Continuous Integration med Jenkins

Linus Tolke
Enea Experts

Föredraget

- Grunderna i mjukvaru-CM
- Trender inom mjukvaruutveckling
- Continuous Integration
- Vad är Jenkins
- Demo
- Jenkins i ArgoUML-projektet
- Problem och lösningar

Föredraget

- **Grunderna i mjukvaru-CM**
- Trender inom mjukvaruutveckling
- Continuous Integration
- Vad är Jenkins
- Demo
- Jenkins i ArgoUML-projektet
- Problem och lösningar

Grunderna i Mjukvaru-CM

Versionshanteringssystem

- Subversion
- CVS
- Dimensions
- Git
- ...

CM-arkiv

För Open Source:

- SourceForge
- Google Code
- Tigris
- GitHub

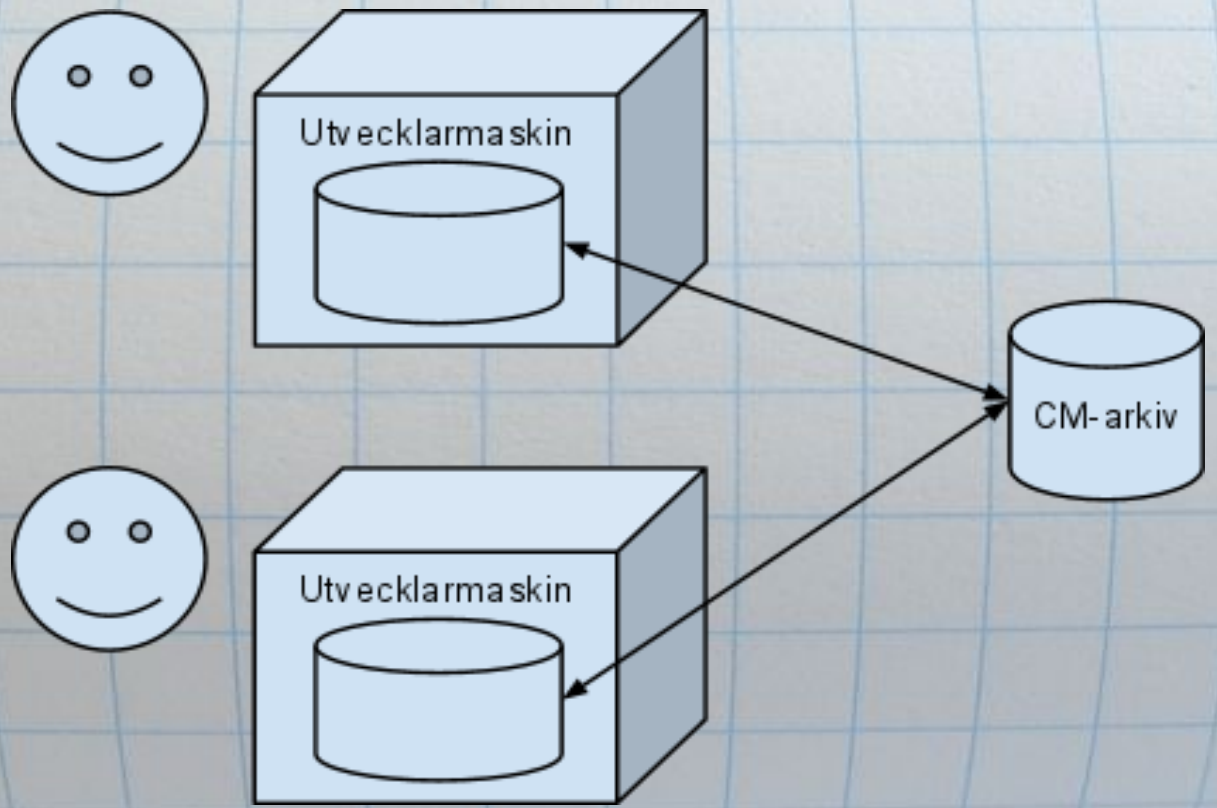
För inte-öppna projekt:

- En server inom företaget
- En köpt tjänst

Utvecklarnas maskiner

Kopplar sig till CM-Arkivet

- över Internet (Open Source, köpt tjänst)
- inom ett slutet nätverk (egen server)



Organisation av källkod

CM-arkivet innehåller

- Koden - för produkten
- Testfallen
- Regler för hur man bygger ihop produkten och kör testfallen (makefiler, konfigurationsfiler, mm).
- Regler för statisk analys, kodningsstil, mm.
- Om man vill säkerställa en stabil miljö kan det också innehålla verktyg för att bygga (kompilator, include-filer för målmiljön, mm). Maven-världen har en mer långtgående lösning på detta.

Organisation av testfall

Unit test

- Testar en klass i taget - stubbar för allt
- Går snabbt att köra - Eclipse kan dresseras till att köra dem när man sparar

Integrationstestfall eller systemtestfall

- Testar flera delar av produkten tillsammans (flera olika klasser, flera olika jar-filer)
- Tar ofta lång tid att köra
- Ibland behövs speciella verktyg (licenser, databaser)

Statisk analys och efterlevnad av kodningsstandard

- Eclipse gör en del och visar direkt, även checkstyle
- Annars tunga verktyg (checkstyle och findbugs för java, lint++ m.fl)

CM-arkivet är integrationspunkten

Integration sker vid varje incheckning

Krav på uppnådd kvalitet vid varje incheckning bestäms av projektet eller produktutvecklingsorganisationen.

Låga krav ger större risk att

- projektdeltagare får felsöka andras fel
- kvalitén urholkas

Höga krav ger

- extraarbete vid varje incheckning
- mycket tid spenderas på att köra testerna om igen för andras ändringar

Exempel på krav för incheckning

- Alla testfall körda med godkänt resultat
- Alla statiska kontroller körda med godkänt resultat

Fungerar bra om

- det går snabbt att köra alla tester och statiska kontroller (< 5 minuter)
- det finns bra verktyg som rapporterar resultatet
- programmet och testfallen är plattformsoberoende (eller plattformen liknar utvecklarmaskinerna)
- alla utvecklingsmaskiner har tillgång till alla verktyg (licenser mm)

Arbete för varje incheckning i exemplet

1. Implementera funktion och testfall.
2. Kompilera och kör testfall.
3. Om du inte är klar eller det är fel gå tillbaka till 1.
4. Gör Rebase.
5. Kör alla statisk analysverktyg som måste köras vid incheckning.
6. Kolla resultatet.
7. Om det är några fel, gå tillbaka till 1.
8. Kör alla testfall som måste köras vid incheckning.
9. Kolla resultatet.
10. Om det blir något fel, felsök och gå tillbaka till 1. såvida det inte är någon annan som orsakat problemet då får du vänta tills han är klar och sedan gå tillbaka till 3.
11. Checka in.

Exempel på lite tuffare krav

- Alla testfall körda med godkänt resultat i alla plattformar

Ofta svårt

- det kan finnas ont om tillgängliga plattformar, simulatorer och licenser
- det blir lätt väldigt många kombinationer av HW * OS * Kompilator
- alla utvecklare eller alla personer som sköter integrationen måste klara av att köra alla dessa kombinationer

Föredraget

- Grunderna i mjukvaru-CM
- **Trender inom mjukvaruutveckling**
- Continuous Integration
- Vad är Jenkins
- Demo
- Jenkins i ArgoUML-projektet
- Problem och lösningar

Trender inom mjukvaruutveckling

- Utvecklarnas IDE:er förväntas göra allt mer och kräver mer och mer av utvecklarens maskin (utnyttjar flera kärnor/cpu:er)
- Testdriven utveckling ger fler och fler testfall och ett krav på att köra dem oftare och snabbare
- Kraven, vid mjukvaruutveckling, ökar. Därmed ökar också kraven på de verktyg som skall hjälpa oss att hitta dessa problem
- Verktygen blir fler och mer avancerade och kodmängden ökar vilket gör att körningen av verktygen tar allt längre tid
- Utvecklingen av datorer i pris/prestanda hinner inte med, speciellt som utvecklarnas maskiner ofta jämförs i pris med datorer som används för enklare administrativa uppgifter

Lösning

- Sänk kraven för varje incheckning
  - Allt kompilerar
  - Modultester körda
  - Snabb syntax-kontroll för kodningsstilen (Eclipse inbyggda)
- Kör tunga och repetitiva tester efter incheckning
  - Alla nödvändiga kombinationer av plattformar
  - Tidskrävande verktyg för statisk analys
- Starta de tunga testerna så snabbt som möjligt efter varje incheckning

Föredraget

- Grunderna i mjukvaru-CM
- Trender inom mjukvaruutveckling
- **Continuous Integration**
- Vad är Jenkins
- Demo
- Jenkins i ArgoUML-projektet
- Problem och lösningar

Continuous Integration

Lösningen uppskruvad

Definition av begreppet:
"A fully automated and reproducible build, including testing, that runs many times a day" Martin Fowler (September 2000)

"A Software engineering practice in which isolated changes are immediately tested and reported on when they are added to a larger code base." http://whatis.com (July 2008)

Fördelar med Continuous Integration

Varje utvecklare integrerar hela tiden eller i alla fall vid varje rebase och varje incheckning - då minskas integrationsproblemen

Ger snabb återkoppling om hur mjukvaran mår.

Det går att se när ett fel introduceras i kodbasen och det kan snabbt korrigeras.

Vi visualiserar produktens kvalitet på flera nivåer.

Föredraget

- Grunderna i mjukvaru-CM
- Trender inom mjukvaruutveckling
- Continuous Integration
- **Vad är Jenkins**
- Demo
- Jenkins i ArgoUML-projektet
- Problem och lösningar

Verktyget Jenkins

Hette förut Hudson
En produkt med följande funktion:

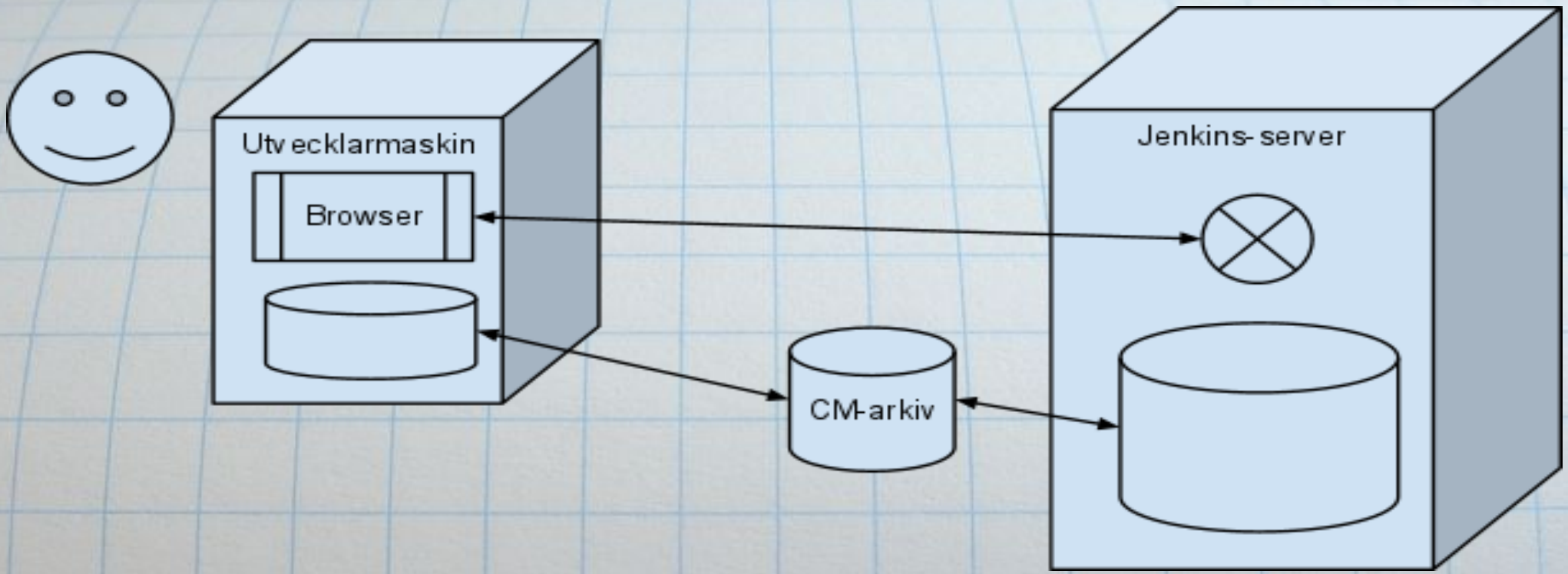
- Bygga kod och köra testfall
  - Polla CM-arkivet
  - Hämta källkod från CM-arkivet
  - Köra skript
  - Lagra resultatet och tillhandahålla på webben
  - Presentera skillnader mellan resultat
- Massor av plug-in:er för olika versionshanteringssystem, kompilatorer, testfallsramverk och verktyg för statisk analys
- Extra bra integration med Maven för att minska mängden nödvändig konfiguration

Ett Open Source-projekt i java som använder maven som byggmiljö

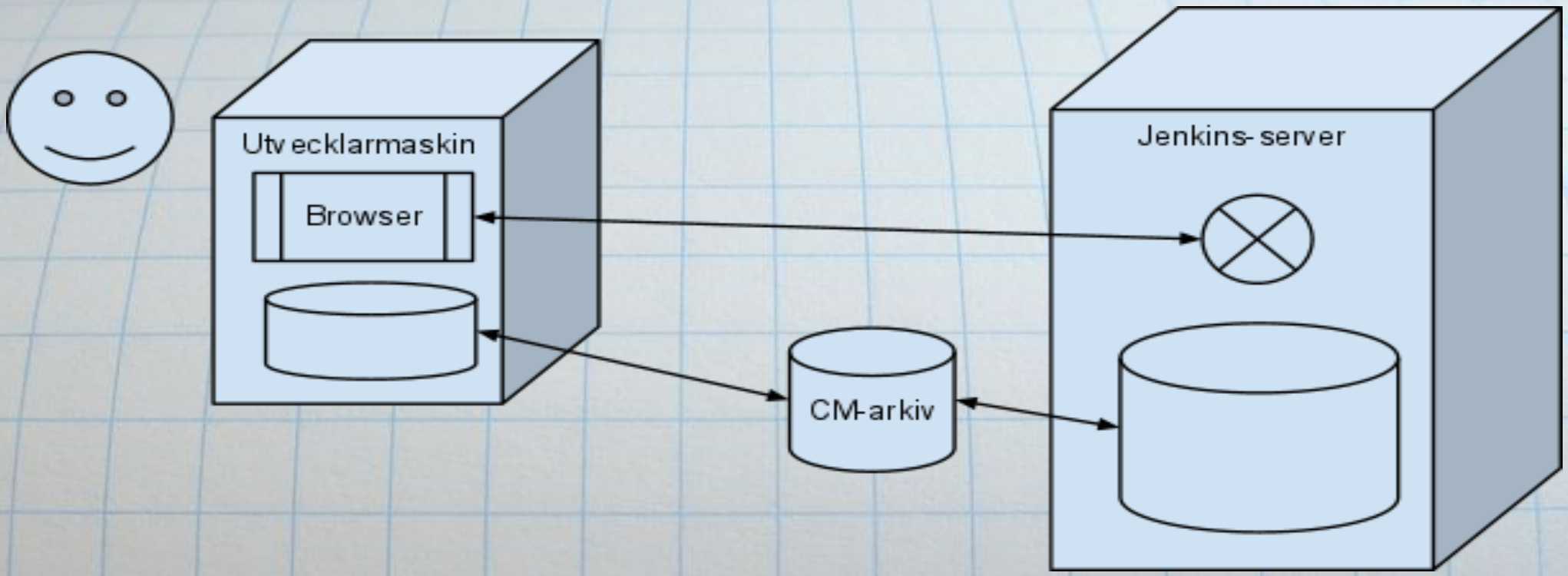
Vilka mål har varit vägledande för utvecklingen av Jenkins

- Få bort tråkigt repetitivt arbete från utvecklaren som annars lätt glömmer det
  - köra automatiserade testfall och statisk analys
- Lätt att installera
- Lätt att konfigurera

Köra testfallen i Jenkins

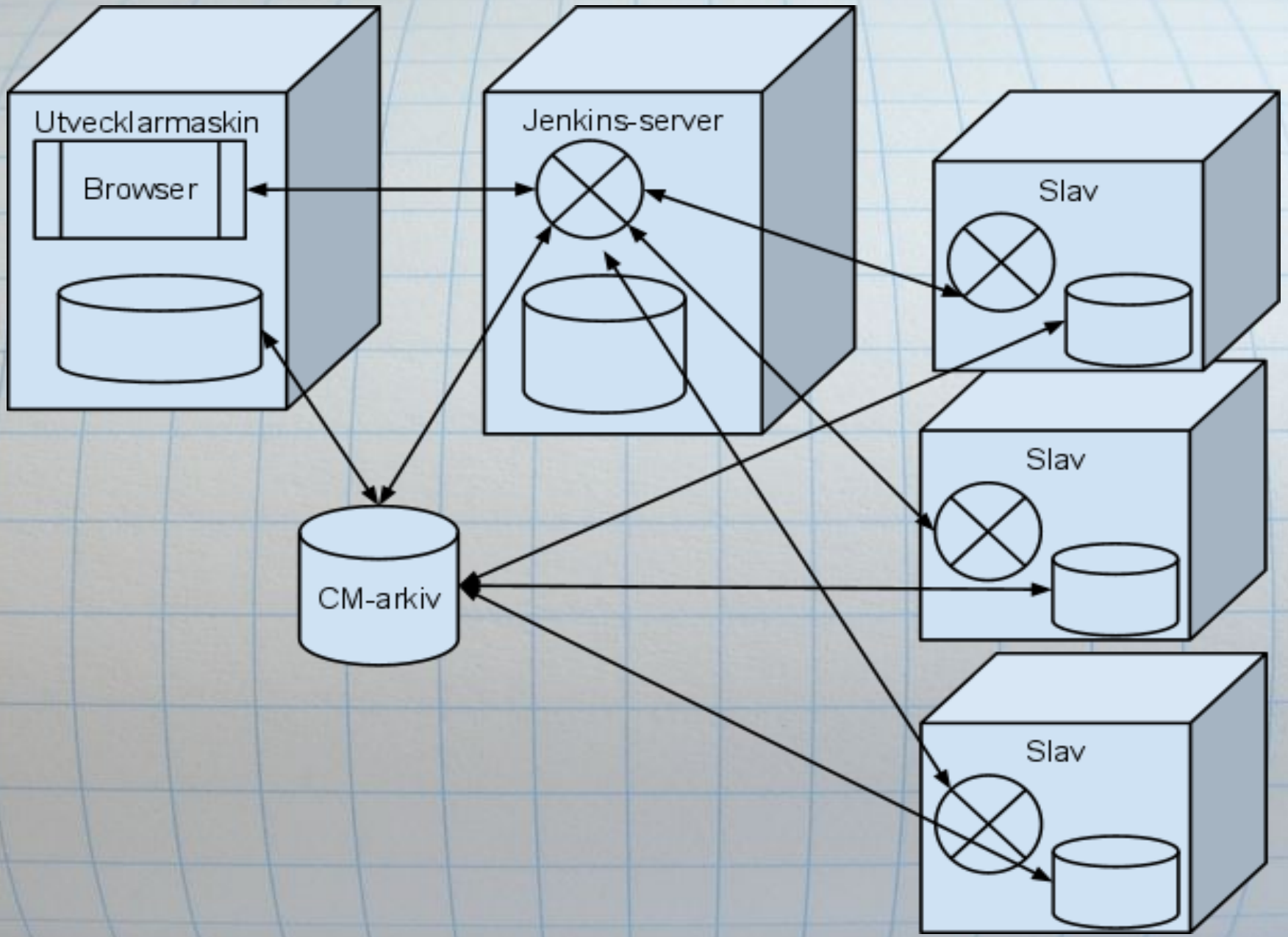


Köra testfallen i Jenkins



- Utvecklaren jobbar vidare
- Utvecklaren kollar läget i nästa naturliga paus
- Jenkins skickar mail om något är fel
- Appar, SMS, Häftiga lampor, tutor och annat som talar om när något måste göras.

Köra testfallen i Jenkins



Föredraget

- Grunderna i mjukvaru-CM
- Trender inom mjukvaruutveckling
- Continuous Integration
- Vad är Jenkins
- **Demo**
- Jenkins i ArgoUML-projektet
- Problem och lösningar

Demo

- Skapa nytt jobb
- Kör det
- Olika vyer in ibland jobben
  - Build history
  - Projekt
  - Byggen
  - Diagram
- Titta på resultatet
  - Olika perspektiv
  - Vad blir enklare med Maven?

Ytterligare en dimension i resultatet

Tidsdimensionen

- Lätt att hitta vilken incheckning som införde problemet
- Se rättade fel
- För statisk analys, lätt att se utvecklingen

Översikt av plug-in

CM-arkivtyper
- Dimensions, Selenium

User management
- Github auth, Mysql auth,
- LDAP email

Notification of results
- IRC, Jabber, Google Calendar, Skype, Twitter

Rapporteringssammanställning
- Cobertura, PMD, findbugs

Föredraget

- Grunderna i mjukvaru-CM
- Trender inom mjukvaruutveckling
- Continuous Integration
- Vad är Jenkins
- Demo
- **Jenkins i ArgoUML-projektet**
- Problem och lösningar

CI och Jenkins i ArgoUML-projektet

- Testresultat på webben sedan september 2003
- "Nightly build" sedan juni 2005
- Hudson sedan mars 2009

Byte till Hudson

- Enklare att uppgradera verktyg (Maven)
- Rapporter får den extra dimensionen
- Enklare publicering av resultatet

Föredraget

- Grunderna i mjukvaru-CM
- Trender inom mjukvaruutveckling
- Continuous Integration
- Vad är Jenkins
- Demo
- Jenkins i ArgoUML-projektet
- **Problem och lösningar**

Potentiella problem
Jenkins föreslagna lösning
ArgoUMLs lösning

Problem: Bygget startar efter halva incheckningen

Kan hända om man kör

- CVS eller Dimensions som inte har atomära incheckningar över flera filer
- Har flera subversion-arkiv som skall byggas och gör incheckningar över flera
- Använder subversion-verktyg eller processer som inte gör atomära incheckningar

Jenkins lösning:

- Jenkins kan vänta en stund efter senaste incheckning innan den startar jobbet

Problem: För många incheckningar

Kan uppstå om:

- De byggmaskiner man har är för få eller för svaga så att de inte hinner med.
- Man har väldigt många utvecklare.

Jenkins lösning:

- Varje bygge startar på det senaste som finns i CM-arkivet. Om incheckningarna är för många kommer inte varje incheckning att byggas.

Annan lösning:

- Köp fler eller kraftigare maskiner.

Problem: Långa skript för att bygga

Kan uppstå om man vidareutvecklar skript direkt i Jenkins

Jenkinsutvecklarens rekommendation:

- Lägg skripten i något CM-arkiv som ett verktyg

Lösning som vi använder i ArgoUML-projektet:

- Hämta ut dem i samband med bygget

Konsekvenser:

- Allt byggs om om man ändrar något av skripten (oftast bra).

Exotiskt problem: Långsam lina

Kan uppstå om:

- Man har många olika byggen mot samma CM-arkiv
- Man har konfigurerat för att radera och checka ut allt för varje bygge
- Långsam lina mellan Jenkins-servern och CM-arkivet
- CM-arkivet har hög belastning

Lösning som vi använder för ArgoUML:

- Kör en spegling av CM-arkivet på Jenkins-servern. Speglingen uppdateras regelbundet, till exempel av ett Jenkins-jobb
- Kör alla jobb mot det speglade CM-arkivet istället för det ordinarie