Testautomatisering

Travis CI, Github, RVM, Bundler

Idag

- FM:
  - Kort om Lab 4 + Inlämning
  - Github + git repetition
  - Hantera Miljö: RVM, Bundler
  - Repetition CI – Hur hänger allt ihop
  - Konkret exempel: Travis CI
  - Vad kan vi göra härnäst?
  - Utvecklingsmiljö vs. Testmiljö vs. Releasemiljö
    - Att tänka på
    - Vad har vi för alternativ för att upprätta en testmiljö?

Inlämning

- Inlämning
  - Individuell
  - Deadline: fredag 17:59
  - Frågor på det mesta vi gått igenom under kursen

- FitNesse
  - Väsensskild från de övriga verktyg vi använt
  - Vi behöver börja skriva cucumber scenarios i eftermiddag…
  - ~~Vi behöver sätta upp en FitNesse Server och skriva Testtabeller i eftermiddag…~~

Lab 4

- FitNesse
  - Min förhoppning…
  - När man väl fått till installation och fixtures är det enkelt att skriva testtabeller i FitNesse (Men jag har lite respekt för installations-fasen).
  - ~~FitNesse är ett krångligt verktyg som jag skall försöka undvika i framtiden…~~

Lab 4

- FitNesse
  - Ej fått igång ett fungerande exempel ännu: Kontakta mig.

Lab 4

- FitNesse
  - Lab 4 - Mina exempel – ni behöver inte nödvändigtvis implementera just dessa exempel.

Lab 4

- Github

Github

- Github
  - Gratis, publikt, git repository

Github

- Github – Exempel
  - Skapa nytt repository
  - Klona det lokalt (git clone)
  - Gör ändring + git add + git commit
  - git push och kika på ändringen på github

Github

- Github: I vårt exempel så kommer källkod + tester finnas på github.

Github

- Github: Service Hooks
  - En service hook innebär lite förenklat ”När en förändring skett – kör den här tjänsten”.
  - Från repository sidan på github: Settings -> Service Hooks

Github

- Mer info: https://help.github.com/

Github

- Vi behövde även hålla reda på miljö
- I detta fall:
  - Ruby + ruby version
  - Gems

RVM

- RVM – Ruby Version Manager
  - Kan användas för att hantera olika versioner av ruby
  - Med RVM kan man t.ex. arbeta i ett ruby 1.9.3 projekt och ett ruby 2.0 projekt på samma dator.

RVM

- RVM – Ruby Version Manager
  - Travis CI använder RVM för att avgöra vilken ruby version som skall installeras innan vi kör vårt rake-script
  - Detta anges i en .travis.yml-fil i roten på projektet.

RVM

- RVM – Exempel

```
language: ruby
rvm:
    - 1.9.3
```

- Mer info: https://rvm.io/

RVM

Bundler

- Vi har nu angivit språk + version
- Kvar är att ange vilka gems vi använder
- I vårt fall: RSpec, Cucumber, Rake

Bundler

- Bundler – en lösning för att hantera gems vi är beroende av
- Vi anger de gems vi är beroende av i en ”Gemfile”

Bundler

- Bundler – exempel

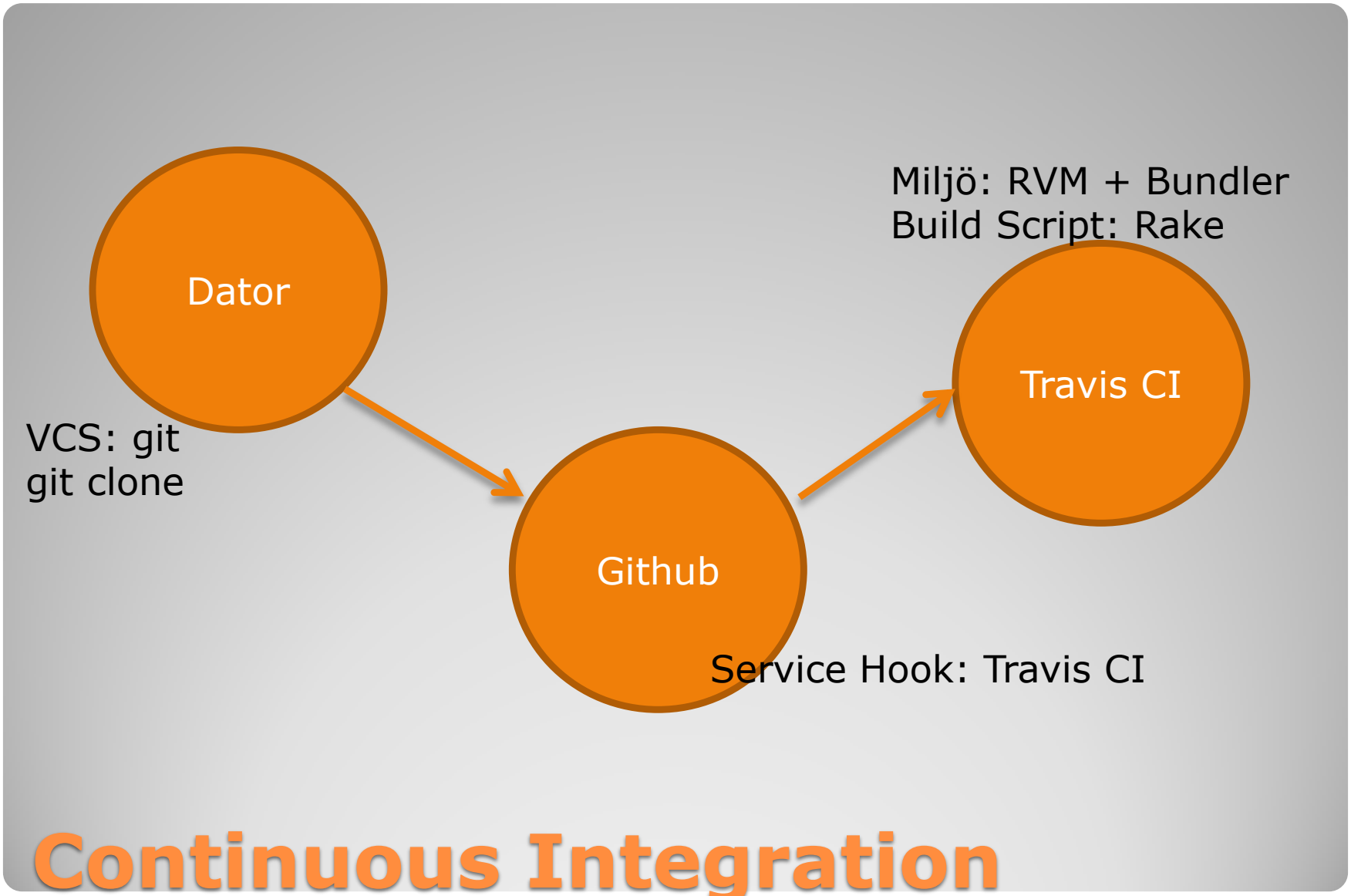
- source: https://rubygems.org
- gem: ”cucumber”
- gem: ”rspec”
- gem: ”rake”

- Mer info: http://gembundler.com/

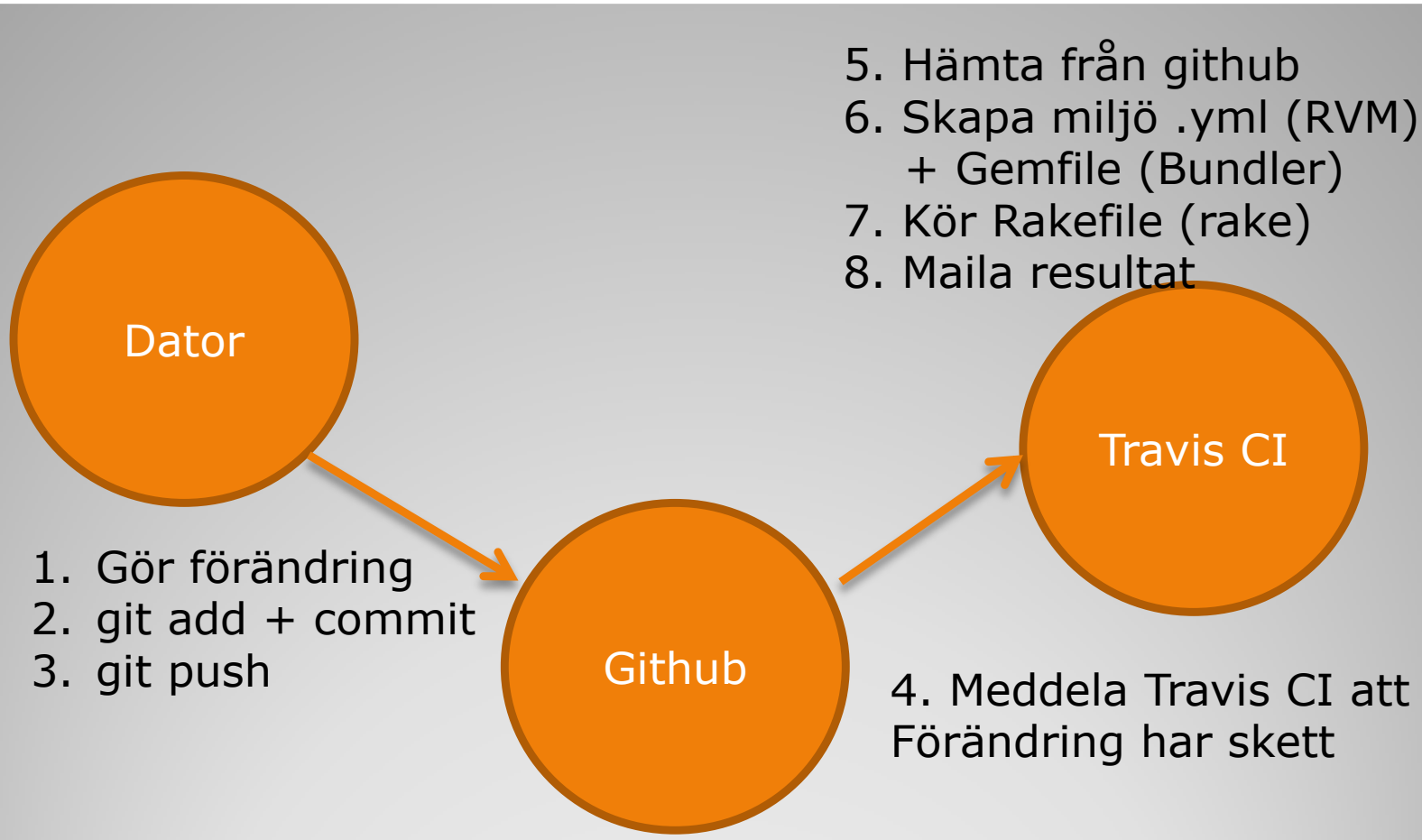
Bundler

- Continuous Integration

Continuous Integration



Continuous Integration



Continuous Integration

- Exempel – Travis CI

Continuous Integration

- Vad är intressant?
  - Vad är CI?
  - Ha ett humm om den övergripande processen

Continuous Integration

Continuous Integration

- Liknande FitNesse
  - Ingen kommer förvänta sig att ni sätter upp en CI server på en eftermiddag
  - Om ni hamnar på ett företag som använder CI kommer lösningen redan vara installerad och på plats.

Travis CI

- Mer info: http://about.travis-ci.org/docs/

Continuous Integration

- Alternativa CI-lösningar:
  - .Net, Cloud Service: appharbour
  - Self hosted: Jenkins
  - Self hosted: CruiseControl
  - Self hosted: TeamCity
  - …+ en uppsjö andra molntjänster och self hosted-lösningar.

- Testa att konfigurera CI för lab 2
  - Skapa Github-konto + lägg upp lab 2 på github
  - Skapa Travis CI-konto + aktivera service hook
  - Lägg till .yml-fil i roten på ert projekt (ruby 1.9.3)
  - Lägg till Gemfile i roten på ert projekt (rake)
  - Lägg till Rakefile i roten på ert projekt (ruby ”path/to/unittest.rb”) – se rakes hemsida
  - Plocka bort watir-testet från lab 2 (alt. sök på headless på travis CIs hemsida + förändra Gemfile + Rakefile)
  - Gör en push till ert repository (kan vara ex. en förändring i info.txt – tänk på att github är publikt)

Continuous Integration

Continuous Integration

- Ni skulle också kunna testa sätta upp en Jenkins-server.
- Jag skulle dock definitivt rekommendera er att experimentera med Travis CI först
- Det är generellt sett mer komplext att konfigurera en self hosted CI-lösning än att använda en molntjänst – iaf. för enklare scenarion.

- Olika miljöer

Miljö

Miljö

- Jag har nämnt några gånger att Virtual Machines är utmärkta testverktyg…
- Dags att prata lite miljö.

Miljö

- Med miljö menar vi:
  - Operativsystem, inkl. inställningar för OS (t.ex. språk)
  - Plattform
    - Ex, Webbapplikation: Apache WebServer, MySQL Databas-server, Ruby on Rails
  - Beroenden för projektet
    - Ex: gemfiles, etc.
  - Ev. hårdvara, drivrutiner eller hårdvaruinställningar
  - Versionsnummer för allt ovanstående ingår i miljön.

Miljö

- Med utvecklingsmiljö så menas den miljö som applikation och tester utvecklas i.

Miljö

- Med produktionsmiljö så menas den miljö där applikationen används skarpt. Det kan vara t.ex. en webbserver för en webbapplikation eller en desktop dator för ett desktop-program.

Miljö

- Med testmiljö så avses den miljö som ett program testas i.

Miljö

- Det är väldigt vanligt att utvecklingsmiljön skiljer sig från Produktionsmiljön
- Detta brukar inte vara hela världen (även om det händer att man råkar ut för otrevliga överraskningar – detta är en av de saker vi vill förhindra mha test).

Miljö

- Det är däremot, för det mesta, väldigt viktigt att Testmiljön stämmer väl överrens med Produktionsmiljö.
- Generellt sett: Ju komplexare program, desto viktigare blir det att testmiljö och produktionsmiljö är lika varandra.
- För affärskritiska produkter vill man generellt ha en testmiljö som är identisk med produktionsmiljön.

Miljö

- Hur åstadkommer vi en testmiljö som är identisk med produktionsmiljön?
- (Jag kommer generellt prata om webbapplikationer här).

- Alt 1:
  - Identisk hårdvara med identisk mjukvarukonfiguration.
  - Detta är relativt dyrt och tidskrävande att underhålla.
  - Det förekommer dock.

Miljö

- Alt 2:
  - Snarlik hårdvaru/mjukvarukonfiguration
  - Fortfarande relativt dyrt.

Miljö

- Alt 3:
  - Virtual Machines
  - Billigt och potentiellt lätt att underhålla
  - Om hårdvaran är en väldigt viktig faktor kan denna lösningen vara problematisk.

Miljö

Miljö

- Alt 4:
  - Vi hostar en webbapplikation i molnet – ex. Amazon EC2
  - Vi kan spinna upp en kopia på vår skarpa miljö för test on demand
  - Relativt billigt och potentiellt lätt att underhålla.
  - Så nära exakt kopia av skarp miljö som vi kan komma.

Miljö

- Min grundrekommendation: Om ingen dedikerad test-server finns; Använd Virtual Machines för test.
- Billiga och steget härifrån till ex. EC2 är inte så långt (det är i grund och botten samma sak).
- Sky som pesten: Att testa native på er ”vardagsburk”.

- Jag har pratat om webbapplikationern som huserar på servern hittills.
- Vid client side tester för webbapplikationer (tester med browsern) så kan ni även här använda Virtual Machines.

Miljö

- Typexempel där detta är smidigt: IE6, IE7, IE8-tester
- (Av erfarenhet: IETester visar inte alla problem och ger ibland false positives).

Miljö

Miljö

- Så hur testar vi desktopapplikationer för ex. Windows som är en väldigt splittrad plattform?
- Ju komplexare (eller hårdvaruberoende) applikation: ju mer tester krävs.
- Detta är ett potentiellt svårt och resursintensivt problem.

- Virtual Machines är behjälpliga även vid test av desktopapplikationer.

Miljö

- Även skapandet av virtual machines går att automatisera.
- Se ex. Vagrant eller Chef.

Miljö

- Avslutande ord

Avslutande ord

- ...

Fin