Tampereen yliopisto
Tampere University

Lecture 14 - recap

Kari Systä, 07.12.2021

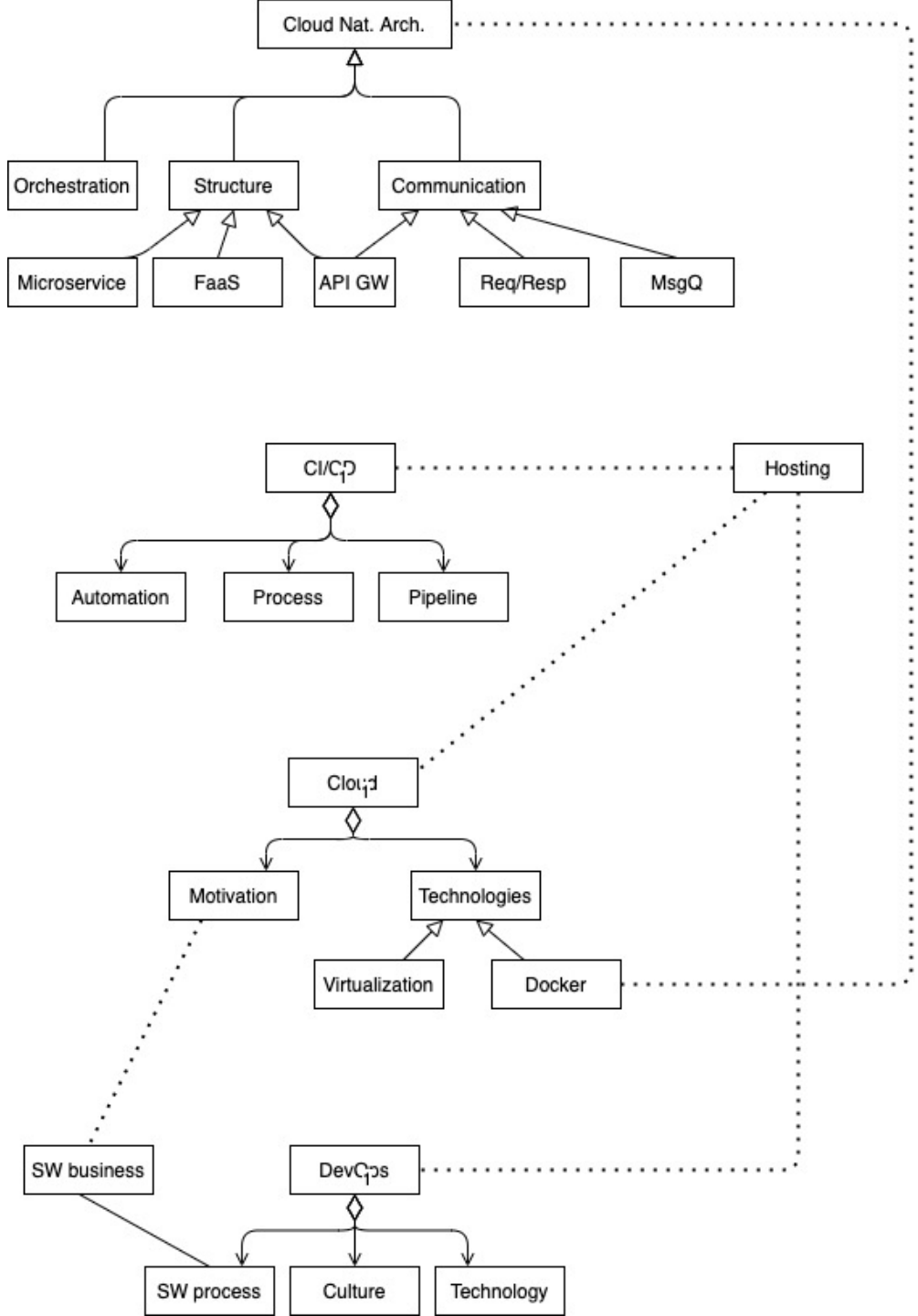
Tampereen yliopisto
Tampere University

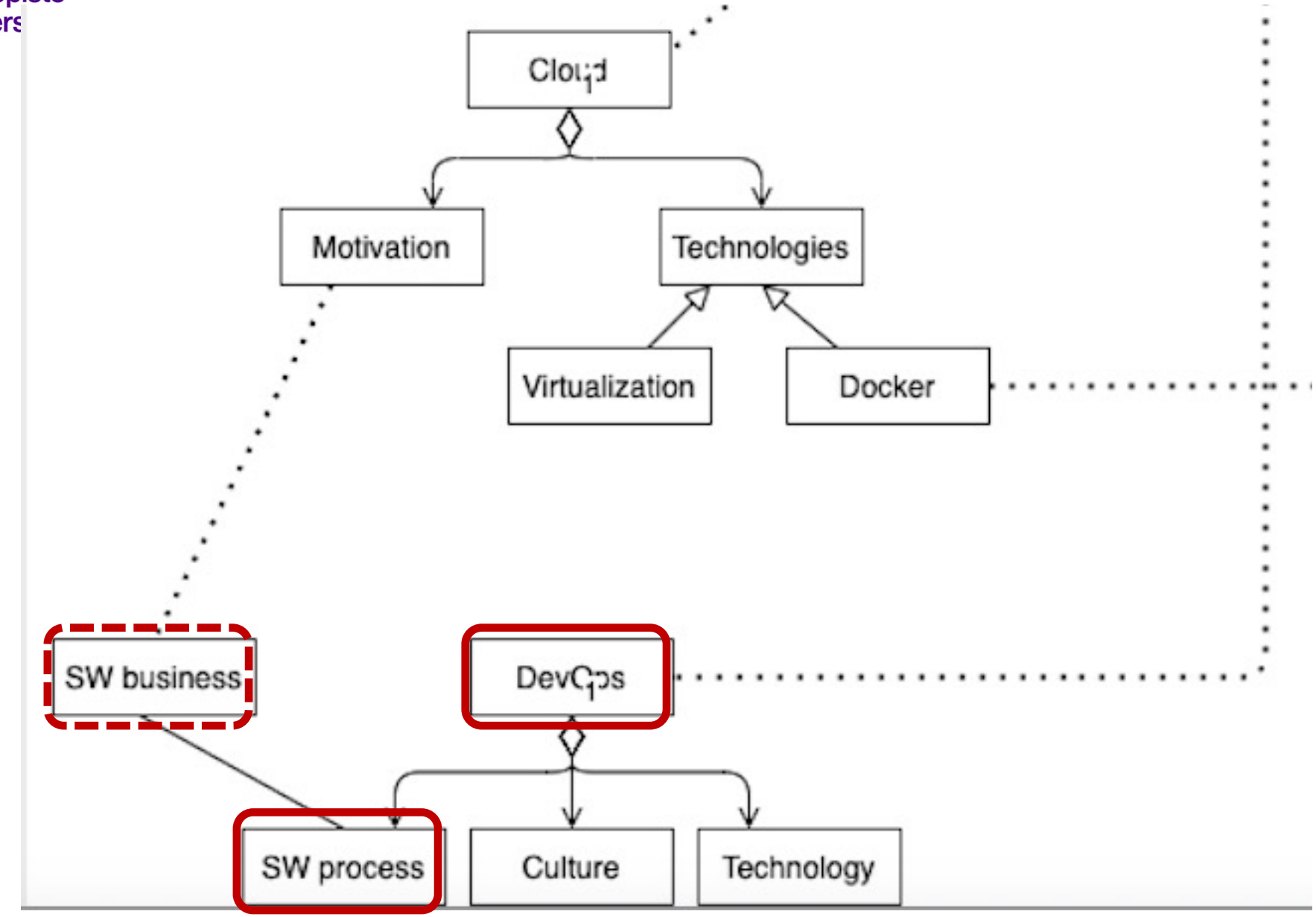
Course status

- Total nbr: 61
- Docker: 59
- Docker Compose: 54
- Message Queue: 40

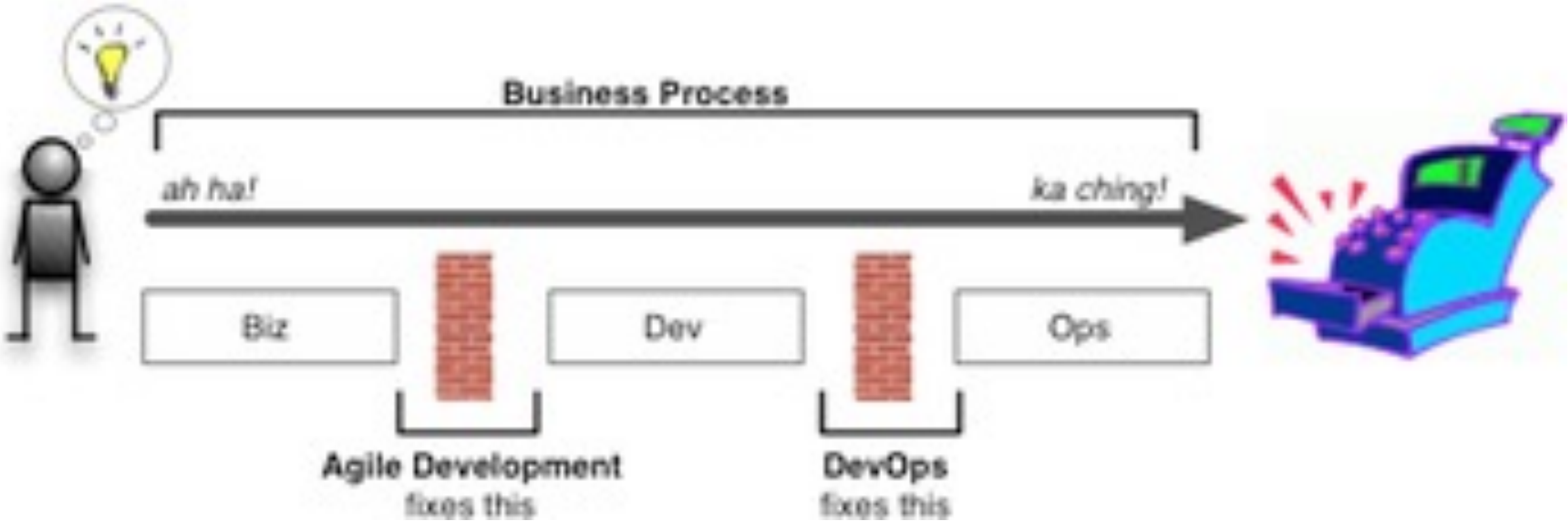
- Ansible: 14
- Project: 0

Tampereen yliopisto
Tampere University




Cloud
Motivation
Technologies
Virtualization
Docker
SW business
DevOps
SW process
Culture
Technology

The lifecycle



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-technical perspective**, DevOps practices are focused on the **automation practices** of software deployment and infrastructure management, specifically automation of configuration management and monitoring.

Tampereen yliopisto
Tampere University

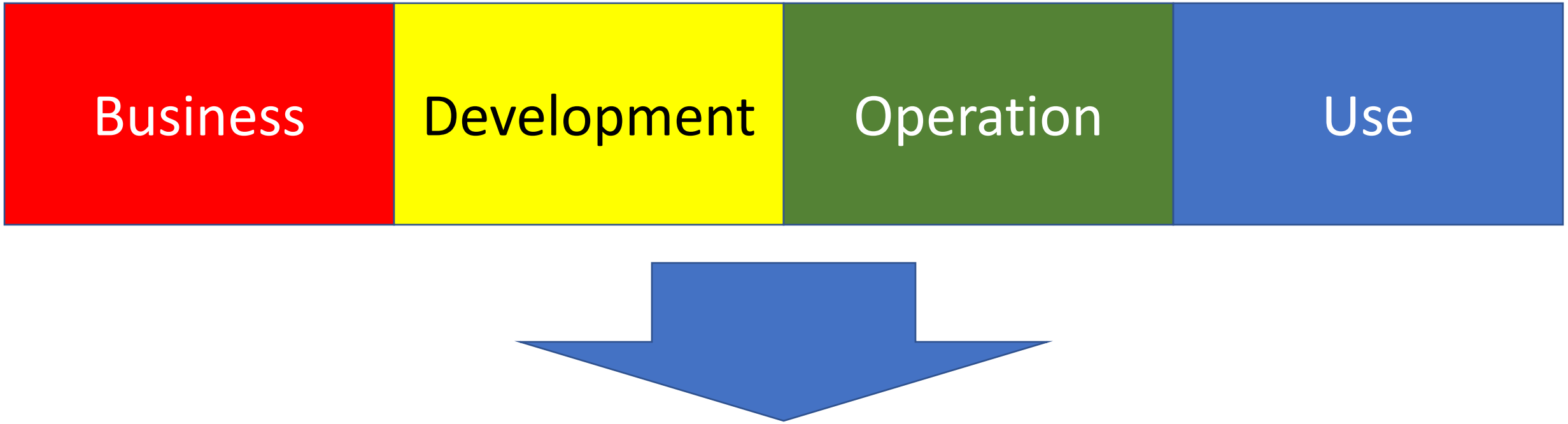
DevOps practices

- Organizational
  - increased scope of responsibilities for developers;
  - intensified cooperation between development and operations.

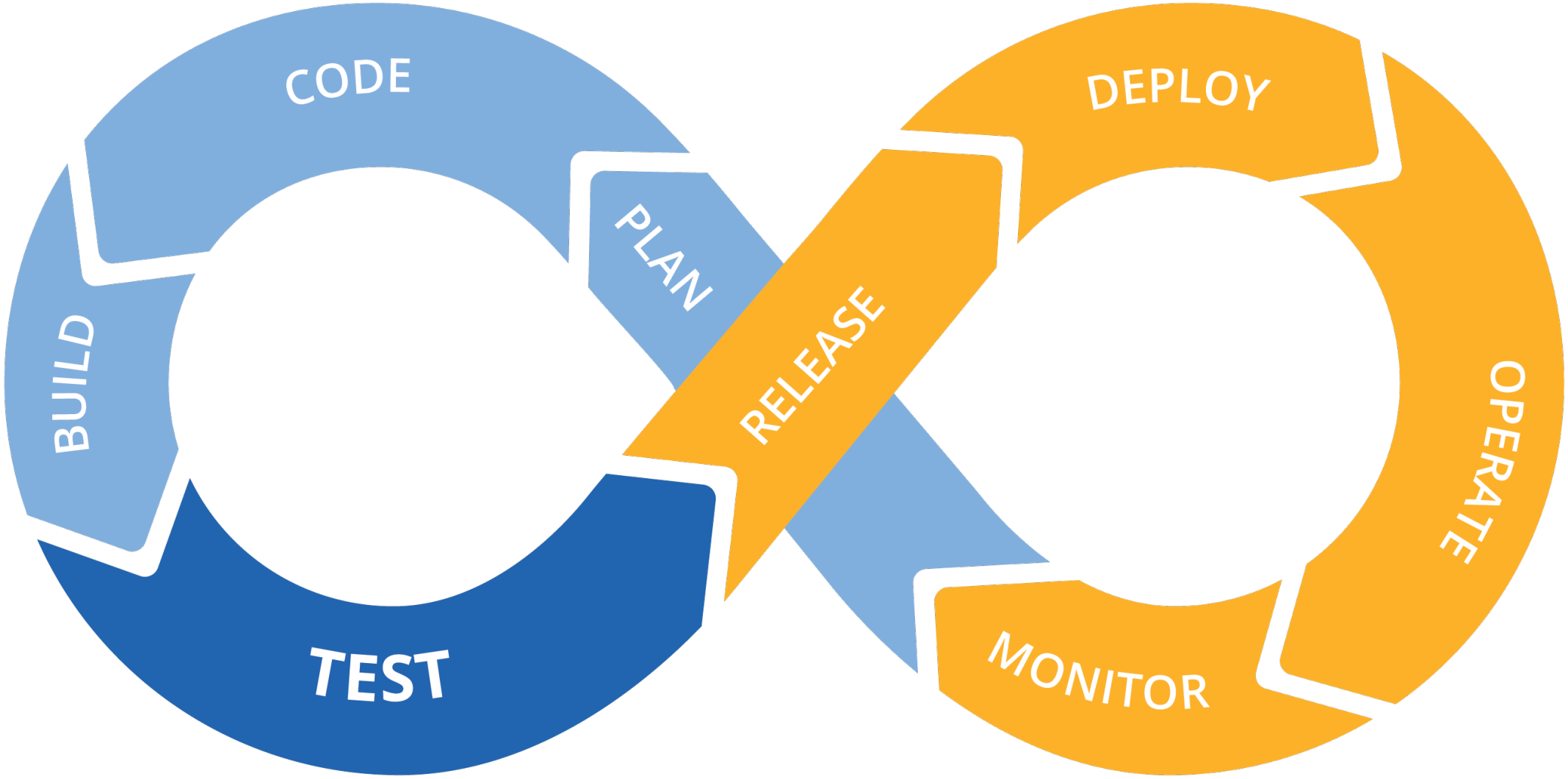
- Technical
  - automation,
  - monitoring
  - measurement

Tampereen yliopisto
Tampere University

Where was the beef?



DevOps



Tampereen yliopisto
Tampere University

DevOps - benefits and challenges

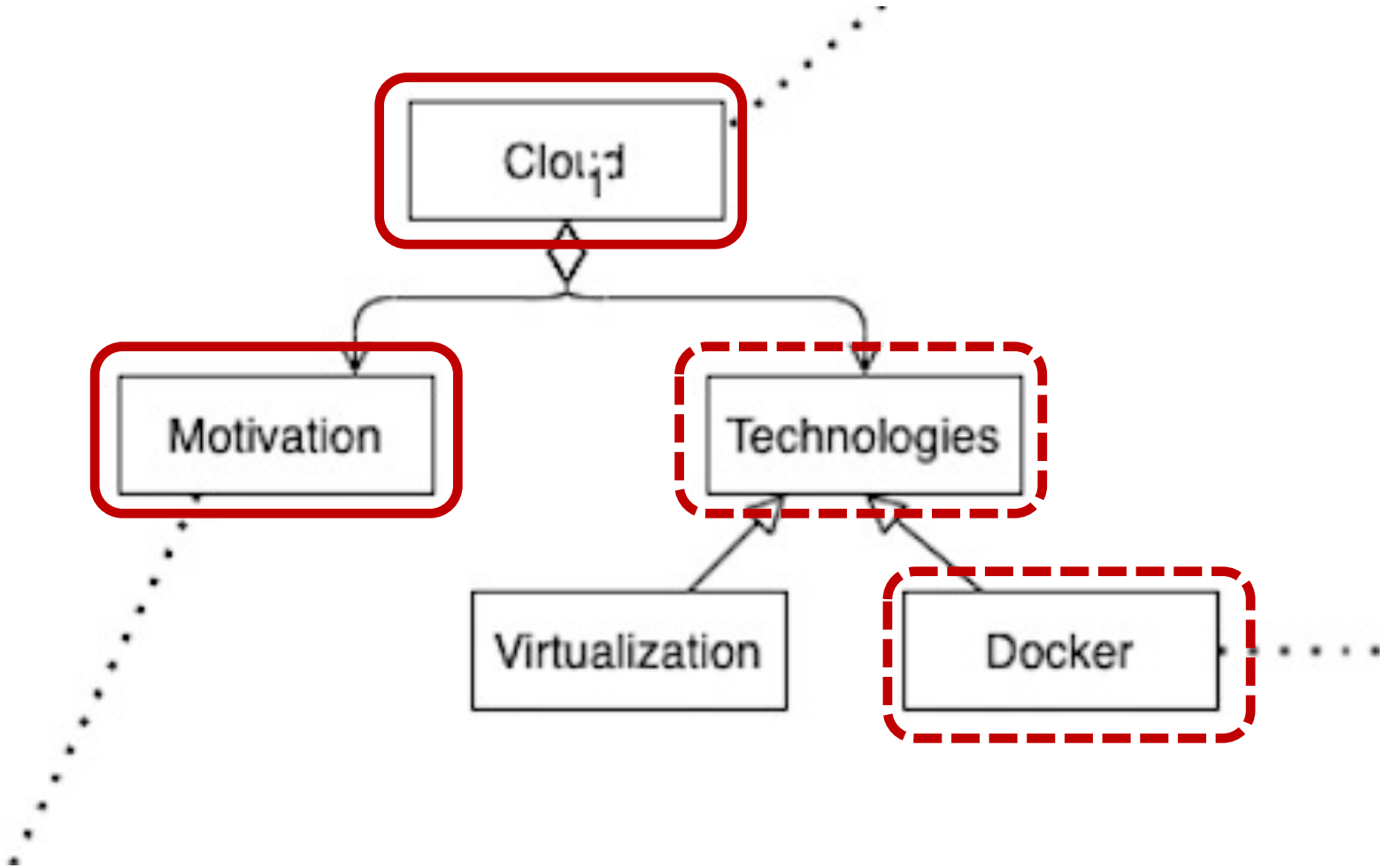
- **improvement in speed (release cycle time)**
- **continuous deployment of system changes**
- **productivity of operations work**
- **improved morale, knowledge and skills**

- **resource constraints;**
- **insufficiencies in infrastructure management;**
- **high demands for required skills and knowledge, and**
- **difficulties in monitoring microservices**

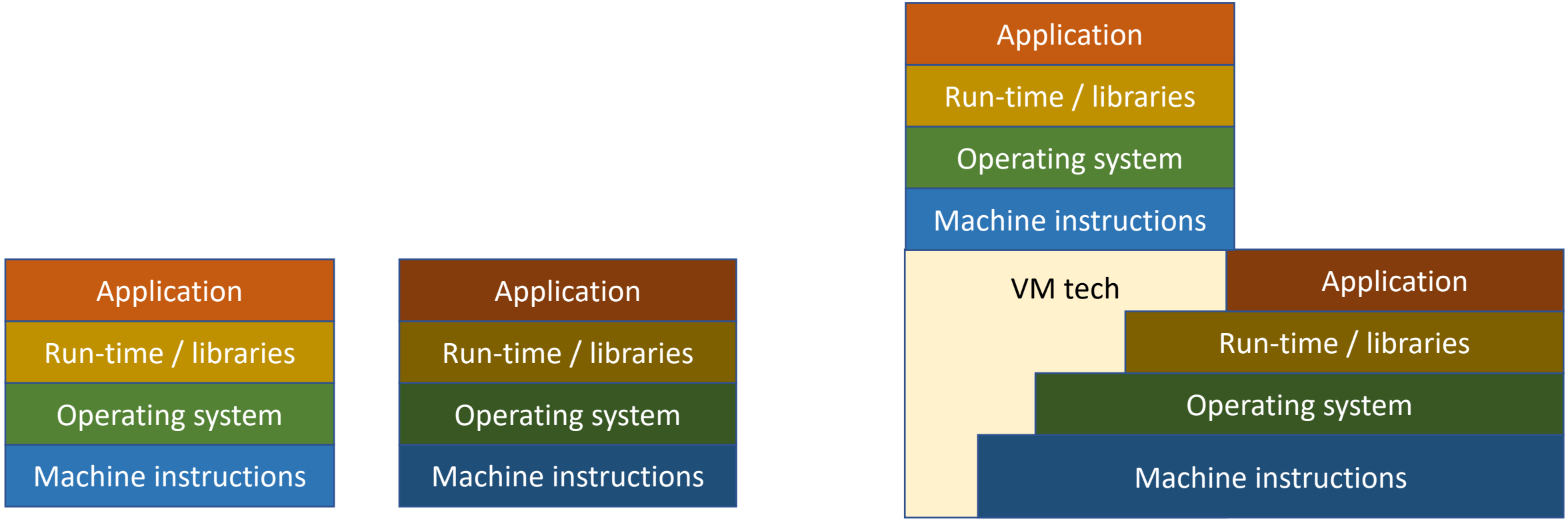
Tampereen yliopisto
Tampere University

Reading material for the exam

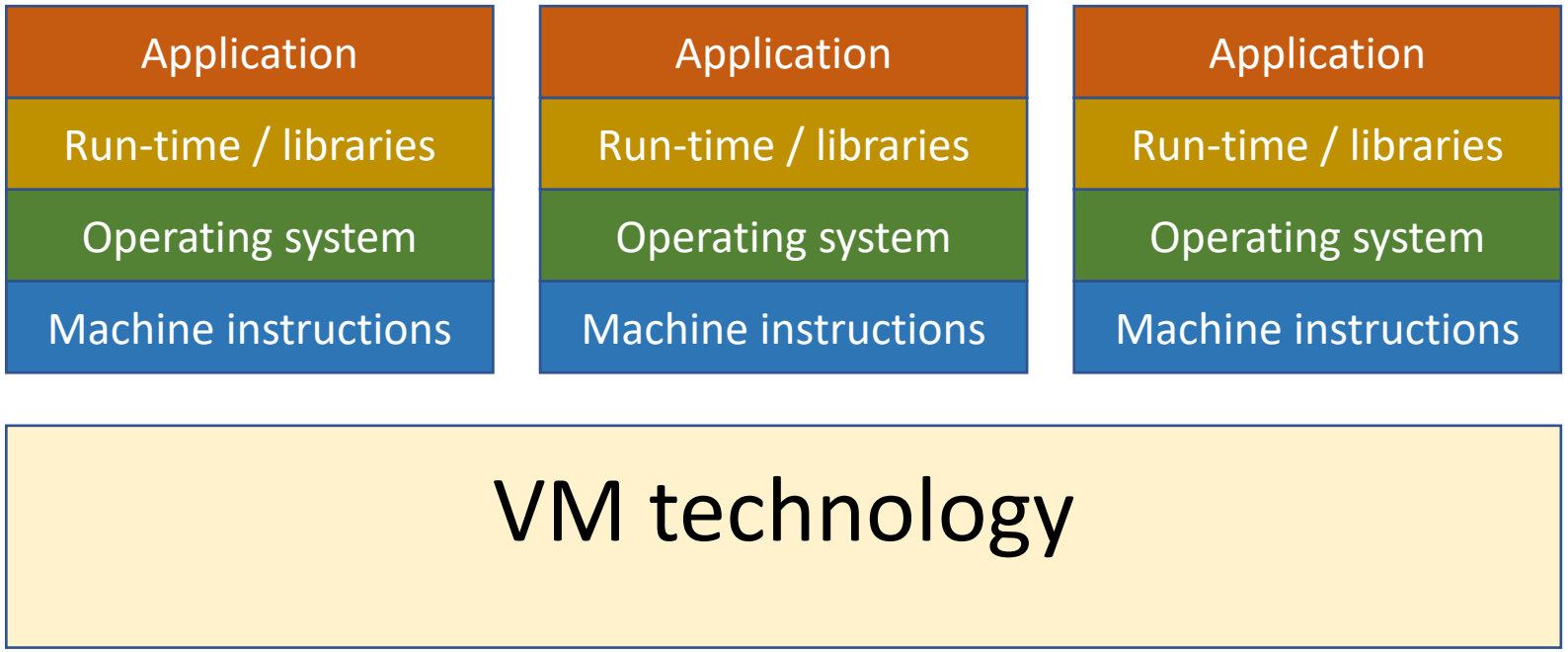
- Lwakatare, Lucy Ellen, Doctoral Dissertation, University of Oulu, 2017, DevOps adoption and implementation in software development practice : concept, practices, benefits and challenges, <http://jultika.oulu.fi/files/isbn9789526217116.pdf>
  - Chapter 2


Cloud
Motivation
Technologies
Virtualization
Docker

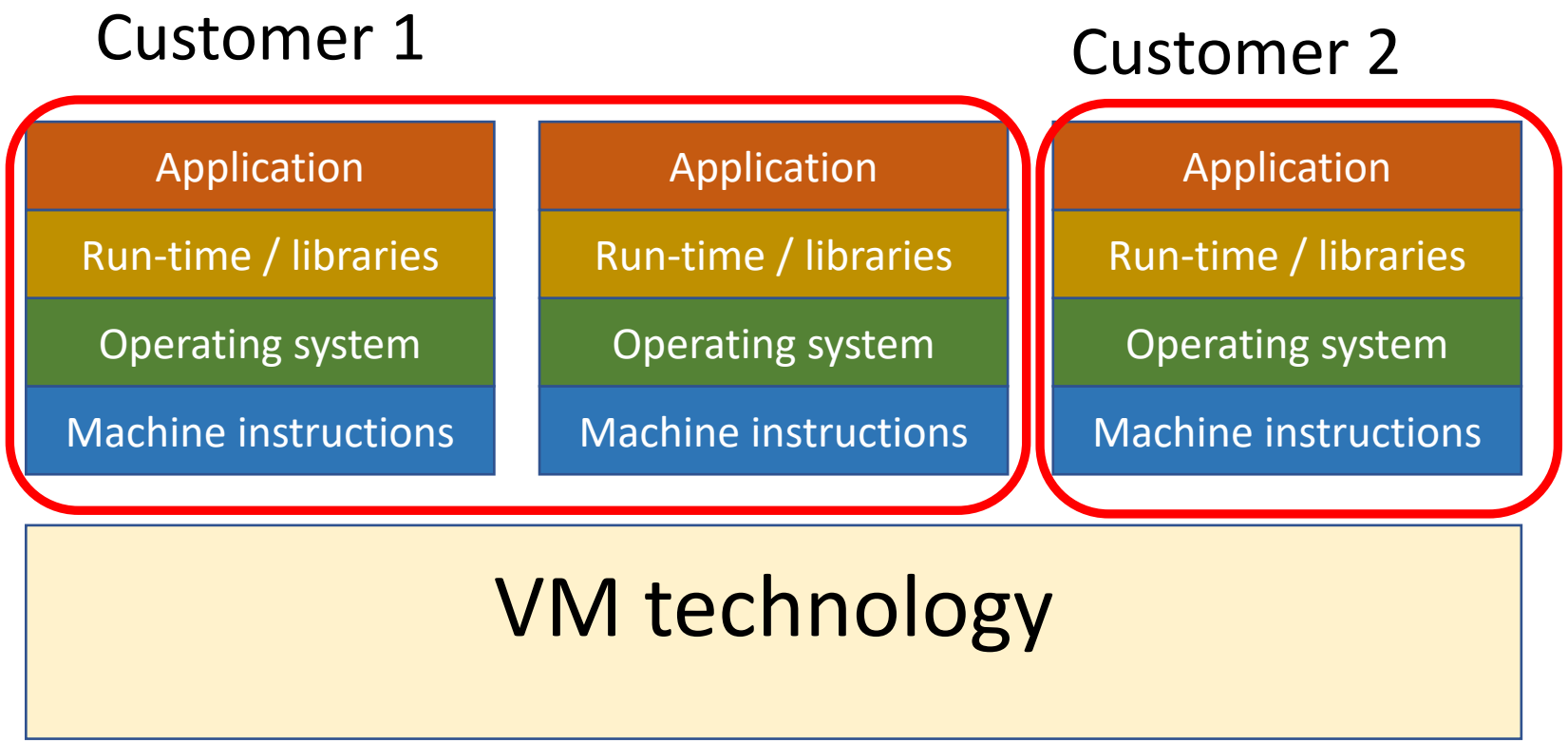
Use case 1: run ”foreign” software



Use case 2: isolate



Use case 3: scale



Tampereen yliopisto
Tampere University

Cloud computing - definition

- In 1997, Professor Ramnath Chellapa of Emory University defined Cloud Computing as the new *'computing paradigm, where the boundaries of computing will be determined by economic rationale, rather than technical limits alone.'*
- *NIST:* Cloud computing is a model for enabling ubiquitous, convenient, **on-demand network access to a shared pool of configurable computing resources** (e.g., networks, servers, storage, applications, and services) that can be **rapidly provisioned and released with minimal management effort** or service provider interaction.
-

Tampereen yliopisto
Tampere University

Essential characteristics 1/2

- *On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
  - Unilaterally?
  - Without human interaction?
- *Broad network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
  - What does the heterogeneous platforms mean in practice?
- *Rapid elasticity.* Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
  - What of scaling is not automatic?

Tampereen yliopisto
Tampere University

Essential characteristics 2/2

- *Resource pooling.* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.
  - Why is this essential?
- *Measured service.* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.
  - Why?

Tampereen yliopisto
Tampere University

Service models

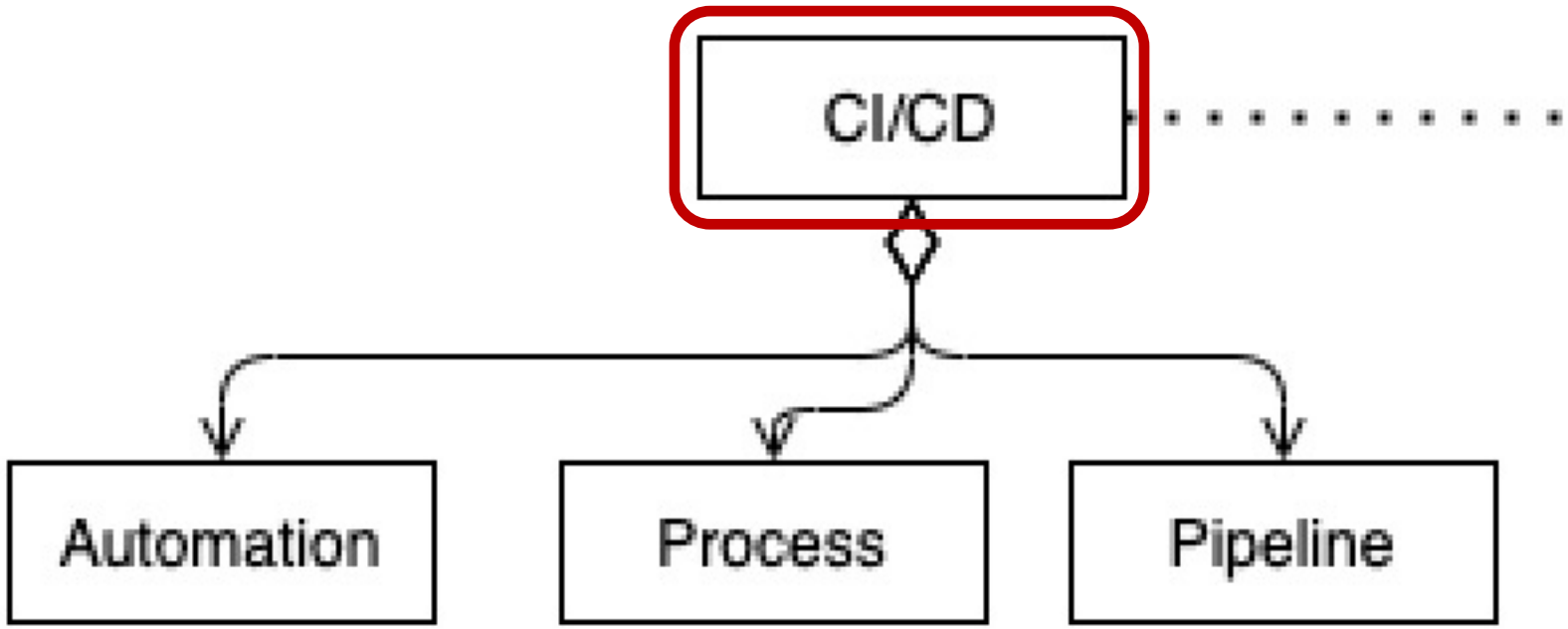
Pets are given names like grumpycat.petstore.com They are unique, lovingly hand raised and cared for When they get ill, you nurse them back to health	Cattle are given numbers like 10200713.cattlerancher.com They are almost identical to other cattle When they get ill, you replace them and get another
Infrastructure is a permanent fixture in the data center	Infrastructure is stateless, ephemeral, and transient
Infrastructure takes days to create, are serviced weekly, maintained for years, and requires migration projects to move	Infrastructure is instantiated, modified, destroyed and recreated in minutes from scratch using automated scripts
Infrastructure is modified during maintenance hours and generally requires special privileges such as root access	Infrastructure uses version-controlled scripts to modify any service without requiring root access or privileged logins
Infrastructure requires several different teams to coordinate and provision the full environment	Infrastructure is self-service with the ability to provision computing, network and storage services with a single click
Infrastructure is static, requiring excess capacity to be dormant for use during peak periods of demands	Infrastructure is elastic and scales automatically, expanding and contracting on-demand to service peak usage periods
Infrastructure is an capital expenditure that charges a fix amount regardless of usage patterns	Infrastructure is a operating expenditure that charges only for services when they are consumed

Tampereen yliopisto
Tampere University

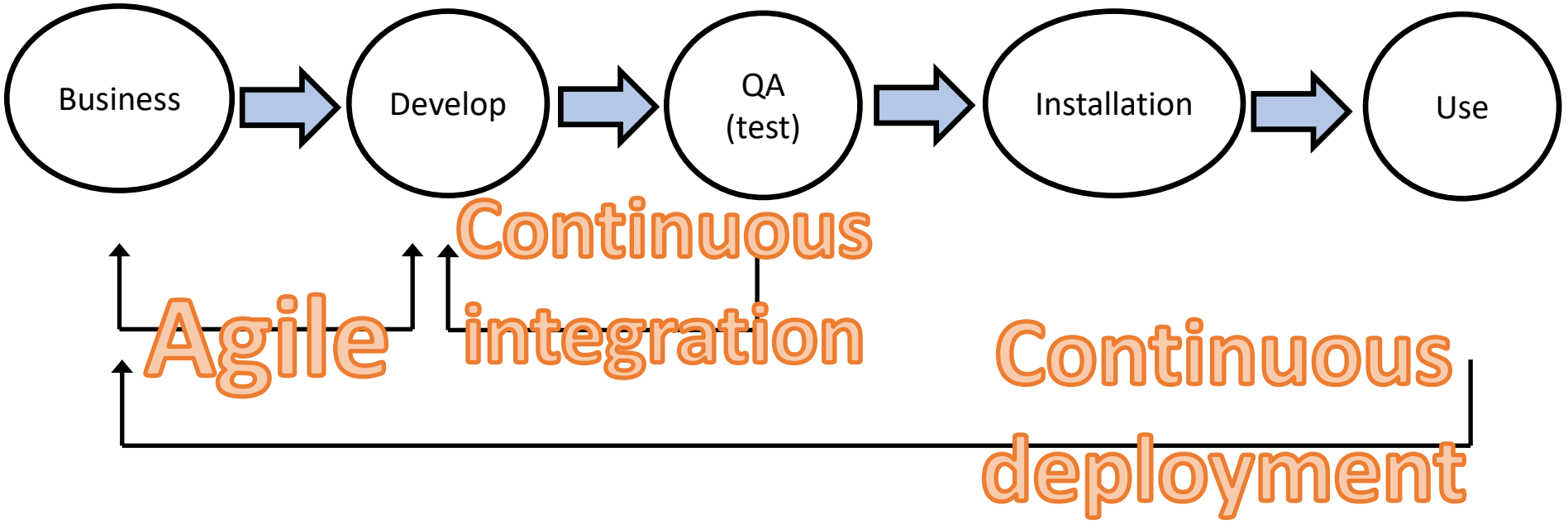
Material for exam

- Peter Mell; Timothy Grance (September 2011). The NIST Definition of Cloud Computing (Technical report). National Institute of Standards and Technology: U.S. Department of Commerce. doi:10.6028/NIST.SP.800-145. Special publication 800-145. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

- Keith D. Foote, A Brief History of Cloud Computing, June 2017, https://www.dataversity.net/brief-history-cloud-computing

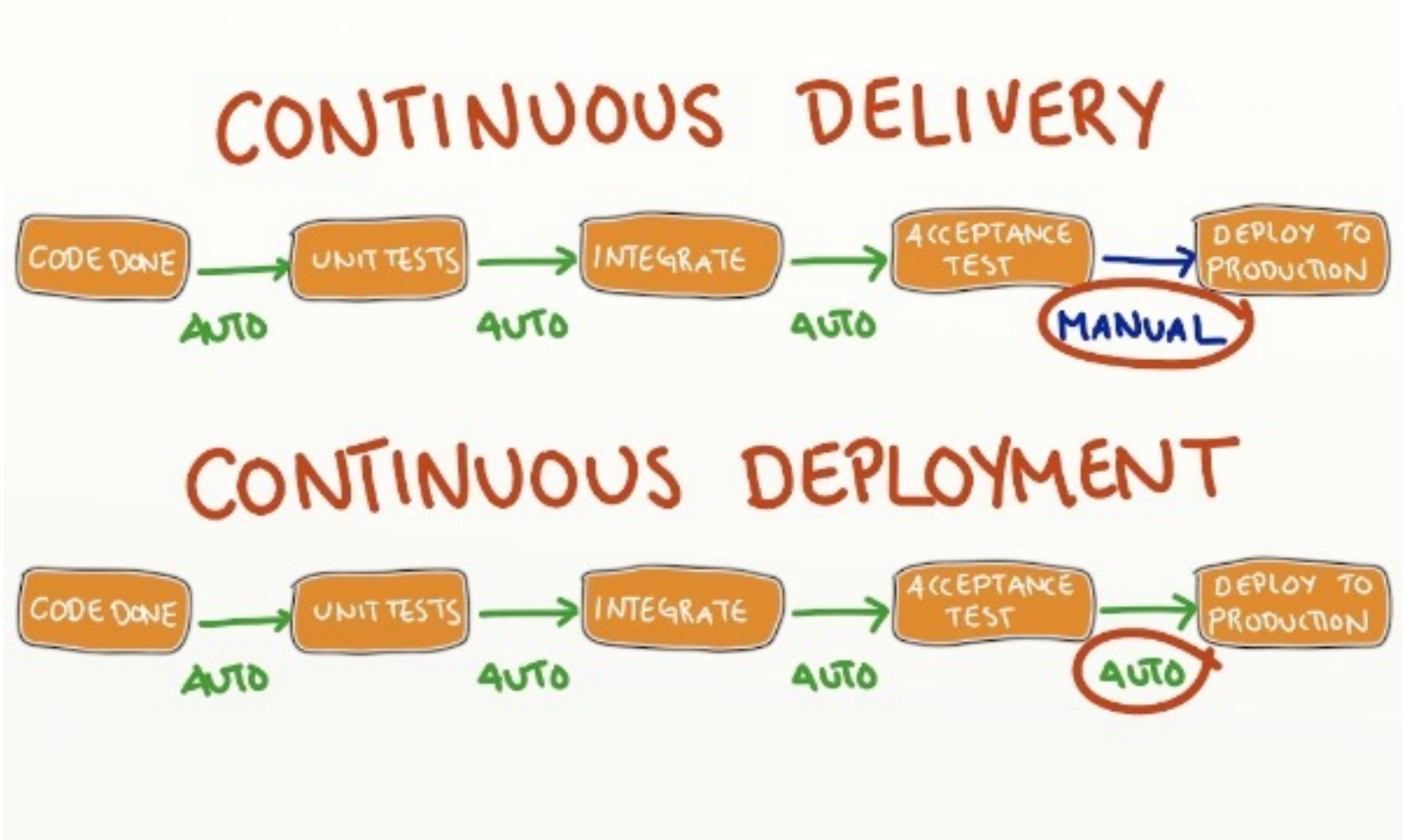

CI/CD
Automation
Process
Pipeline

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

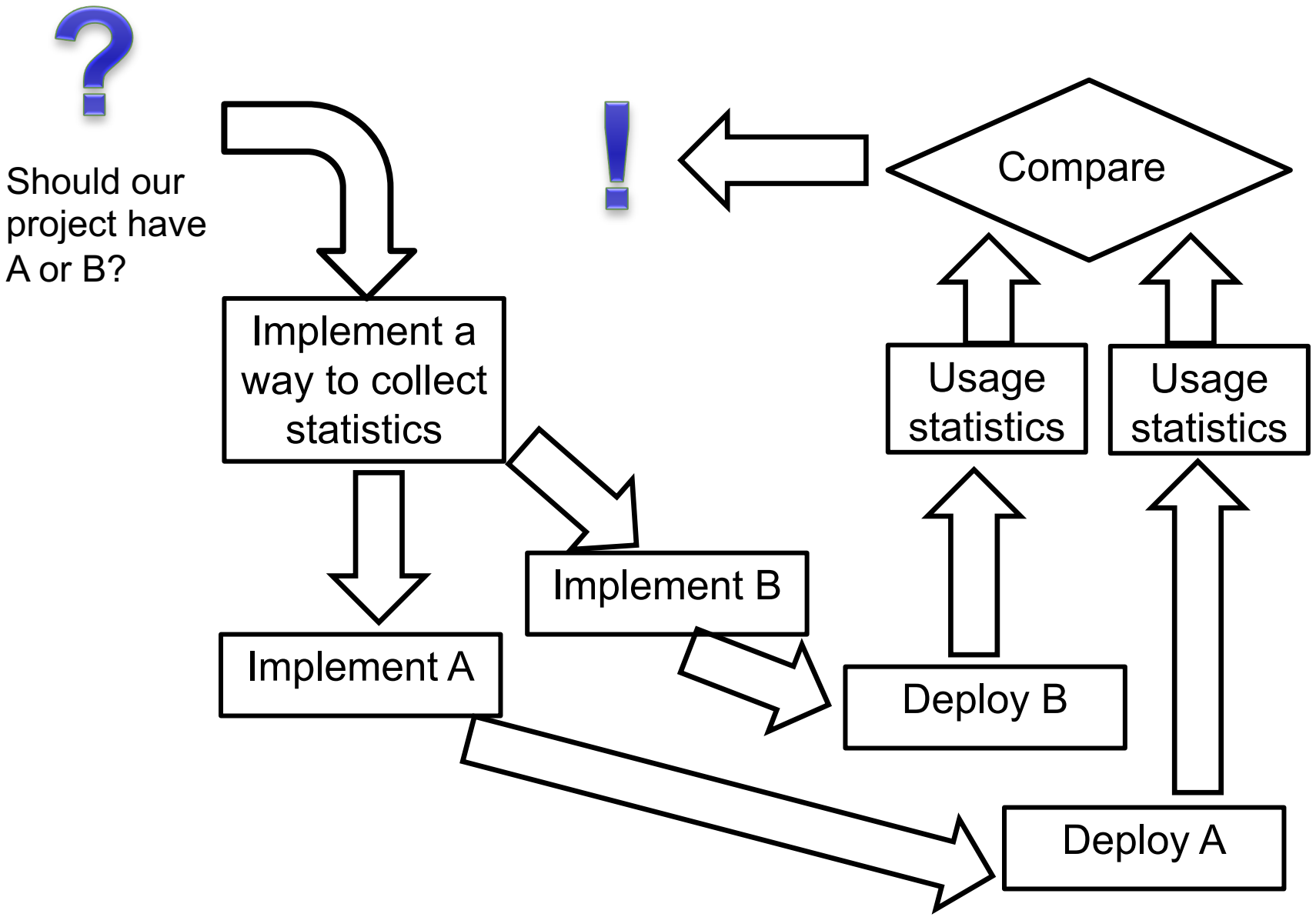


Continuous delivery and deployment

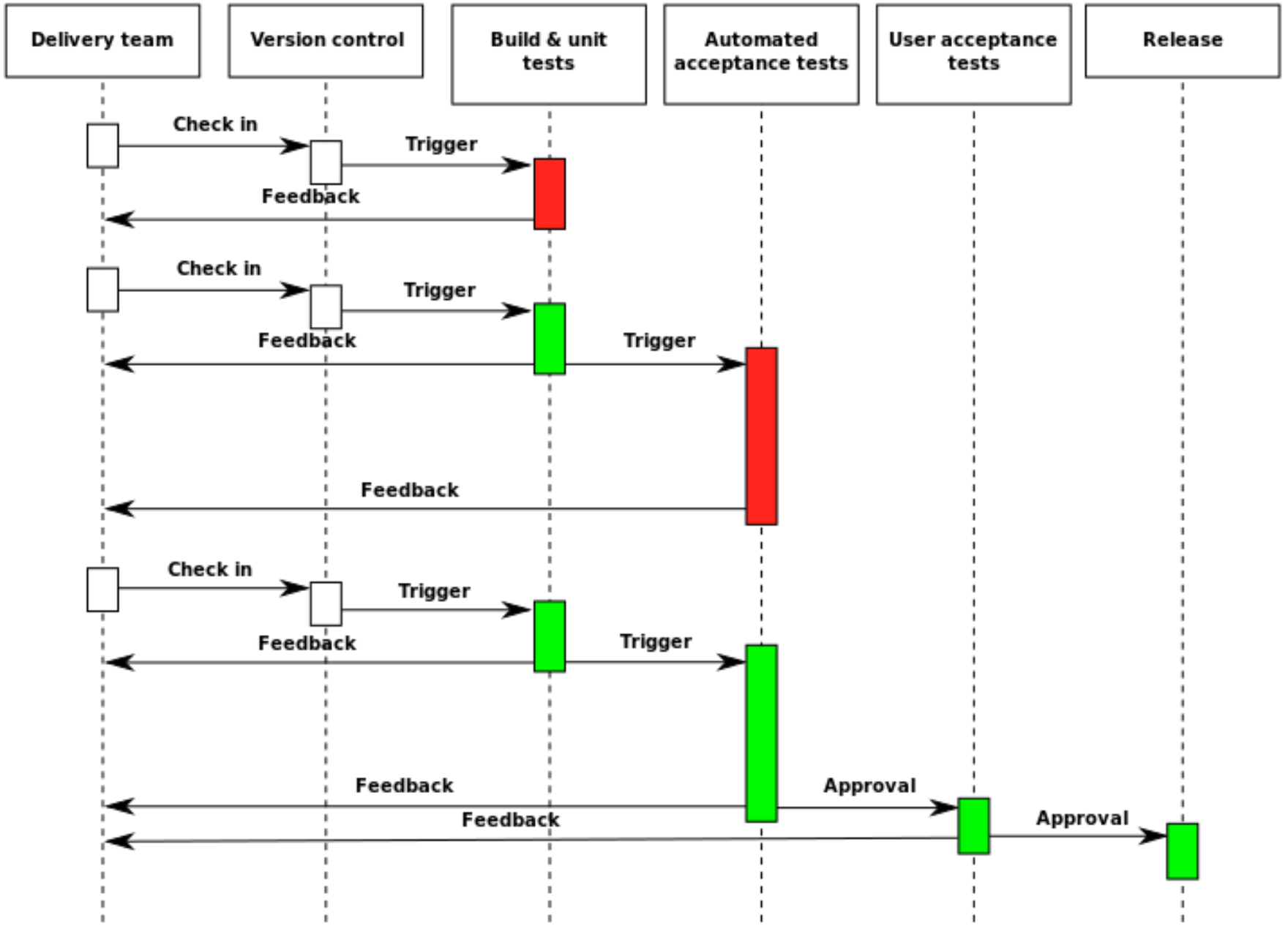
(http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment)



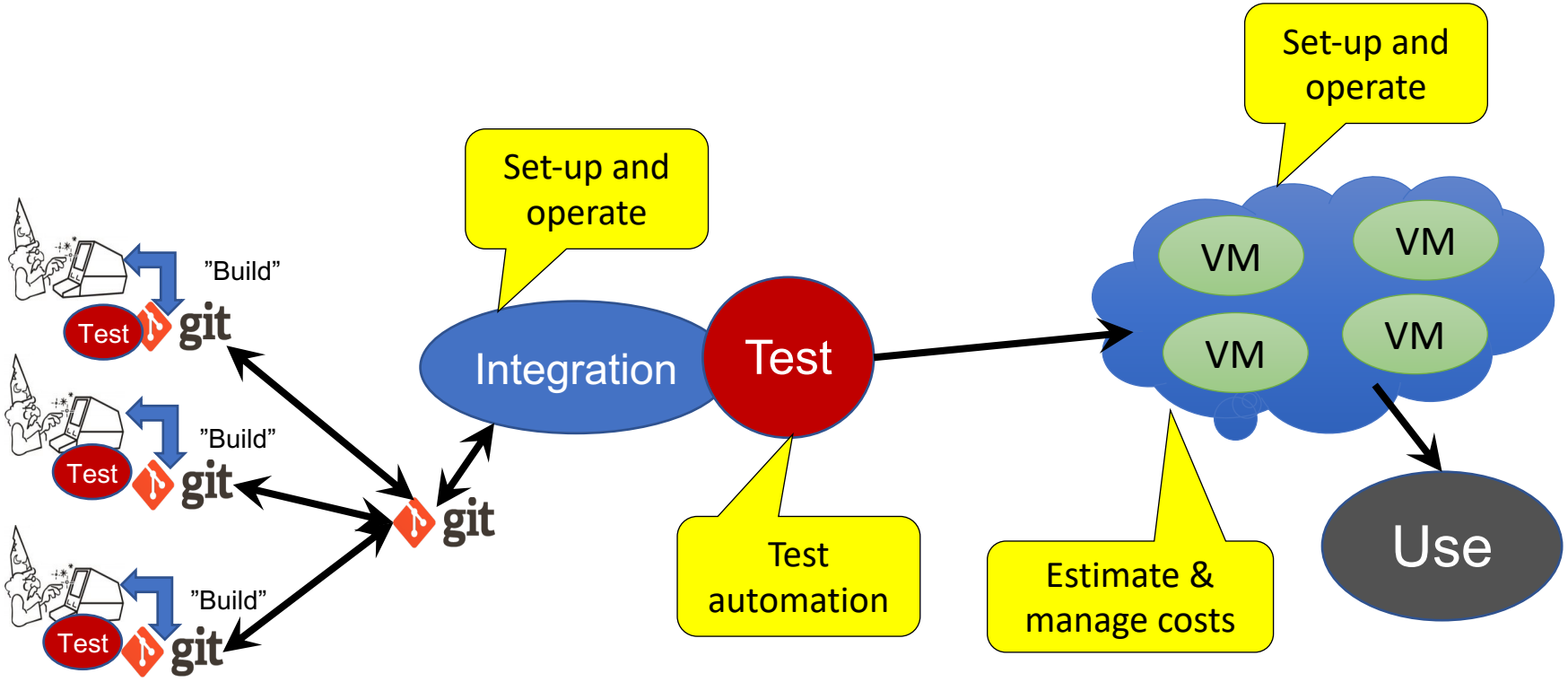
A/B Testing


Should our project have A or B?
Implement a way to collect statistics
Implement A
Implement B
Deploy B
Deploy A
Usage statistics
Usage statistics
Compare

Deployment pipeline (a possible example)



What does it really take to run CD?



Tampereen yliopisto
Tampere University

CI – essential practices
(according to Humbley and Farley)

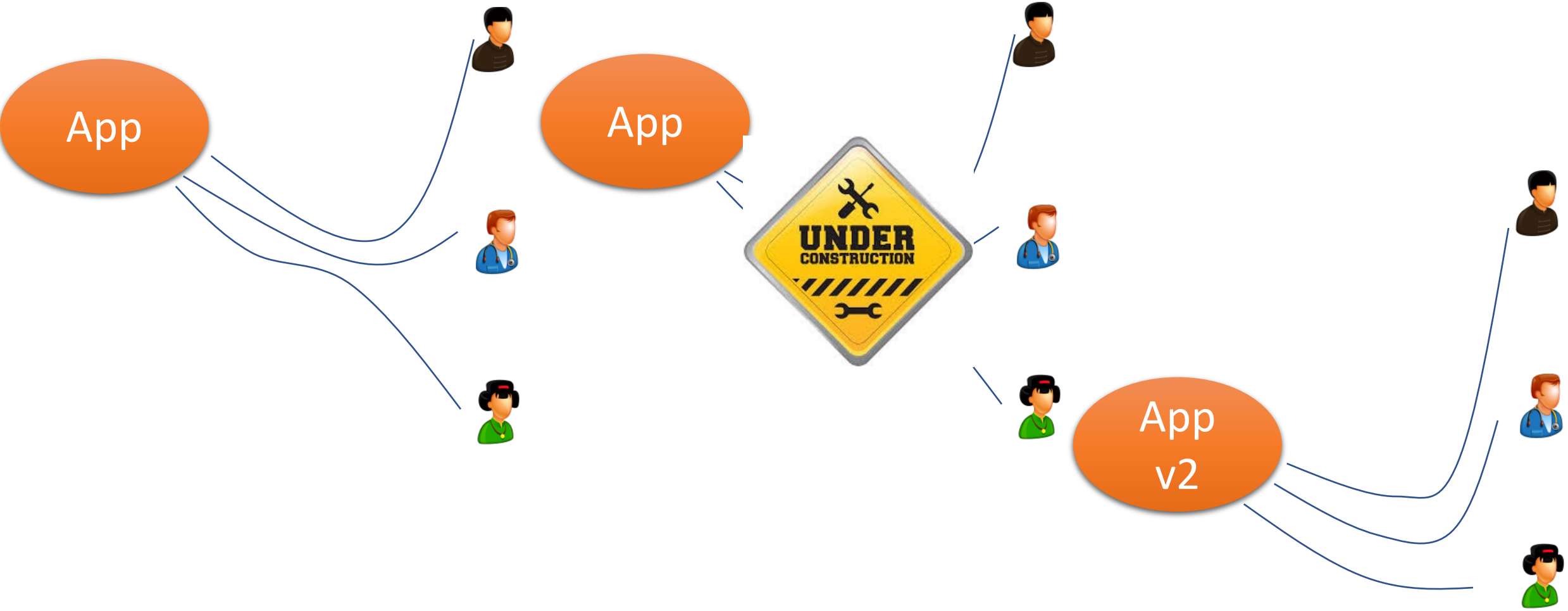
- Don’t check in on a broken code
- Always run all commits tests locally before committing, or get your CI server to do it for you
- Wait for commit tests to pass before moving on
- Never go home on a broken build
- Always be prepared to revert to the previous revisions
- Time-box fixing before reverting
- 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

A possible strategy to deploy a new version?



Tampereen yliopisto
Tampere University

Deployment strategies

- Basic Deployment (aka Suicide)
- Rolling Deployment
- BlueGreen Deployment
- Canary Releases

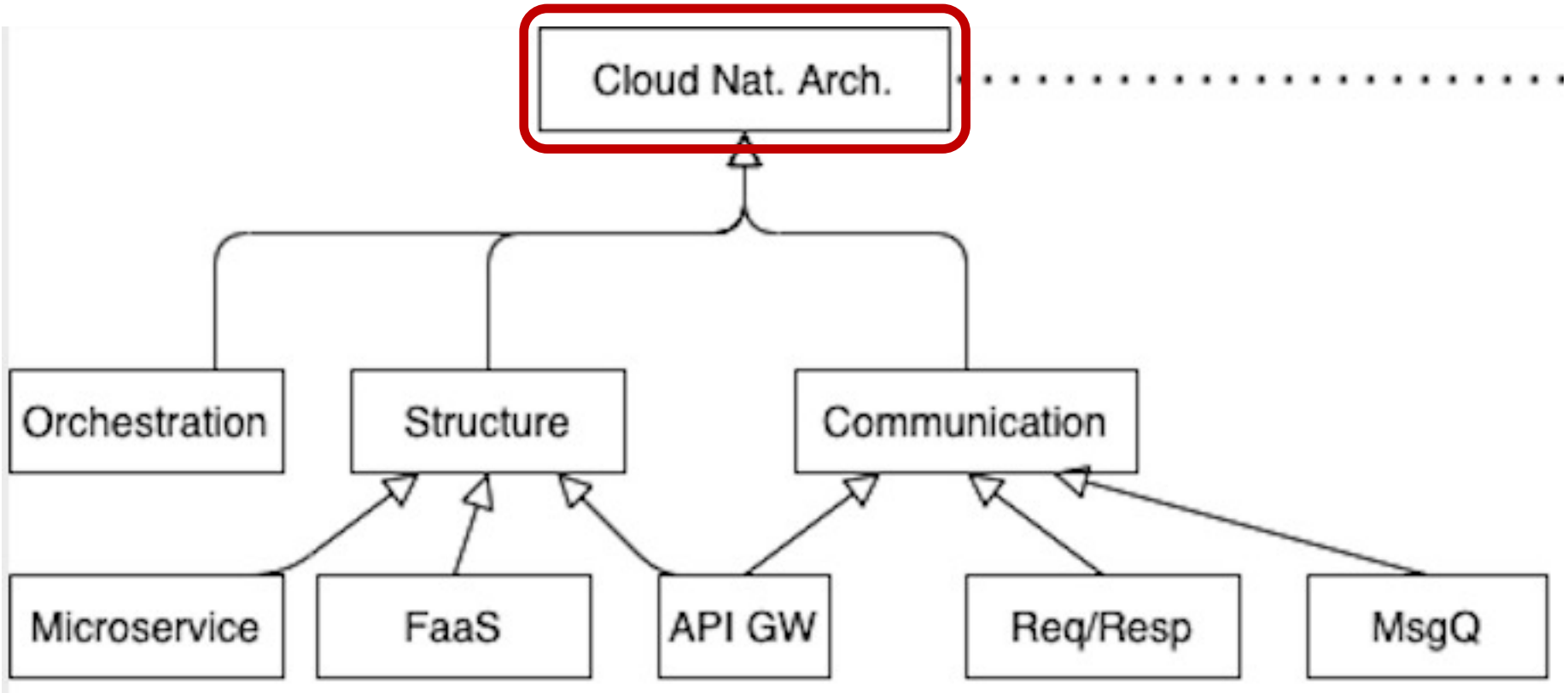
- Note: corrected link: https://harness.io/blog/blue-green-canary-deployment-strategies/

Tampereen yliopisto
Tampere University

Reading material for exam

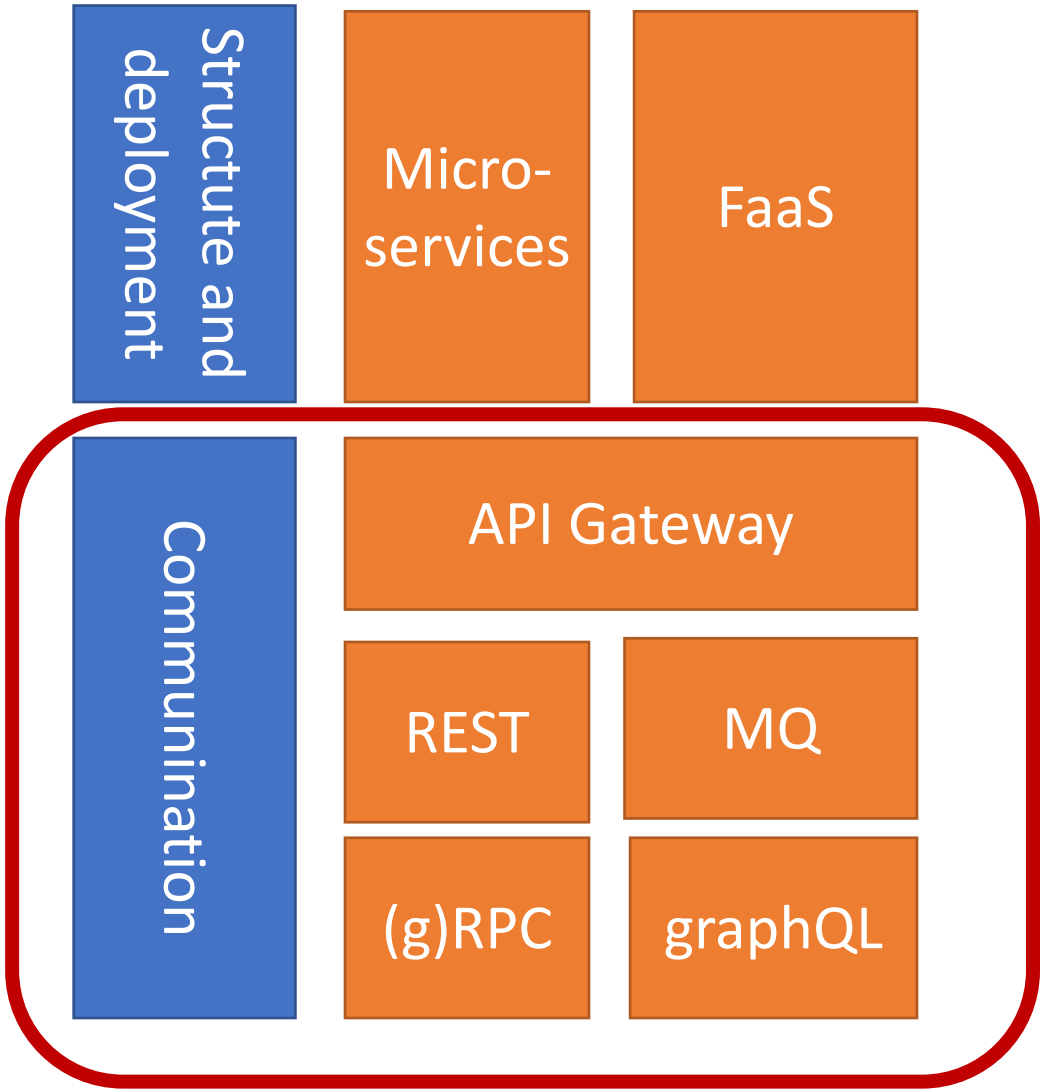
- https://continuousdelivery.com

- M. Leppänen *et al*., "The highways and country roads to continuous deployment," in *IEEE Software*, vol. 32, no. 2, pp. 64-72, Mar.-Apr. 2015, doi: 10.1109/MS.2015.50, https://ieeexplore.ieee.org/document/7057604


Cloud Nat. Arch.
Orchestration
Structure
Communication
Microservice
FaaS
API GW
Req/Resp
MsgQ

Tampereen yliopisto
Tampere University

More about cloud-native architectures



Tampereen yliopisto
Tampere University

s ces

Can be used in many ways

Designed for intependent

Standard ways to document
Designed for intependent

Practically none on top of Network infra

Practically none on top of Network infra

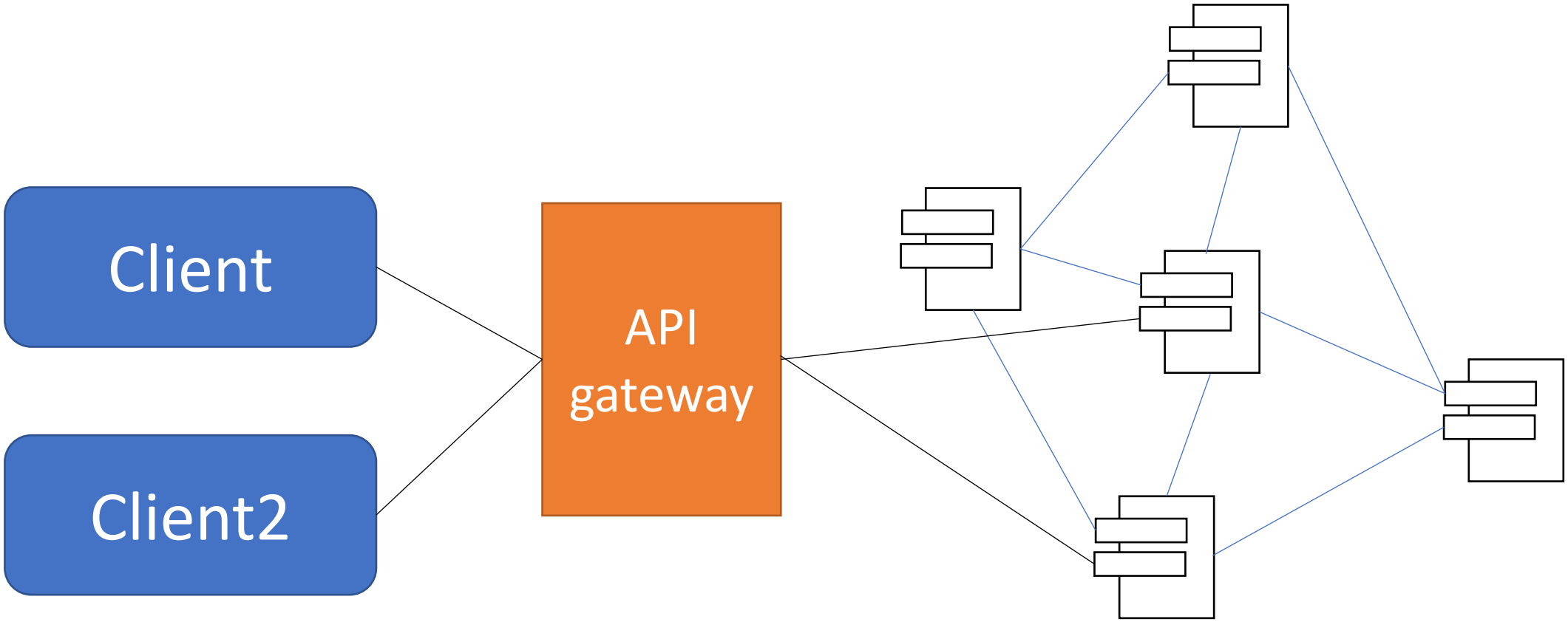
	Independent development	Independen deployment	Minimum ce manageme
REST			
gRPC			
Message queue			

No standards: need to be agreed on

The queue even suppors interrupts

The message queue need to be maintained

How about external calls?



Tampereen yliopisto
Tampere University

RECALL Interface segregation principle

“many client-specific interfaces are better than one general-purpose interface.”

“Make fine grained interfaces that are client specific”

“Clients should not be forced to depend upon methods they do not use”

- Big system with many dependencies = small change causes changed everywhere
- Large interfaces are split to smaller and role-base interfaces.

  ⇒changes do not affect everybody

  ⇒New features are easier to add

  ⇒Interfaces are easier to learn

Tampereen yliopisto
Tampere University

Recap

David S. Linthicum, Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points Between, IEEE Cloud Computing, December 2017

- **Performance**. You'll typically be able to access provide better performance than is possible with nonnative features. For example, you can deal with an input/output (I/O) system that works with autoscaling and loadbalancing features.
- **Efficiency**. Cloud-native applications' use of cloud-native features and application programming interfaces (APIs) should provide more efficient use of underlying resources. That translates to better performance and/or lower operating costs.
- **Cost**. Applications that are more efficient typically cost less to run. Cloud providers send you a monthly bill based upon the amount of resources consumed, so if you can do more with less, you save on dollars spent.
- **Scalability**. Because you write the applications to the native cloud interfaces, you have direct access to the autoscaling and load-balancing features of the cloud platform.

https://martinfowler.com/articles/microservices.html

the microservice architectural style is an approach to developing a single application as a suite of small services each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

I. Nadareishvili et al., Microservice Architecture: Aligning Principles, Practices, and Culture, O'Reilly, 2016.

- small
- messaging enabled,
- bounded by contexts,
- autonomously developed,
- independently deployable,
- decentralized, and
- built and released with automated processes.

Tampereen yliopisto
Tampere University

Some links

- 10 Key Attributes of Cloud-native Applications, <https://thenewstack.io/10-key-attributes-of-cloud-native-applications/>
- What are cloud-native applications? <https://opensource.com/article/18/7/what-are-cloud-native-apps>
- Native cloud application (NCA), <https://searchitoperations.techtarget.com/definition/native-cloud-application-NCA>
- Understanding cloud-native applications, <https://www.redhat.com/en/topics/cloud-native-apps>
- David S. Linthicum, Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points Between, IEEE Cloud Computing, December 2017.

Tampereen yliopisto
Tampere University

Some links

- 10 Key Attributes of Cloud-native Applications, <https://thenewstack.io/10-key-attributes-of-cloud-native-applications/>
- What are cloud native applications?

1. Packaged as lightweight containers
2. Developed with best-of-breed languages and frameworks
3. Designed as loosely coupled microservices
4. Centered around APIs for interaction and collaboration
5. Architected with a clean separation of stateless and stateful services
6. Isolated from server and operating system dependencies
7. Deployed on self-service, elastic, cloud infrastructure
8. Managed through agile DevOps processes
9. Automated capabilities
10. Defined, policy-driven resource allocation

pps>

ud-

n: The
ecember

Do you really want to keep
your containers running all the time
if you need to pay for it?

Do you really want to operate
and maintain your containers –
your developers could also
do something else.

Tampereen yliopisto
Tampere University

Serverless computing

Baldini et all: Serverless Computing:

Current Trends and Open Problems, Research Advances in Cloud Computing, Springer, 2017.

A cloud-native platform

for

- short-running, stateless computation
- event driven applications

which

- scale up and down instantly and automatically

and

- charge for actual usage and high granulatity

Tampereen yliopisto
Tampere University

Stateful vs stateless computation

- If a service has an internal state it is difficult to
  - Scale it
  - Move it to other server or other hosting system

  => Stateless Services are subject to cloud-specific optimizations
- The internal state my be
  - volatile or
  - non-volatile
  - ... in memory, file local to container,

- Serverless / FaaS

Tampereen yliopisto
Tampere University

7R's of cloud Micration

Replace
with imilar or improved but SaaS

Reuse
in the new SaaS version

Refactor
towards cloud-native architecture

Replatform
by using cloud services

Rehost
to a VM

Retain

Retire

Tampereen yliopisto
Tampere University

What it means to be Cloud Native approach — the CNCF way

https://medium.com/developingnodes/what-it-means-to-be-cloud-native-approach-the-cncf-way-9e8ab99d4923

1. Containerization

- **Docker container** image is a lightweight, standalone, executable package of software that includes everything needed to run an application.

2. CI/CD

3. Orchestration

- **Kubernetes** is the market-leading orchestration solution.

4. Observability & Analysis

- Monitoring, logging, and tracing

5. Service MESH

6. Networking and Policy

- Flexibility with authorization, admission control and data filtering

7. Distributed Database

- When you need more resiliency and scalability than you can get from a single database

8. Messaging

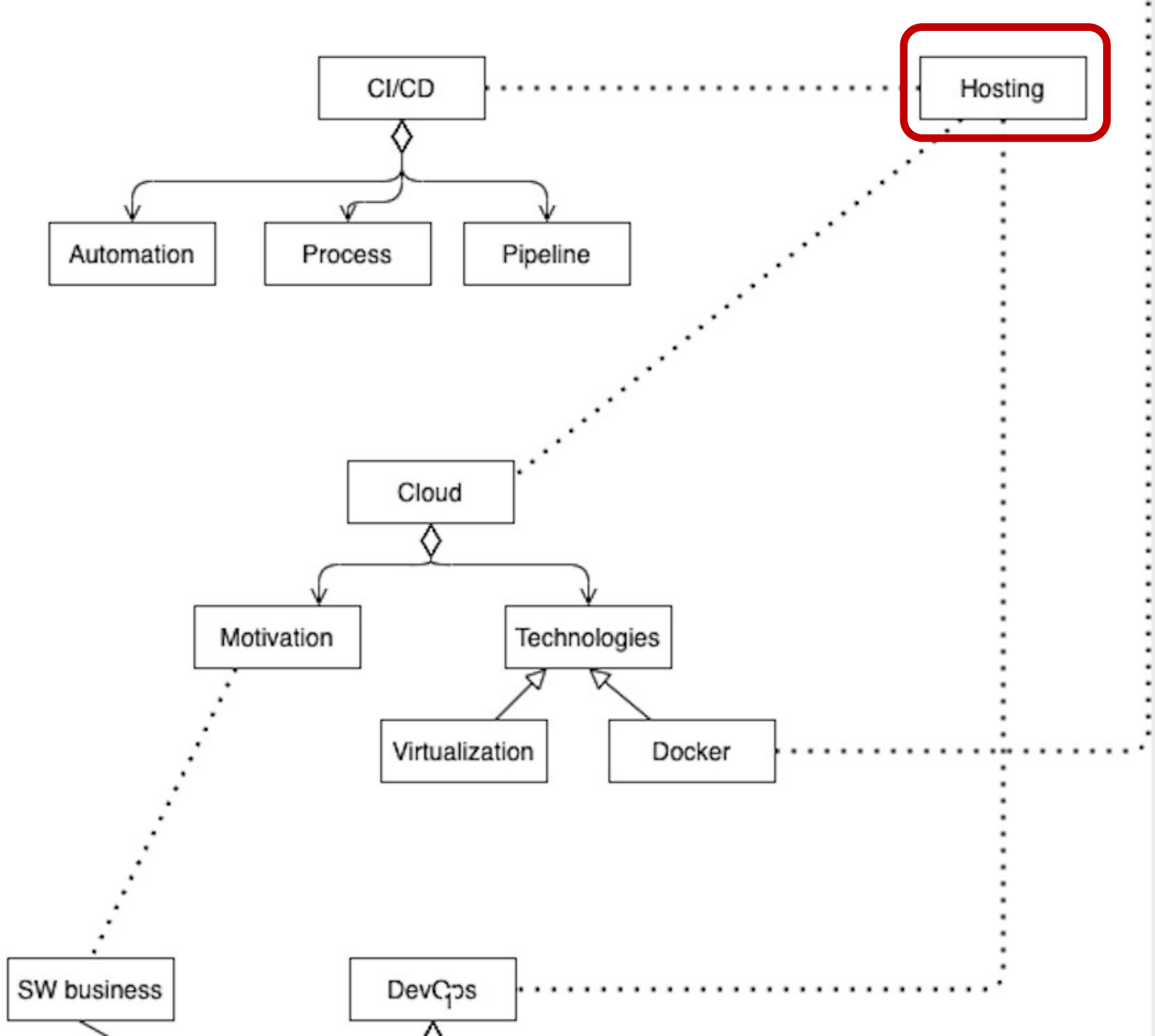
9. Container registry and runtimes

10. Sofware distribution

Tampereen yliopisto
Tampere University

Reading material

- D. S. Linthicum, "Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points Between," in *IEEE Cloud Computing*, vol. 4, no. 5, pp. 12-14, September/October 2017, doi: 10.1109/MCC.2017.4250932. https://ieeexplore.ieee.org/document/8125545/
- N. C. Mendonça, C. Box, C. Manolache and L. Ryan, "The Monolith Strikes Back: Why Istio Migrated From Microservices to a Monolithic Architecture," in *IEEE Software*, vol. 38, no. 5, pp. 17-22, Sept.-Oct. 2021, doi: 10.1109/MS.2021.3080335. https://ieeexplore.ieee.org/document/9520758
- Baldini et all: Serverless Computing: Current Trends and Open Problems, Research Advanges in Cloud Computing, Springer, 2017. https://arxiv.org/pdf/1706.03178.pdf


CI/CD
Hosting
Automation
Process
Pipeline
Cloud
Motivation
Technologies
Virtualization
Docker
SW business
DevOps

Tampereen yliopisto
Tampere University

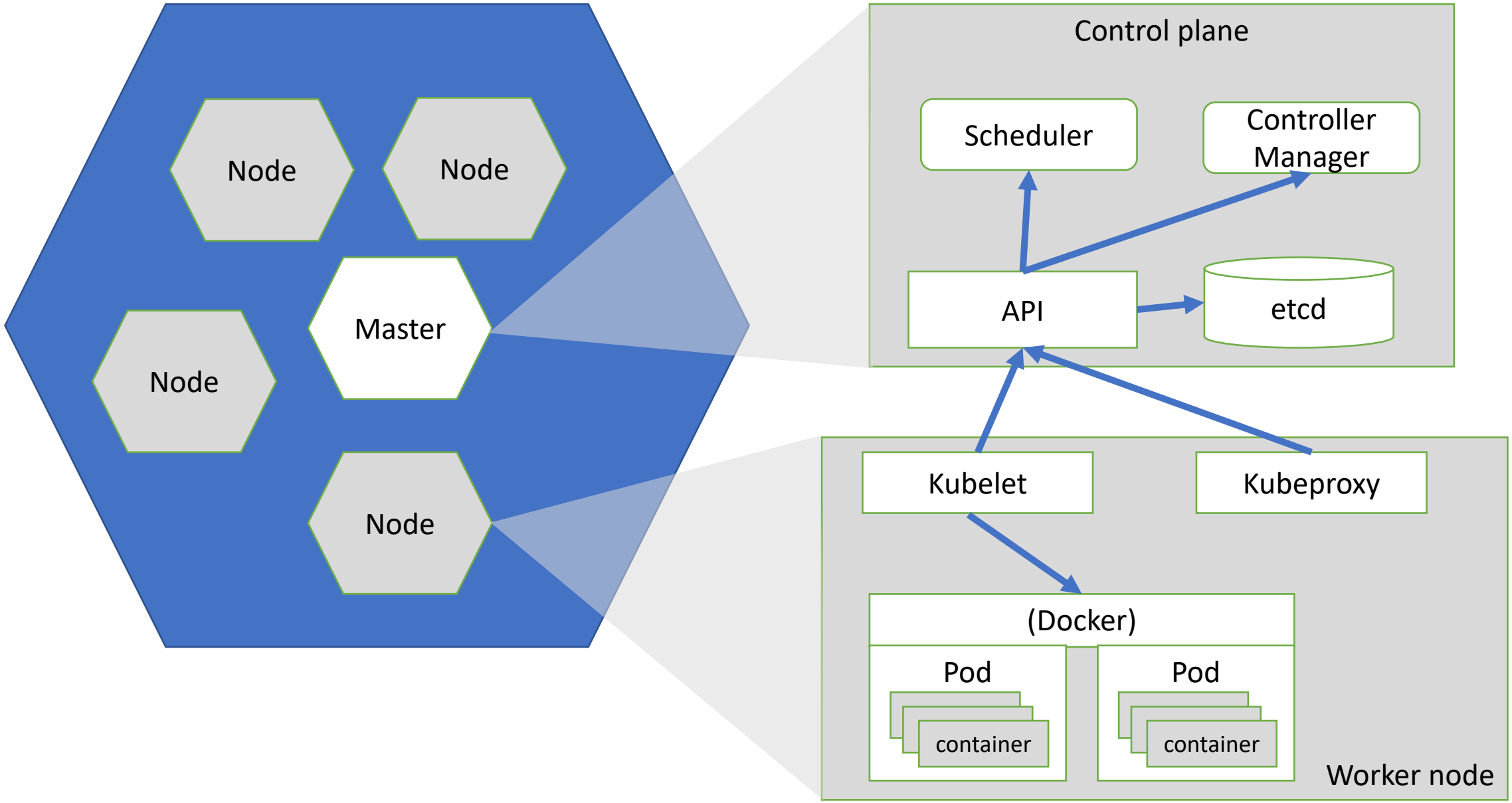
Infrastructure as code

From: https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code

Infrastructure as Code (IaC) is

- the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model,
- using the same versioning as DevOps team uses for source code.
- Like the principle that the same source code generates the same binary, an IaC model generates the same environment every time it is applied.
- IaC is a key DevOps practice and is used in conjunction with continuous delivery.

Kubernetes Kluster



Tampereen yliopisto
Tampere University

Example from https://aws.amazon.com/ec2/pricing/ (as of 13.09.2021)

- Free tier
  - AWS Free Tier includes 750 hours of Linux and Windows t2.micro instances, (t3.micro for the regions in which t2.micro is unavailable) each month for one year. To stay within the Free Tier, use only EC2 Micro instances.
- On-Demand
  - With On-Demand instances, you pay for compute capacity by the hour or the second depending on which instances you run.
- Spot instances
  - Amazon EC2 Spot instances allow you to request spare Amazon EC2 computing capacity for up to 90% off the On-Demand price.
- Reserved Instances
  - provide you with a significant discount (up to 72%) compared to On-Demand Instance pricing. In addition, when Reserved Instances are assigned to a specific Availability Zone, they provide a capacity reservation, giving you additional confidence in your ability to launch instances when you need them.

Name	vCPUs	Memory (GiB)	Baseline Performance/v CPU	CPU Credits earned/hr	Network burst bandwidth (Gbps)	EBS burst bandwidth (Mbps)	On-Demand Price/hr*	1-yr Reserved Instance Effective Hourly*	3-yr Reserved Instance Effective Hourly*
t3.nano	2	0.5	5%	6	5	Up to 2,085	$0.0052	$0.003	$0.002
t3.micro	2	1.0	10%	12	5	Up to 2,085	$0.0104	$0.006	$0.005
t3.small	2	2.0	20%	24	5	Up to 2,085	$0.0209	$0.012	$0.008
t3.medium	**2**	**4.0**	**20%**	**24**	**5**	**Up to 2,085**	**$0.0418**	**$0.025**	**$0.017**
t3.large	2	8.0	30%	36	5	Up to 2,780	$0.0835	$0.05	$0.036
t3.xlarge	4	16.0	40%	96	5	Up to 2,780	$0.1670	$0.099	$0.067
t3.2xlarge	8	32.0	40%	192	5	Up to 2,780	$0.3341	$0.199	$0.133

Lets calculate a bit

One year plan

- Reserved Instance Price/hr*: 0.025$
- There are 24*365 = 8760 hours / year

⇒219 $ / year

Three year plan

- Reserved Instance : 0.017
- 26295h

=> 447$ / 3 years

On demand

- Price/hr*: 0.0418$
- There are 24*365 = 8760 hours / year

⇒353 $ / year

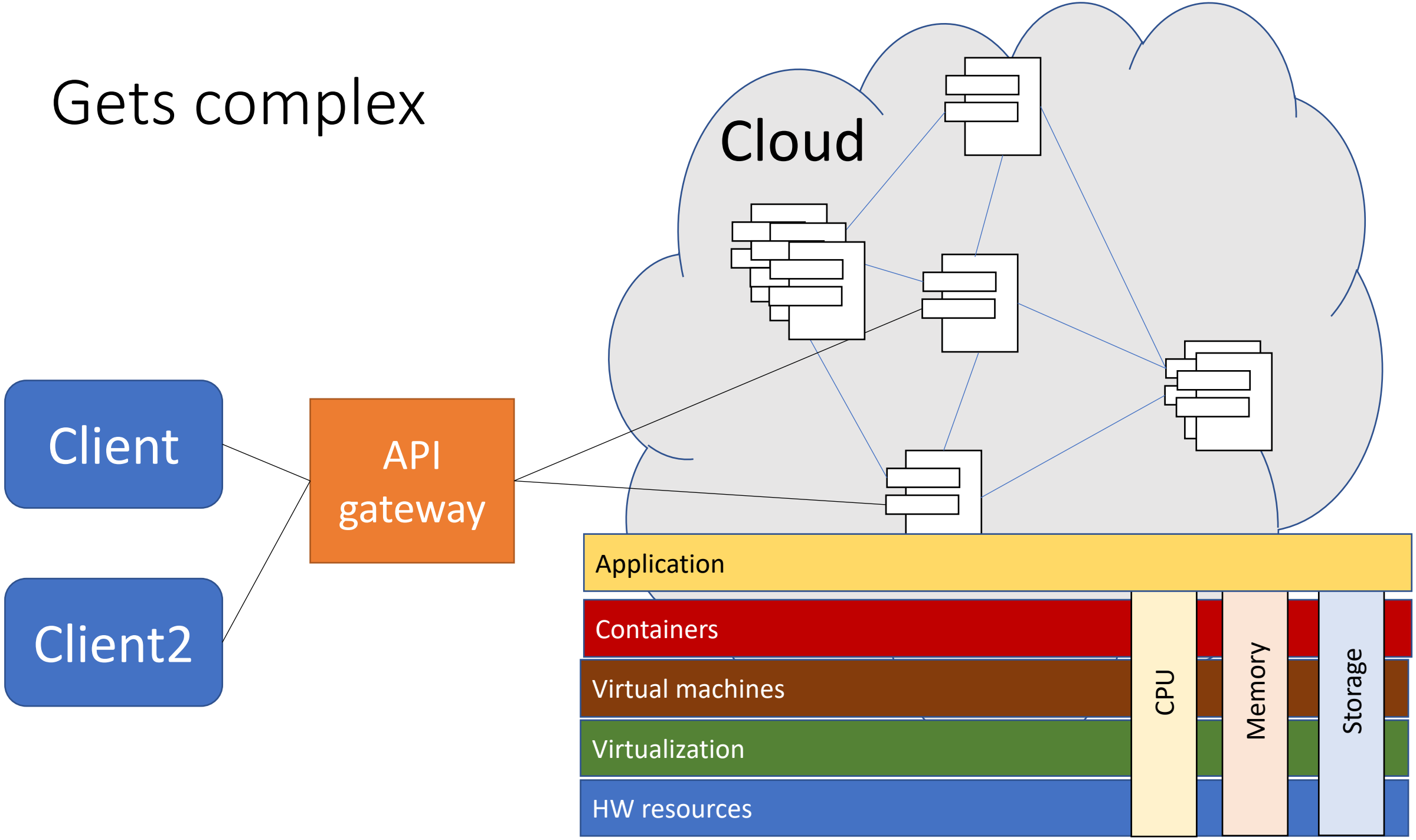
⇒1061$ / 3 years

So

1y: if you use more than 15h/day

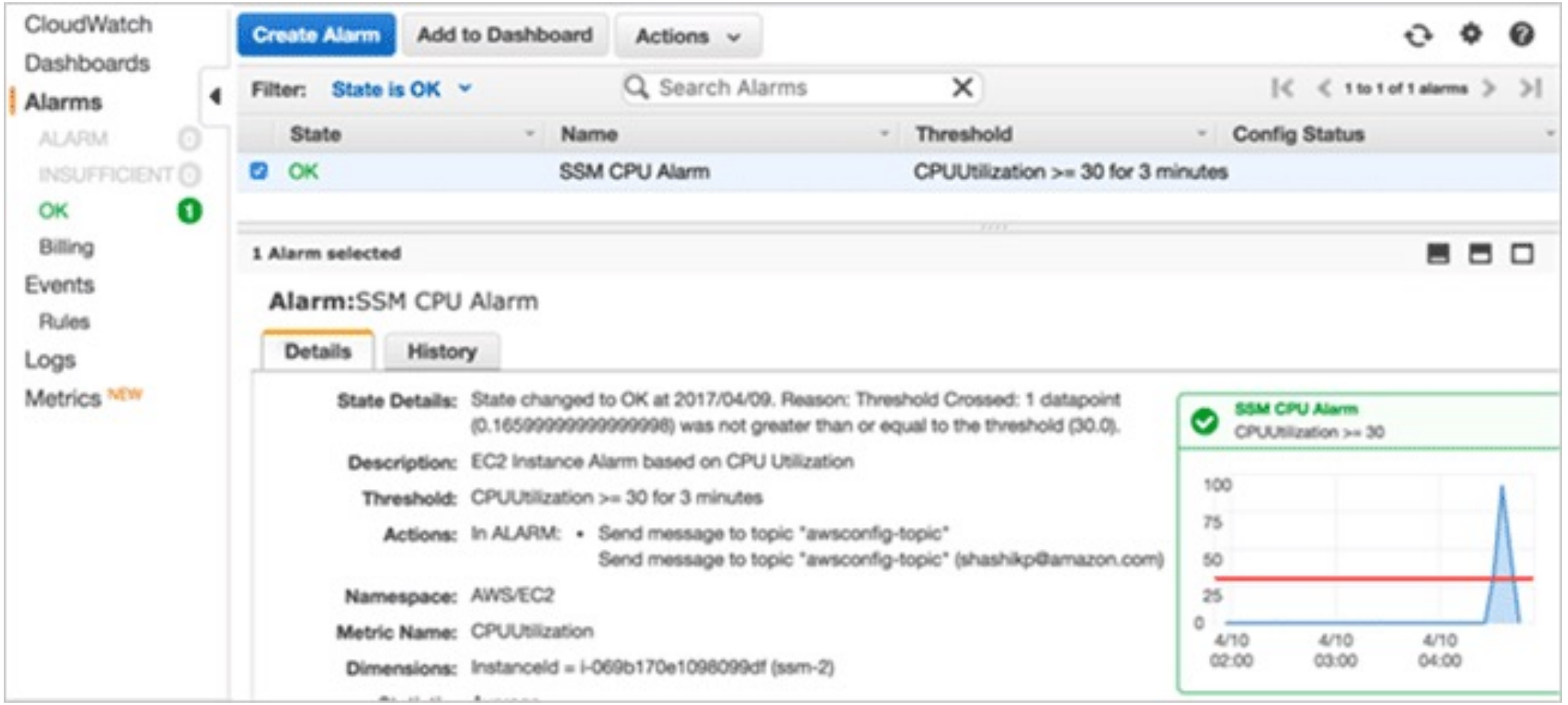
3y: if you use more than 10h/day

Gets complex

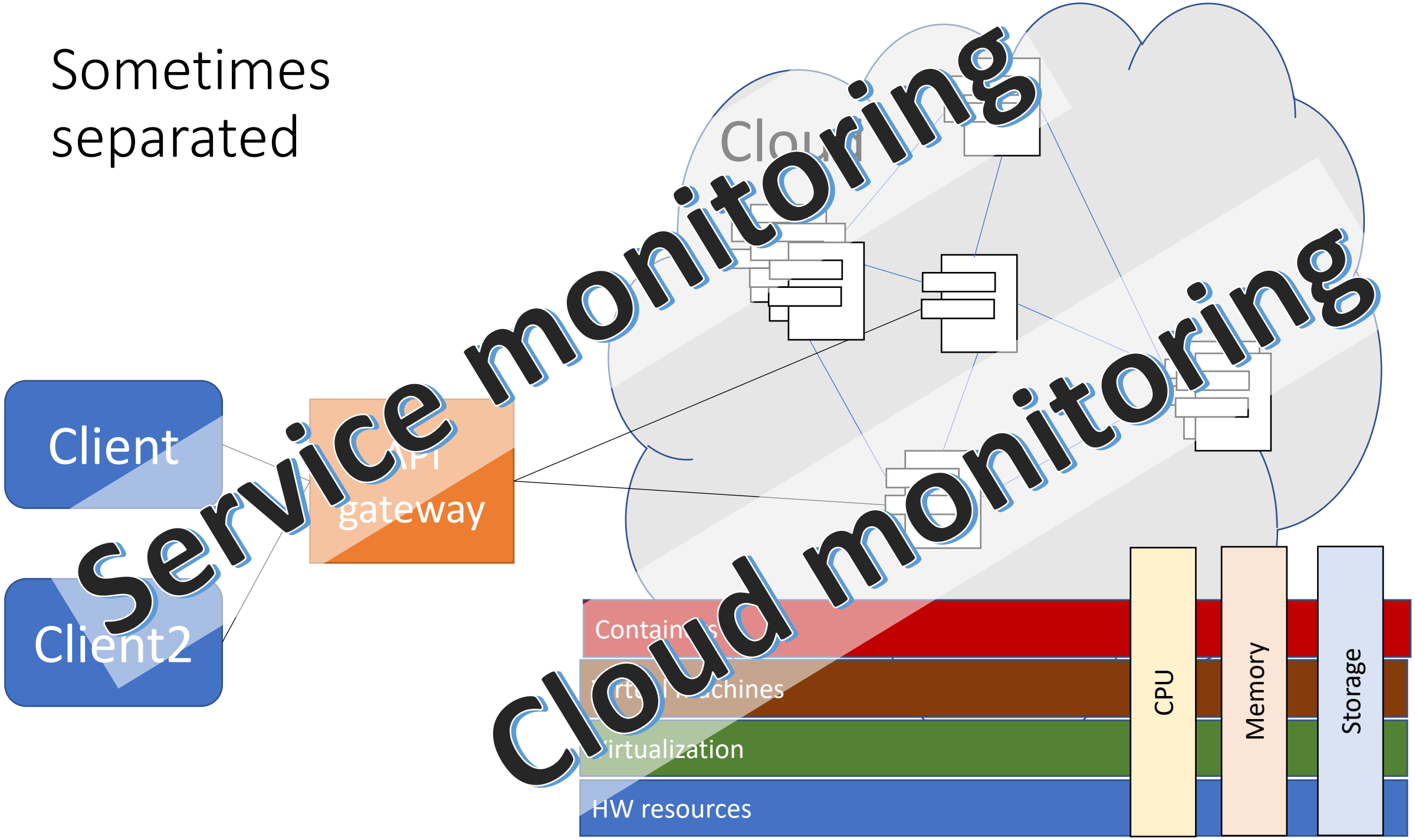


Tampereen yliopisto
Tampere University

Example: Amazon CloudWatch



Sometimes separated



Tampereen yliopisto
Tampere University

How prepare to exam

- Lecture videos & slides can be used as background material
- Read
  - [Chapter 2] Lwakatare, Lucy Ellen, Doctoral Dissertation, University of Oulu, 2017, DevOps adoption and implementation in software development practice : concept, practices, benefits and challenges, http://jultika.oulu.fi/files/isbn9789526217116.pdf [Chapter 2]
  - Peter Mell; Timothy Grance (September 2011). The NIST Definition of Cloud Computing (Technical report). National Institute of Standards and Technology: U.S. Department of Commerce. doi:10.6028/NIST.SP.800-145. Special publication 800-145. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
  - Keith D. Foote, A Brief History of Cloud Computing, June 2017, https://www.dataversity.net/brief-history-cloud-computing
  - https://continuousdelivery.com
  - M. Leppänen et al., "The highways and country roads to continuous deployment," in IEEE Software, vol. 32, no. 2, pp. 64-72, Mar.-Apr. 2015, doi: 10.1109/MS.2015.50, https://ieeexplore.ieee.org/document/7057604
  - D. S. Linthicum, "Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points Between," in IEEE Cloud Computing, vol. 4, no. 5, pp. 12-14, September/October 2017, doi: 10.1109/MCC.2017.4250932. https://ieeexplore.ieee.org/document/8125545/
  - N. C. Mendonça, C. Box, C. Manolache and L. Ryan, "The Monolith Strikes Back: Why Istio Migrated From Microservices to a Monolithic Architecture," in IEEE Software, vol. 38, no. 5, pp. 17-22, Sept.-Oct. 2021, doi: 10.1109/MS.2021.3080335. https://ieeexplore.ieee.org/document/9520758
  - Baldini et all: Serverless Computing: Current Trends and Open Problems, Research Advanges in Cloud Computing, Springer, 2017. https://arxiv.org/pdf/1706.03178.pdf