiassa työstetty eteenpäin tai ketkä ryhmän jäsenet ovat olleet aktiivisimpia.

Osa Gitlabin kaavioista ja muista työkaluista on selkeästi suunnattu suuremmille projekteille, kuin mitä tietojenkäsittelyn koulutusohjelmassa yleensä on tehty. Esimerkiksi projektien haarautumisien ja projektissa käytettyjen kielien kaaviot eivät välttämättä ole niin hyödyllisiä. Projektien päivityksiin ja projektin jäsenten aktiivisuuteen liittyvät kaaviot taas voidaan nähdä erittäin hyödyllisiksi myös pienempien projektien seurantaan. Online –version ja Community Edition –version välillä ei projektinseurannan kannalta ollut juurikaan eroja, joten niiden välillä valitseminen riippuu enemmän käytännön tarpeista.

Versionhallinta ei ollut ennen opinnäytetyötä itselleni ollenkaan tuttu aihe, joten siitä tuli opittua paljon opinnäytetyön työstämisen ohessa. Asioita, joita yksin ei voinut testata liittyivät ryhmiin ja sen asetuksiin. Esimerkiksi on vaikea sanoa miten ryhmät kannattaa säätää, mikäli palvelua käytetään opetuksessa. Kannattaako esimerkiksi tehdä niin, että opettaja luo kaikki ryhmät ja liittää opiskelijat niihin, vai olisiko järkevämpää, että opiskelijat luovat itse ryhmät ja liittävät opettajan ryhmänjäseneksi. Tämän testaamiseen vaadittaisiin useampaa henkilöä.

Gitlab voidaan nähdä parhaimpana vaihtoehtona muihin versionhallintapalveluihin verrattuna. Sen ilmaisversiossa on kaikki tarpeelliset ominaisuudet, jotka muista versionhallintapalveluista löytyvät vain maksullisista versioista. Jos esimerkiksi on harkittu Githubin käyttöönottoa, Gitlabin ilmaisversiossa on kaikki projektinasetuksiin liittyvät ominaisuudet, jotka löytyvät vain Githubin maksullisessa versiossa.

# 9 LÄHTEET


Better explained. (n.d). *Intro to Distributed Version Control (Illustrated).* Haettu 10.1.2017 osoitteesta https://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/

Black Duck Open Hub. (2017). *Compare Repositories*. Haettu 17.1.2017 osoitteesta https://www.openhub.net/repositories/compare

Chacon, S. & Straub, B. (2014). *Pro Git*. Apress. Haettu 10.1.2017 osoitteesta https://progit2.s3.amazonaws.com/en/2016-03-22-f3531/progit-en.1084.pdf

Degeler, A. (2014). *How GitHub rival GitLab is building a business with just 0.1% paying customers* 4.6.2014. Haettu 18.1.2017 osoitteesta https://thenextweb.com/insider/2014/06/04/github-rival-gitlab-building-business-just-0-1-paying-customers/#.tnw_JNxJP8mA

GitHub (2017). *GitHub is how people build software*. Haettu 18.1.2017 osoitteesta https://github.com/about

Lenkkeri, J. (2013). *Versionhallinta teoriassa*. Haettu 10.1.2017 osoitteesta www.sugif.fi/arkisto2013/CVS_esitys.pptx

Luontola, E & Paksula, M. (2009). *Versionhallinta*. Tietojenkäsittelytieteen laitos. Helsingin yliopisto. Haettu 12.1.2017 osoitteesta https://www.cs.helsinki.fi/u/paksula/versionhallinta/s09/git.pdf

Medium (2016). *GitHub vs. Bitbucket vs. GitLab vs. Coding*. Haettu 18.1.2017 osoitteesta https://medium.com/flow-ci/github-vs-bitbucket-vs-gitlab-vs-coding-7cf2b43888a1

O'Grady, S. (2015). *DVCS and Git Usage in 2015*. Blogijulkaisu 24.11.2015. Haettu 16.1.2017 osoitteesta http://redmonk.com/sogrady/2015/11/24/dvcs-and-git-2015/

Technology Conversations (2015). *GitHub vs GitLab vs BitBucket Server (Formerly Stash)* 16.10.2015. Haettu 18.1.2017 osoitteesta https://technologyconversations.com/2015/10/16/github-vs-gitlabs-vs-bitbucket-server-formerly-stash/

Van der Voort, J (2015). *Bitbucket vs. GitLab.com* 15.4.2015. Haettu 19.1.2017 osoitteesta https://about.gitlab.com/2015/04/15/bitbucket-vs-gitlab-com/

Viikon valo (2013). *GitHub*. Blogijulkaisu 2.2.2013. Haettu 17.1.2017 osoitteesta http://viikonvalo.fi/GitHub/