Versionshantering
i ett jättelitet nötskal på 10 minuter

Henrik Henriksson

LINKÖPING UNIVERSITY

Versionshantering?

Alla versionshanterar dokument och kod:

- `rapport_v0.4.docx`
- `rapport_v0.5.pdf`
- `buggfixad-kod.cpp`
- `rapport_v1.0.docx`
- `rport_v0.9-FINAL.doc`
- `kompl-v2.docx`
- `färdig.7.pdf`
- `jättefärdig.2.pdf`
- `raport_v1.1.docx`
- `v1.0-inlämnad.rtf`

Versionshantering?

Alla versionshanterar dokument och kod:

- `rapport_v0.4.docx`
- `rapport_v0.5.pdf`
- `buggfixad-kod.cpp`
- `rapport_v1.0.docx`
- `rport_v0.9-FINAL.doc`
- `kompl-v2.docx`
- `färdig.7.pdf`
- `jättefärdig.2.pdf`
- `raport_v1.1.docx`
- `v1.0-inlämnad.rtf`

Effektiv versionshantering kräver processer och verktyg!

Vad vill vi få ut från versionshantering?

- ☐ Låta oss ändra saker utan att förstöra tidigare arbete
- ☐ Se vad som förändrats mellan två versioner
- ☐ Hjälpa oss arbeta flera personer i samma projekt
- ☐ Ha ett bra workflow
- ☐ Ha en översikt över projektets historia
- ☐ Veta att vårt arbete är säkert
- ☐ Veta att tidigare versioner inte har ändrats

Vad vill vi få ut från versionshantering?

Git hjälper oss!

- ☑ Låta oss ändra saker utan att förstöra tidigare arbete
- ☑ Se vad som förändrats mellan två versioner
- ☑ Hjälpa oss arbeta flera personer i samma projekt
- ☑ Ha ett bra workflow
- ☑ Veta att vårt arbete är säkert
- ☑ Veta att tidigare versioner inte har ändrats
- ☑ Ha en översikt över projektets historia

Vad är Git?

- Git är ett verktyg för versionshantering som ursprungligen utvecklades av Linus Torvalds för att hantera Linuxkärnans källkod.
- Sedan Git släpptes 2005 har det snabbt blivit något av en industristandard för versionshantering av kod.
- "Riktad acyklisk graf över ändringar"
- Används oftast genom ett kommandoradsgränssnitt.
- Det har utvecklats många plattformar och verktyg kring Git, till exempel GitHub, Bitbucket och GitLab.

Vad borde man versionshantera?

Det mesta som ska "levereras" brukar vara en bra idé att versionshantera:

- Kod
- Dokument
- Kandidatrapport

Vissa saker är ibland olämpliga att hantera i Git:

- Databaser
- Binärdata

Hur ska man versionshantera?

Processer är viktiga!

Välj hur ni vill jobba tidigt, men var inte rädda för att ändra på det! Olika arbetssätt har olika tradeoffs, hitta det som fungerar för ert projekt.

- Enklast möjliga: alla jobbar på samma gren - Lättdrivet, men svårkontrollerat.
- Vanligt: *Feature branches* - mer tungdrivet, men mer kontroll.

Resurser för versionshantering

GitLab

Vid LiU finns *GitLab*, `gitlab.liu.se`, en platform för mjukvaruutveckling med Git.

- Delade Git-repon
- Merge Requests
- Issue tracking
- Integrerad CI/CD för automatiska tester
- Publicering av statiska webbsidor (“Pages”)
- Kravhantering, DevOps, Code-review, Boards, diverse planeringsverktyg, webhooks, testrapporter, paketrepo ....

Grafiska klienter

Det finns grafiska klienter för Git, men de kan vara knepiga att använda om man inte redan vet hur kommandoradsgränssnittet fungerar.

Rekommendation: Börja med kommandoradsgränssnittet, när man känner sig bekväm med det så kan man hitta en grafisk klient man tycker om.

Resurser för att lära sig Git

Git på mer än 10 minuter

- GitLab har ofta bra dokumentation
  - `https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html`
  - `https://docs.gitlab.com/ee/gitlab-basics/`
- När man känner sig lite bekväm med Git kan det här föredraget från Linus Torvalds vara intressant

  `https://www.youtube.com/watch?v=4XpnKHJAok8`
- Skapa en 'lekrepo' och testa saker!

Blandade tips

- Prata med varandra om hur ni vill använda Git, och se till så att ni håller interna utbildningar så att alla hänger med.
- Om man vill testa köra ett konstigt Git-kommando, ta en kopia på hela mappen med repot först!
- 'Protected branches' i GitLab skyddar en från att ta bort all kod - aktivera genast!
- GitLab-CI med automatiska tester är underbart, utnyttja att det finns!
- Börja enkelt! `commit`, `push` och `pull` räcker oftast!
- `git status`

Lycka till!

www.liu.se

LINKÖPING UNIVERSITY