Testautomatisering

CI, Git, Rake

Idag

- FM:
  - Lab 4 - snabbutvärdering
  - Blackbox/Whitebox Testing
  - Performance/Load Testing
  - Snabbintro, Continuous Integration
  - Git - crashcourse
  - Rake – crashcourse
  - Inlämningsuppgift – deadline fredag 17:59

- Lab 4 - Snabbutvärdering
  - 1. Hur många timmar har du lagt?
  - 2. Hur många ytterligare timmar kommer du lägga?
  - 3. Svårighet: För Lätt / Lagom / För Svår
  - 4. Lärdommar - "Jag ser... ":
    - Ingen praktisk nytta (-)
    - Viss praktisk nytta (0)
    - Stor praktisk nytta (+)

Lab 4 Utvärdering

Blackbox/Whitebox

- Blackbox testing
  - Tester som sker utan hänsyn till- eller kännedom om implementationsdetaljer i det som testas
- Whitebox testing (Glassbox testing)
  - Tester som sker med kännedom om implementationsdetaljer i det som testas

Blackbox/Whitebox

- Blackbox testing
  - Fördel: Implementationsdetaljer är ofta ointressant. Här testar vi istället beteende.
  - Generellt sett kräver blackbox testing mindre analys än whitebox testing – eftersom vi har färre aspekter att ta hänsyn till.
- Whitebox testing
  - Fördel: Vi kan testa gränserna mellan två komponenter i ett system
  - Det är ofta i gränserna mellan två komponenter som oönskat beteende uppstår.

- Performance/Load Testing

Performance/Load Testing

Performance/Load Testing

- Performance/Load Testing
- Behöver ofta skräddarsys för en enskild applikation

Performance/Load Testing

- Performance/Load Testing
- En strategi för enkla Performance-tester:
- Skriv någon form av integrations eller end-to-end test. (T.ex. GUI-test med Watir).

Performance/Load Testing

- Performance/Load Testing
- Lägg ev. till tidsmätning i era tester
- clock_start = Time.now
- #… gör något intressant här
- clock_end = Time.now
- elapsed_time = clock_end – clock_start

Performance/Load Testing

- Performance/Load Testing
- Starta ex. 100 instanser av ert test-script och kör dessa mot applikationen ifråga.
- Samla in tidsmätningar
- Kontrollera hur responsen är i programmet.

Performance/Load Testing

- Performance/Load Testing
- På detta sätt kan vi ex. enkelt mäta om ”applikationen klarar 1000 samtidiga inloggningar, tiden för en enskild inloggning får inte överskrida 5 sekunder under last”.

- Continuous Integration

Snabbintro - CI

- Grundproblemet – vi har:
  - Källkod
  - Test för källkoden

Snabbintro - CI

- Automatisera körning av dessa test
  - Alt 1: Varje gång en förändring sker
    - Ex: Vid rättning av en bugg
  - Alt 2: Schemalagt
    - Ex: Varje natt 02:00

Snabbintro - CI

- För detta krävs
  - Källkod
  - Tester
  - När skall automatisering ske?
  - Vad skall ske?
  - I vilken miljö skall detta ske?
  - Var finns källkod/tester?

Snabbintro - CI

Snabbintro - CI

- Continuous Integration (CI):
  - När skall automatisering ske?
  - Vad skall ske?
  - I vilken miljö skall detta ske?
  - Var finns källkod/tester?

Snabbintro - CI

- Continuous Integration:
  - När skall automatisering ske? (Policy)
  - Vad skall ske? (Script)
  - I vilken miljö skall detta ske? (Miljöhantering)
  - Var finns källkod/tester? (Versionshantering)

Snabbintro - CI

- Continuous Integration:
  - När skall automatisering ske?
    - Vid Checkin – Travis CI/Github
  - Vad skall ske?
    - Testscript - Rake
  - I vilken miljö skall detta ske?
    - RVM + Bundler
  - Var finns källkod/tester?
    - Github

Snabbintro - CI

- Continuous Integration:
  - Imorgon kikar vi på:
  - När skall automatisering ske?
    - Vid Checkin – Travis CI/Github
  - I vilken miljö skall detta ske?
    - RVM + Bundler

Snabbintro - CI

- Continuous Integration:
  - Idag:
  - Vad skall ske?
    - Testscript - Rake
  - Var finns källkod/tester?
    - Github (git)

- Continuous Integration:
  - Var finns källkod/tester?
    - Github (git)

Crashcourse - git

- Versionshantering
  - (Source Code Management (SCM), Version Control System (VCS), Version Control, etc)
  - Problem 1: Vi vill kunna se historik för snabbt förändrliga digitala dokument.
  - Problem 2: Vi vill kunna redigera redigera gemensamma digitala dokument samtidigt utan att information försvinner.

Crashcourse - git

- Versionshantering
  - (Source Code Management (SCM), Version Control System (VCS), Version Control, etc)
  - git, hg (Mercurial), SVN, Team Foundation, etc. (Legacy: CVS, Source Safe)

Crashcourse - git

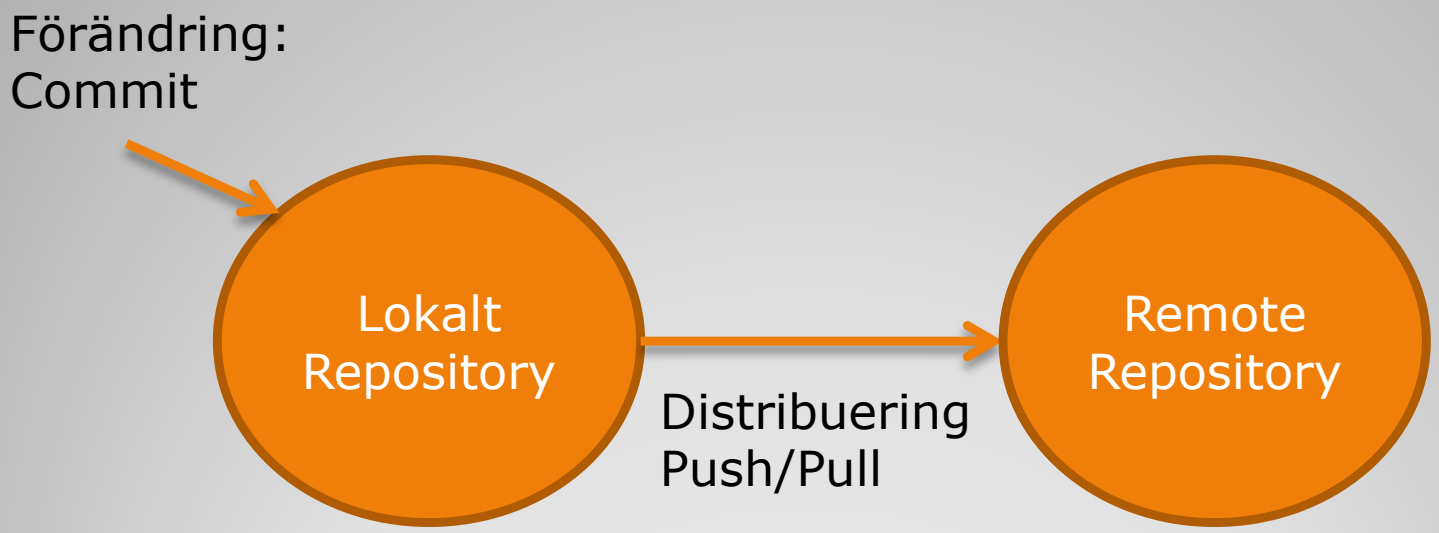
- Versionshantering
  - Varför?
    - Vi kommer inte skriva automatiserade tester utan någon form av VC
    - Vi kommer inte skriva källkod utan någon form av VC
    - Vi kommer hämta källkod/tester från ett VC repository
    - Vi kommer inte köra CI utan någon form av VC
    - (Funkar även utmärkt för sammarbete kring andra digitala dokument)

Crashcourse - git

Crashcourse - git

- Versionshantering
  - Git?
    - Modernt
    - Distribuerat
    - Används för stora projekt
    - (Väldigt likt hg – ett annat stort VCS)

- Git i ett nötskal – The Big Picture

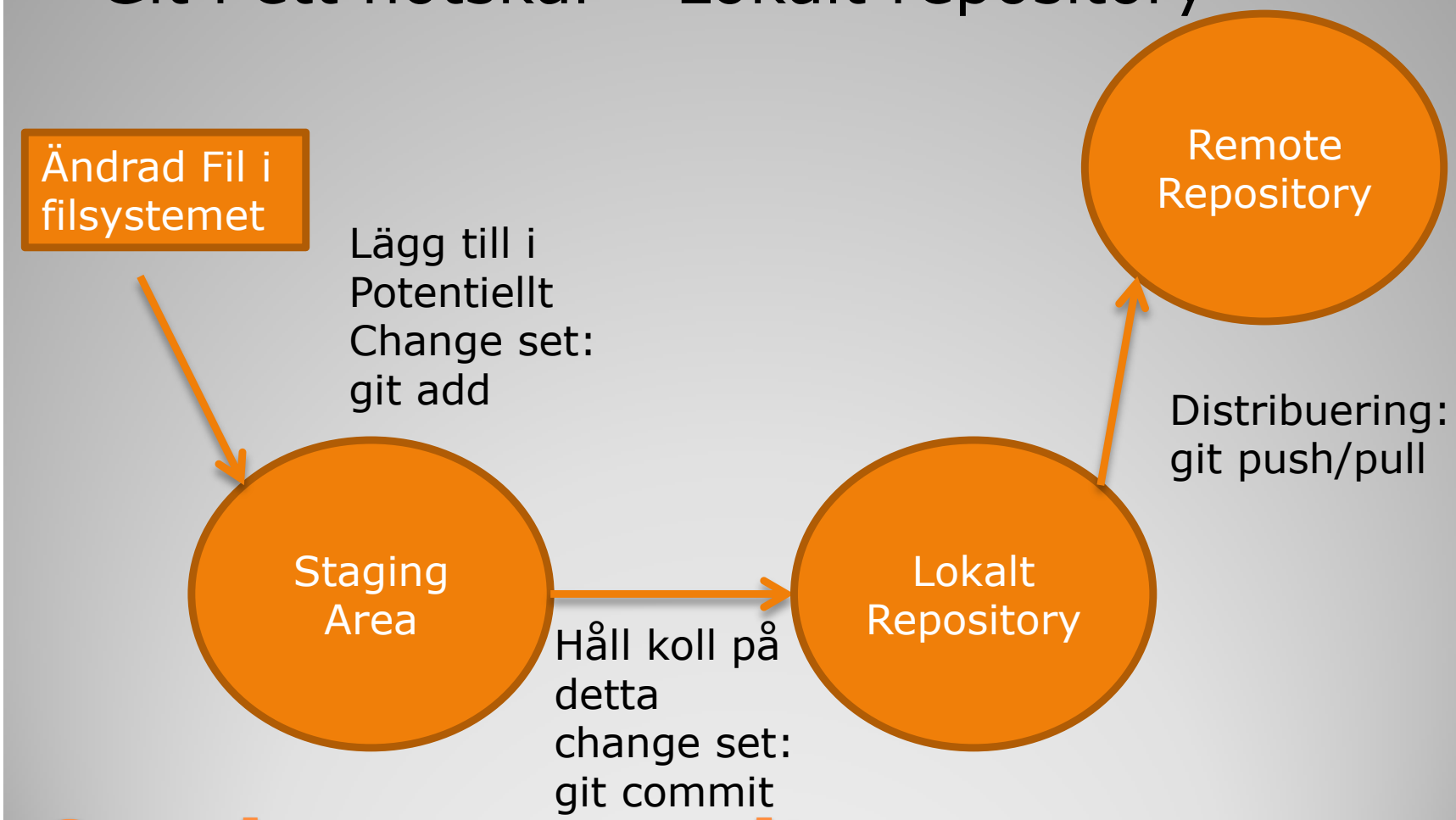


Crashcourse - git

- Förändringar
  - Ett antal förändringar som hör ihop samlas i ett change set
  - Ex: En buggfix kanske leder till att vi göra två förändringar på två olika rader i källkoden – dessa två förändringar buntas ihop till ett change set som vi kan döpa till t.ex. ”Buggfix – Issue: 1274”

Crashcourse - git

- Git i ett nötskal – Lokalt repository



Crashcourse - git

- Git kommandon:
  - git init: skapa nytt repository
  - git clone: skapa en kopia på ett befintligt repository

- Git kommandon:
  - git add: lägg till dessa förändringar till staging arean
  - git commit: skapa ett change set utifrån staging arean

Crashcourse - git

- Git kommandon:
  - git pull: hämta förändringar från remote repository (ex. github)
  - git push: skicka mina förändringar till remote repository (ex. github)

Crashcourse - git

- Git kommandon:
  - git checkout: hämta ut ett visst changeset

Crashcourse - git

- Git kommandon:
  - Viktigast av allt:
  - git status: Vad har jag för förändringar i staging area och local repository

Crashcourse - git

- Git kommandon:
  - git log: hur ser historiken för ett repository ut.
  - *Lägg märke till att varje commit har ett unikt id (hash) – detta gör att vi kan referera till specifika commits när vi t.ex. gör en checkout.

Crashcourse - git

- Installation
  - Xubuntu: sudo apt-get install git
  - Windows: http://windows.github.com/
  - Mac: http://git-scm.com/downloads

Crashcourse - git

- Git
  - Exempel

Crashcourse - git

Crashcourse - git

- Git tutorial:
  - http://www.codeschool.com/courses/try-git
  - http://learn.github.com/p/intro.html
  - Jag har även en enklare git-övning som jag lägger upp på kursbloggen

- Rake

Crashcourse - rake

Crashcourse - rake

- Rake – Ruby Make
  - Rake är till för att köra andra script
  - Ex: Installationsscript
  - Ex: Migrationsscript
  - För oss: Test script

Crashcourse - rake

- Rake – Ruby Make
  - Vi kommer titta på att använda rake för att köra rspec och cucumber tester

Crashcourse - rake

- Rake – Ruby Make
  - Rake script läggs i en ”Rakefile”
  - Detta gör att t.ex. en continuous integration lösning kan köra våra tester – den behöver inte hålla reda på var våra tester ligger och vad de heter – den behöver bara köra Rakefile.

Crashcourse - rake

- Rake – Ruby Make
  - I grund och botten:
  - Ett ruby script som utför en eller flera tasks

Crashcourse - rake

- Rake – Ruby Make

```
task :default => [:test]

task :test do
    ruby "test/unittest.rb"
end
```

Crashcourse - rake

- Rake – Ruby Make
- Föregående bild:
  - Utför uppgiften ”:task”
  - ”:task”-uppgiften består i att köra ruby på filen unittest.rb som ligger i test/-katalogen

Crashcourse - rake

- Rake – Ruby Make
- Installation
  - gem install rake
  - Xubuntu: sudo gem install rake

Crashcourse - rake

- Rake – Ruby Make
  - Exempel

Crashcourse - rake

- Rake – Ruby Make
  - Mer info:
    - http://rake.rubyforge.org/
  - Rake + cucumber:
    - https://github.com/cucumber/cucumber/wiki/Using-Rake
  - Rake + rspec:
    - https://www.relishapp.com/rspec/rspec-core/docs/command-line/rake-task

Nästa gång

- RVM
- Continuous Integration (Travis CI)
- Utvecklingsmiljö vs. Testmiljö vs. Releasemiljö

- ...

Fin