Tampere University

Lecture 7 – cloud native architectures

Kari Systä, 12.10.2021

Content

- General course topics
- Cloud native architectures
  - Motivations – including examples from our earlier lectures
  - Principles and definitions
  - An example: microservices

- Next week
  - More about microservices
  - Function as a service

Tampere University

Course matters

- News on course staff
  - Petri Kannisto has too many other tasks and leaves the course
  - Instead
    - Petri Rantanen (petri.rantanen@tuni.fi) joins the team
    - Fabiano Pecorelli will also give some help

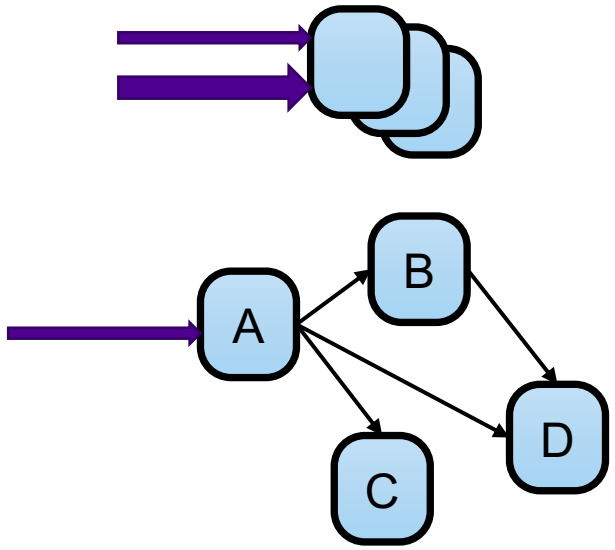
Tampere University

Participants (11.10.2021)

	28.9	5.10	12.10
• Total number of enrolled students:	72		
• Answers to survey:	63	64	
• Current participants:	60	60	
• Docker exercise:	54	59	59
• Docker compose exercise:	17	50	54
• Ansible exercise:		2	2

What are typical cloud applications

- Networks of containers!



Logically like:

```
A() {
    B();
    C();
    D();
}
```

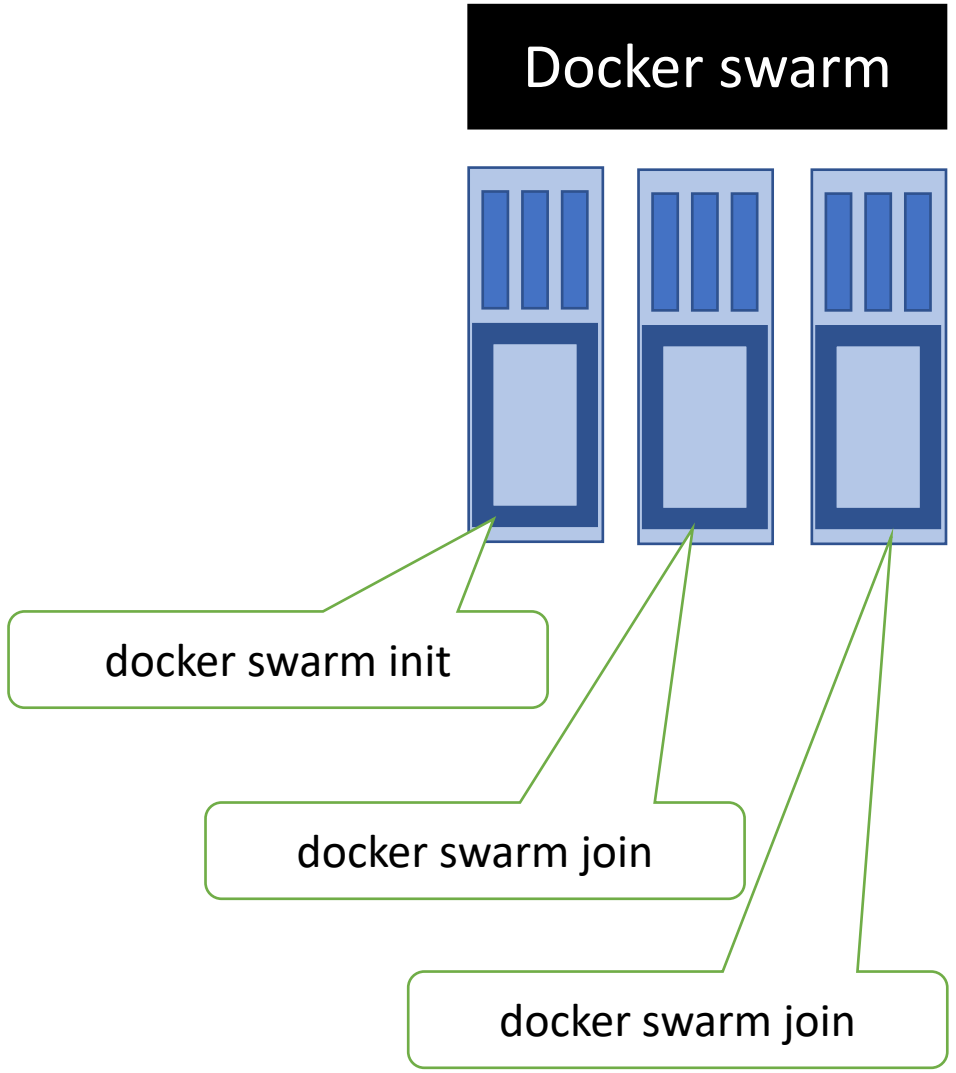
But implemented as
inter-process communication.

```
A() {
    http.get(B:80);
    http.get(C:80);
    http.get(D:80);
}
```

Often considered as corner-stone of cloud-native

Docker Swarm

- Clustering for scalability
- A swarm is a group of host running docker in swarm mode
- A host can be either a *manager* or *worker*
- Workers run *services*
- Manager assigns tasks to worker nodes
  - *Load balancing*

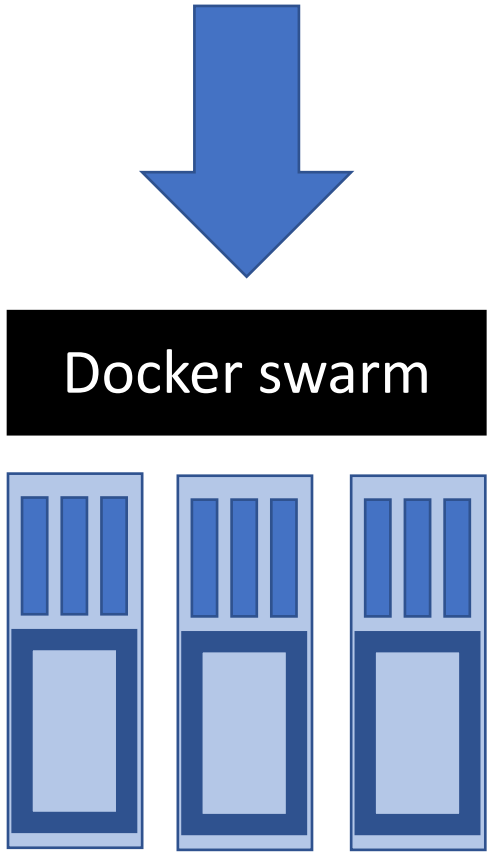


Scalability is important, too

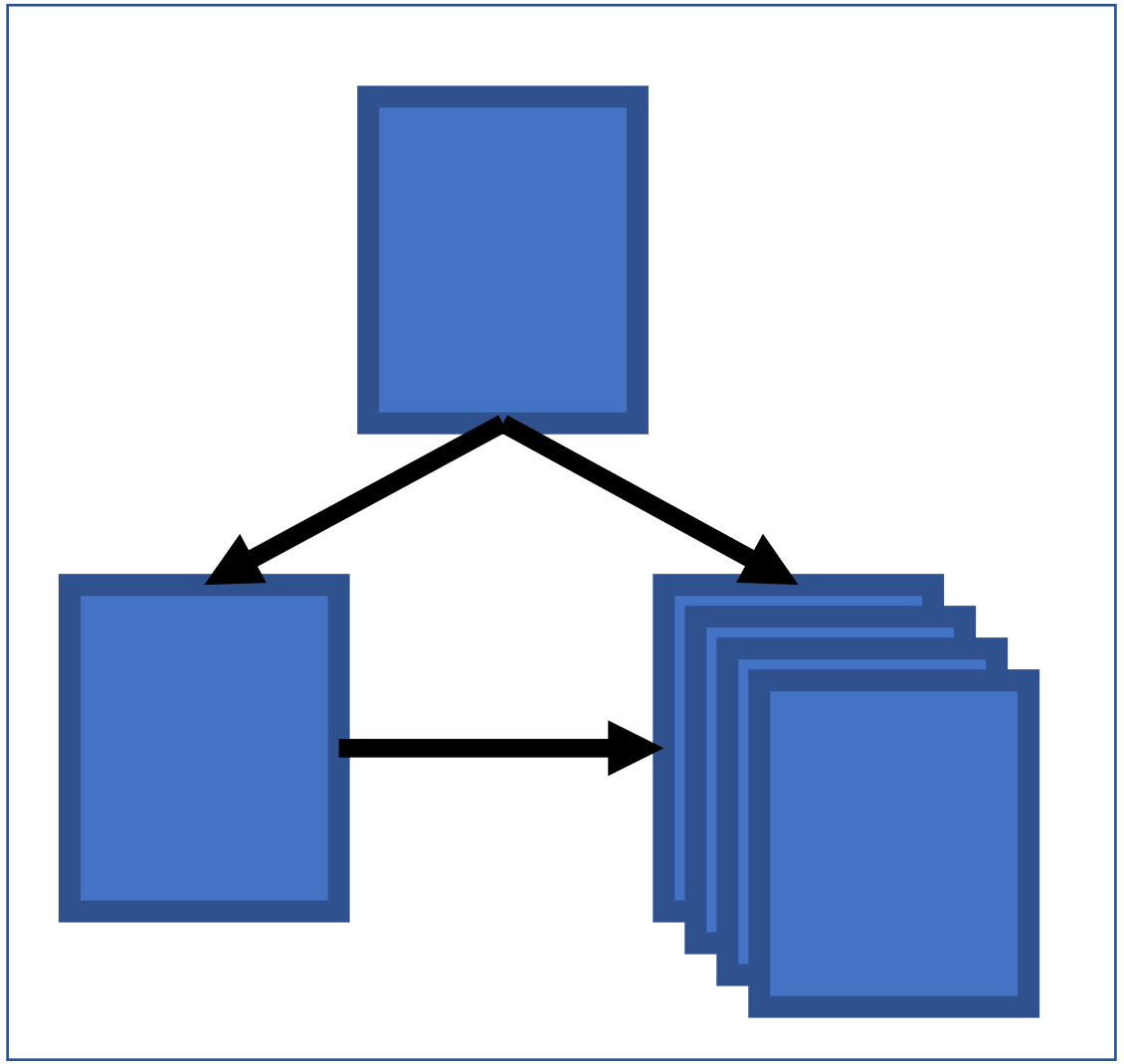
Tampereen yliopisto
Tampere University

Docker swarm - docker compose both support cloud-native

Orchestration

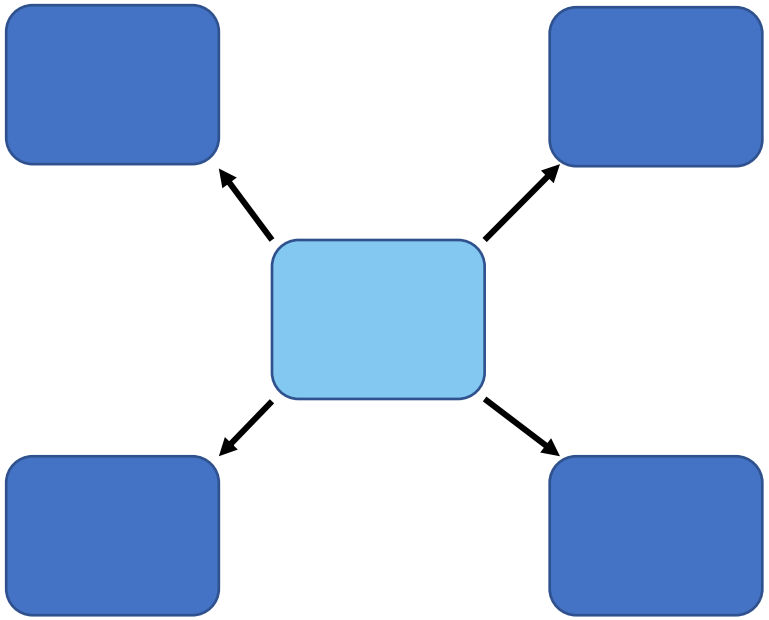


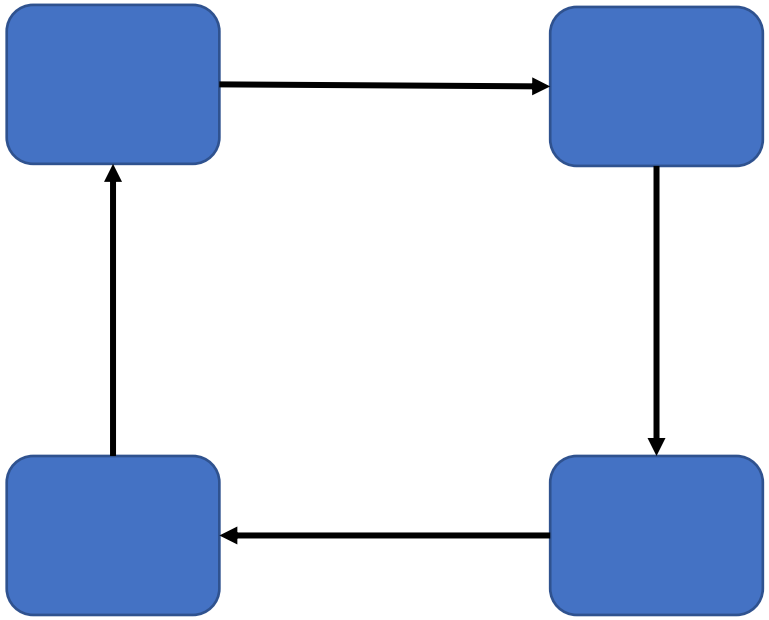
Docker compose



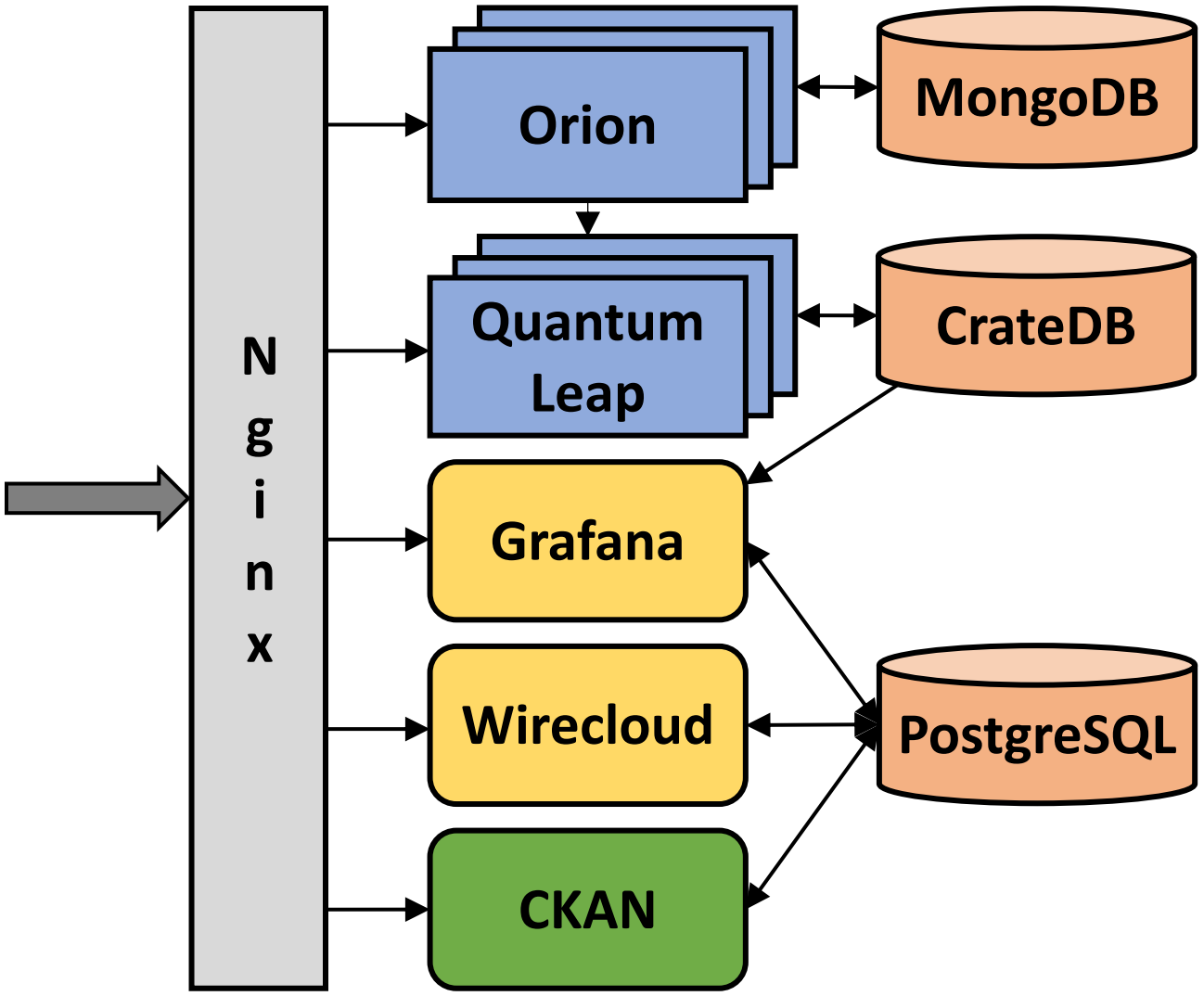
Tampereen yliopisto
Tampere University

Cloud-native should support Orchestration vs Choreography





FIWARE platform architecture



FIWARE Core Component

Dashboard Component

Data Management Component

Database

Access control, proxy server

FIWARE access control components (Keyrock, Wilma and AuthZForce) are not included in this document.

Tampereen yliopisto
Tampere University

Cloud-native applications and architectures

Tampereen yliopisto
Tampere University

Some definitions

- If an app is "cloud-native," it’s specifically designed to provide a consistent development and automated management experience across private, public, and hybrid clouds.
- A native cloud application (NCA) is a program that is designed specifically for a cloud computing architecture.
  - NCAs are designed to take advantage of cloud computing frameworks,
  - which are **composed of loosely-coupled cloud services**.
  - That means that developers must break down tasks into separate **services that can run on several servers in different locations**.
  - Because the infrastructure that supports a native cloud app does not run locally, NCAs must be **planned with redundancy** in mind so the application can withstand equipment failure and be able to re-map IP addresses automatically should hardware fail.

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 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

Tampereen yliopisto
Tampere University

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 provide better performance than is possible with non-native features. For example, you can deal with an input/output (I/O) system that works with autoscaling and load balancing 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 and 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.

Tampereen yliopisto
Tampere University

Microservices

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.

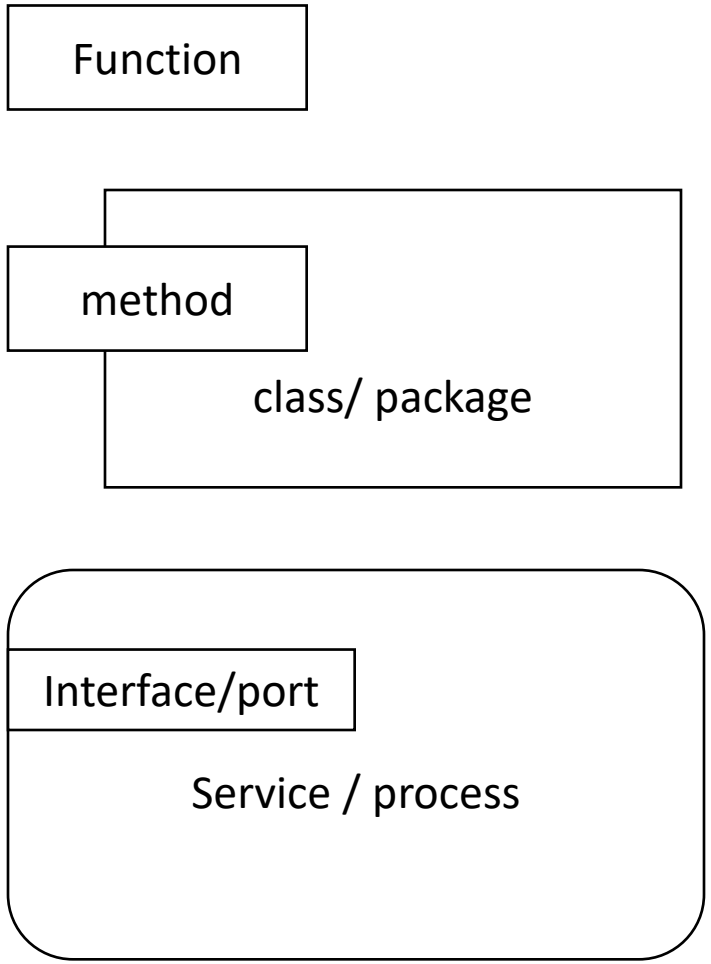
I. Nadareishvili et al., Microservice Architecture: Aligning Principles, Practices, and Culture, O'Reilly, 2016.

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

Tampereen yliopisto
Tampere University

Microservices

- Modular and decomposed ?
- Service-oriented ?
- Distributed ?
- Message-oriented ?
- Independently developed ?
- Independently deployed ?


Function
method
class/ package
Interface/port
Service / process

Tampereen yliopisto
Tampere University

What do you remember from the “Software Engineering Methodologies” course?

- Properties of an Agile (e.g. Scrum) team?

- Self-organizing ?
- Cross-functional ?
- Co-located ?
- 7 +- 2/4 ?

Independent development

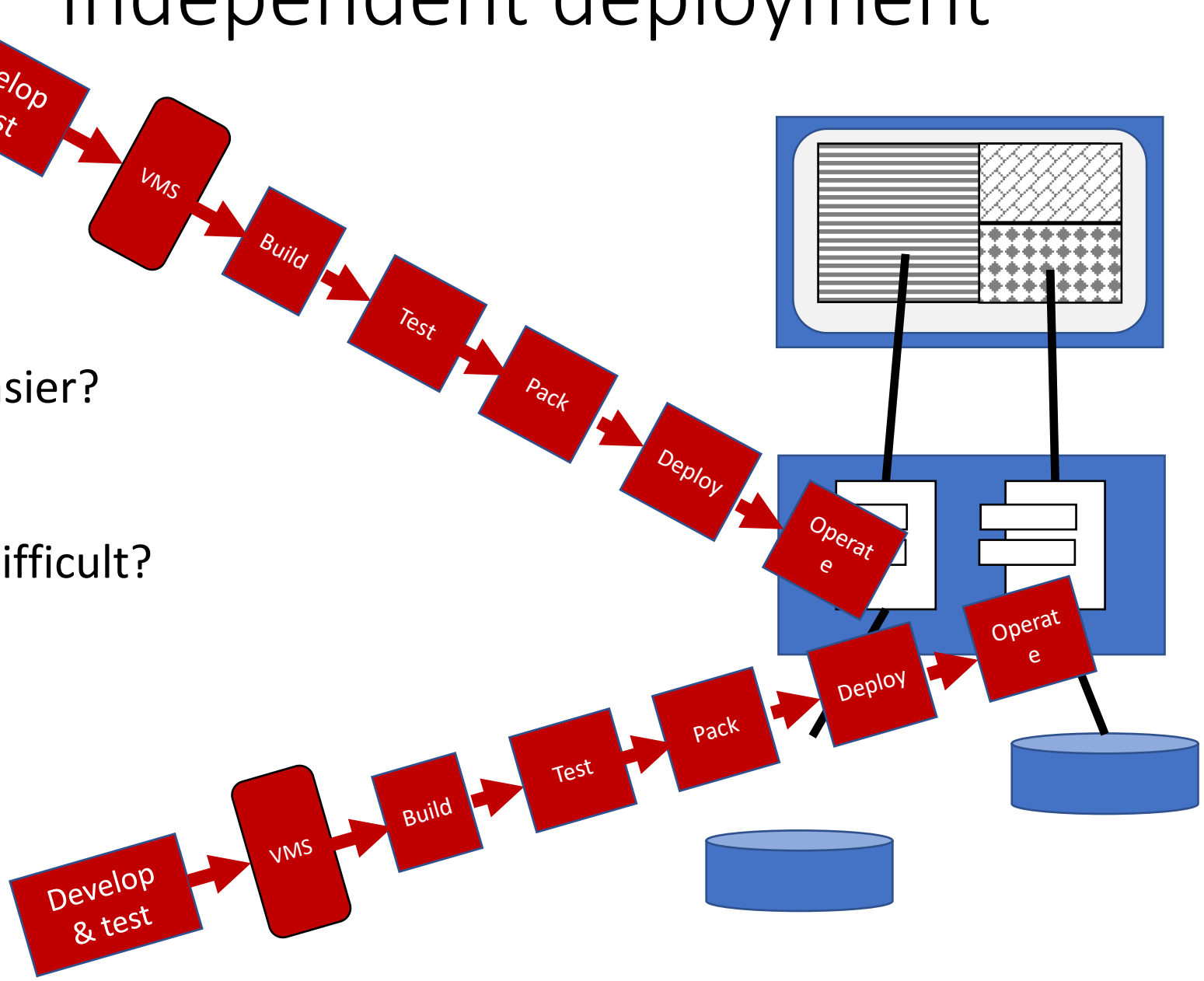
- Separate team
  - Two pizza rule
  - Scrum propose 7±2
  - Small team is more effcient
- Independently selected
  - Run-time
  - Libraries
  - Programming language
- Or, arrogant developers want to use their own favorite?

Tampereen yliopisto
Tampere University

Independent deployment

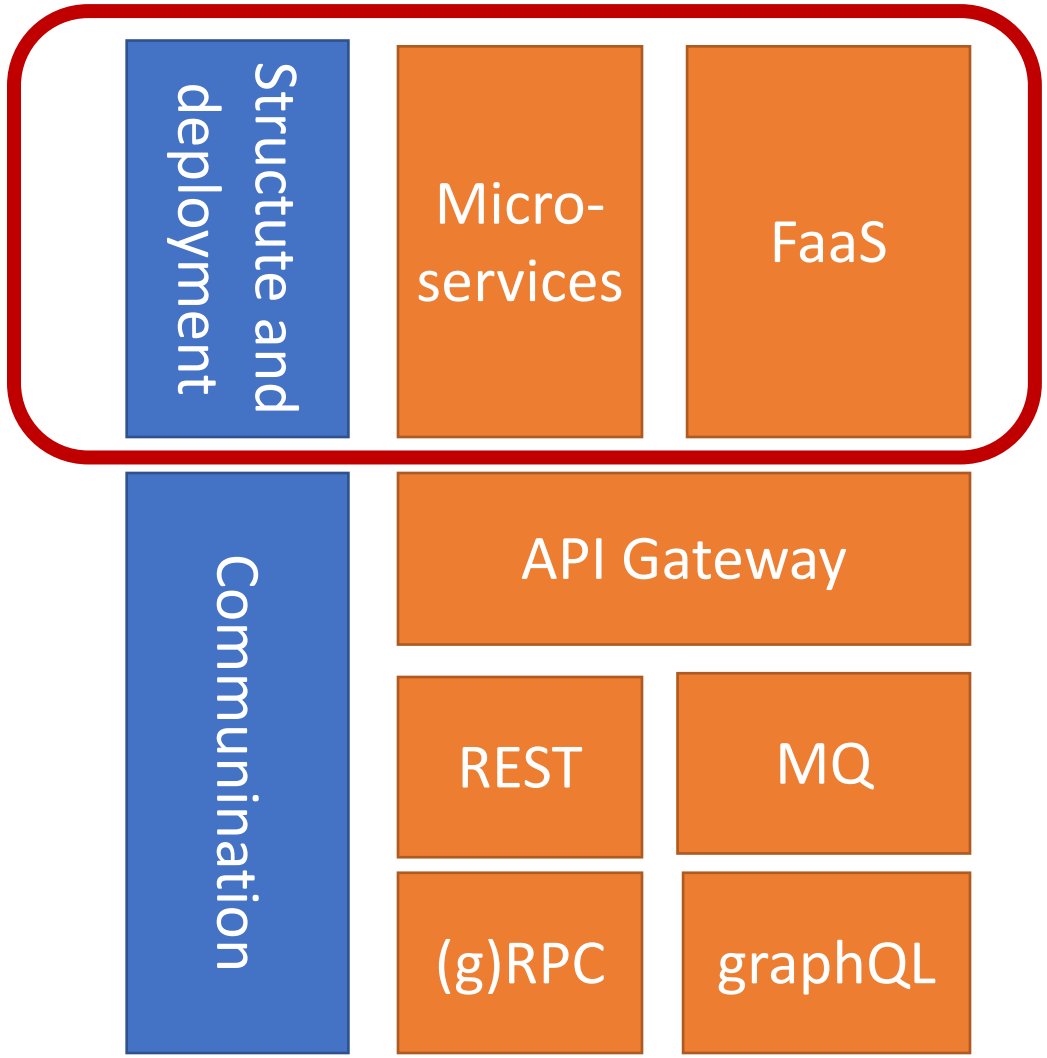
What becomes easier?

What gets more difficult?



Tampereen yliopisto
Tampere University

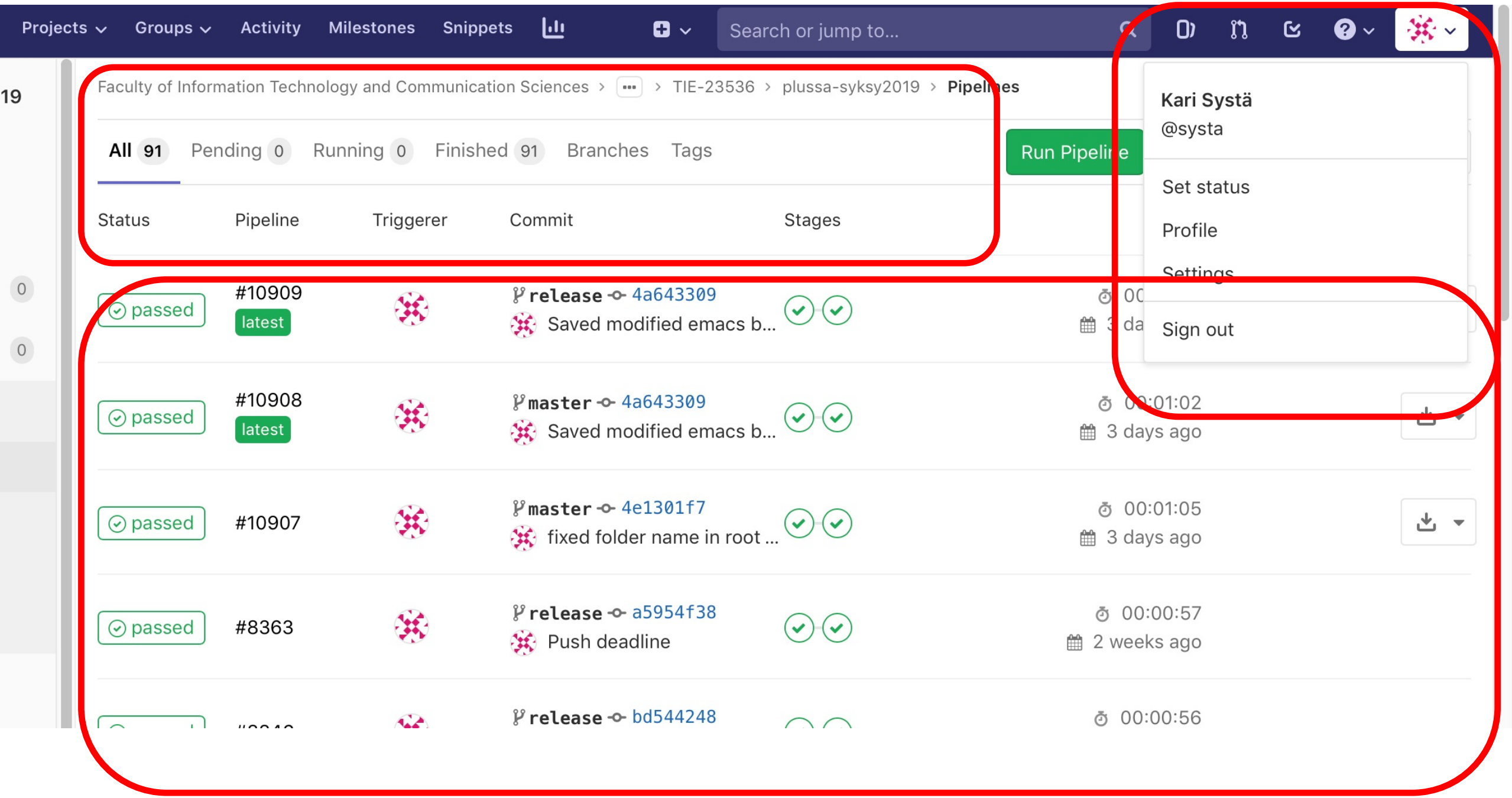
More about cloud-native architectures



Tampereen yliopisto
Tampere University

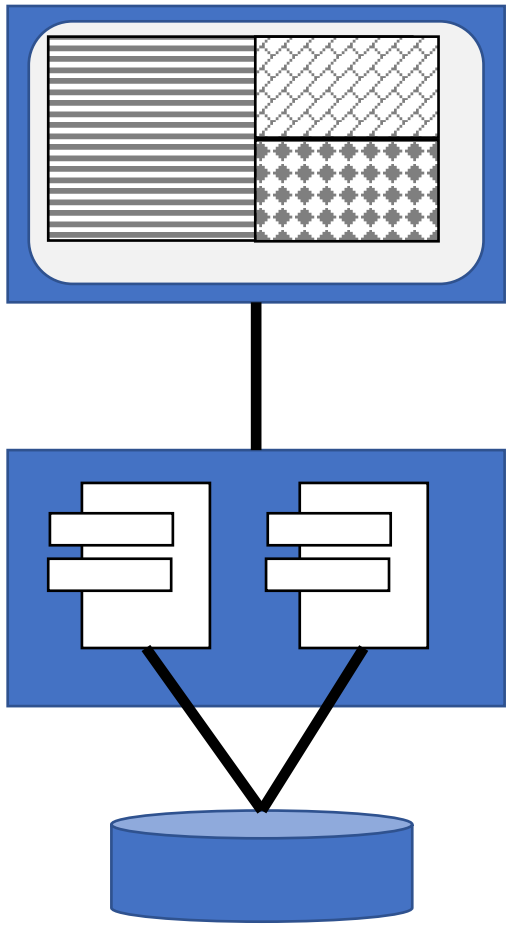
Build around business capabilities?

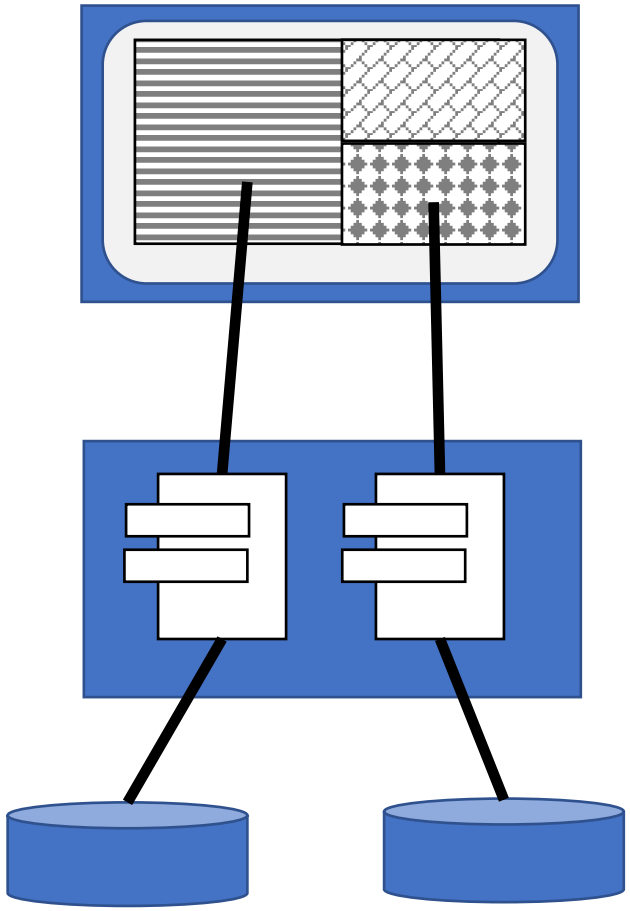
- A way to split monolith to micro services
- Traditional SOA way is to look at static or dynamic dependencies
  - And minimize inter-service calls
- Should support independent development:
  - *Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.*
  - *-- Melvyn Conway, 1967*
- Should support independent deployment
  - Testing
  - Timing

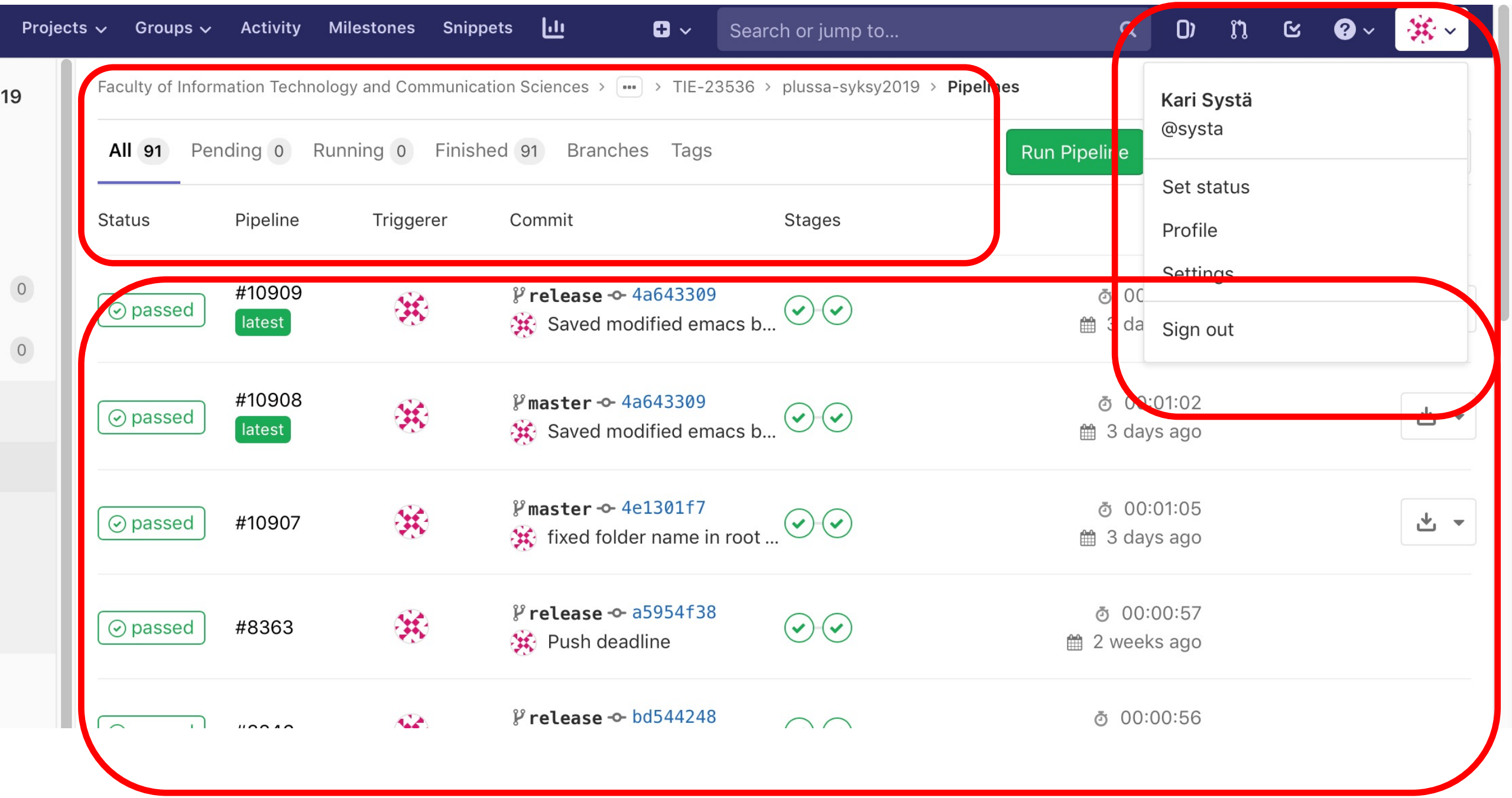

Projects
Groups
Activity
Milestones
Snippets
Search or jump to...
Faculty of Information Technology and Communication Sciences
TIE-23536
plussa-syksy2019
Pipelines
Kari Systä
@systa
Set status
Profile
Settings
Sign out
All 91
Pending 0
Running 0
Finished 91
Branches
Tags
Run Pipeline
Status
Pipeline
Triggerer
Commit
Stages
passed
#10909
latest
release 4a643309
Saved modified emacs b...
passed
#10908
latest
master 4a643309
Saved modified emacs b...
00:01:02
3 days ago
passed
#10907
master 4e1301f7
fixed folder name in root ...
00:01:05
3 days ago
passed
#8363
release a5954f38
Push deadline
00:00:57
2 weeks ago
release bd544248
00:00:56

Tampereen yliopisto
Tampere University

Build around business capabilities?






Projects
Groups
Activity
Milestones
Snippets
Search or jump to...
Faculty of Information Technology and Communication Sciences
TIE-23536
plussa-syksy2019
Pipelines
Kari Systä
@systa
Set status
Profile
Settings
Sign out
All 91
Pending 0
Running 0
Finished 91
Branches
Tags
Run Pipeline
Status
Pipeline
Triggerer
Commit
Stages
passed
#10909
latest
release 4a643309
Saved modified emacs b...
passed
#10908
latest
master 4a643309
Saved modified emacs b...
00:01:02
3 days ago
passed
#10907
master 4e1301f7
fixed folder name in root ...
00:01:05
3 days ago
passed
#8363
release a5954f38
Push deadline
00:00:57
2 weeks ago
release bd544248
00:00:56

Tampereen yliopisto
Tampere University

Homework

- Read arcticle “The Monolith Strikes Back: Why Istio Migrated From Microservices to a Monolithic Architecture” https://ieeexplore.ieee.org/document/9520758 (access requires VPN connection to TUNI)
- And prepare a list of microservice drawbacks you find from the article