

# **Versionhallinta ja Git**

Osa 1

# Sisältö

---

- Johdanto versionhallintaan.
- Git, GitHub ja GitLab.
- Git-järjestelmän tärkeimmät osat.
- Työprosessi Git-versionhallinnassa.
- Gitin asennus omalle koneelle.
  - Windows-järjestelmässä huomioitavaa.
- Harjoitustyön versionhallinnan aloitus.

# Johdanto

---

- Versionhallinta (version control) tarkoittaa tiedostojen eri kehitysvaiheiden (versioiden) tallentamista siten, että versioiden sisältö voidaan kuvailla (metadata) samoin kuin versioiden väliset erot.
  - Versionhallinta ei liity pelkästään ohjelmointiin, koska mistä tahansa dokumentista voidaan, ja on usein toivottavaa, tehdä versioita.
  - Ensimmäisillä ohjelmointikursseilla versionhallinnan hyödyt käyvät ilmi vähitellen harjoitustöiden yhteydessä.
    - Harjoitustyön tekeminen voi edetä siten, että siitä tehdään jokin osa valmiiksi, jonka jälkeen uusi versio tallennetaan omin versiota kuvaavin kommentein varustettuna vaikkapa omaan hakemistoonsa, minkä jälkeen työtä jatketaan. Samalla saadaan varmuuskopioita.

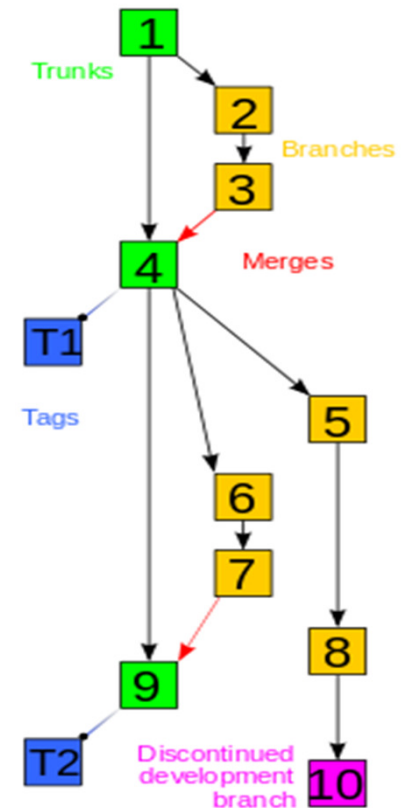
# Johdanto

---

- Versionhallintaan voidaan käyttää erillistä ohjelmistoa, versionhallintajärjestelmää.
  - Joissakin ohjelmistoissa on sisäänrakennettua versionhallintaa. Ohjelma voi palauttaa aiemman version peruuttamistoiminnolla (undo) tai se voi säilyttää muutaman edellisen version tiedostosta tallentamalla ne eri nimille.
- Järjestelmällinen versionhallinta on tärkeä osa laajempia ohjelmistoprojekteja, joissa saattaa työskennellä tuhansia ihmisiä.
  - Ryhmätyöskentely ei olisi mahdollista ellei eri henkilöiden eri aikoihin tekemiä muutoksia ohjelmistojen eri ja **samoihin** osiin voida hallinnoida ja yhdistää.

# Johdanto

- Ohjelmistojen kehitys etenee usein rinnakkain siten, että osaa ohjelmasta työstetään omassa sivuhaarassaan (branch) samalla, kun varsinainen työ etenee ohjelman päähaarassa (master, trunk).
- Sivuhaara yhdistetään (merge) päähaaraan, jos sivuhaaran tavoitteet saavutettiin.
  - Haarojen yhdistäminen voi olla vaikeaa.
- Tärkeät versiot voidaan nimetä tagilla.



Wikipedia.  
[https://commons.wikimedia.org/wiki/File:Revision\\_controlled\\_project\\_visualization-2010-24-02.svg](https://commons.wikimedia.org/wiki/File:Revision_controlled_project_visualization-2010-24-02.svg) (Luettu viimeksi 12.3.2019.)

# Git

---

- Perinteisen versiohallintajärjestelmän keskiössä on palvelin, joka säilöö keskitetysti eri versiot tietovarastoon (engl. repository, atk repo) ja pitää yllä versiohistoriaa.
  - Palvelimen asiakkailla on tyypillisesti käytettävissä vain osa projektin tiedostoista ja työskentely ilman palvelinta ei ole mahdollista.
- **Git** on hajautettu versionhallintajärjestelmä, jonka säilöö koko tietovaraston ja kaikki sen historiatiedot paikallisesti.
  - Projektin jäsenet voivat ohjelmoida paikallisesti ja yhdistää versionsa myöhemmin, jos palvelin on saavuttamattomissa.
  - Mahdollistaa perinteiseen palvelimeen sidottuun versionhallintaan verrattuna joustavampia työskentelytapoja.

# Git

---

- Git-ohjelmiston (2005–) luotiin Linux-käyttöjärjestelmän ytimen kehitystyön versionhallintaan.
  - Alkuperäinen kehittäjä ja ylläpitäjä Linus Torvalds.
  - Nimi viittaa osin Torvaldsiin (“ääliö”) ja osin varhaisen Gitin ominaisuuksiin (tyhmä, yksinkertainen, toimii toisinaan).
    - Nimi ei ole tässä enne; Git on oikeasti toimiva versionhallintajärjestelmä, jota käytetään laajalti avoimen lähdekoodin projekteissa ja ohjelmistoteollisuudessa.
  - Suunnittelun tavoitteena oli nopeus, koska Linux-ydintä kehitetään ahkerasti.
  - Taustalla UNIX-työkaluajattelu: Git on joukko työkaluja, joista kukin osaa jonkin asian hyvin.
-

# GitHub ja GitLab

---

- GitHub ja GitLab ovat ilmaisia verkkopalveluja, jotka tarjoavat helppokäyttöisen käyttöliittymän tietovarastoja hallinnoivaan palvelimeen.
  - Projektin versioita voidaan tallentaa GitHubin ja Gitlabin hallinnoimaan etätietovarastoon paikallisesta Git-tietovarastosta ja päinvastoin.
  - Sisältävät ryhmätyöskentelyä ja projektinhallintaa helpottavia graafisia työkaluja.
  - Paikallaan erityisesti usean henkilön projekteissa.
  - Etävarastoon tallennetut omat ohjelmat ovat nykyisin töitä haettaessa osa ansioluetteloa.



# GitHub ja GitLab

---

- Hervannan kampuksella on opetuskäyttöön tarkoitettu yksityinen GitLab-asennus (Tuni-GitLab), jota on käytetään tällä kurssilla.
- **Älä jaa harjoitustyötäsi kurssin aikana julkisesti** esimerkiksi GitLab-palvelussa.
  - Harjoitustyön saa julkaista maailmalle vasta lukuvuoden 2020–2021 alussa. Harjoitustyön saa tallentaa julkiseen etätietovarastoon aiemmin, jos tietovarasto on suojattu salasanalla.

# Git-järjestelmän tärkeimmät osat

---

- **Työhakemisto** (työpuu, työalue, worktree), jossa ovat projektin työstettävän version tiedostot ja hakemistot.
- **Valmistelualue** (indeksi, index, stage, cache), jonne liitetään seuraavaan versioon tallennettavat muokatut tiedostot.
- **Paikallinen tietovarasto**, johon tallennetaan versiot ja versiohistoria.
  - Sijaitsee työhakemiston *.git*-hakemistossa.
  - Uuden version tiedot luetaan valmistelualueelta ja tallennetaan paikalliseen varastoon.
- **Välivarasto** (stash) on “jemma”, jonne voi tallentaa väliaikaisesti.

# Git-järjestelmän tärkeimmät osat

---

- **Etätietovarasto** on ulkoinen kohde (esimerkiksi GitLab), jonne paikallisten tietovarastojen sisältöä voidaan tallennetaan muun muassa synkronointia varten.
- Etävarastoon tallentaminen on kurssin harjoitustyön tapaisissa yhden hengen projekteissa tärkeää lähinnä varmuuskopioinnin ja GitLabiin tutustumisen kannalta.
  - Etätietovarasto on keskeisessä asemassa kolmannen lukuvuoden Projektityö-kurssin kaltaisissa monen hengen projekteissa.
- Uudet projektit perustetaan usein etätietovarastosta kloonamalla.

# Työprosessi Git-versionhallinnassa

---

1. Perusta paikallisesti uusi työhakemisto tai kloonaa se etätietovarastosta.
2. Ohjelmoi eli muokkaa tiedostoja, joista osa tai kaikki ovat Gitin seuraamia, kunnes sopiva välitavoite on saavutettu.
3. Lisää seuraavaan versioon haluamasi tiedostot valmistelualueelle.
4. Tee uusi versio (commit).
5. Palaa 2. vaiheeseen niin pitkään kuin ohjelmoit tällä kertaa.
6. Tallenna työsi lopuksi etätietovarastoon.

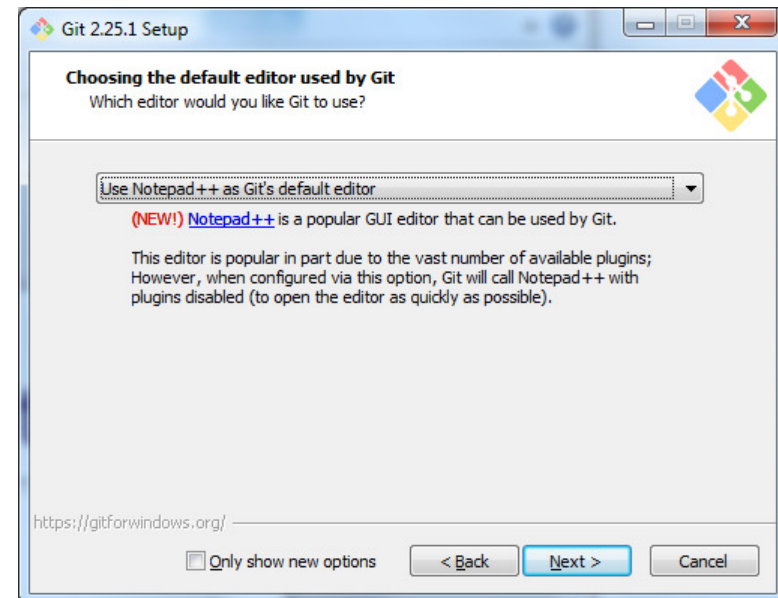
# Gitin asennus omalle koneelle

---

- Virallisimmat versiot löytyvät osoitteesta: <https://git-scm.com/downloads>, jossa on Git-toteutukset ja asennusohjeet yleisimmille käyttöjärjestelmille.
  - Asennusohjelma pyytää valitsemaan tekstieditorin, koska Git käynnistää sen joissakin tilanteissa automaattisesti.
  - Valitse jokin tuttu editori, jotta Gitin käyttö ei muutu tarpeettoman vaikeaksi hankalan editorin vuoksi.
  - Valintoja voi olla paljon, mutta usein oletusvalinta on niistä paras.
- Asenna siten, että voit käyttää Gitiä komentoiikkunan kautta.
- Näet onko Git jo asennettu koneellesi antamalla komentoiikkunassa komennon `git`

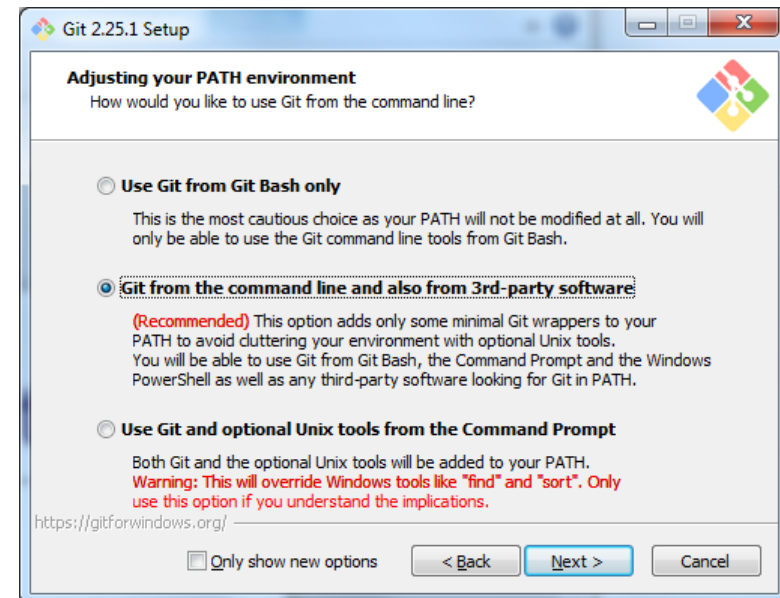
# Windows-järjestelmässä huomioitavaa

- Valitse editoriksi Vim tai Nano vain, jos tunnet ne ennestään.
  - Asennusohjelman listalla suosittuja Windows-editoreja valmiina, vaikka kaikkia niitä ei ole asennettu koneellesi. Keskeytä Git-asennus ja asenna editori, jos suosikkieditorisi on listalla, mutta ei koneellasi. Voit valita editoritiedoston myös käsin (“Select other editor as Git's default editor”).



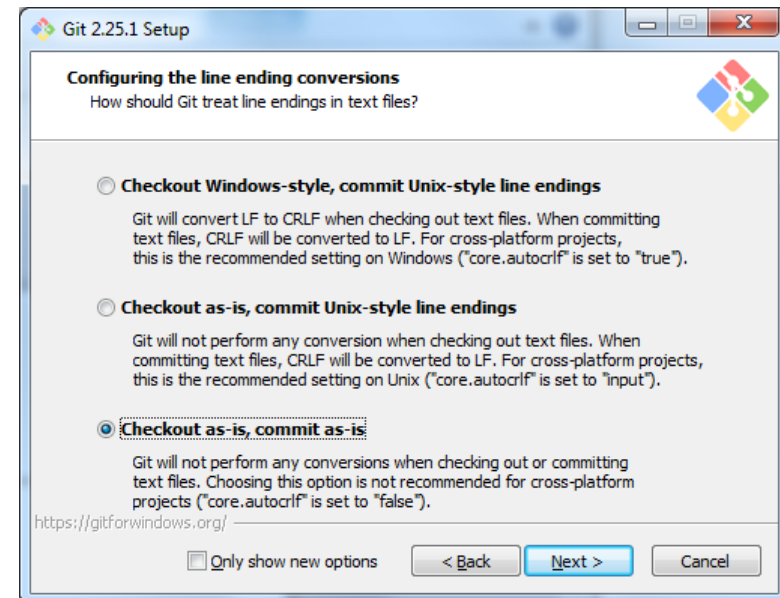
# Windows-järjestelmässä huomioitavaa

- Valitse kohdassa “Adjusting your PATH environment” oletus eli kohta “Git from command line and also from 3rd-party software”, jotta voit käyttää Gitiä tuttuun tapaan Windowsin komento-ikkunasta.
  - Valitse jotain muuta ainoastaan, jos \*nix-käyttöjärjestelmät ja Bash ovat hyvin hallussa.



# Windows-järjestelmässä huomioitavaa

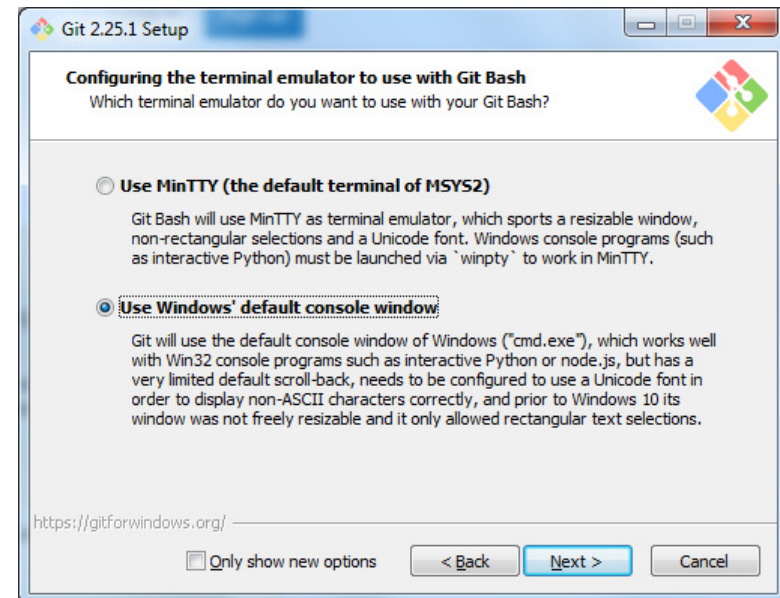
- Valitse kohdassa “Configuring the line ending conversions” eli “Checkout as-is, commit as-is”.
- Tarvitset jotain muuta vain, jos ohjelmoit sekä Windowsissa että jossain \*nix-käyttöjärjestelmässä.





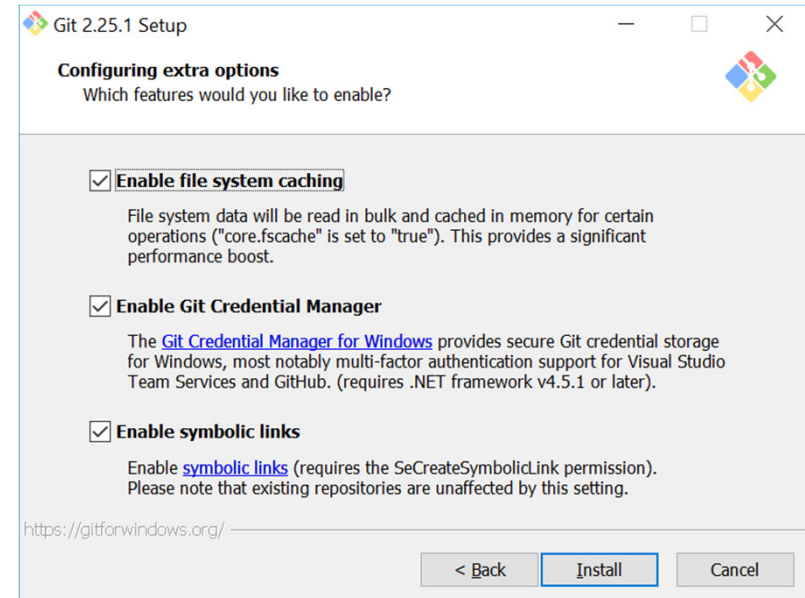
# Windows-järjestelmässä huomioitavaa

- Komentoikkunan valinnassa “Configuring the terminal emulator to use with Git Bash” oletusvalinta “**Use Windows’ default console windows**” on ainoa oikea, jos haluat Gitille tutun ja turvallisen Windows-komentoikkunan.
  - Vaitse muuta ainoastaan, jos aikomuksesi käyttää Bash-komentotulkkia.



# Windows-järjestelmässä huomioitavaa

- Valitsemalla “Enable Git Credential manager” etätietovaraston käyttö nopeutuu, koska voit tallentaa kirjautumistietosi Windowsin salasananhallintaan (Credential Manager).
  - Tuni-GitLabiin kirjaudutaan Tuni-tunnuksella ja sen salasanalla. Harkitse siksi tarkkaan onko kirjautumistietojen tallentaminen tarpeen.



- Ole tarkkana, jos päätät tallentaa kirjautumistietosi. Jos salasananhallintaan tallentuu väärä salasana, pääset etävarastoon vasta, kun olet käynyt korjaamassa salasanan.

# Gitin asetukset

---

- Aseta aluksi Gitiin oma nimesi ja sähköpostiosoitteesi, koska versionhallintaan säilötään aina tiedot version tallentajasta.
  - `global`-valinta asettaa tiedot kaikille projekteille. Ilman tätä valintaa sinulla on oltava tietovarasto, jossa teet asetukset.

```
git config --global user.name "Etunimi Sukunimi"
```

```
git config --global user.email "etunimi.sukunimi@tuni.fi"
```

- Voit tarvittaessa vaihtaa esimerkiksi editorin Gitin `config`-komennolla. Esimerkiksi:

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

- Komento `git config --list` tulostaa asetuksesi näytölle.

# Harjoitustyön versionhallinnan aloitus

---

- Tutustutaan Gitiin aloittamalla harjoitustyön versionhallinta.
  - Seuraavassa käsitellyt komennot käydään tarkemmin läpi luentorungon seuraavassa luvussa.
  - Vaikka Git-komennot annetaan Windows-järjestelmässä, ne toimivat samoin myös Mac- ja Linux-järjestelmissä.
    - Käyttöjärjestelmien omissa komennoissa on luonnollisesti edelleen eroja.
  - Tietovaraston perustamisessa on huomioitava projektihakemisto, jos harjoitustyön tekemiseen käytetään NetBeansia. Myös tästä kerrotaan myöhemmin lisää.
    - Gitiin voi tutustua nyt komentoikkunassa, vaikka tekisit projektisi myöhemmin NetBeansilla.

# 1) Tietovaraston perustaminen

---

- Perustetaan paikallinen tietovarasto kloonamalla se Tuni-GitLabin etätietovarastostasi.
  - Etätietovarasto sisältää valmiina annetut rajapinnat, joitakin hakemistoja ja kaksi tärkeää tiedostoa.
- Avaa komentoiikkuna ja siirry tämän kurssin työt sisältävään hakemistoon *x*, joka voi olla esimerkiksi *oope2\_2020*.
- Kloonaa tietovarasto hakemistossa *x* komennolla `git clone https://course-gitlab.tuni.fi/oope2-2020/tunnus`, missä tunnus on Tuni-tunnuksesi.
  - Anna Gitille Tuni-tunnuksesi ja sen salasana. Ole erityisen tarkka salasanan kanssa, jos tallennat kirjautumistietosi.

# 1) Tietovaraston perustaminen

---

- Git luo *tunnus*-nimisen työhakemiston ja perustaa sinne paikallisen tietovaraston, jos kirjautumistietosi olivat oikeelliset.
- Esimerkki kloonauskomennosta ja Gitin tulosteista kloonaamisen aikana:

```
>git clone https://course-gitlab.tuni.fi/oope2-2020/oo000000
Cloning into 'oo000000'...
Username for 'https://course-gitlab.tuni.fi': oo000000
Password for 'https://oo000000@course-gitlab.tuni.fi':
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 16 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (16/16), done.
```

# 1) Tietovaraston perustaminen

---

- Voit halutessasi nimetä *tunnus*-hakemiston uudelleen, koska Git ei ole kiinnostunut työhakemiston nimestä, vaan sen sisällöstä.
  - Tässä esimerkissä hakemiston nimeä ei muuteta.
- Siirry työhakemistoon `cd`-komennolla. Esimerkiksi `cd oo00000`.
- Listaa hakemiston sisältö. Näet kätkeytyt tiedostot ja hakemistot Windowsissa komennolla `dir /ah` ja Mac- ja Linux-järjestelmissä komennolla `ls -a`

# 1) Tietovaraston perustaminen

---

- Työhakemistossa tulisi olla seuraavaa:
  - Kätkeyty *.git*-hakemisto, joka sisältää projektin tietovaraston.
  - *.gitignore*-tiedosto, joka määrittelee mitä tietoja ei tule pitää versionhallinnassa. Tämä on kätkeyty Mac- ja Linux-järjestelmissä.
  - *README.md*-tiedosto, johon kirjoitetaan kuvaus projektista.
  - *Harjoitustyo*-hakemisto, jonka alla on alihakemisto rajapinnat sisältävälle *harjoitustyo.apulaiset*-pakkaukselle sekä (lähes) tyhjä alihakemistot *harjoitustyo.dokumentit*- ja *harjoitustyo.omalista*-pakkauksille.
    - *Dokumentit*- ja *omalista*-hakemistoissa on tyhjä *.keep*-tiedosto, koska Git ei tallenna versionhallintaan täysin tyhjiä hakemistoja. Nämä tiedostot voi halutessaan poistaa, kun hakemistoissa on muita tiedostoja.



## 2.1) Päivitetään README.md

---

- GitLab ja GitHub ymmärtävät kevennettyä merkintäkieltä (lightweigh markup language, LML), jolla voidaan esittää HTML-kielen keskeisimpiä muotoiluja yksinkertaistetussa muodossa.
  - Verkkopalvelut tulostavat LML-tiedostot HTML-muodossa.
  - LML-kielestä on useita versioita, joista Markdown on yleinen Git-projekteissa.
- Markdown-merkintäkielessä esimerkiksi lihavointi ilmaistaan näin `**tämä lihavoituu GitLabissa**` ja HTML-kielellä pitemmin `<b>tämä lihavoituu selaimessa</b>`.

## 2.1) Päivitetään README.md

---

- Perinteisesti tietovarastoissa on projektista kertova LML-kielellä kirjoitettu *README.md*-tiedosto.
- *README.md*-tiedostossa on tällä hetkellä opettajan kirjoittamaa tekstiä. Lisätään sinne jotain omin sanoin sanottua.
- Tiedostoa voi muokata tekstieditorilla.
- GitLab- ja GitHub näyttävät tiedoston automaattisesti HTML-muodossa. Markdown-tiedostojen katseluun on myös olemassa ohjelmia ja lisäosia.

## 2.1) Päivitetään README.md

---

## Yleistä

WETO käyttää aina alkuperäistä `_harjoitustyo.apulaiset_`-pakkausta. Pakkauksen tiedostoja ei siksi ole syytä muuttaa ellei ole Mac- tai Linux-käyttäjä (katso alla).

Projektipohjan juuressa on `_.gitignore_`-tiedosto, jolla estetään tavukoodin tallennus versionhallintaan. Lisää siihen tarvittaessa estoja.

## Mac- ja Linux-käyttäjät

...

### Yleistä

WETO käyttää aina alkuperäistä `harjoitustyo.apulaiset`-pakkausta. Pakkauksen tiedostoja ei siksi ole syytä muuttaa ellei ole Mac- tai Linux-käyttäjä (katso alla).

Projektipohjan juuressa on `.gitignore`-tiedosto, jolla estetään tavukoodin tallennus versionhallintaan. Lisää siihen tarvittaessa estoja.

### Mac- ja Linux-käyttäjät

*README.md*  
editorissa.

*README.md*  
Tuni-GitLabissa.

## 2.2) Lisätään ajoluokka

---

- Harjoitustyön ajoluokka on *Oope2HT* ja se sijaitsee kaikkien pakkausten ulkopuolella työhakemistossa.
- Luodaan *main*-metodin sisältävä *Oope2HT.java*-tiedosto ja tallennetaan se työhakemistoon.

### 3) Siirretään muutokset valmistelualueelle

---

- Tarkistetaan aluksi tietovaraston tila komennolla `git status`.
- Gitin tulosteesta nähdään, että *README.md*:n muutoksia ei ole viety tietovarastoon ja *Oope2HT.java* ei ole vielä mukana versionhallinnassa.

```
>git status
```

```
On branch master
```

```
Your branch is up to date with 'origin/master'.
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git checkout -- <file>..." to discard changes in working directory)
```

```
    modified:   README.md
```

```
Untracked files:
```

```
  (use "git add <file>..." to include in what will be committed)
```

```
    Oope2HT.java
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

---

### 3) Siirretään muutokset valmistelualueelle

---

- Lisätään muutokset tietovarastoon komennolla `git add --a11`, joka lisää varastoon kaiken muuttuneen ja uuden tiedon.
  - Add-komennolla voidaan lisätä myös yksittäisiä kohteita.
- Tarkistetaan vielä `status`-komennolla, että lisäys onnistui.

```
>git status
```

```
On branch master
```

```
Your branch is up to date with 'origin/master'.
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
new file:   Oope2HT.java
```

```
modified:   README.md
```

## 4) Tehdään uusi versio

---

- Tallennetaan lopuksi versio eli “tehdään commit” tai “kommitoidaan”, jonka jälkeen muutokset ovat versionhallinnassa.
- Tiedosto tallennetaan tietovarastoon kommentin kera `commit`-komennolla joko siten, että kommentti kirjoitetaan komentorivillä:

```
git commit -m "Aloitettiin oman projektin versionhallinta."
```

tai siten, että annetaan pelkkä `commit`-komento, jolloin Git avaa editorin kommentin kirjoittamista varten.

- Windowsissa suositellaan jälkimmäistä tapaa, jotta kommentti saadaan varmasti utf-8-muodossa tietovarastoon.

## 4) Tehdään uusi versio

---

- Kommentin tulee olla melko lyhyt, mutta kuitenkin lukijalleen hyödyllinen.
- Versiohistoriaa voi tutkia komennolla: `git log`
- Tulosteesta nähdään kuka tallensi version ja milloin versio tallennettiin. Tulosteessa on annettu myös SHA-1-tiiviste, joka yksilöi version. Kaksi ensimmäistä versiota on tehnyt vastuopettaja ja kolmas versio on opiskelijan tekemä. Vastuopettaja on antanut projektipohjille versionumerot.



## 4) Tehdään uusi versio

---

```
>git log
```

```
commit 22ae358eccfde6202ca9854a05a950ad531db549 (HEAD -> master)
Author: Onni Opiskelija <onni.opiskelija@tuni.fi>
Date:   Tue Mar 10 13:26:56 2020 +0200
```

Aloitettiin oman projektin versionhallinta. README.md päivitetty ja ajoluokka luotu.

```
commit 966e7726744d1e57aed10ef891b7be3bd983d10a (tag: v1.1, origin/master,
origin/HEAD)
Author: Jorma Laurikkala <jorma.laurikkala@tuni.fi>
Date:   Tue Mar 10 11:17:09 2020 +0200
```

Lisätty pakolliset pakkaushakemistot. Hakemistoissa on tyhjä .keep-tiedosto, koska Git ei tallenna tyhjiä hakemistoja.

```
commit 390cc81e7dee9c197b0cf8d706b7a6d691bad8cf (tag: v1.0)
Author: Jorma Laurikkala <jorma.laurikkala@tuni.fi>
Date:   Thu Mar 5 21:43:04 2020 +0200
```

Vastuupettaja loi projektipohjan

## 6) Ladataan projekti etävarastoon

---

- Siirretään projekti Tuni-Gitlabin etätietovarastoon.
  - Tarkista aluksi, että `origin`-alias liittyy oikeaan paikkaan eli URL:iin, josta etävarasto kloonattiin:  

```
>git remote --verbose  
origin https://course-gitlab.tuni.fi/oope2-2020/oo00000 (fetch)  
origin https://course-gitlab.tuni.fi/oope2-2020/oo00000 (push)
```
  - Anna tämän jälkeen komento `git push origin master`, joka päivittää paikalliseen tietovarastoon tehdyt muutokset etätietovarastoon.
    - Anna Tuni-tunnus ja sen salasana.
  - Voit käydä katsomassa etävaraston sisältöä kirjautumalla osoitteessa [https://course-gitlab.tuni.fi/users/sign\\_in](https://course-gitlab.tuni.fi/users/sign_in).
-