Tampere University

Lecture 5: continuous deployment

Outline of today

- Course matters
- Discussion about ”homework”
- Continuous Delivery & Deployment

Tampere University

Participants (28.09.2021)

- Total number of enrolled students: 72
- Answers to survey: 63
- Current participants: 60
- Docker exercise: 54
- Docker compose exercise: 17

Tampere University

Next

Next exercise timetable:

- 29.09/05.10/12.10: Ansible exercise
- 05.10/12.10/19.10: Message Queue exercise

Will be a bit delayed. Follow the email!

Homework from last week

- What are the things that I did not mention in last week's lecture

- "*Volumes*, on the other hand, are physical areas of disk space shared between the host and a container, or even between containers. In other words, **a volume is a shared directory in the host**, visible from some or all containers."
  - On point was forgotten

- "Similarly, ***networks*** **define the communication rules between containers, and between a container and the host**. Common network zones will make containers' services discoverable by each other, while private zones will segregate them in virtual sandboxes."
  - Discoverable?
  - Segregate?

Tampere University

Example (~same as last week)

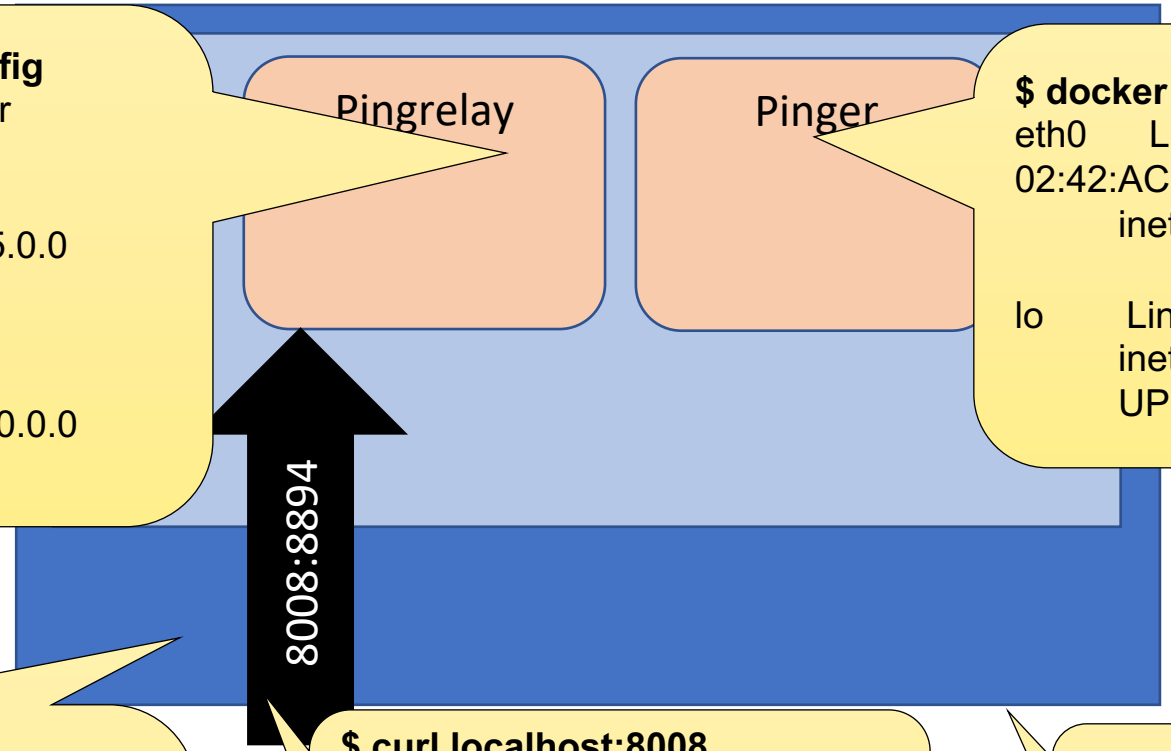
```
version: '3'
services:
  pinger:
      build: pinger
      image: "pinger"
#      ports:
#          - "8893:8893"
      networks:
         - pingnet
      volumes:
         - ./data:/data
      environment:
         ServiceName: service_2
  pingrelay:
      build: "pingrelay"
      ports:
         - "8008:8894"
      networks:
         - pingnet
      volumes:
         - ./data:/data
      environment:
         ServiceName: service_1
networks:
    pingnet:
volumes:
  data: {}
```

```
more pinger/Dockerfile
FROM node:6.10.0-alpine

# Never run processes as root!
USER root

# Copy application itself
COPY . /home
WORKDIR /home
# Set port on which to run the node process:
ENV PORT=8893
# Expose port 8893:
EXPOSE 8893

CMD node pinger.js
```



$ docker exec c81d9d91915d ifconfig
eth0 Link encap:Ethernet HWaddr 02:42:AC:12:00:03
inet addr:**172.18.0.3**
Bcast:172.18.255.255 Mask:255.255.0.0
UP BROADCAST RUNNING

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK

$ docker exec beb1f83a47d4 ifconfig
eth0 Link encap:Ethernet HWaddr 02:42:AC:12:00:02
inet addr:**172.18.0.2**

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING

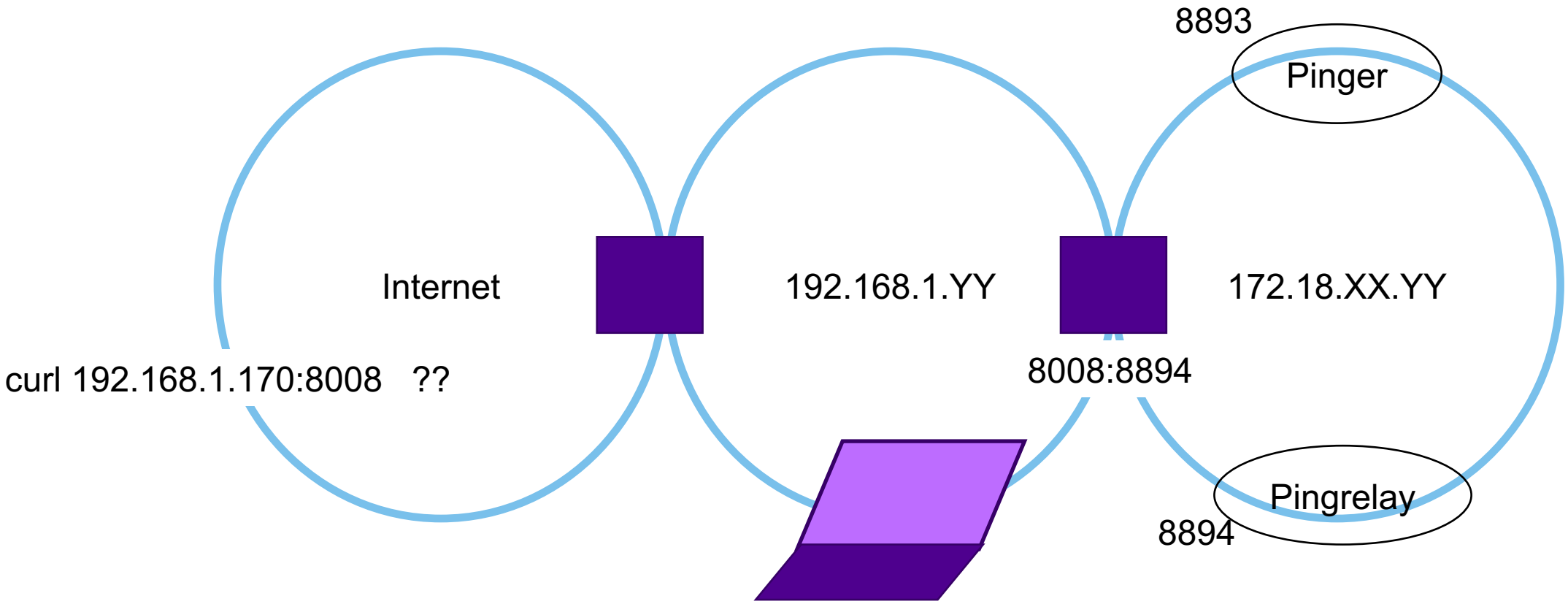
$ ifconfig
br-00e4ab43a2d4:
flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet **172.18.0.1** netmask 255.255.0.0
broadcast 172.18.255.255
inet6 fe80::42:9bff:fece:ce18 prefixlen 64
scopeid 0x20<link>

$ curl localhost:8008
Hello from ::ffff:172.18.0.1:33196
to ::ffff:172.18.0.3:8894
Hello from ::ffff:172.18.0.3:50838
to ::ffff:172.18.0.2:8893

$ curl 172.18.0.2:8893
Hello from ::ffff:172.18.0.1:41232
to ::ffff:172.18.0.2:8893

$ curl 172.18.0.3:8894
Hello from ::ffff:172.18.0.1:33204
to ::ffff:172.18.0.3:8894
Hello from ::ffff:172.18.0.3:50846
to ::ffff:172.18.0.2:8893

wks-91544-mac:~ systa**$ curl 192.168.1.170:8008**
Hello from ::ffff:192.168.1.32:63068
to ::ffff:172.18.0.3:8894
Hello from ::ffff:172.18.0.3:50852
to ::ffff:172.18.0.2:8893


8893
Pinger
Internet
192.168.1.YY
172.18.XX.YY
curl 192.168.1.170:8008 ??
8008:8894
Pingrelay
8894

Tampereen yliopisto
Tampere University

Continuous Delivery and Deployment

Tampereen yliopisto
Tampere University

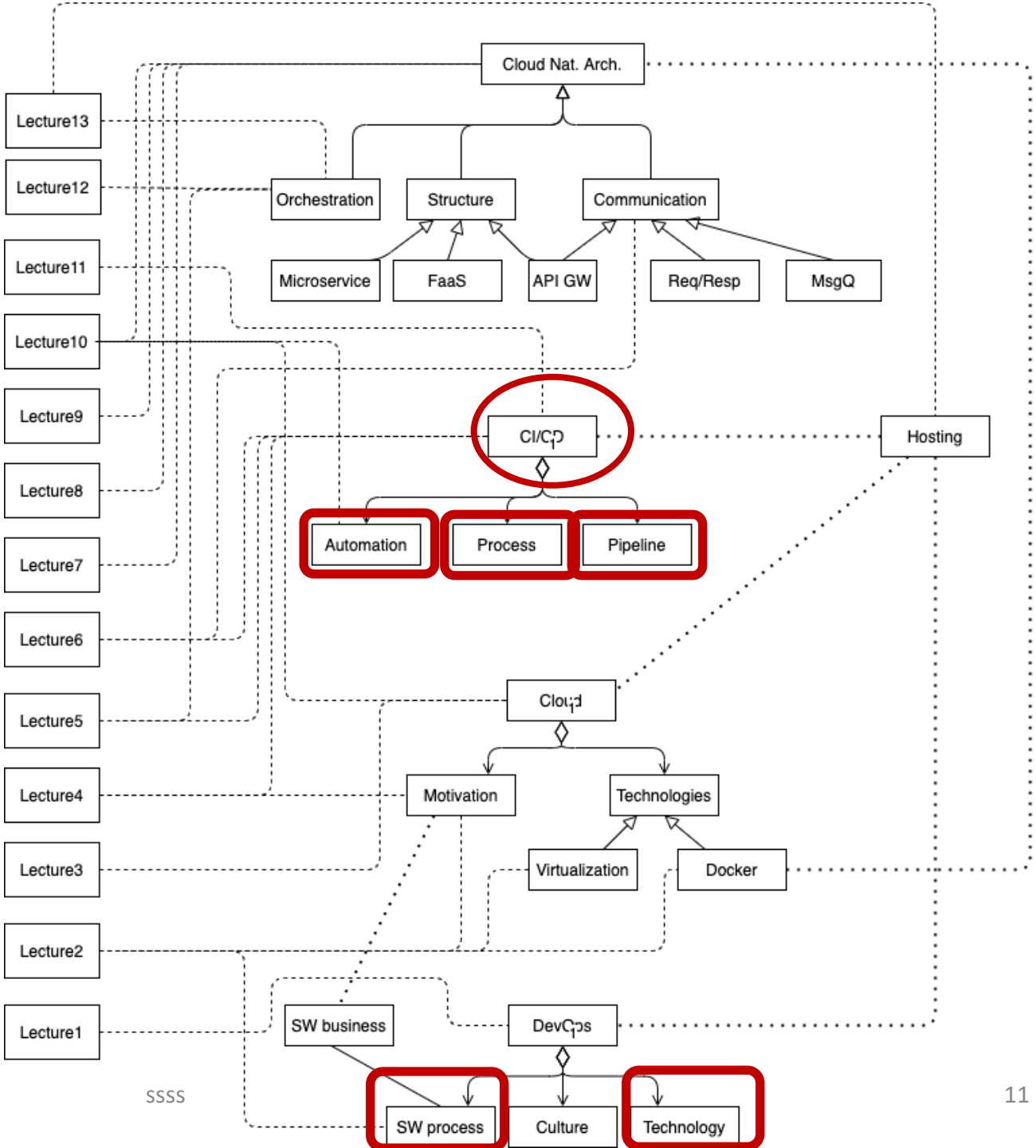
What is DevOps (there are several definitions)

- Lucy Lwakatare:
  - DevOps is a concept that embodies a **cultural and mindset change** that is substantiated with a **set of practices** to encourage **cross-disciplinary collaboration between software development and IT operations** within a software company. The main purpose for the collaboration is to enable the **fast release of quality software changes** while simultaneously **operating resilient systems**.
  - From a **socio-technica perspective**, DevOps practices are focused on the **automation practices** o nfrastructure management, specifically a on management and monitoring.

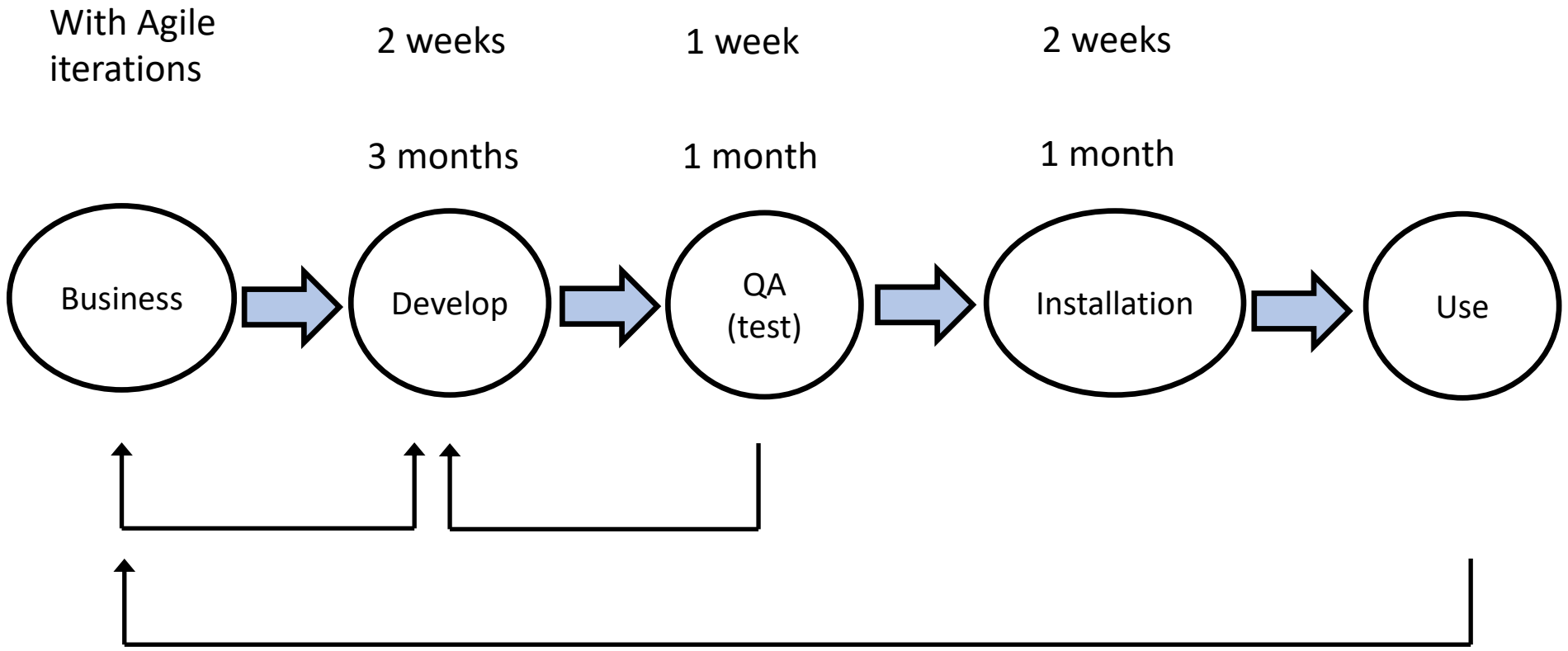
Continuous Delivery

Tampereen yliopisto
Tampere University

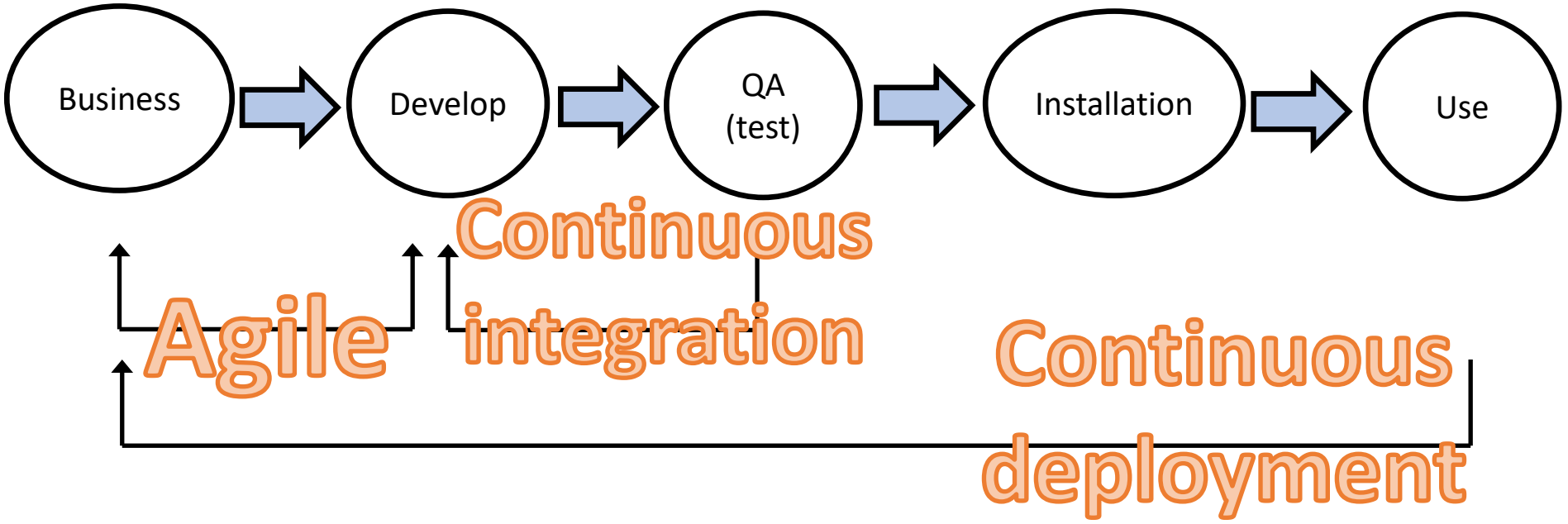
Relation to our content



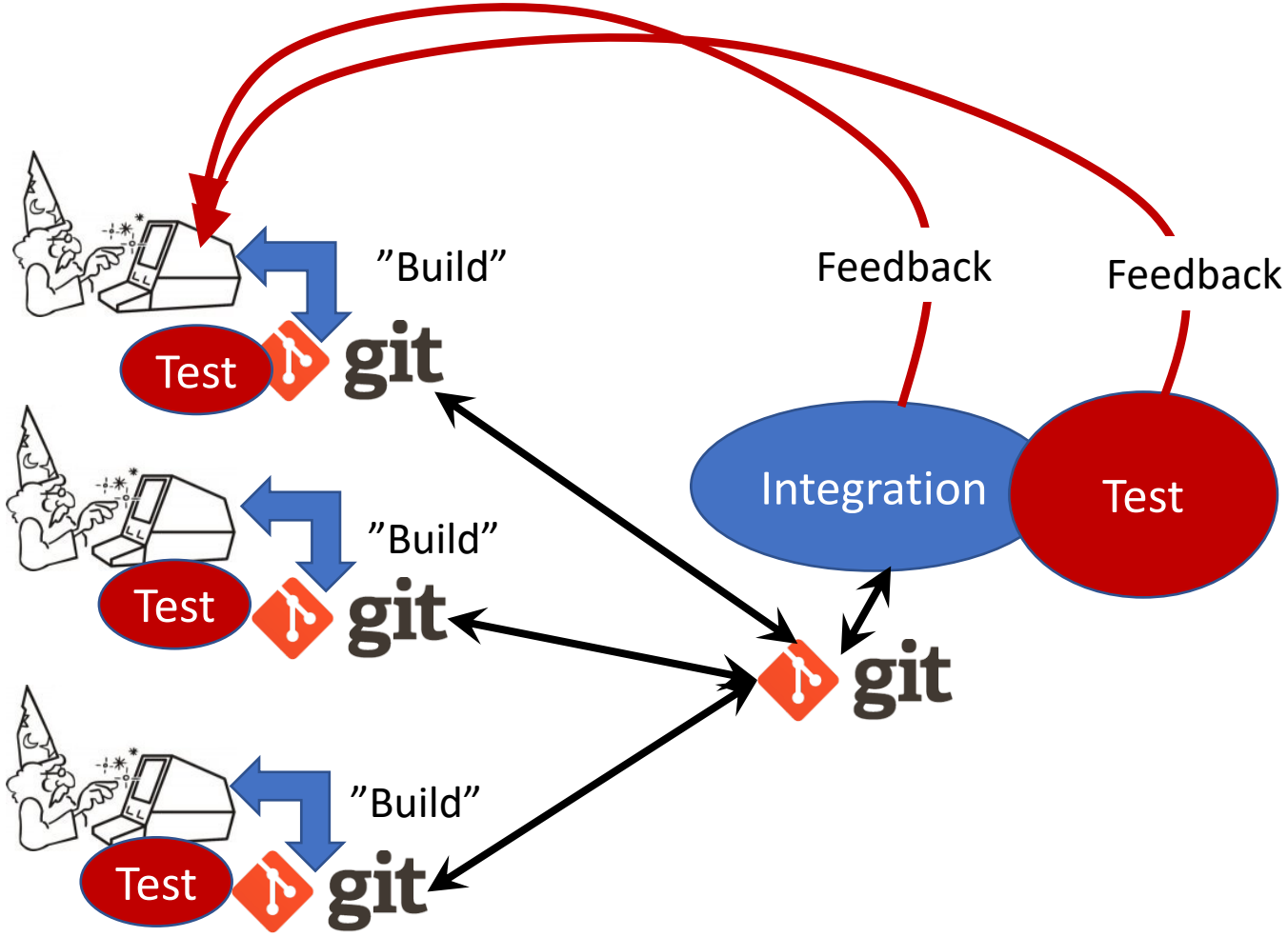
Feedback in traditional development
(Case: Internet-based service; based on slide by Antti Tirilä)



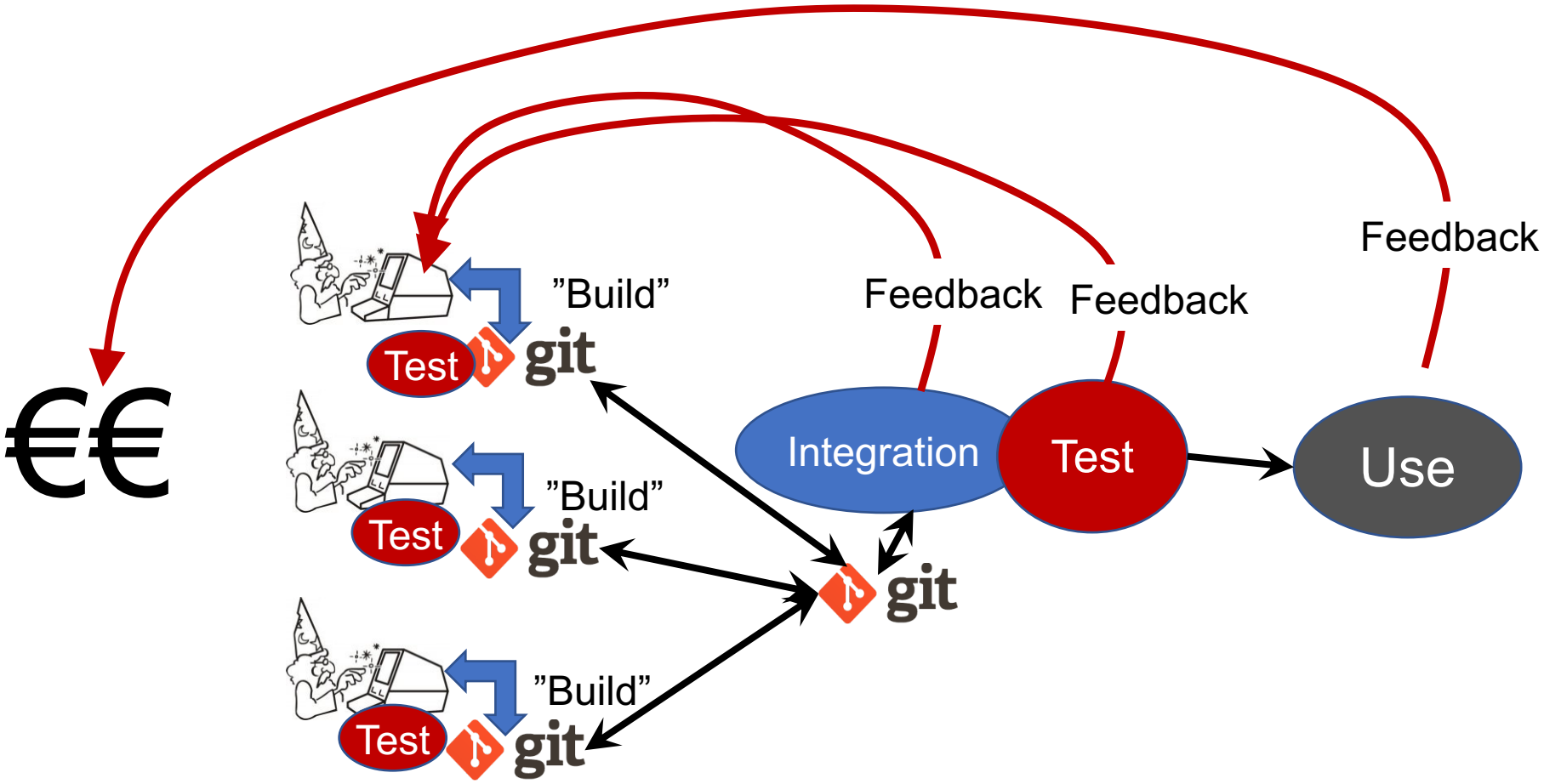
Feedback in traditional development (Case: Internet-based service; based on slide by Antti Tirilä)



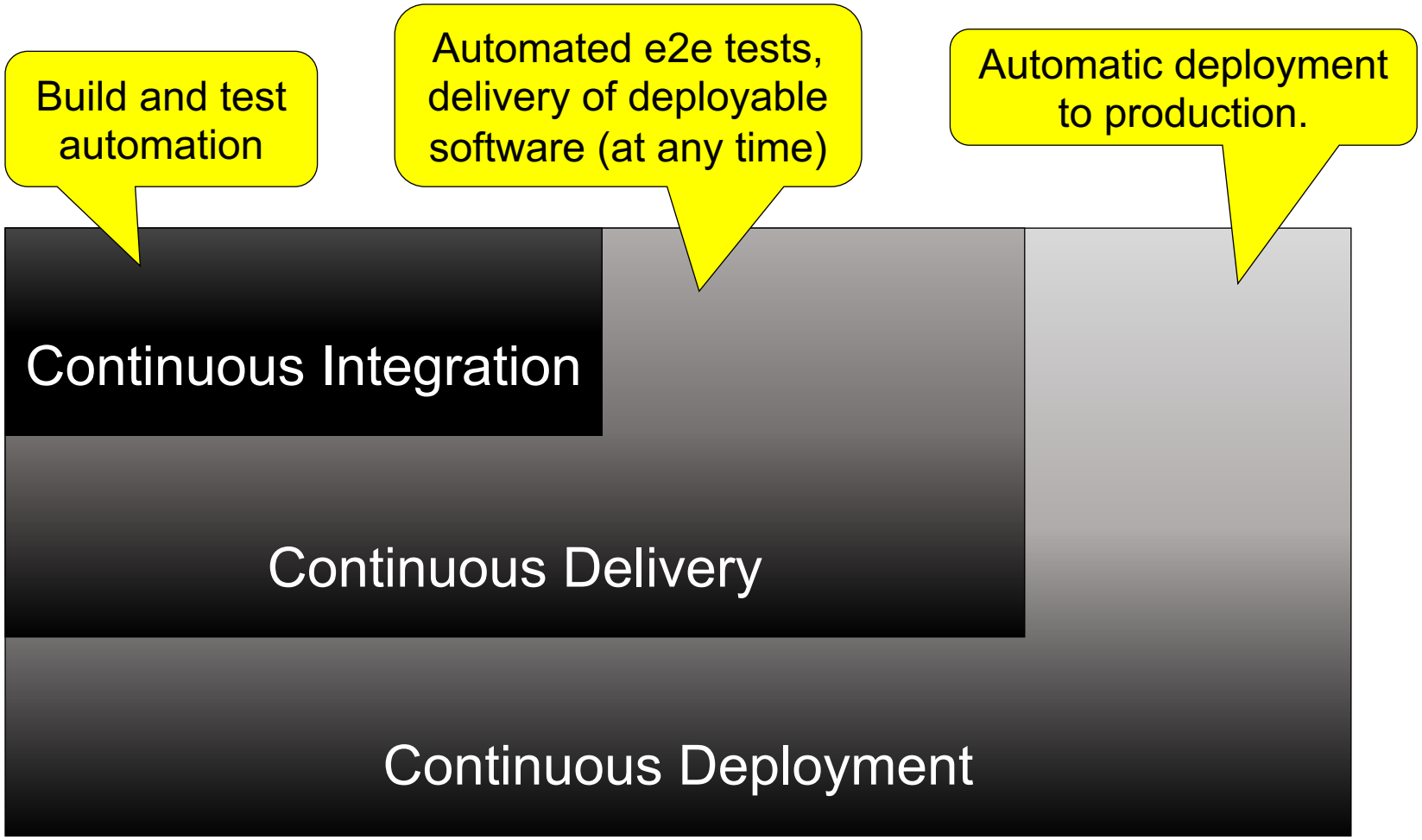
Continuous integration



Continuous deployment

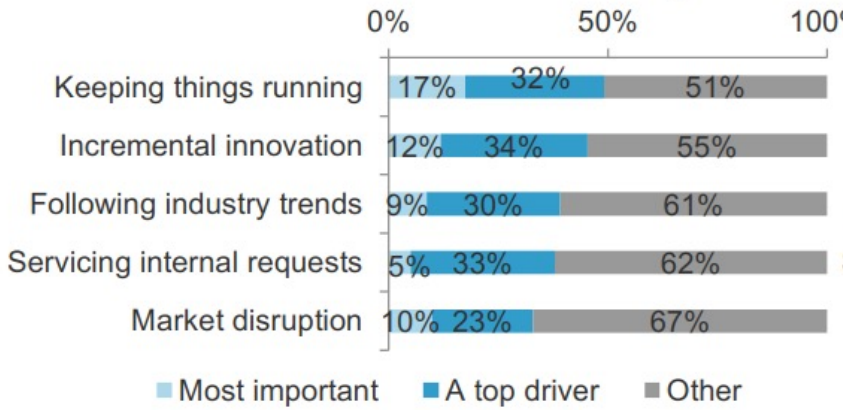


Continuous X

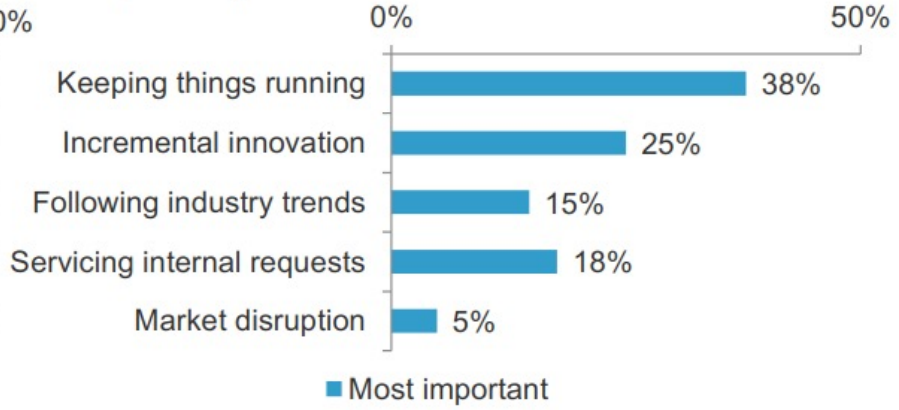


From Forrester report: Continuous Delivery: A Maturity Assessment Model: Building Competitive Advantage With Software Through A Continuous Delivery Process, 2013

"Please rank the importance of the business drivers behind your software development investments within your business area (current)."

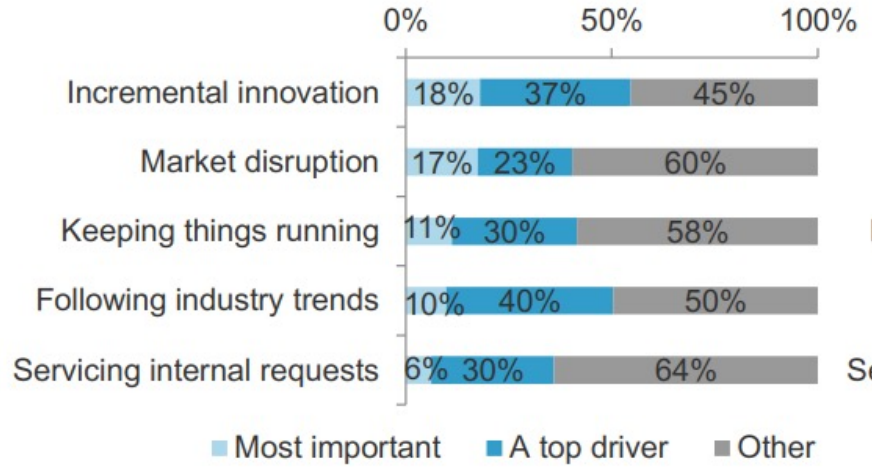


Base: 161 business decision-makers

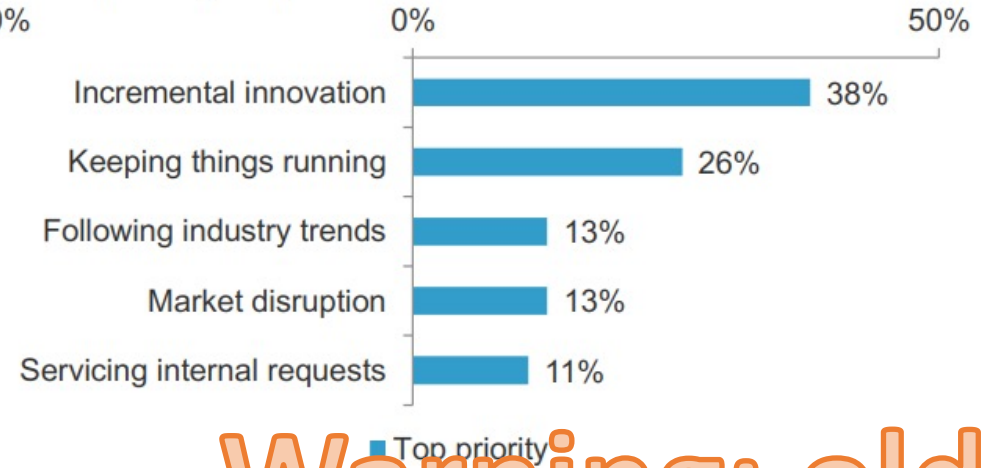


Base: 164 IT executives and managers

"Please rank the importance of the business drivers behind your software development investments within your business area (in two years)."



Base: 161 business decision-makers



Base: 164 IT executives and managers

Source: A commissioned study conducted by Forrester Consulting on behalf of Thoughtworks, September 2012

Warning: old stuff

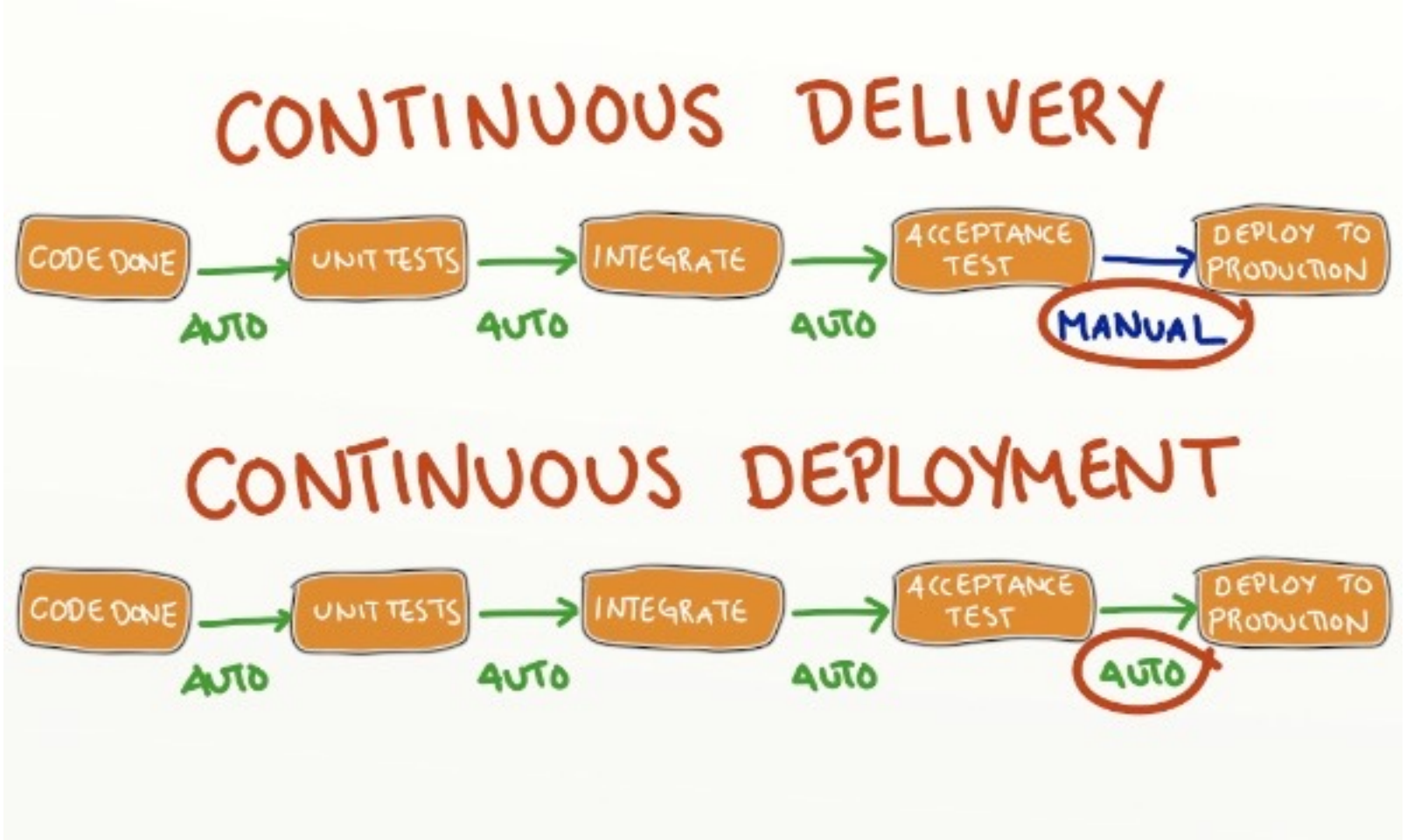
Google trends after that 2003

DevOps

Continuous deployment

Continuous delivery and deployment

(http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment)



Tampereen yliopisto
Tampere University

Main principles (https://continuousdelivery.com/principles/)

- Build quality in
- Work in small batches
- Computers perform repetitive tasks, people solve problems
- Relentlessly pursue continuous improvement
- Everyone is responsible

Sound familiar from somewhere?

Tampereen yliopisto
Tampere University

CI – essential practices
(according to Humbley and Farley)

- Don’t check in on a broken code
- Always run all commits tests locally before committing, or get your CI server to do it for you
- Wait for commit tests to pass before moving on
- Never go home on a broken build
- Always be prepared to revert to the previous revisions
- Time-box fixing before reverting
- Don’t comment out failing tests
- Take responsible for all breakages that result from your changes
- Test-driven development

Tampereen yliopisto
Tampere University

Deployment essential pract.
(according to Humbley and Farley)

- Only build your binaries once
- Deploy the same way to every environment
- Smoke-test your deployments
- Deploy to copy of production
- Each change should propagate through the pipeline instantly
- If any part of pipeline fails, stop the line

Tampereen yliopisto
Tampere University

Reported HP case-study
(https://continuousdelivery.com/evidence-case-studies/)

They had three high-level goals:

- Create a single platform to support all devices
- Increase quality and reduce the amount of stabilization required prior to release
- Reduce the amount of time spent on planning

A key element in achieving these goals was implementing continuous delivery, with a particular focus on:

- The practice of continuous integration
- Significant investment in test automation
- Creating a hardware simulator so that tests could be run on a virtual platform
- Reproduction of test failures on developer workstations

Tampereen yliopisto
Tampere University

Reported HP case-study
(https://continuousdelivery.com/evidence-case-studies/)

They

- C
- I
- F

A

a p

- T
- S
- C
- F

Results:

- Overall development costs were reduced by ~40%.
- Programs under development increased by ~140%.
- Development costs per program went down 78%.
- Resources driving innovation increased eightfold.

Tampereen yliopisto
Tampere University

Let’s speculate the contribution of each

They had three high-level goals:

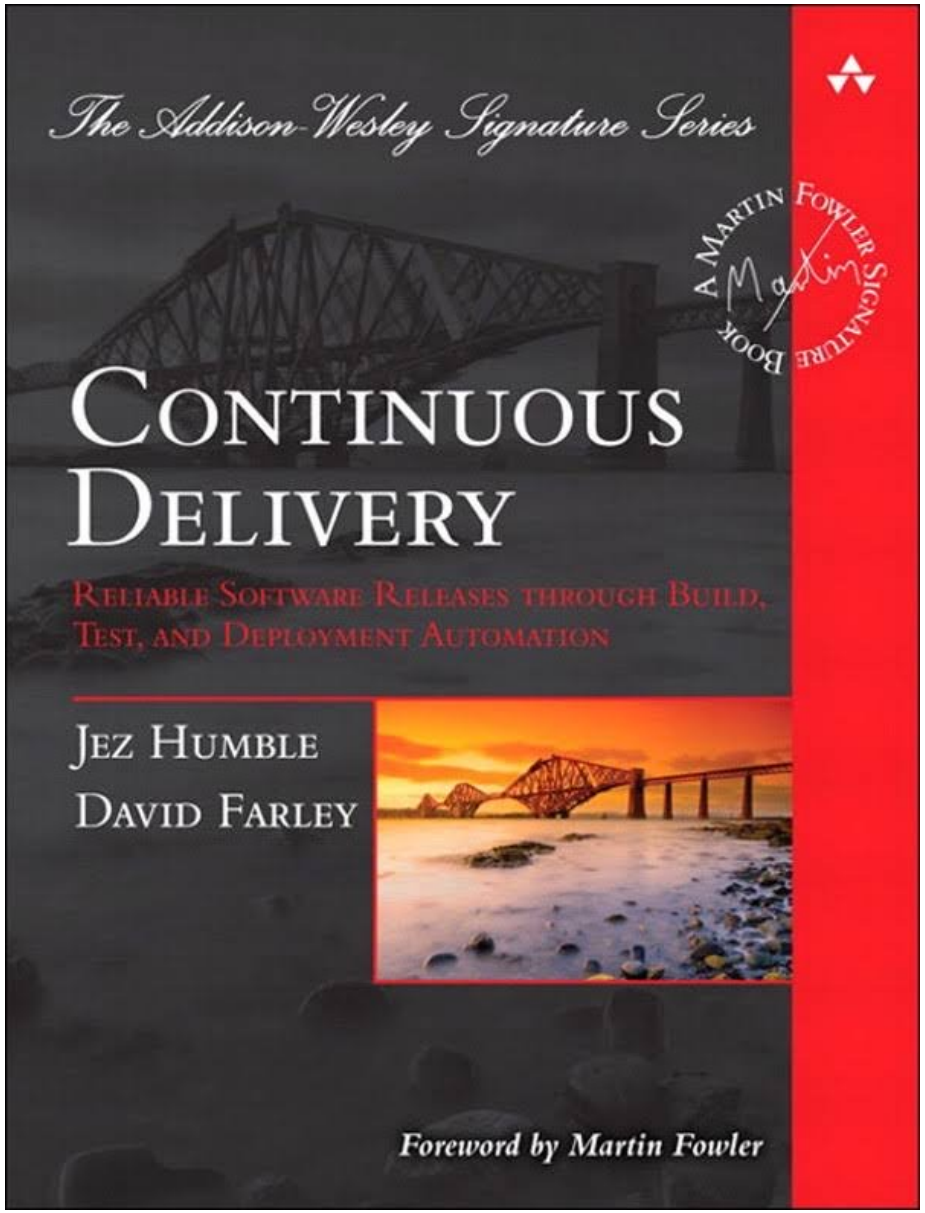
- Create a single platform to support all devices
- Increase quality and reduce the amount of stabilization required prior to release
- Reduce the amount of time spent on planning

A key element in achieving these goals was implementing continuous delivery, with a particular focus on:

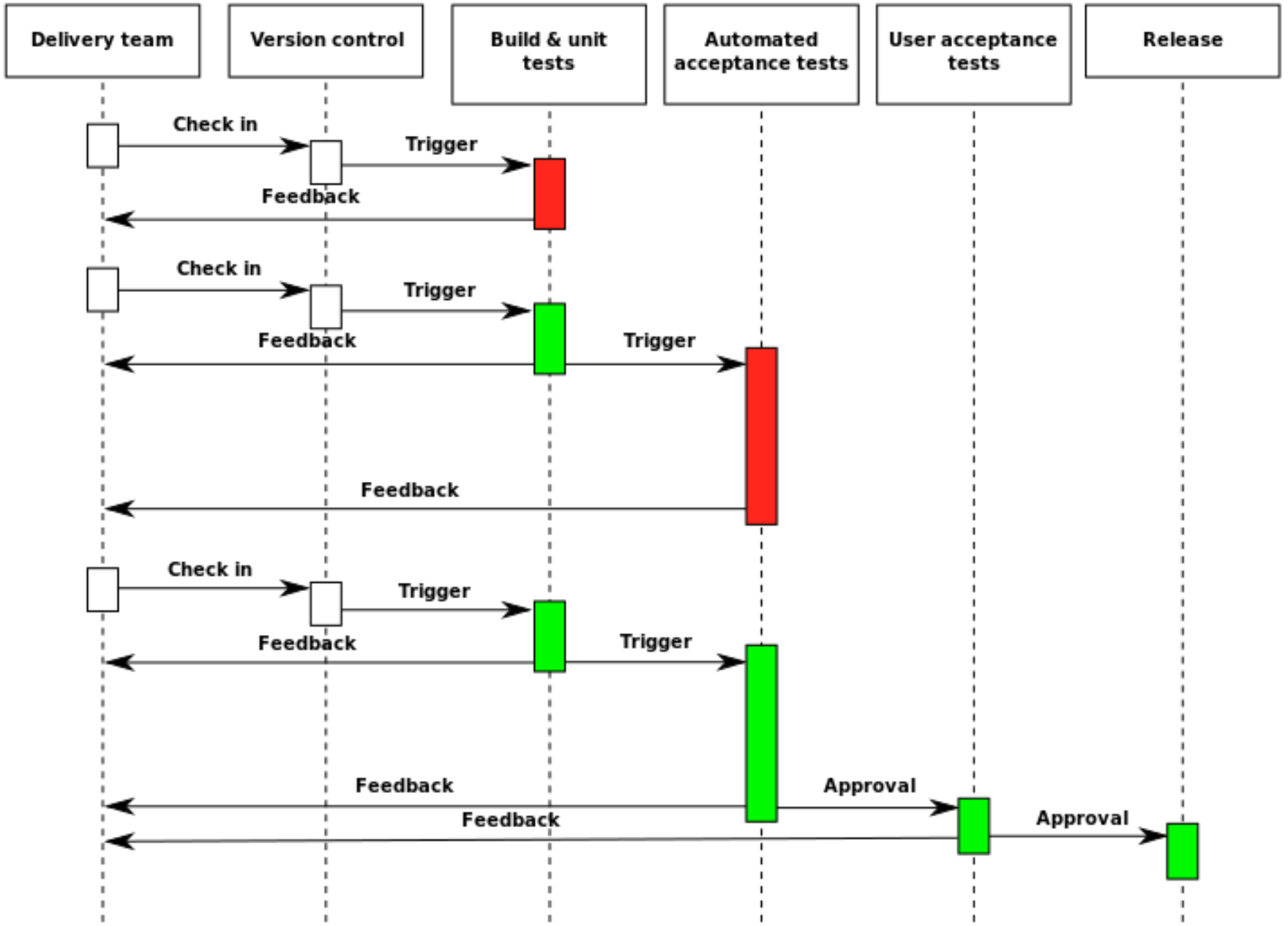
- The practice of continuous integration
- Significant investment in test automation
- Creating a hardware simulator so that tests could be run on a virtual platform
- Reproduction of test failures on developer workstations

Tampereen yliopisto
Tampere University

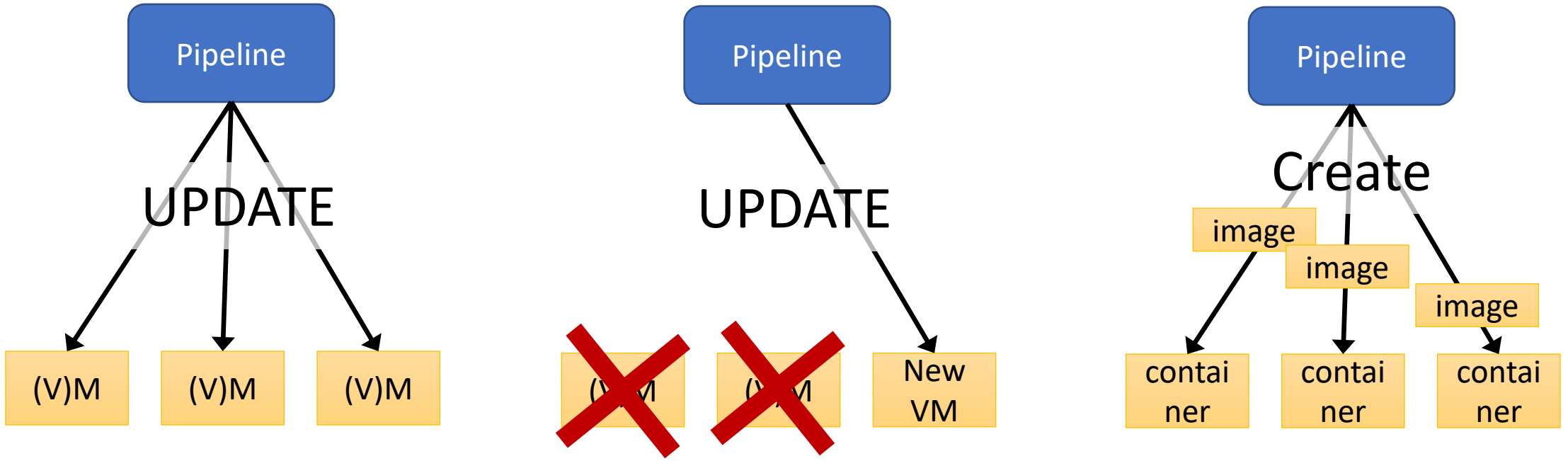
CD: Some technical material



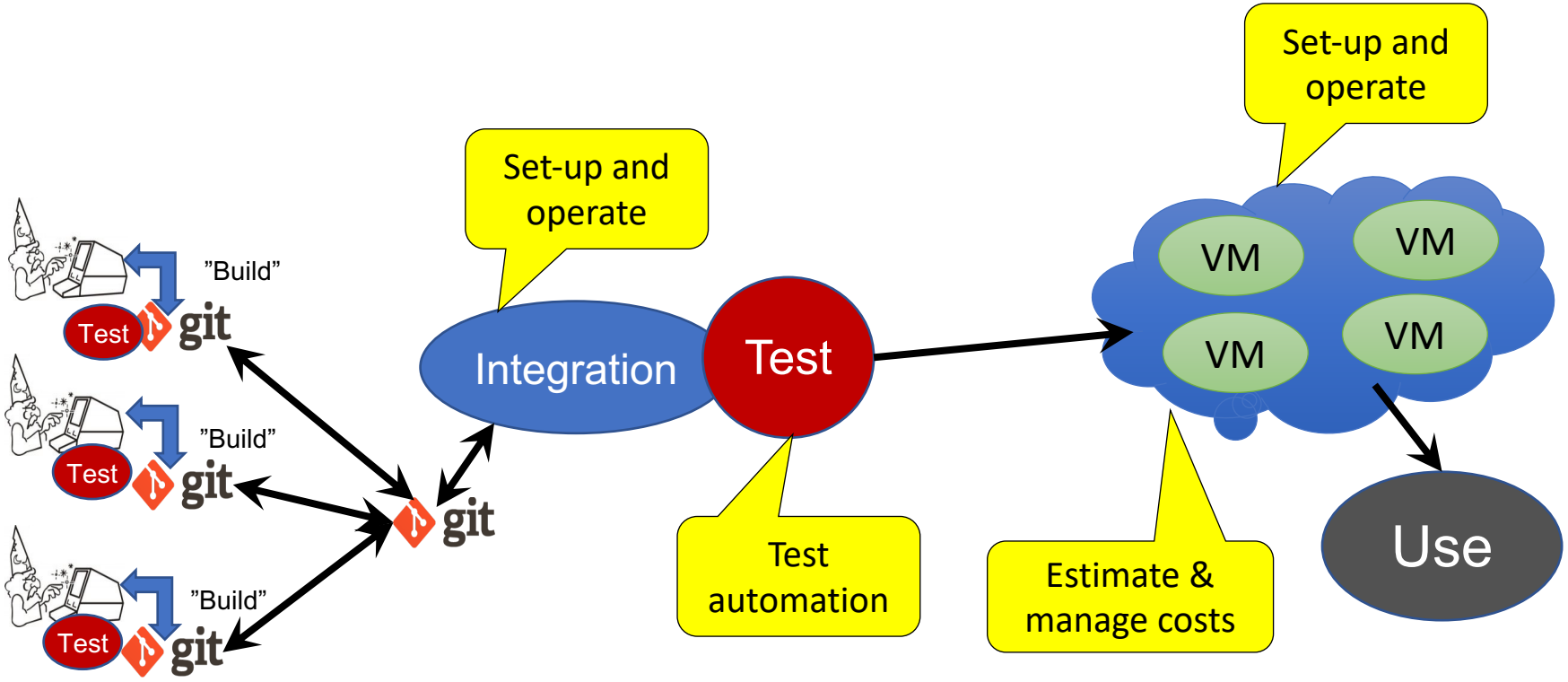
Deployment pipeline (a possible example)



Hypothesis of possible approaches

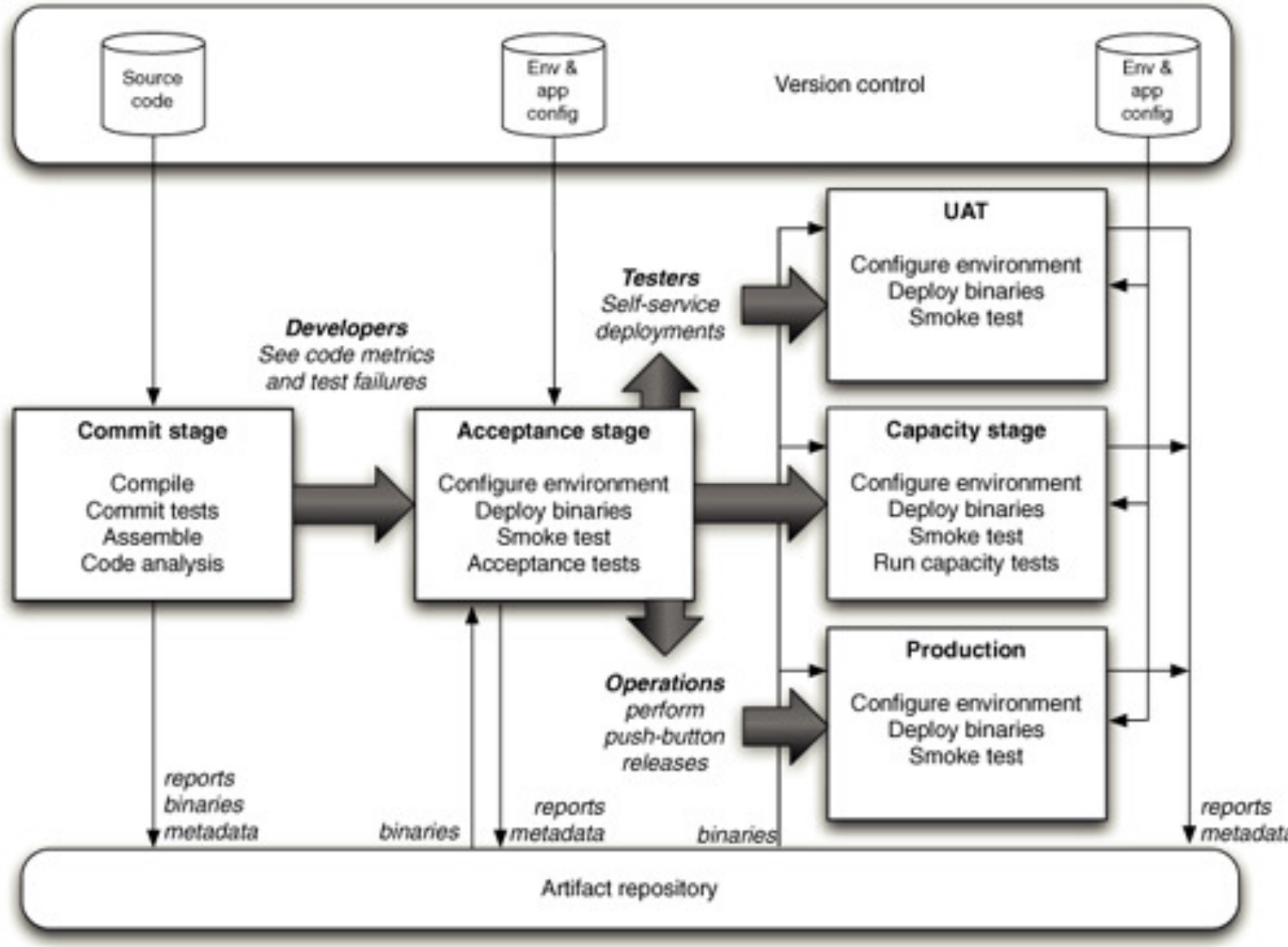


What does it really take to run CD?



Tampereen yliopisto
Tampere University

Artifact repository

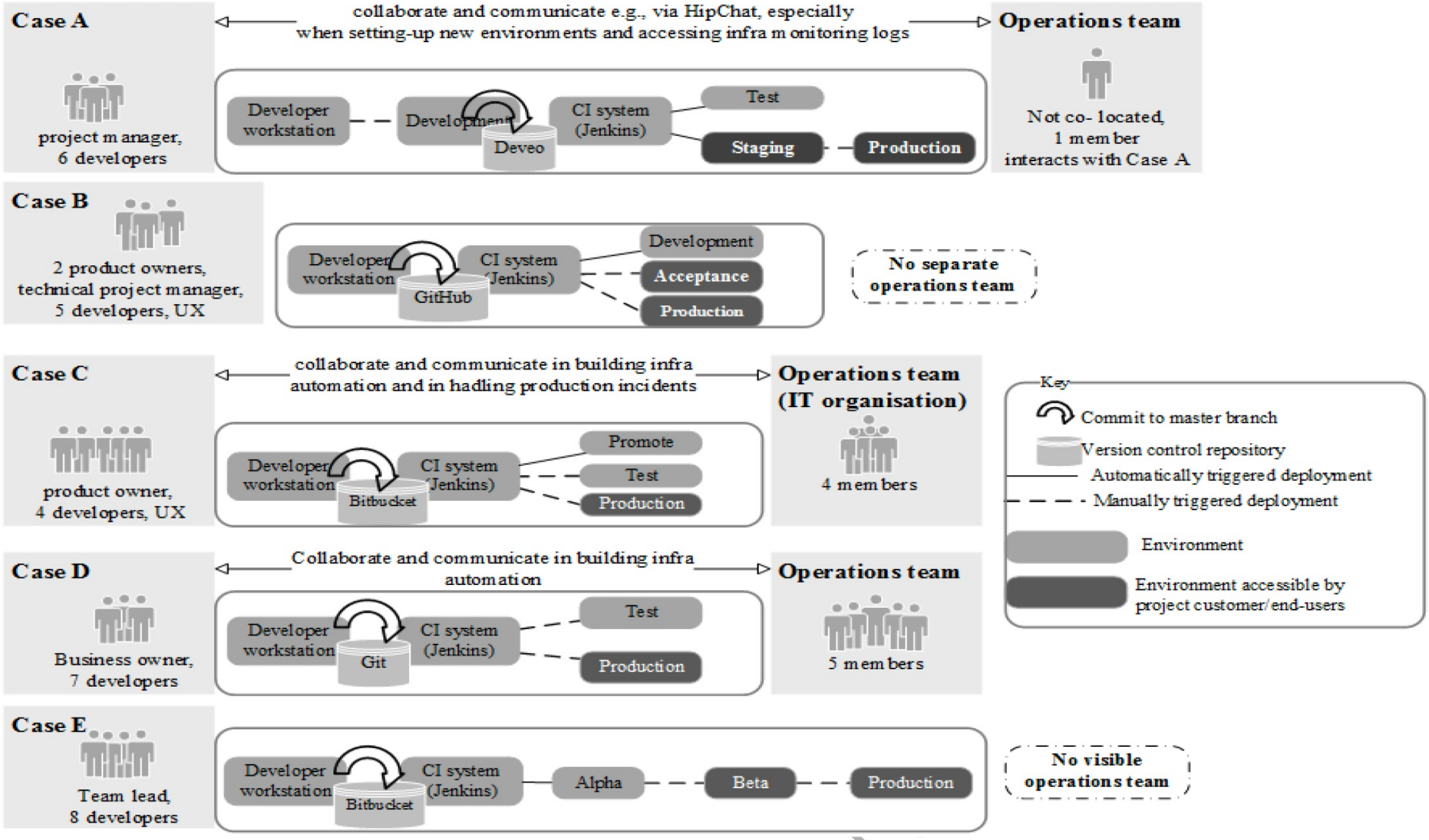


Tampereen yliopisto
Tampere University

Couple of Finnish studies

Tampereen yliopisto
Tampere University

Lwakatare , Kilamo , Karvonen, Sauvola , Heikkilä, Itkonen, Kuvaja, Mikkonen, Oivo & Lassenius:
DevOps in practice : A multiple case study of five companies, Information and Software Technology , vol. 114 , pp. 217-230 . https://doi.org/10.1016/j.infsof.2019.06.010


Case A
project manager, 6 developers
collaborate and communicate e.g., via HipChat, especially when setting-up new environments and accessing infra monitoring logs
Operations team
Not co- located, 1 member interacts with Case A
Developer workstation
Development
Deveo
CI system (Jenkins)
Test
Staging
Production
Case B
2 product owners, technical project manager, 5 developers, UX
Developer workstation
GitHub
CI system (Jenkins)
Development
Acceptance
Production
No separate operations team
Case C
product owner, 4 developers, UX
collaborate and communicate in building infra automation and in hadling production incidents
Operations team (IT organisation)
4 members
Developer workstation
Bitbucket
CI system (Jenkins)
Promote
Test
Production
Case D
Business owner, 7 developers
Collaborate and communicate in building infra automation
Operations team
5 members
Developer workstation
Git
CI system (Jenkins)
Test
Production
Case E
Team lead, 8 developers
Developer workstation
Bitbucket
CI system (Jenkins)
Alpha
Beta
Production
No visible operations team
Key
Commit to master branch
Version control repository
Automatically triggered deployment
Manually triggered deployment
Environment
Environment accessible by project customer/end-users

Tampereen yliopisto
Tampere University

Perceived benefits

- **Improved delivery speed** of software changes Improved speed in the development and deployment of software changes to production environment.
- **Improved productivity in operations** work. Decreased communication problems, bureaucracy, waiting overhead due to removal of manual deployment hand-offs and organisational boundaries; Lowered human error in deployment due to automation and making explicit knowledge of operation-related tasks to software development
- **Improvements in quality**. Increased confidence in deployments and reduction of deployment risk and stress; Improved code quality; Improved product value to customer resulting from production feedback about users and usage.
- **Improvements in organisational-wide culture** and mind-set. Enrichment and wider dissemination of DevOps in the company through discussions and dedicated training groups 'communities of practice'

Tampereen yliopisto
Tampere University

Perceived challenges

- Insufficiencies in infrastructure automation
- High demand for skills and knowledge
- Project and resource constraints
- Difficulties in monitoring, especially for microservice-based applications and in determining useful metrics
- Difficulties in determining a right balance between the speed of new functionality and quality.

Tampereen yliopisto
Tampere University

Summary of the findings

(i) software development team attaining ownership and responsibility to deploy software changes in production is crucial in DevOps.

(ii) toolchain usage and support in deployment pipeline activities accelerates the delivery of software changes, bug fixes and handling of production incidents. (ii) the delivery speed to production is affected by context factors, such as manual approvals by the product owner

(iii) steep learning curve for new skills is experienced by both software developers and operations staff, who also have to cope with working under pressure.

Tampereen yliopisto
Tampere University

Leppänen, Mäkinen, Pagels, Eloranta, Itkonen, Mäntylä, Männistö
The highways and country roads to continuous deployment,
IEEE Software, vol. 32, no. 2, pp. 64-72, Mar.-Apr. 2015.
doi: 10.1109/MS.2015.50

” Interviews with 15 information and communications technology companies revealed the benefits of and obstacles to continuous deployment. Despite understanding the benefits, none of the companies adopted a fully automatic deployment pipeline.”

Tampereen yliopisto
Tampere University

State of the practice (2014)

- Only one company had completely automatic pipeline to deployable product; no one really to production
- Fastest time from code change to production
  - 5min – 4 weeks
    (for web application developers longest time was 1 day)
- Cycle-time to potentially deployable software
  - 20min – 1 months
- Full deployment cycle
  - 1 hour – 1.5 years

Tampereen yliopisto
Tampere University

Perceived benefits 1/2

- Faster feedback
  - to development
  - From users to decision making
- More Frequent Releases
  - ” less waste because the features weren’t waiting in the development pipeline to be released.”
- Improved Quality and Productivity
  - robust automated deployment with a comprehensive test suite
  - reduced scope for each release

Tampereen yliopisto
Tampere University

Perceived benefits 2/2

- Improved Customer Satisfaction
  - new product features provided better customer service
  - (reported by 5 out of 15 interviewed organisations)
- Effort Savings
  - three interviewees reported
  - automation saved time
- Closer Connection between Development and Operations
  - only one reported !

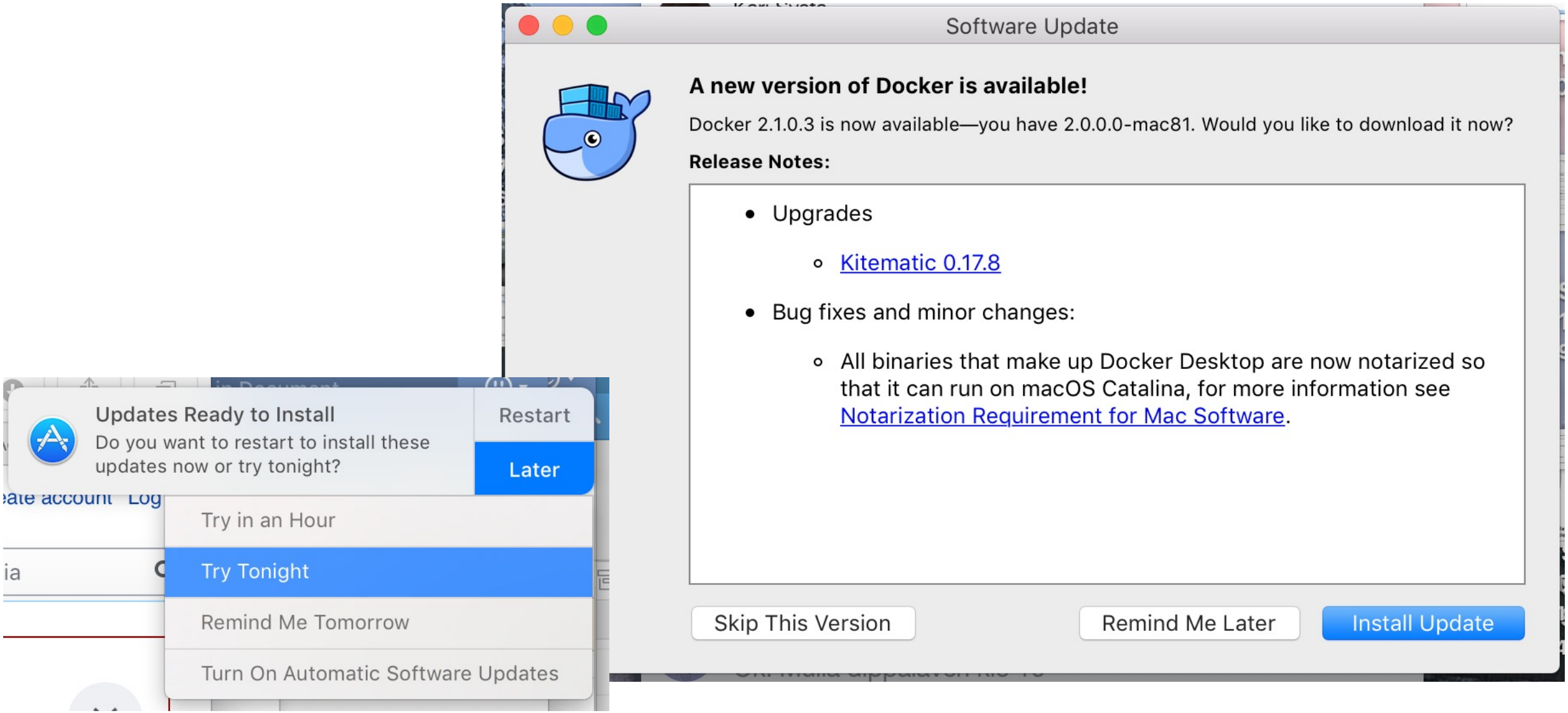
Tampereen yliopisto
Tampere University

Obstacles 1/2

- Resistance to Change
  - Organization culture, management, social relations, …
- Customer Preferences
  - Might be reluctant to deal with more frequent releases
- Domain Constraints
  - Telecom, Medical, Embedded, …
  - Distribution channels
- Developer Trust and Confidence
  - Proficiency and knowledge of typical continuous-deployment practices
  - Reliable automated testing (… even browser-bases apps)

Tampereen yliopisto
Tampere University

About resistance



Tampereen yliopisto
Tampere University

Obstacles 2/2

- Legacy Code Considerations
  - Quality has decreased over time
  - Not be designed to be automatically tested
- Duration, Size, and Structure
  - Effort to create the pipe-line and tests is big
  - In big projects the execution of tests will also take time
- Different Development and Production Environments
  - Especially ”embedded”
- Manual and Nonfunctional Testing