Guðlaugur Stefán Egilsson

Samfelld afhending

Continuous integration - delivery - deployment

Spurningarnar

- Hvað þetta er
- Af hverju það er mikilvægt
- Hvað þarf til
- Hvenær það er praktískt
- Og hvenær ekki

Hvað er samfelld afhending?

- Alsjálfvirk afhendingarpípa
- Samsett af að lágmarki
  - Útgáfustjórnun (Version Control)
  - Byggingarumhverfi (Build environment)
  - Byggingarferli (Build script)
  - Sjálfvirkri uppfærslu á keyrsluumhverfum (dev/test/staging/production)

Virki sem er hægt að reisa

- Sjálfvirk “offline” próf
  - Unit tests, end-to-end tests, integration/functional tests

- Álagspróf með raungögnum
  - Vandamálið er yfirleitt að geta stýrt tímaþættinum

- Sjálfprófanir í prófana og raunumhverfi
  - e. Module self-tests.

- Vaxandi uppsetning (e. Paritioned deployment)
  - Möguleiki ef notendum er skipt í hópa
  - Cluster uppsetningar

- Sjálfvirkar mælingar á lykiltölfræði (KPA)

- Bakka sjálfvirkt út úr uppfærslu

Continuous Delivery

Útgáfustýrt - rekjanlegt

1%

Þýðir/byggist

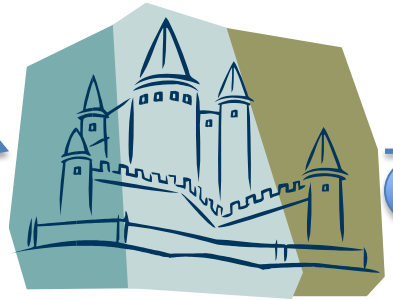
2%

95%

98%

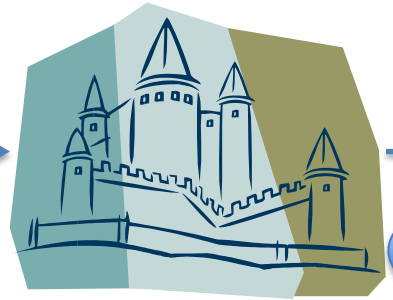
Samþáttunarpróf keyra 100%

Einingapróf keyra 100%



Takkinn

99.5%



99.9x%

Virknipróf keyra 100%

Sjálfprófanir og lykil tölfræði í lagi

Af hverju er þetta mikilvægt?

- Fækka samþáttunarvandamálum. Í flestum tilfellum hverfa þau alveg.
- Hraðari endurgjöf. Færð mikið af upplýsingum um hvað þú varst að gera bara með “build passed”
- Fullkomlega fyrirsjáanlegt og endurtakanlegt ferli við þýðingu, prófanir og uppsetningu.
- Engin handavinna við að uppfæra kerfi

Tæknileg geta – hvað þarf til?

- Að kunna að forrita og hanna…
- Kunna að skrifa einingapróf
  - Helst með Test Driven Development
  - Fókuseruð samþáttunarpróf
- Kunna að skrifa virknipróf
  - T.d. DOM fókuseruð test m. Selenium, TestDriver
- “Build kerfi”
  - Ant, NAnt, MSBuild, Maven, Gradle, Rake, Grunt…
- Kunna á Continuous Integration tól
  - TeamCity, Jenkins/Hudson, Bamboo, AntHill…
- Fyrir skalanleika
  - Module repositories & dependency management
  - Maven, NuGet, NPM, RubyGems, etc
- Admin kunnátta
  - Geta sett upp stýrikerfi, öll keyrsluumhverfi (DB, vefþjóna, þjónustur, notendur og réttindi, o.s. frv)
  - Kunna að “scripta” umhverfið
    - Afritun milli véla, stjórnun á services/daemons ofl.
  - Config mgmt tól, t.d. Chef.

Hvenær er þetta praktískt?

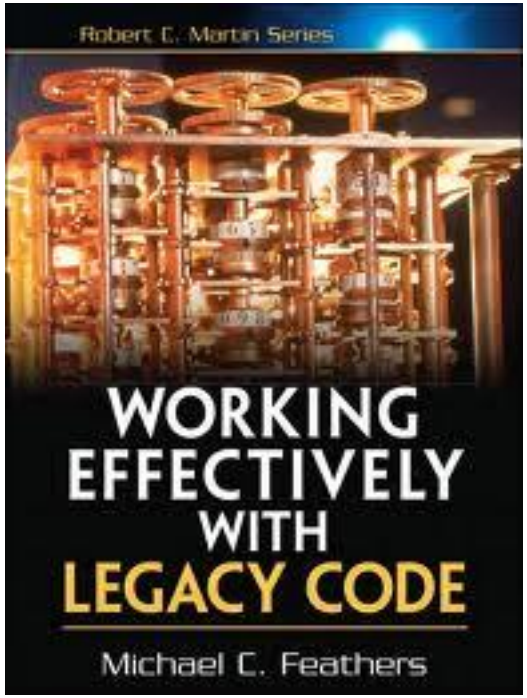
	Integration	Delivery	Deployment
Vefur	Alltaf	Alltaf	Yfirleitt (*1)
“Desktop”	Alltaf (*2)	Yfirleitt(*3)	Stundum (*3)
Mobile	Yfirleitt(*4)	Fer eftir dreifiaðila	Fer eftir dreifiaðila
Embedded	Yfirleitt(*4)	Sjaldan	???

Miðað er við nýtt kerfi þróað frá grunni með núverandi tæknilega getu innan Spretts.
Hvað er praktískt fer mikið eftir reynslu og þekkingu.

*1 Þetta er eingöngu spurning um sjálfsöryggi og traust, sem er fall af hæfni og “test coverage”
*2 Hversu mikið af sjálfvirkum prófum er hægt að byggja inn er mjög misjafnt eftir umhverfum.
*3 Afhendingarleiðir fyrir desktop hugbúnað eru mjög misjafnar. Continuous delivery/deployment krefst “auto update” útfærslu, ýmist með eða án staðfestingar notanda.
*4 Hversu nytsamlegt þetta er, fer eftir hversu ítarlegt hermiumhverfi er til staðar

Að byrja í kerfinu mínu…

- Finna lítinn part af kerfinu sem gæti hentað í samfellda samþáttun
- Byggja upp þennan part af kerfinu með sjálfvirknivæðingu í huga
- Nýta núverandi gæðaferla – og auka sjálfvirknivæðingu smám saman, sér í lagi unit test
- Eingöngu setja upp build frá buildserver– hætta að setja upp build beint frá þróunarvélum
- Auka tíðni á uppsetningum í dev/test – markmiðið er að gera þær að ekki-frétt
- Bæta við “Deploy” takka fyrir raunumhverfi. Að ýta á takkann er sameiginleg ákvörðun hagsmunaaðila
- Endurbæta ferli þar til traust er nægilegt til að að hætta með takkann, og hafa ferlið alfarið sjálfvirkt



- The key test is that a business sponsor could request that the current development version of the software can be deployed into production at a moment's notice - and nobody would bat an eyelid, let alone panic.

  - Martin Fowler