Lecture 10

Automation

Kari Systä

03.11.2020

Tampereen yliopisto
Tampere University

Schedule for coming weeks

Week	Lecture	Plussa exercises (deadlines)
10/45	03.11 Testing and testing automation	
11/46	10.11 Guest Lecture, CD pipeline at cargotec	
12/47	17.11 Deployment, hosting and monitoring	
13/48	24.11 Introduction of some popular tools	
14/40	01.12 Recap	

Content of today

- Recap about gitlab CI
- Summary of Cloud Native
- Automation in the pipeline

Tampereen yliopisto
Tampere University

Example from:
https://docs.gitlab.com/ee/user/project/pages/getting_started/pages_from_scratch.html

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

Tampereen yliopisto
Tampere University

Is this correct?

Why not?

Empty

Before script

Test

Before script

deploy

```
gem install bundler
bundle install
```

```
gem install bundler
bundle install
```

```
bundle exec jekyll build -d test
```

```
bundle exec jekyll build -d build
```

Tampereen yliopisto
Tampere University

This is correct visualization!

Empty

Before script

Test

If branch != master

If branch == master

Before script

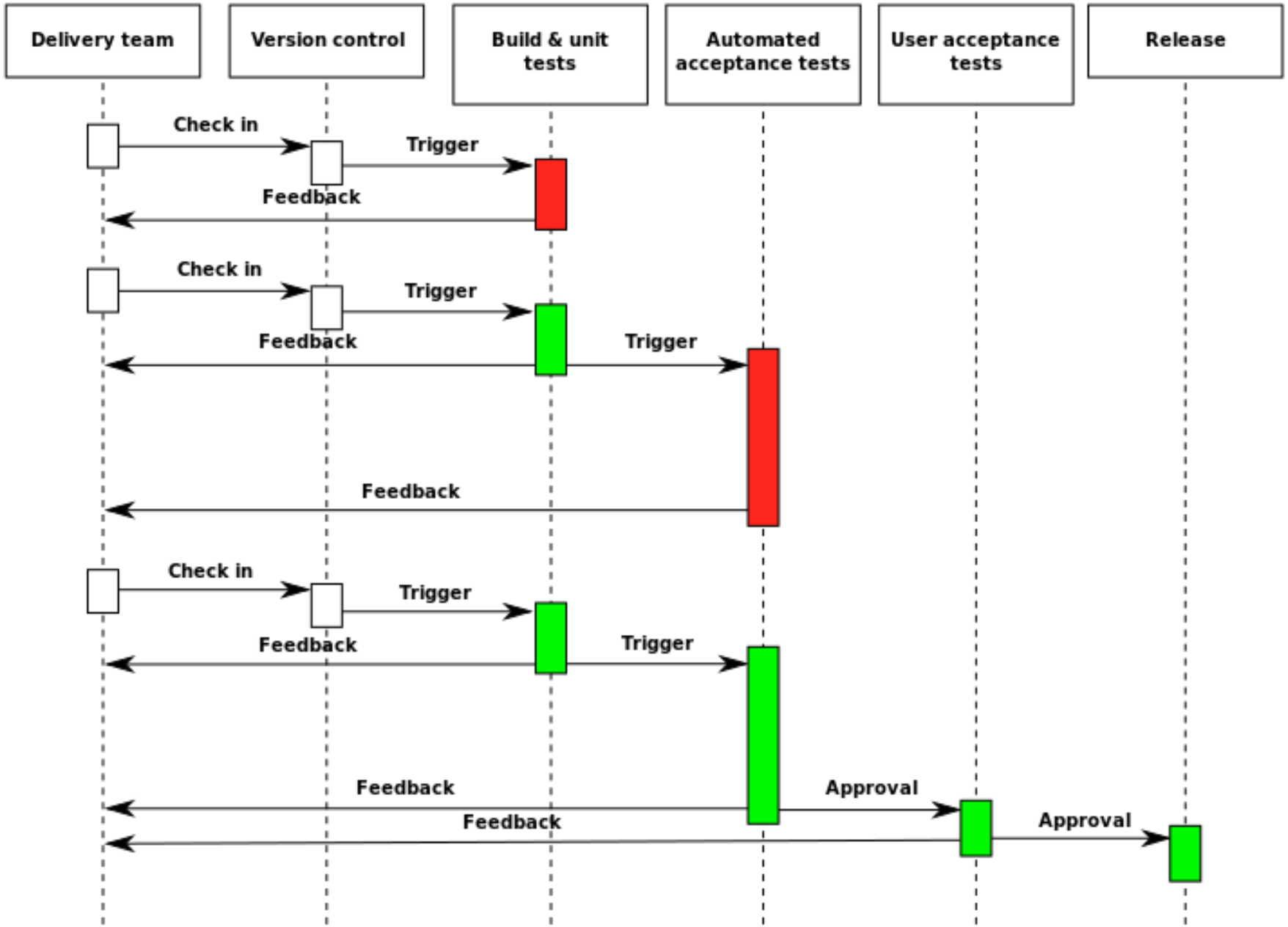
deploy

DevOps practices

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

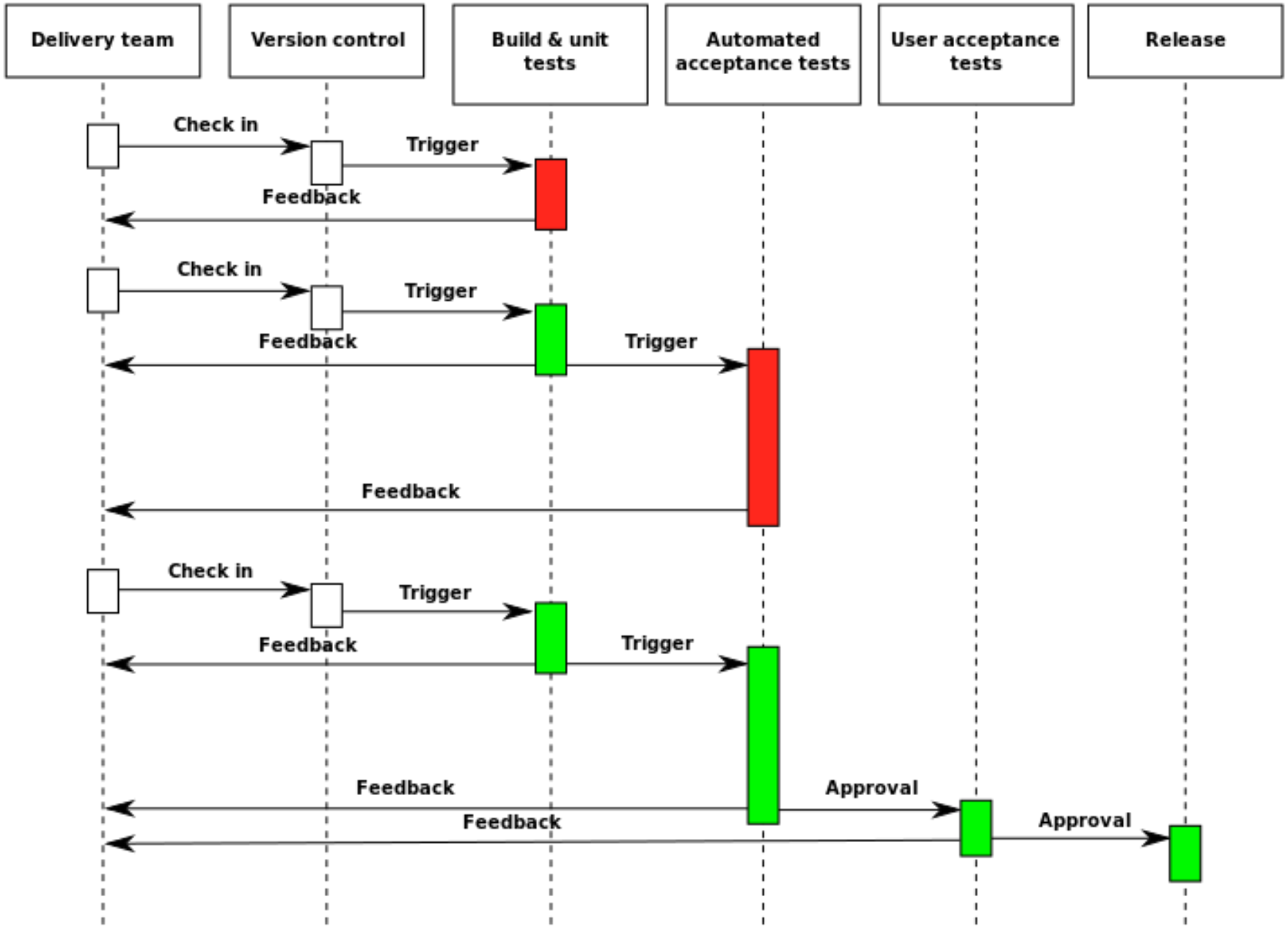
- Technical
  - **automation,**
  - monitoring
  - measurement

Deployment pipeline (a possible example)



About automation

Deployment pipeline (a possible example)



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.

Benefits of automation

- Prevent errors
- Is repeatable
- No need to write documentation
- Enables collaboration because everything is explicit in scripts
- Expertise encapsulated in scripts
- Manual work is boring
- Fast and relentless feedback
- Risk management: Automated checking and auditing

Automation includes

- Building
  -> no command-line tools needed
- Testing
  -> run frequently
- Other quality analysis
  -> less manual inspection needed;
- Deployment
  -> VMs and containers created automatically
  -> configuration management
- Database tools
  -> initialization
  -> management
- Scaling

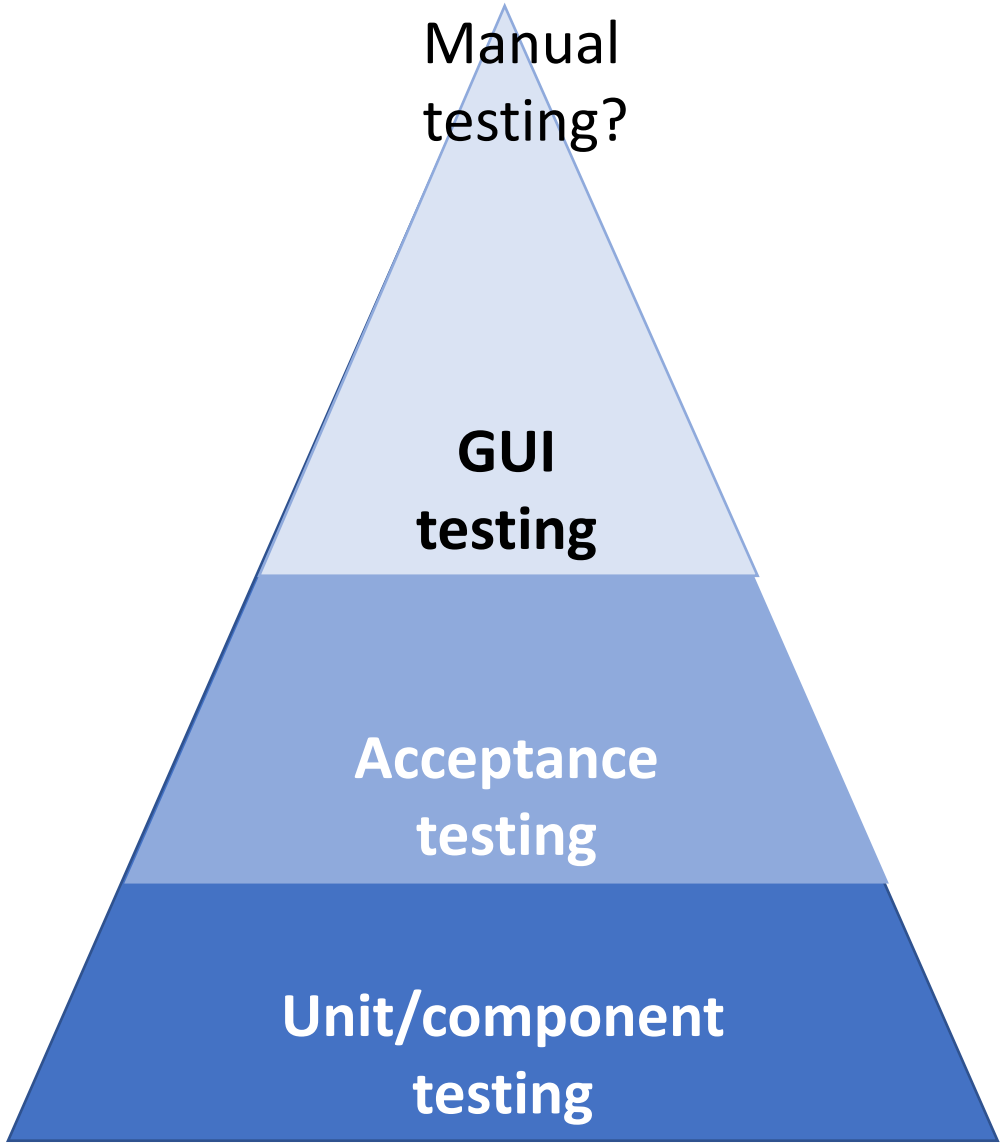
Automated tests

- A common practice in CI and CD
- Does not invent the test (usually);
  - test are designed and implemented manually but
  - executed automatically
- Tests need to maintained
- Software needs to be testable
- Not a silver bullet for testing, but necessary helper in CI/CD

Testability

- Testbed can command the software
- Tests can investigate state and results
- Proper architecture and coding style helps
  e.g. Standard getters and setters
- Well-defined APIs

Tampereen yliopisto
Tampere University

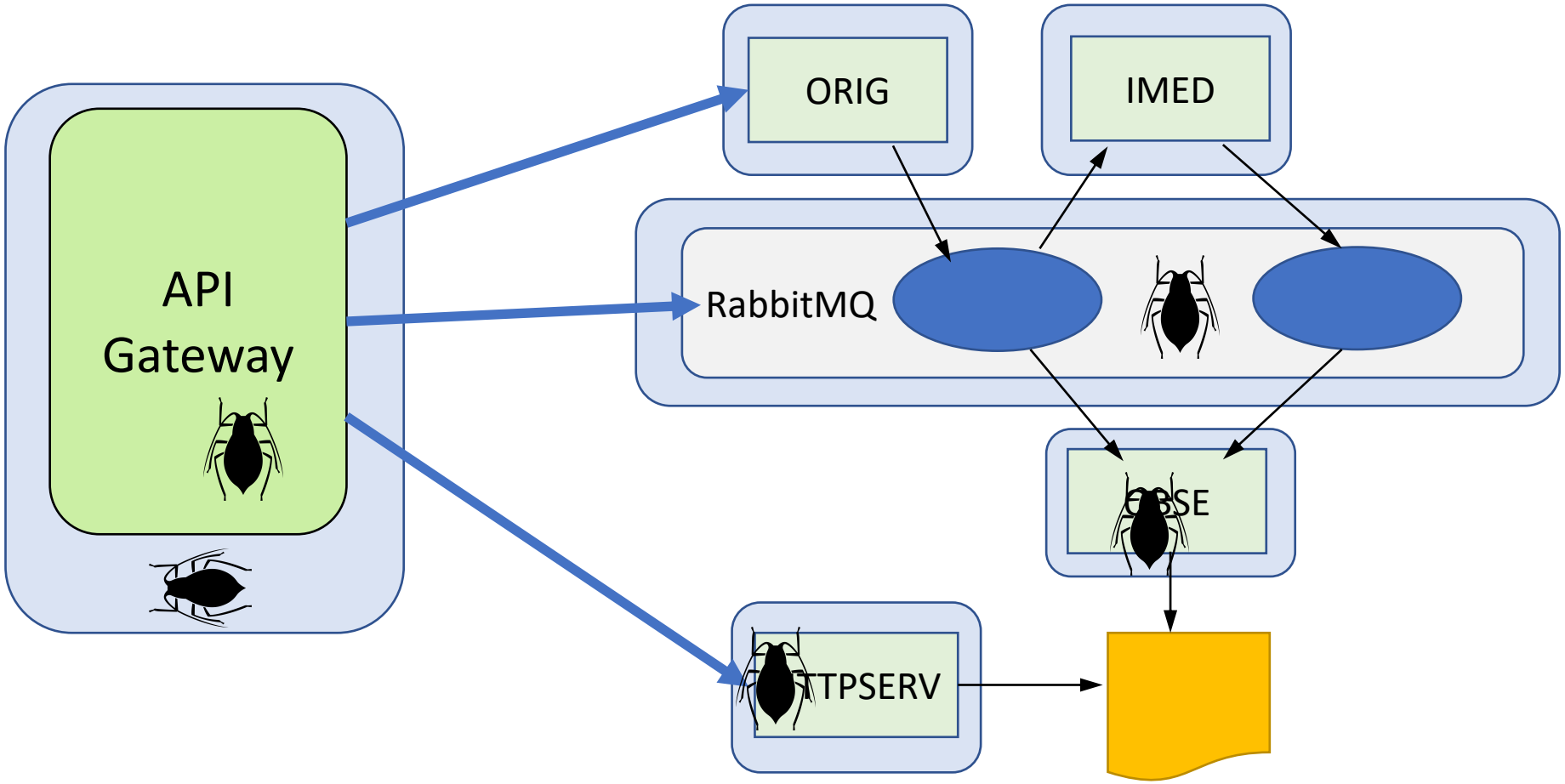


Automated acceptance tests

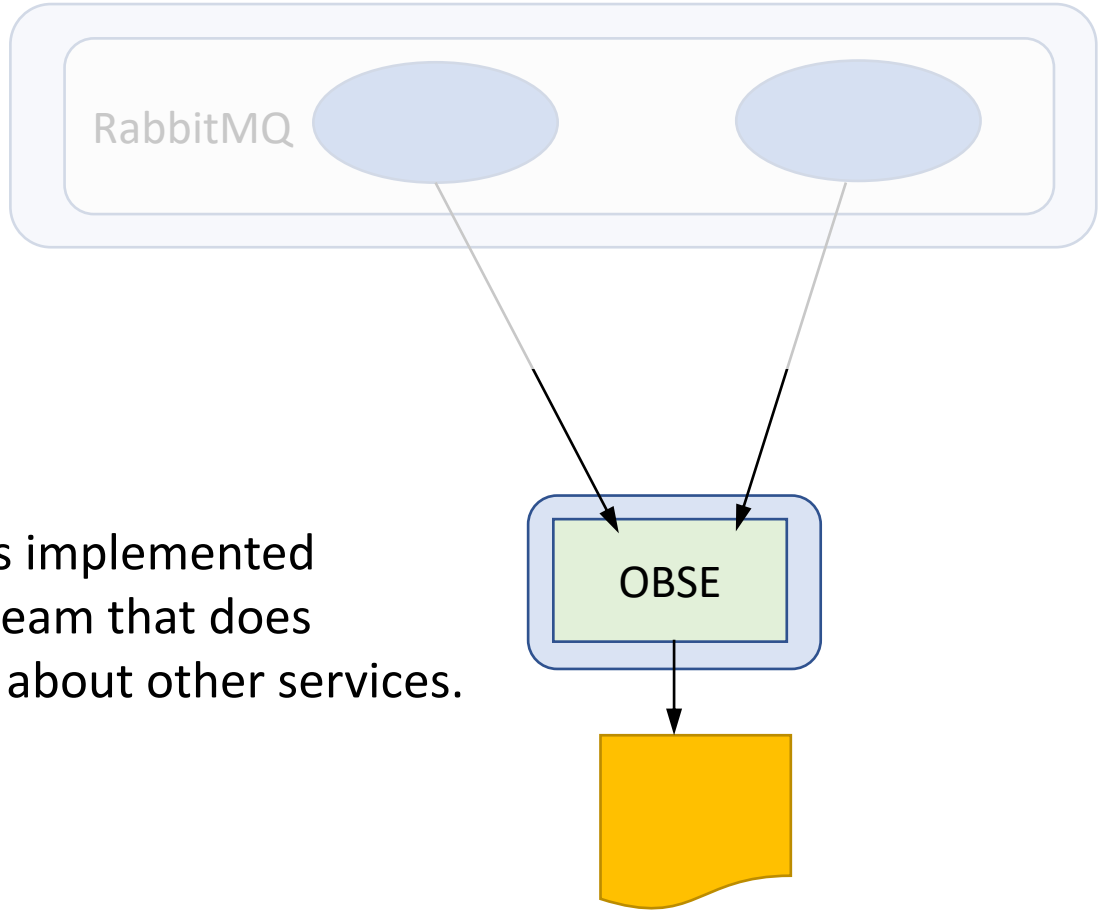
- Acceptance tests do not test everything but is an essential “gate” if deployment is automated.
- Some best practices (according to Humbley and Farley):
  - Test in realistic environment(s)
  - Acceptence tests are owned by the whole team (no separate team for it)
  - Developers should be able to run the tests in their own dev environment)
  - Tie to business value – not to technical solution of the system
- Nonfunctional testing
  - Capacity, scalability
  - Code quality analysis

Tampereen yliopisto
Tampere University

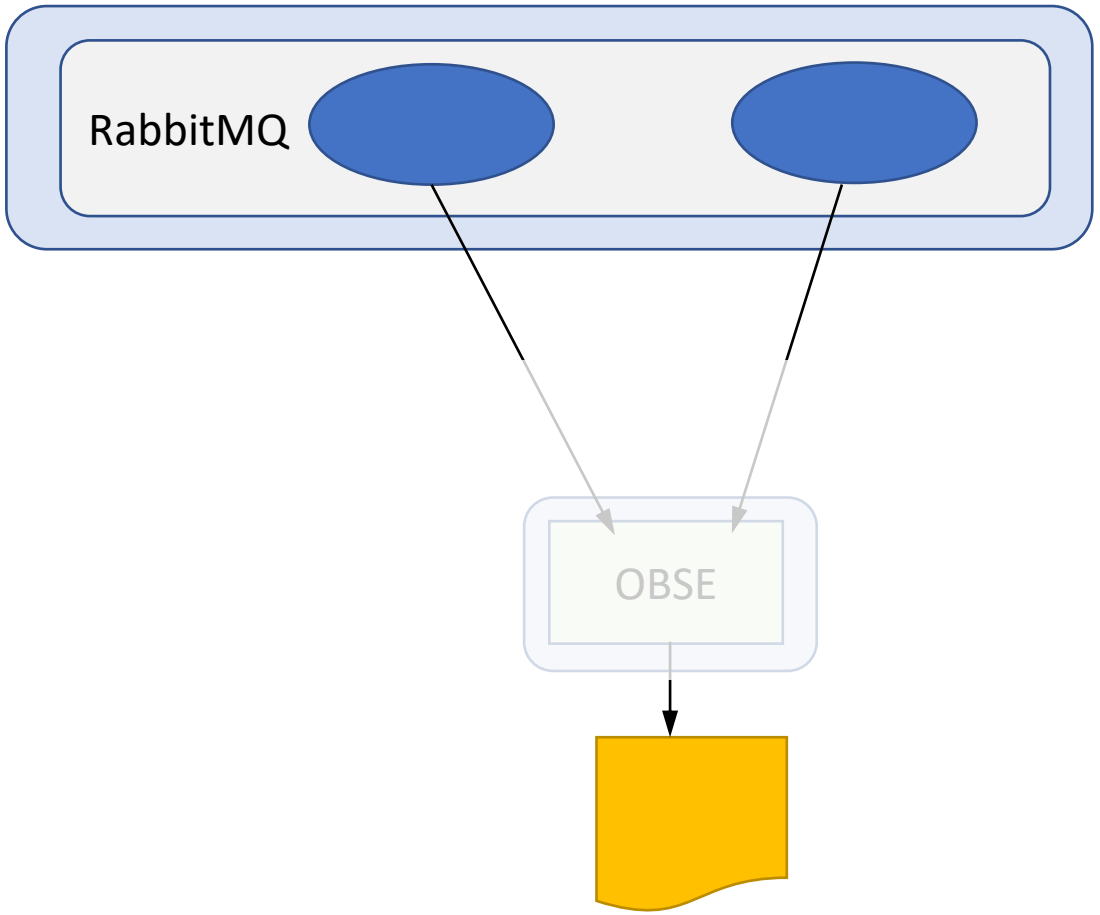
Testing cloud-native is difficult
And debugging even more difficult



Tampereen yliopisto
Tampere University



What if OBSE is implemented
by a separate team that does
not now much about other services.

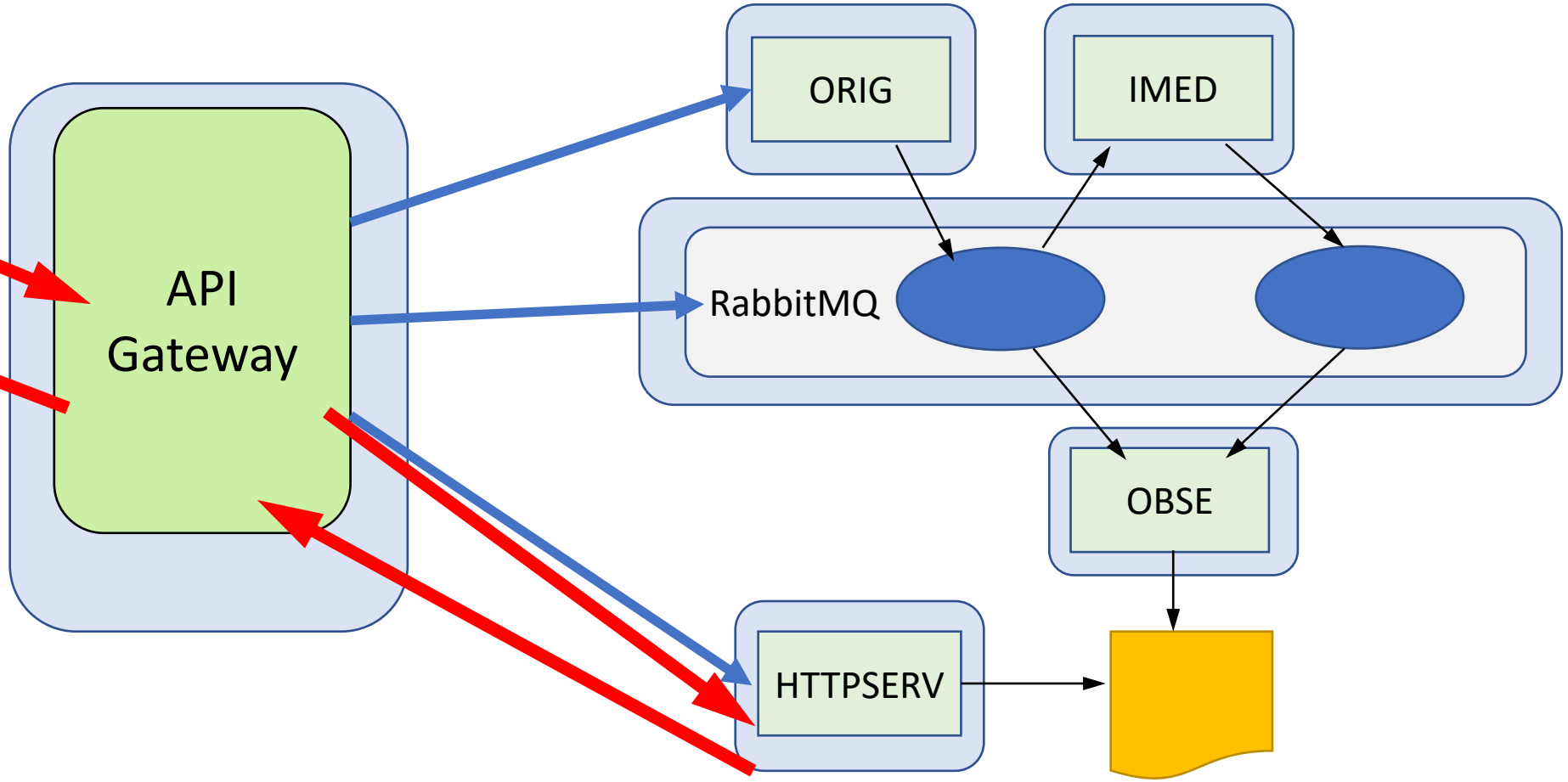

RabbitMQ
OBSE

Tampereen yliopisto
Tampere University

GET /messages

2020-10-01T06:35:01.373Z
Topic my.o: MSG_1

Correct?

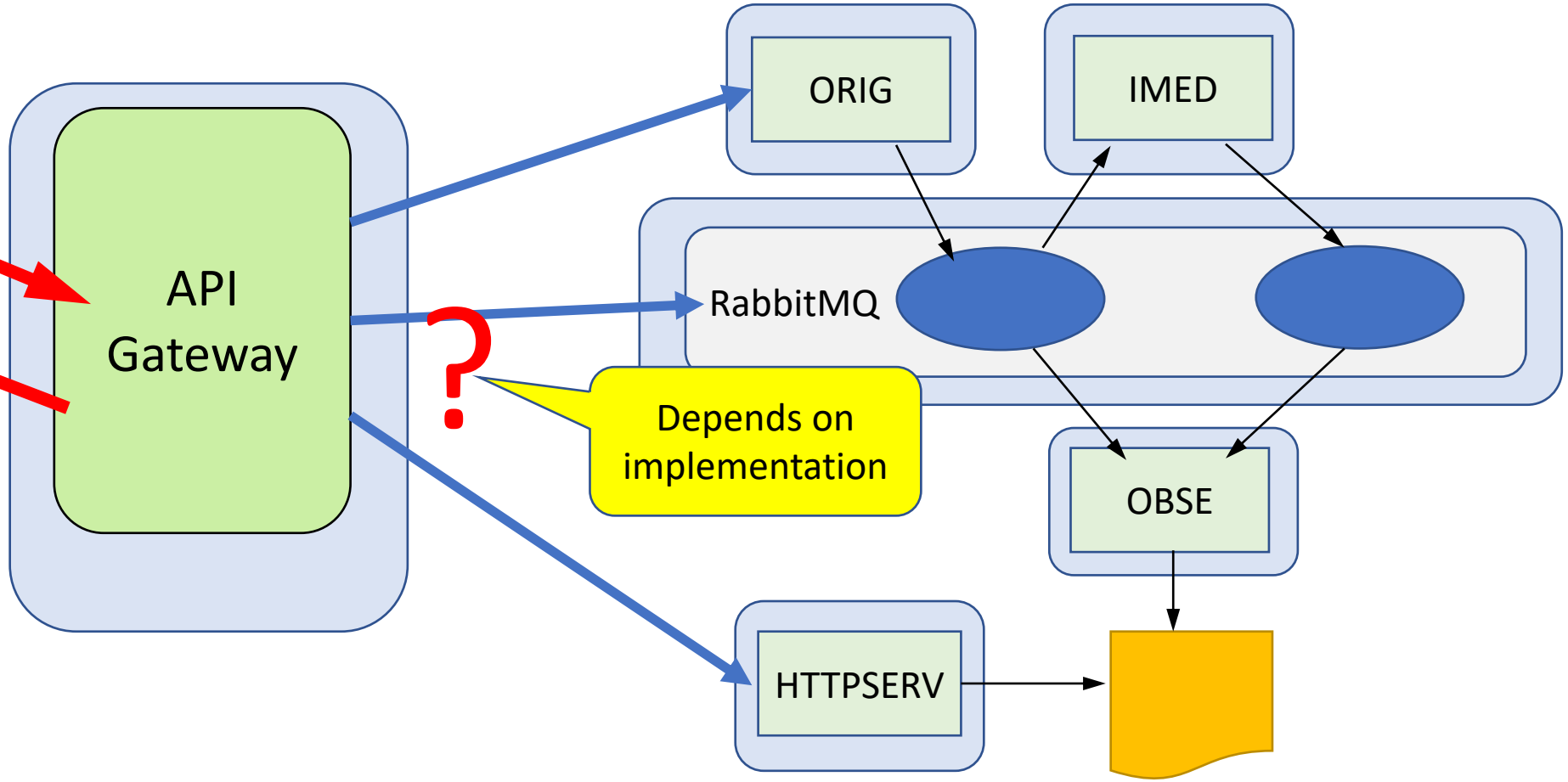


Tampereen yliopisto
Tampere University

GET /run-log

2020-11-01T06:35:01.373Z: INIT
2020-11-01T06:40:01.373Z:
PAUSED
2020-11-01T06:40:01.373Z:
RUNNING

Correct?



Testing microservices

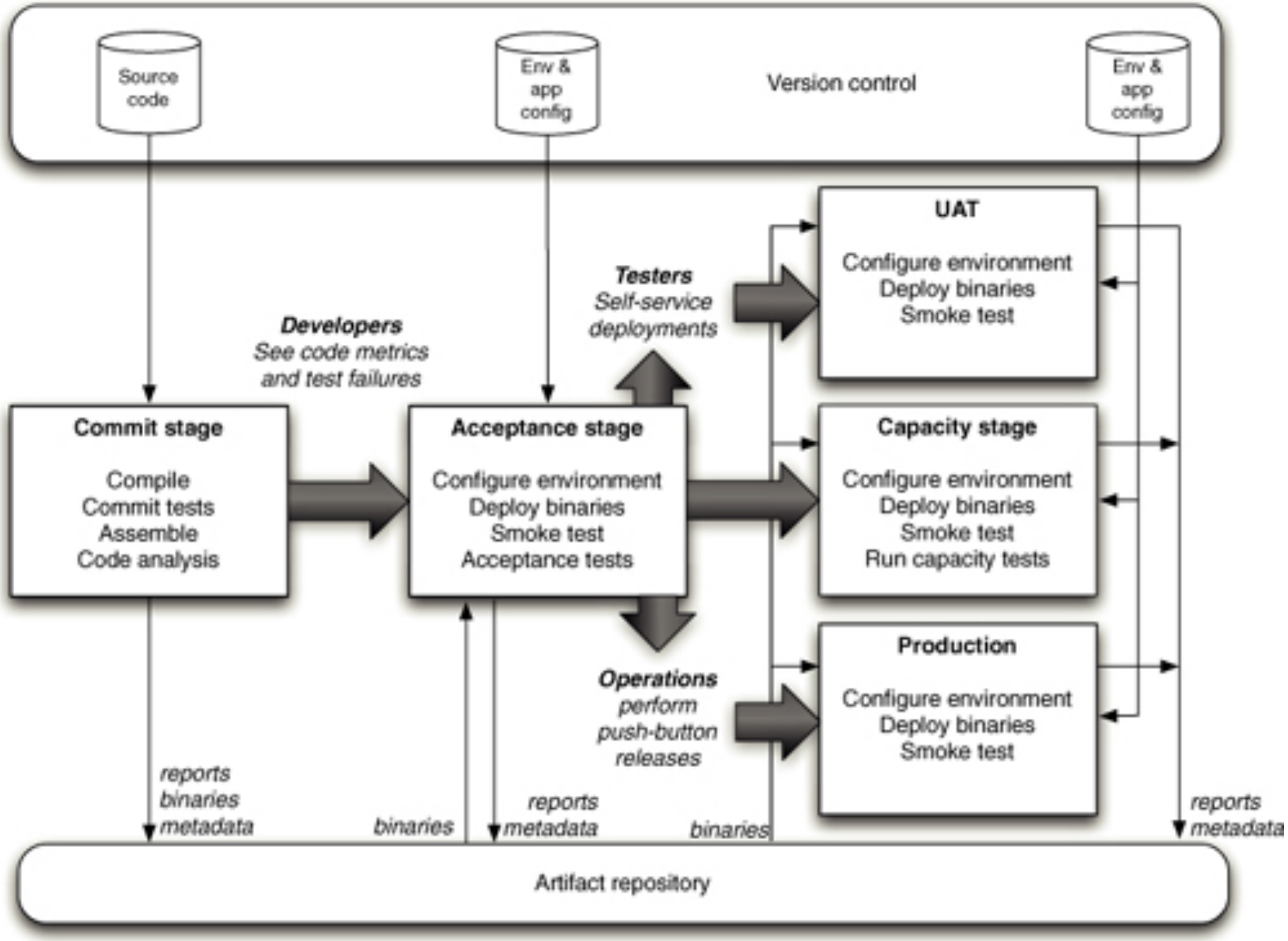
(https://www.infoq.com/articles/twelve-testing-techniques-microservices-intro/)

Key takeaways

- Because a microservice architecture relies more on over-the-wire (remote) dependencies and less on in-process components, your testing strategy and test environments need to adapt to these changes.
- When testing monoliths using existing techniques like service virtualization, you do not have to test everything together; instead, you can divide and conquer, and test individual modules or coherent groups of components.
- When working with microservices, there are also several more options available, because microservices are deployed typically in environments that use containers like Docker.
- You will need to manage the interdependent components in order to test microservices in a cost and time effective way. You can use test doubles in your microservice tests that pretend to be real dependencies for the purpose of the test.

Tampereen yliopisto
Tampere University

Artefact repository

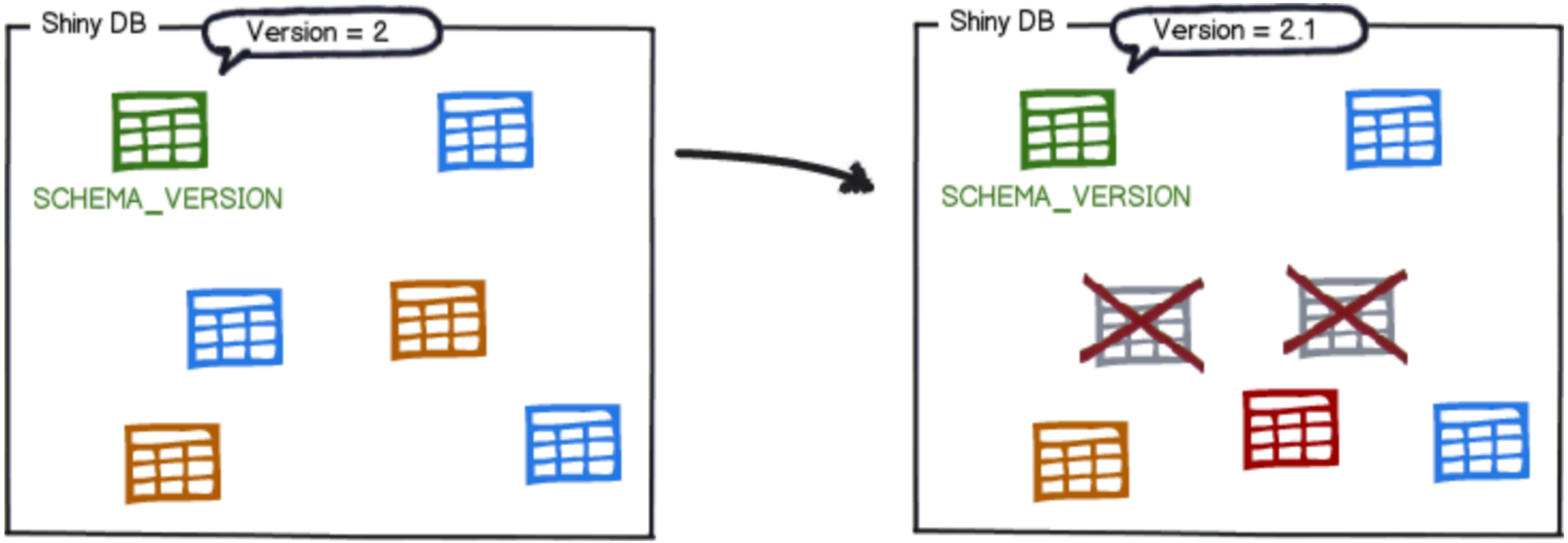


Tampereen yliopisto
Tampere University

Example of data base automation https://flywaydb.org

” Flyway is an open-source database migration tool. It strongly favors simplicity and convention over configuration. It is based around just 6 basic commands: Migrate, Clean, Info, Validate, Baseline and Repair. Migrations can be written in SQL (database-specific syntax (such as PL/SQL, T-SQL, …) is supported) or Java (for advanced data transformations or dealing with LOBs).”

They are then **sorted by version number** and **executed in order**:



The **schema history table** is **updated** accordingly:

flyway_schema_history

installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1	Initial Setup	SQL	V1__Initial_Setup.sql	1996767037	axel	2016-02-04 22:23:00.0	546	true
2	2	First Changes	SQL	V2__First_Changes.sql	1279644856	axel	2016-02-06 09:18:00.0	127	true
3	2.1	Refactoring	JDBC	V2_1__Refactoring		axel	2016-02-10 17:45:05.4	251	true

Automation challenges

- ”…provisioning scripts were considered error-prone and, according to developers, they did not work in some environments…”
- ”…automation of the network in was said to be difficult in addition to dealing with legacy system…”
- ”Networks are pretty hard. Some of the databases are pretty hard too because the old relational databases haven’t been designed to be clustered…”

Tampereen yliopisto
Tampere University

Automation scripts are programs Infrastructure as code

- ”Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.”
- three approaches to IaC: declarative (functional) vs. imperative (procedural) vs. intelligent (environment aware)

```
tasks:
- name: ensure apache is at the
        latest version
  yum:
    name: httpd
    state: latest
- name: ensure that postgresql is started
    service:
      name: postgresql
      state: started
```

```
apt-get install …
```

Tampereen yliopisto
Tampere University

Infrastructure as code

All SW engineering principles should be applied.

- Testing
- Maintenance
- Documentation
- Version and configuration management

- Bugs may stop the whole engine

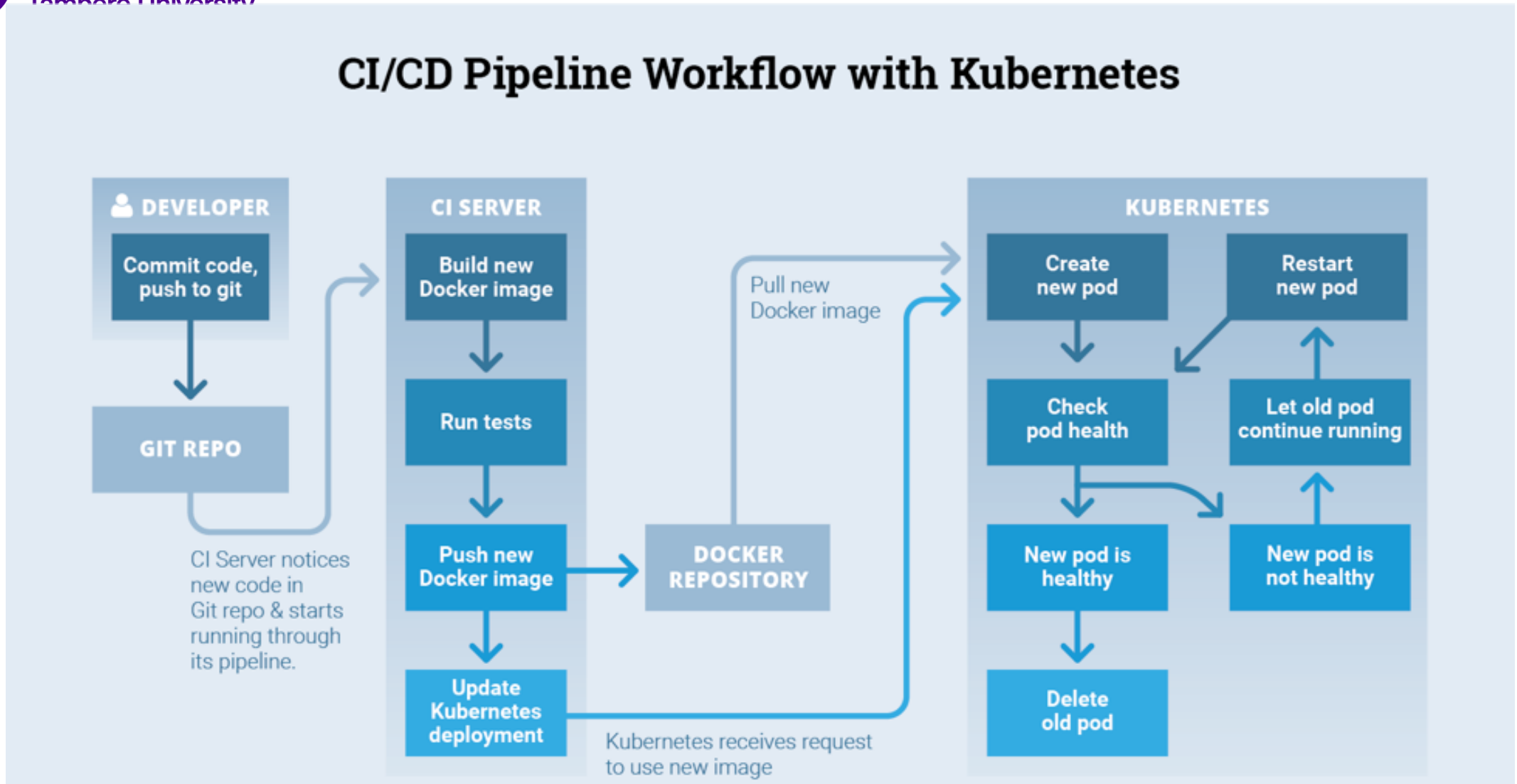
Tampereen yliopisto
Tampere University

Huge number or tools available

- https://digital.ai/periodic-table-of-devops-tools
- https://landscape.cncf.io

Summary of cloud native

One proposal



Fine but drawn by a Kubernetes consultancy company

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

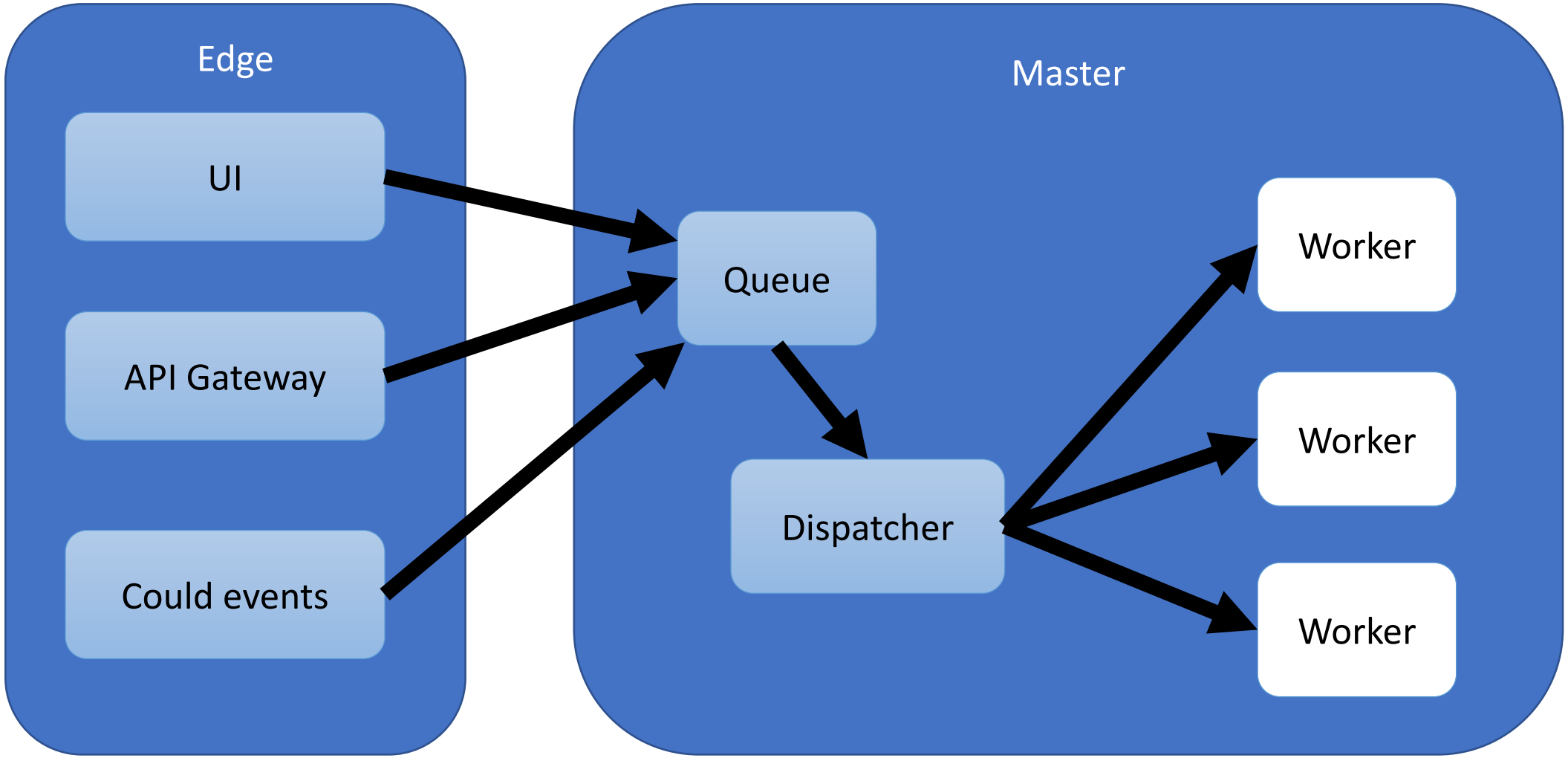
8. Messaging

9. Container registry and runtimes

10. Sofware distribution

Serverless

Baldini et al: Serverless Computing: Current Trends and Open Problems



Tampereen yliopisto
Tampere University

Microservices vs. Serverless/FaaS
(They are different – do not call serveless microservices)

- Microservice
  - Small services running in their own process and communicating with lightweight services
  - Can be stateful

- Serverless / FaaS
  - Short term execution triggered by a request, then closes down
  - For stateless computing

Tampereen yliopisto
Tampere University

Some comparison

	Microservice	Serverless / FaaS
Bug hunting	Easier (but not easy)	Difficult
Infrastructure code	May be complex	Minimal or even non-existent
Scaling	Need to be implemented	Automatic
Performance	Good	Possible cold-start issues
Running cost	May include cost of idle time	Pay only per use

Tampereen yliopisto
Tampere University

Microfront-ends

Projects Groups Activity Milestones Snippets Search or jump to...

Faculty of Information Technology and Communication Sciences › ··· › TIE-23536 › plussa-syksy2019 › Pipelines

All 91 Pending 0 Running 0 Finished 91 Branches Tags

Run Pipeline

Kari Systä
@systa

Set status
Profile
Settings
Sign out

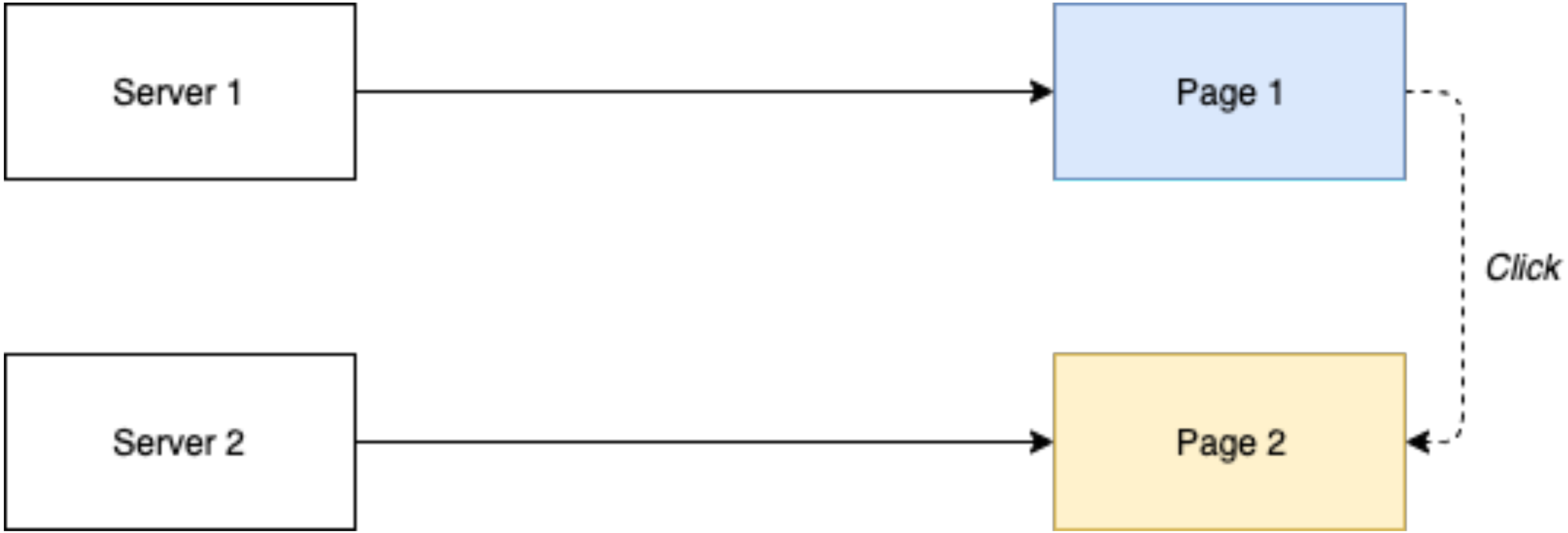
Status	Pipeline	Triggerer	Commit	Stages		
passed	#10909 latest		release 4a643309 Saved modified emacs b...		00 3 da	
passed	#10908 latest		master 4a643309 Saved modified emacs b...		00:01:02 3 days ago	
passed	#10907		master 4e1301f7 fixed folder name in root ...		00:01:05 3 days ago	
passed	#8363		release a5954f38 Push deadline		00:00:57 2 weeks ago	
			release bd544248		00:00:56	

Tampereen yliopisto
Tampere University

Alternative architectures

(from https://morioh.com/p/ee1b48c9de16)

1. Web Approach

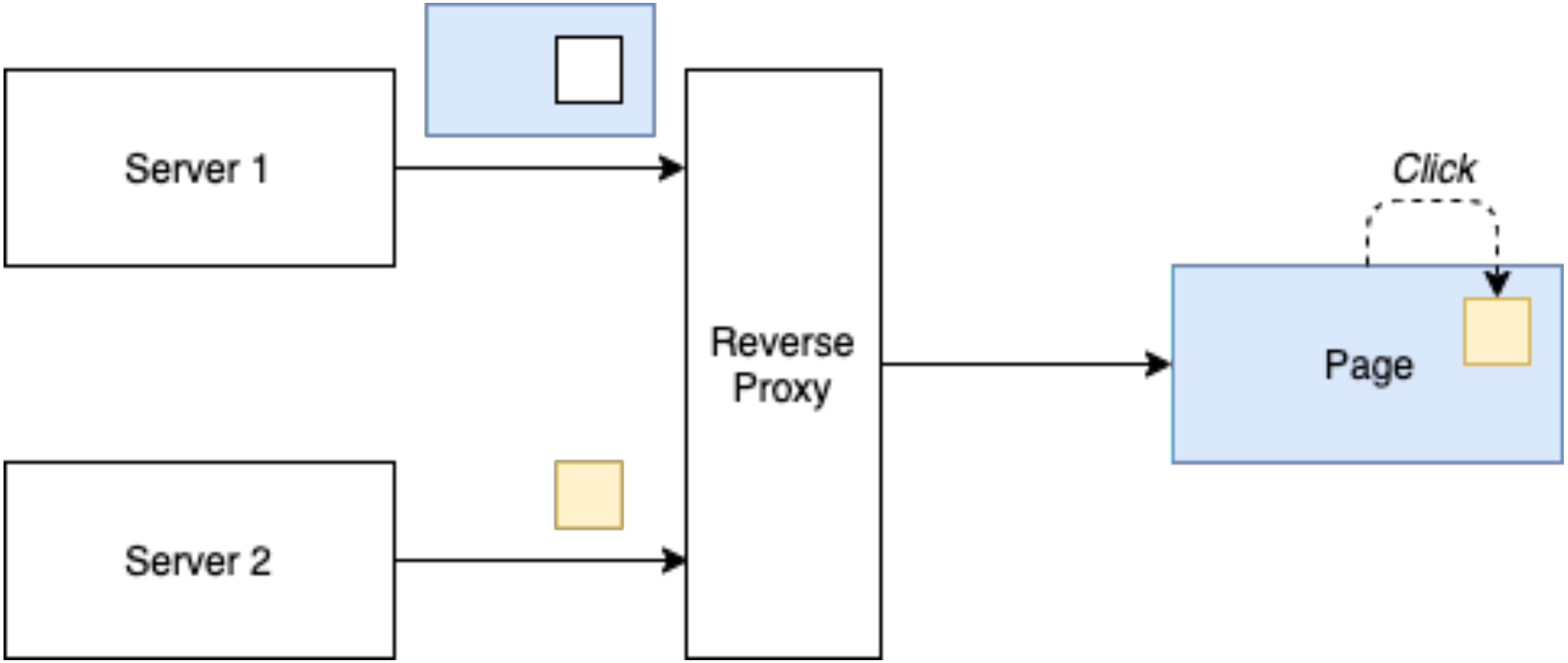


Tampereen yliopisto
Tampere University

Alternative architectures

(from https://morioh.com/p/ee1b48c9de16)

2. Server-side composition

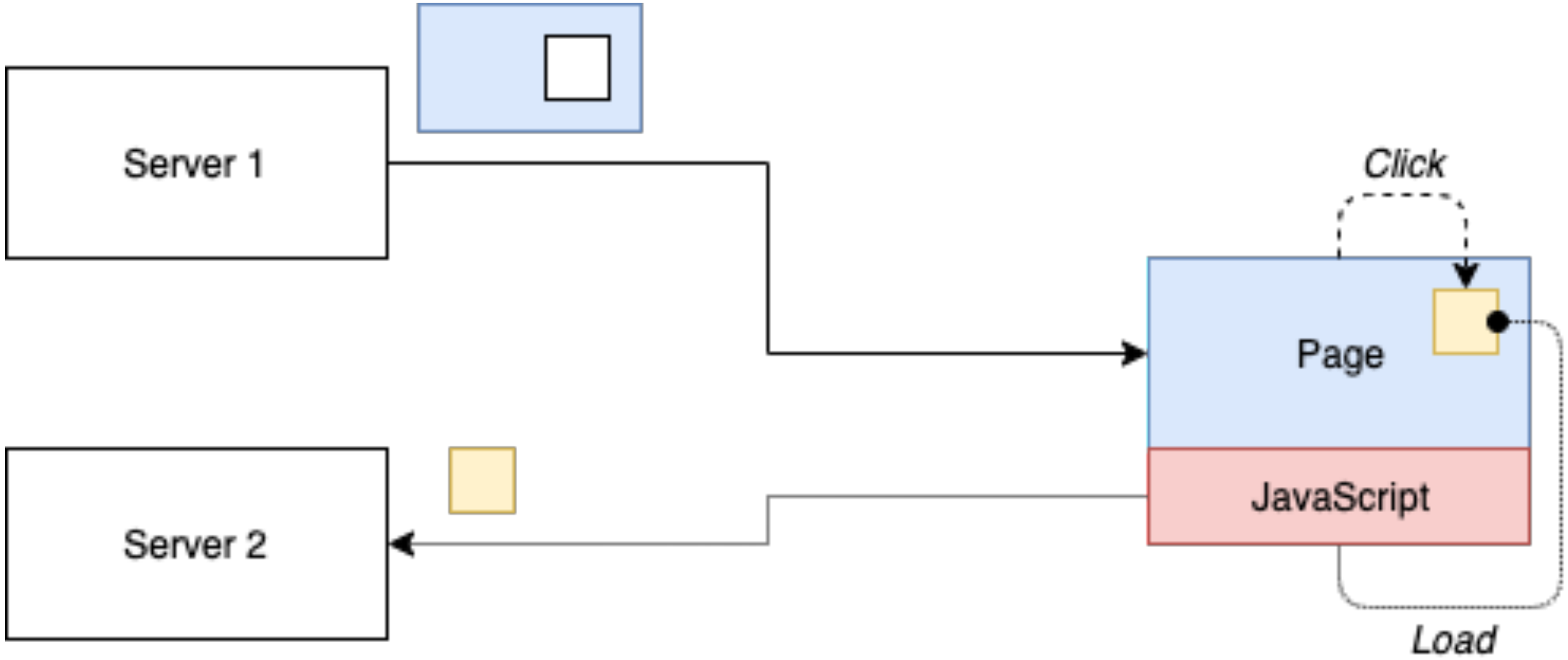


Tampereen yliopisto
Tampere University

Alternative architectures

(from https://morioh.com/p/ee1b48c9de16)

3. Client-side composition

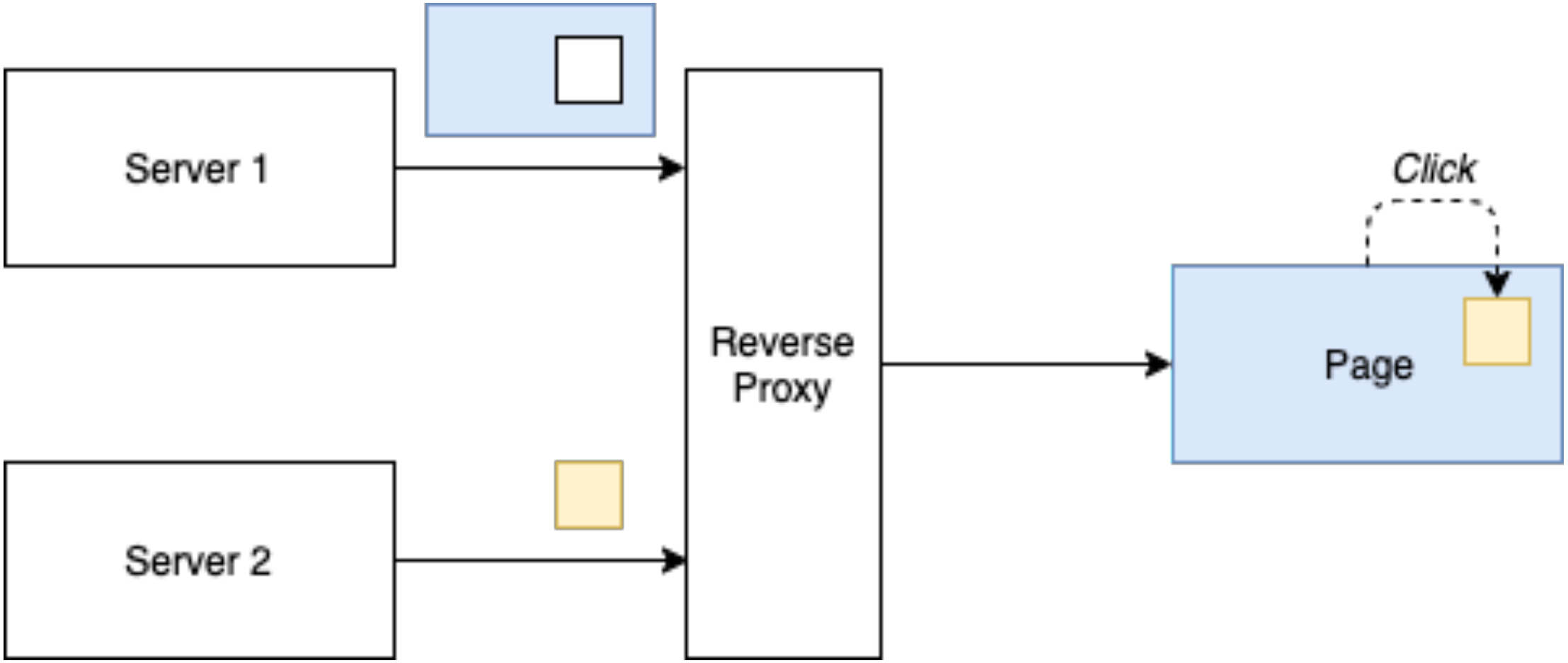


Tampereen yliopisto
Tampere University

Alternative architectures

(from https://morioh.com/p/ee1b48c9de16)

4. Client-side rendering

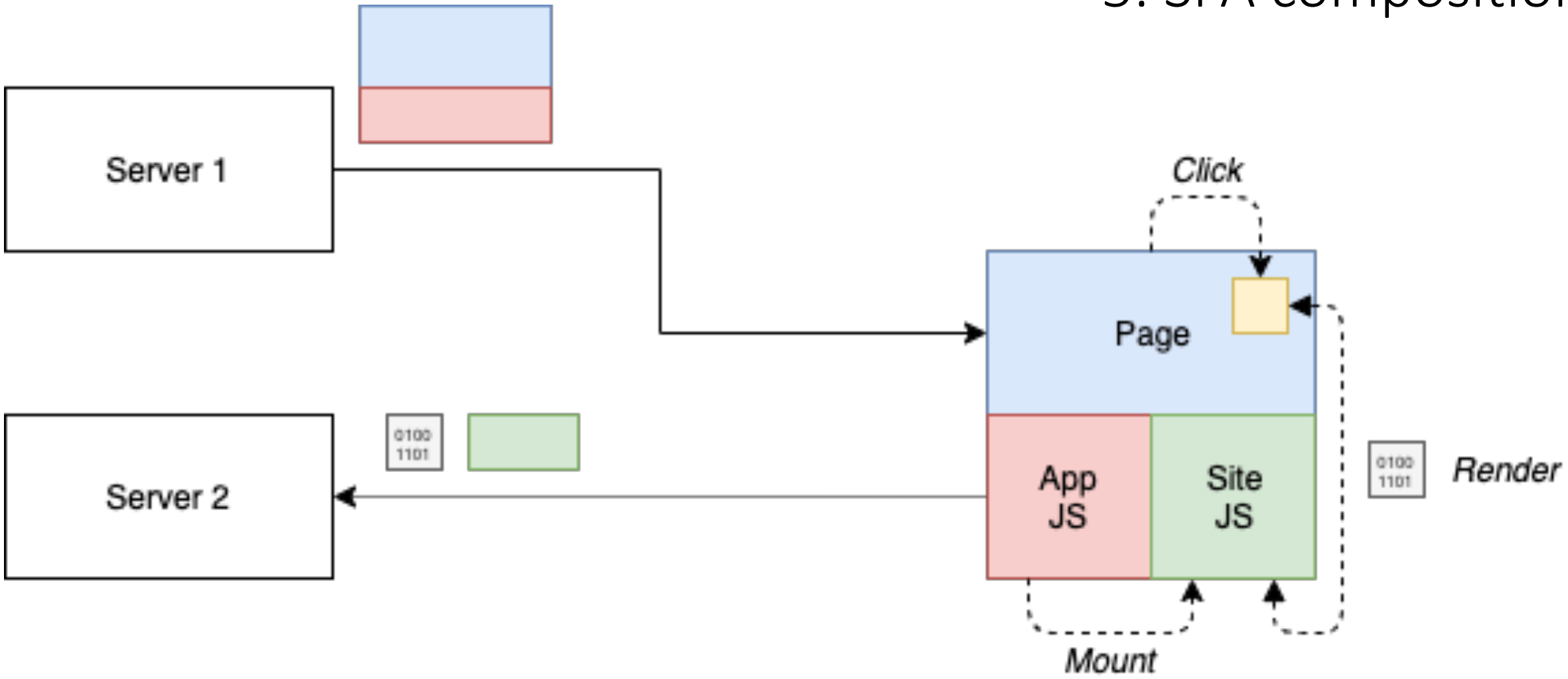


Tampereen yliopisto
Tampere University

Alternative architectures

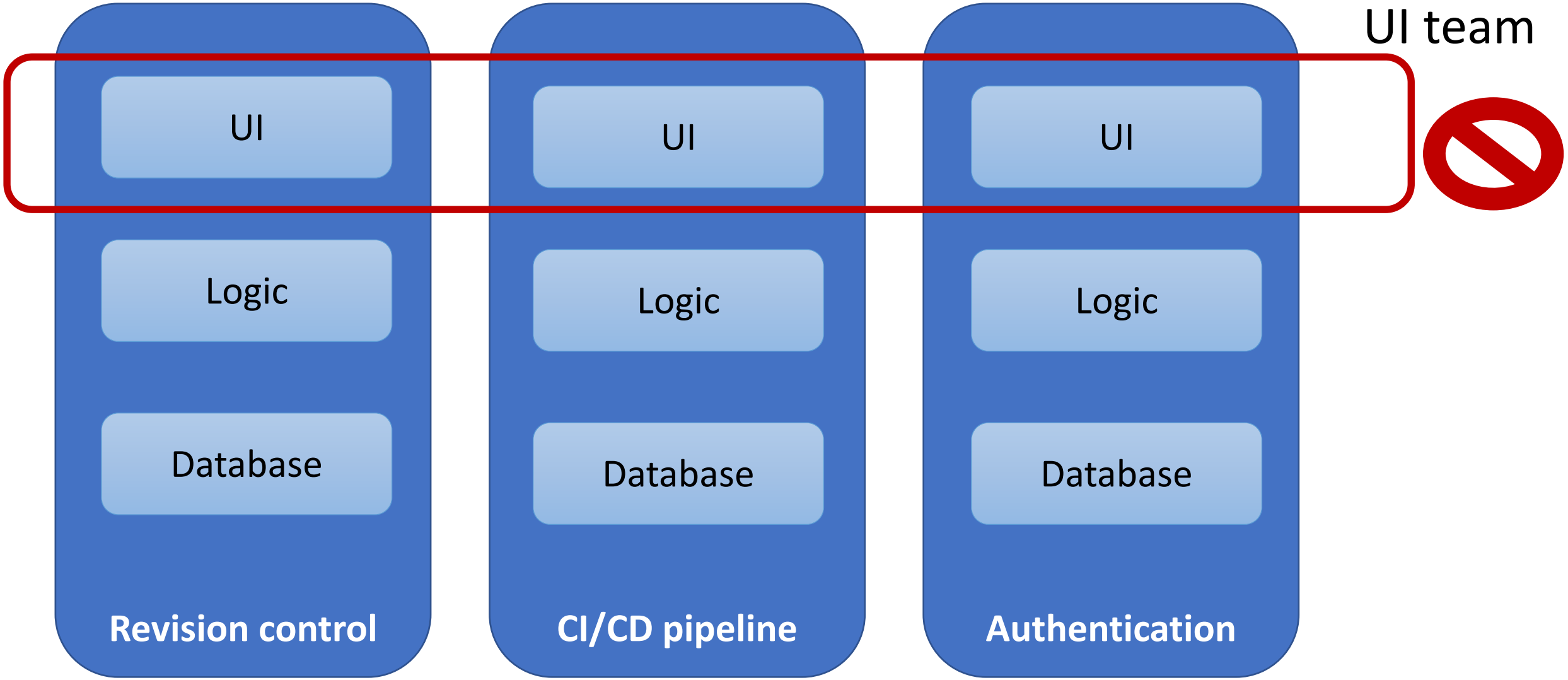
(from https://morioh.com/p/ee1b48c9de16)

5. SPA composition



Tampereen yliopisto
Tampere University

Organization and process issues



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

Reminder

https://thenewstack.io/10-key-attributes-of-cloud-native-applications/

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