Lecture 06

Rest of DevOps, CD, Deployment, Dependency Management

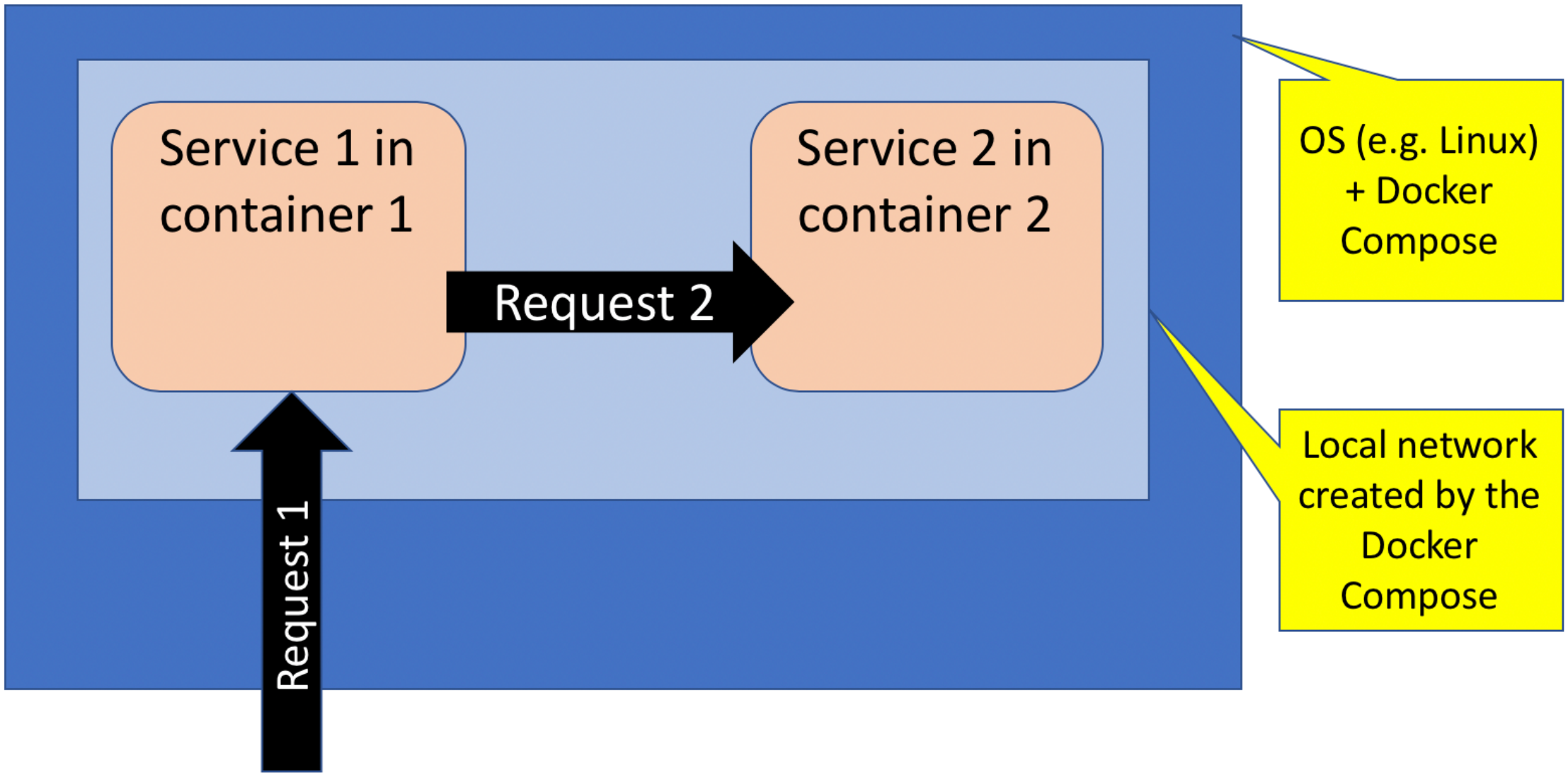
Kari Systä

29.09.2020

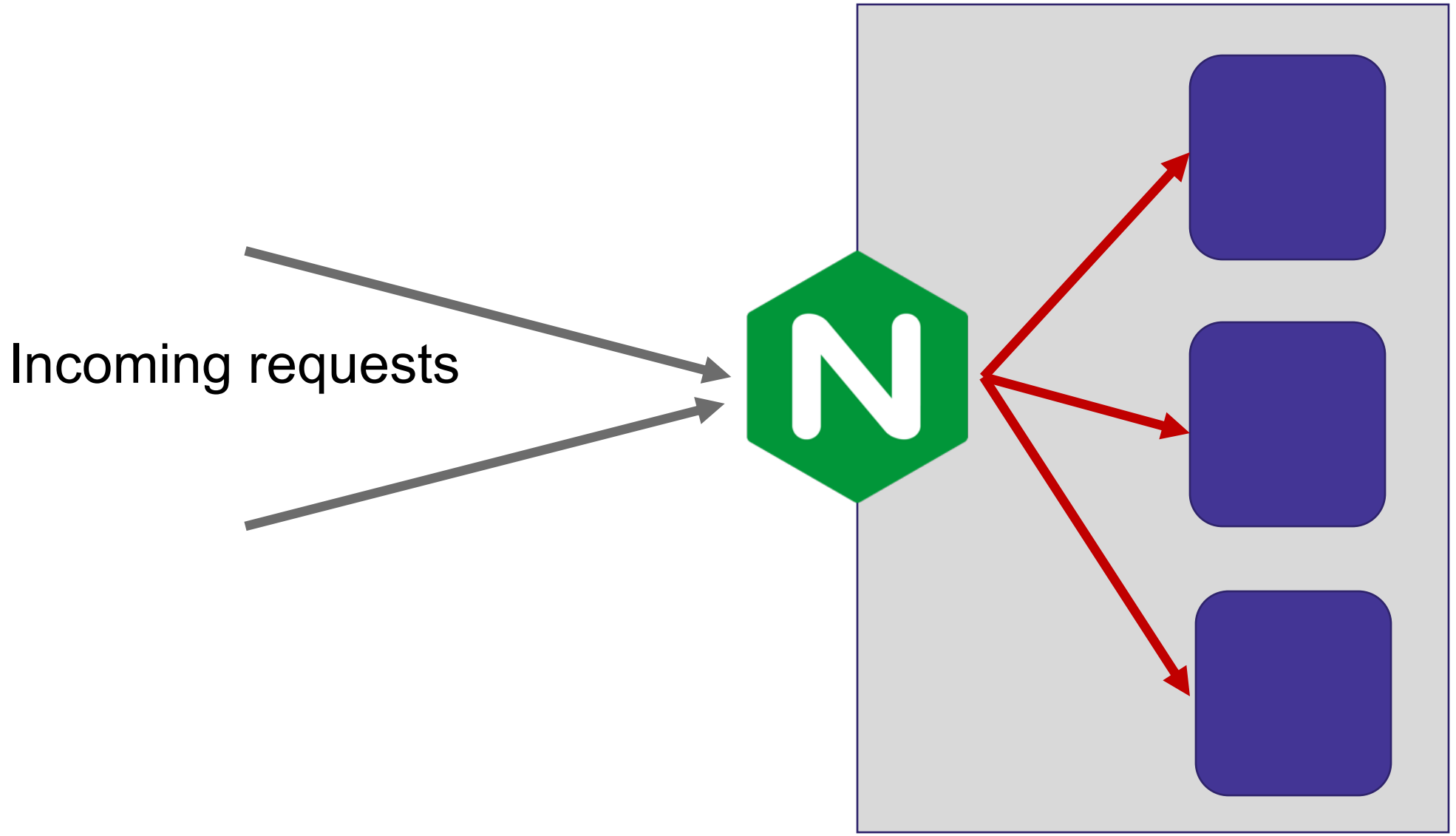
Tampereen yliopisto
Tampere University

Schedule for coming weeks

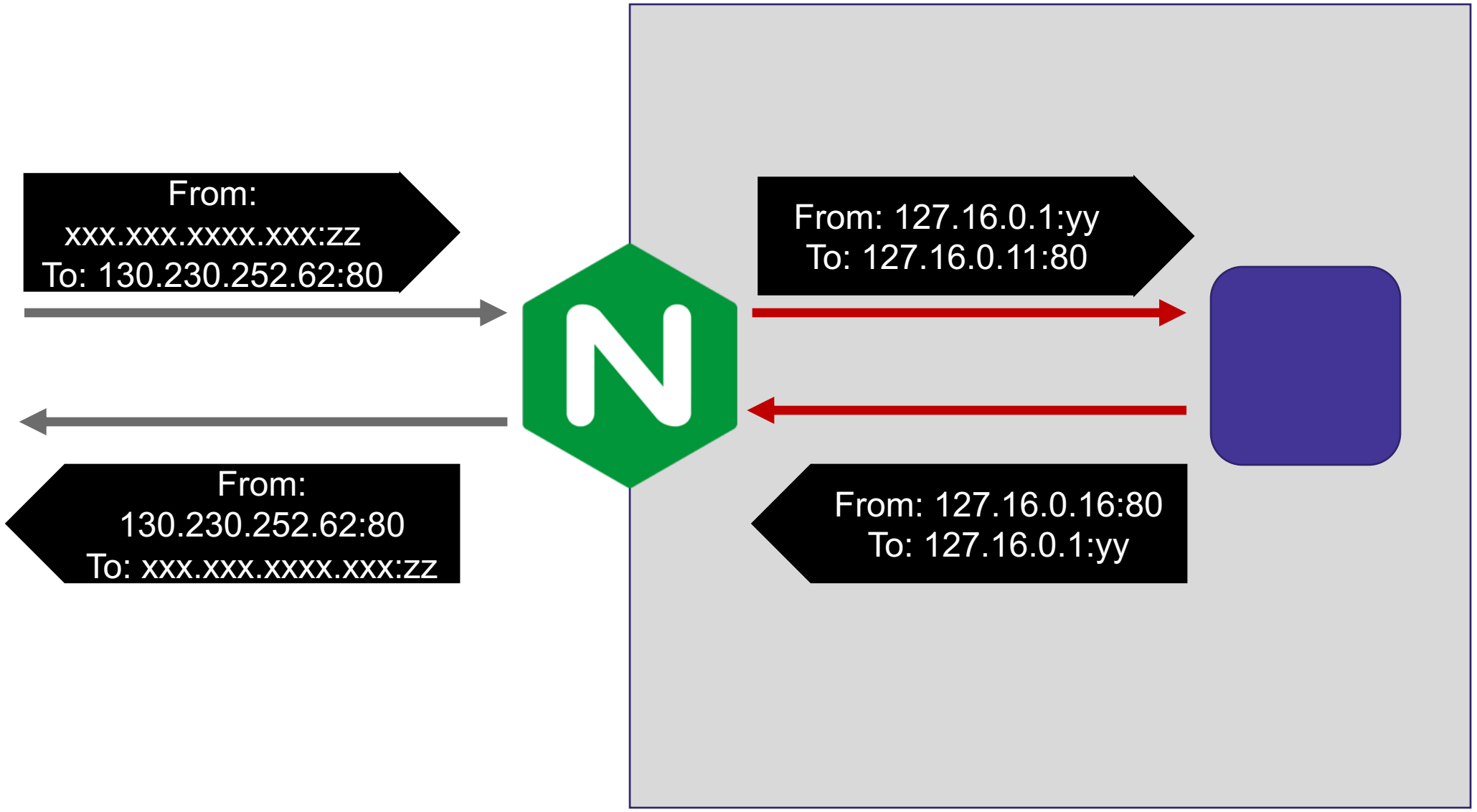
Week	Lecture	Plussa exercises (deadlines)
4 / 38	15.09 Continuous deployment, what and why	17.09 Docker compose e. opens
5 / 39	22.09 Continuous deployment, tools and techniques	
6 / 40	29.09 CD; Issues on cloud-SW: isolation, dependency management etc	01.10 Docker compose e. closes
7/ 41	06.10 Cloud-native architectures, part 1.	05.10 Next exercise opens
X/42	Exam week	
8/43	20.10 Cloud-native architectures, part 1.	19.10 Next exercise closes 20.10 Project instructions opens


Service 1 in container 1
Service 2 in container 2
Request 2
Request 1
OS (e.g. Linux) + Docker Compose
Local network created by the Docker Compose

Yet another technology example: NGINX


Incoming requests

Get
130.230.252.62

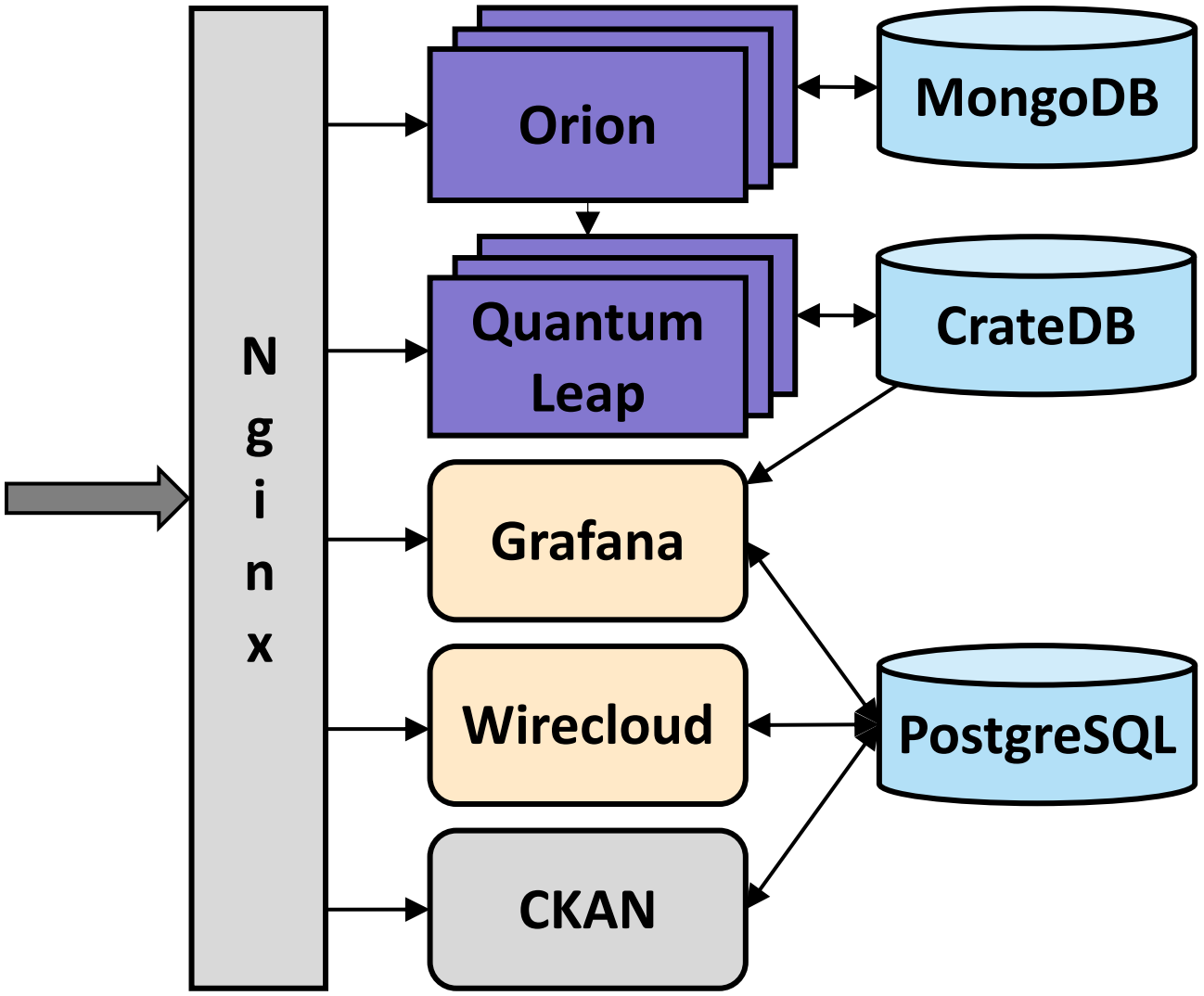


But NGINX is a lot more

- Load balancer
- Content cache
- Firewall
- Authentication service
- Reverse proxy
- Network Address Translator (NAT)

Tampereen yliopisto
Tampere University

FIWARE platform architecture



FIWARE Core Component

Dashboard Component

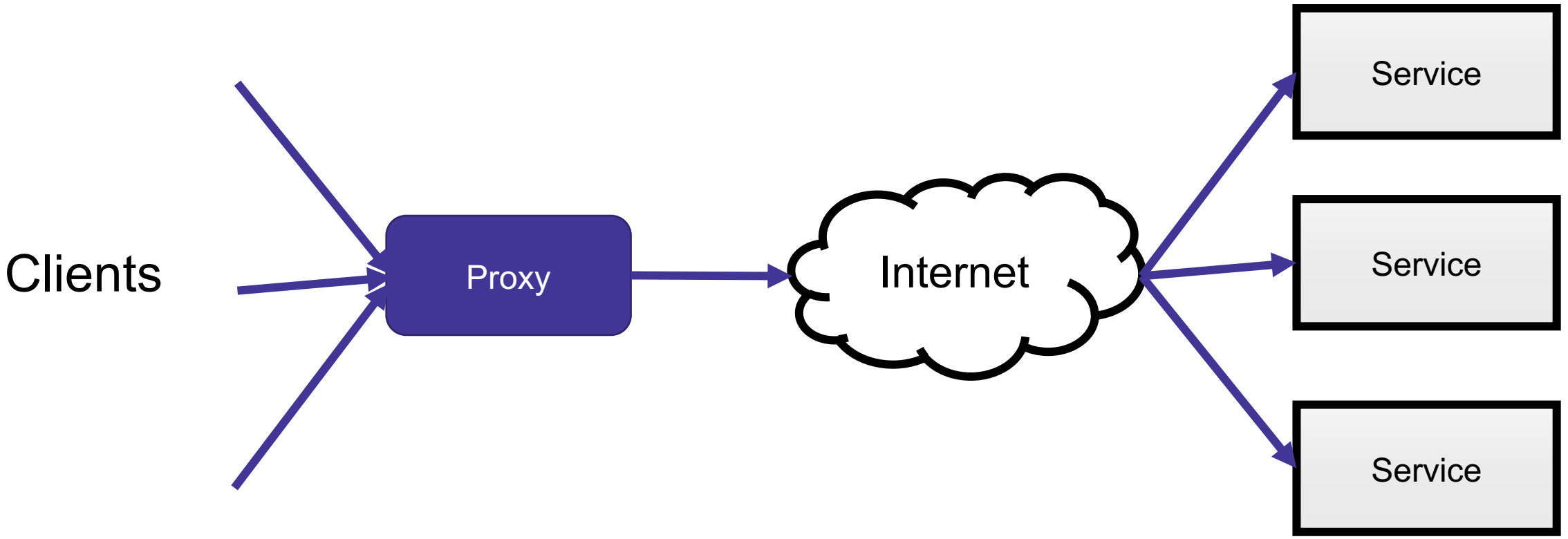
Data Management Component

Database

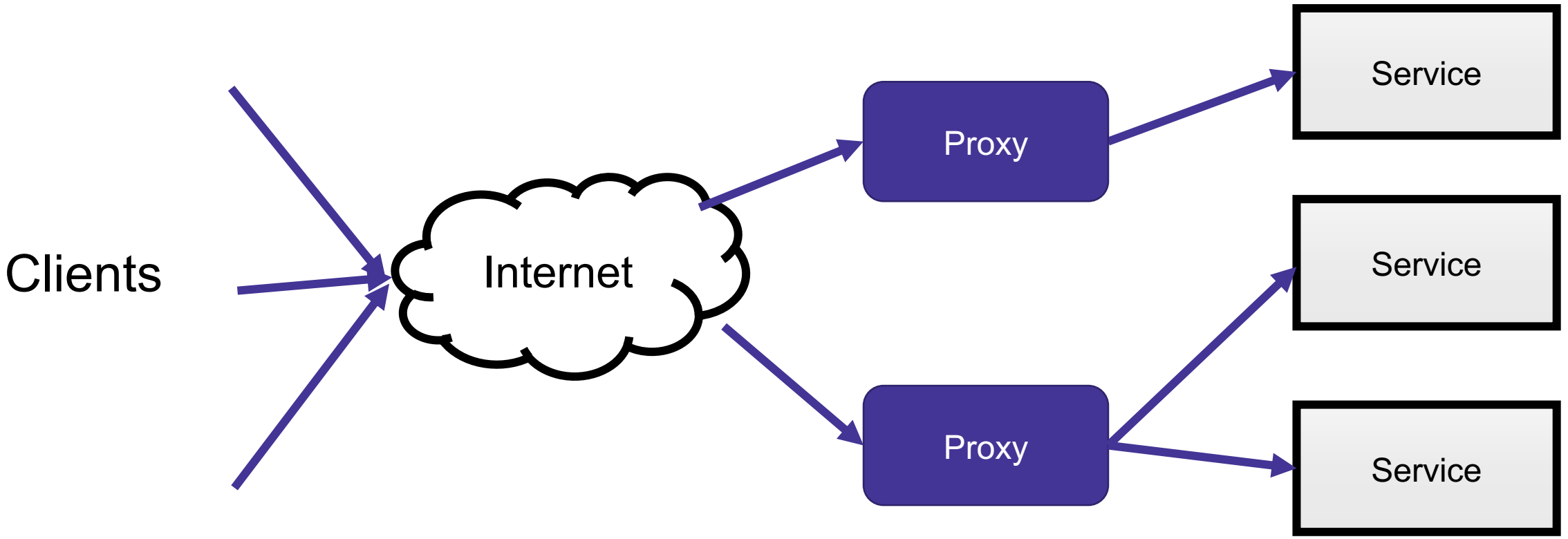
Access control, proxy server

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

Forward proxy



Reverse proxy



Tampereen yliopisto
Tampere University

Maturity models in software engineering

- The first welknown was
  - Capability Maturity Model developed by Software Engineering Institute at Carnegie Mellon University in 1986
  - Five levels:
    - *Initial* (chaotic, ad hoc, individual heroics) - the starting point for use of a new or undocumented repeat process.
    - *Repeatable* - the process is at least documented sufficiently such that repeating the same steps may be attempted.
    - *Defined* - the process is defined/confirmed as a standard business process
    - *Capable* - the process is quantitatively managed in accordance with agreed-upon metrics.
    - *Efficient* - process management includes deliberate process optimization/improvement.
- Practical meaning may be questioned, but there has been many followers.

Tampereen yliopisto
Tampere University

Maturity models

(https://developer.ibm.com/urbancode/docs/continuous-delivery-maturity-model/)

- **Base**: The base level is enough to “be on the model”. The team has left fully manual processes behind.
- **Beginner**: At the beginner level, the team is trying to adopt some ECD practices in earnest but is still performing them at a rudimentary level.
- **Intermediate**: Practices are somewhat mature and are delivering fewer errors and more efficiency. For many teams, Intermediate practices may be sufficient.
- **Advanced**: The team is doing something well beyond what most of the rest of the industry and is seeing a great deal of efficiency and error prevention as a result.
- **Extreme**: Elements within the Extreme category are ones that are expensive to achieve but for some teams should be their target. Put another way, most organizations would be crazy to implement them, while this minority would be crazy to not implement them.

Tampereen yliopisto
Tampere University

Another

(https://www.infoq.com/articles/Continuous-Delivery-Maturity-Model/)

Base … started to prioritize work in backlogs, have some process defined which is rudimentarily documented and developers are practicing frequent commits into version control.

Beginner … teams stabilize over projects and the organization has typically begun to remove boundaries by including test with development. Multiple backlogs are naturally consolidated into one per team and basic agile methods are adopted ….

Intermediate … extended team collaboration when e.g. DBA, CM and Operations are beginning to be a part of the team or at least frequently consulted by the team. Multiple processes are consolidated and all changes, bugs, new features, emergency fixes, etc, follow the same path to production. Decisions are decentralized to the team and component ownership…

Advanced … team will have the competence and confidence it needs to be responsible for changes all the way to production. Continuous improvement mechanisms are in place … releases of functionality can be disconnected from the actual deployment, which gives the projects a somewhat different role. A project can focus on producing requirements for one or multiple teams and when all or enough of those have been verified and deployed to production the project can plan and organize the actual release to users separately.

Expert …some organizations choose to make a bigger effort and form complete cross functional teams that can be completely autonomous. With extremely short cycle time and a mature delivery pipeline, such organizations have the confidence to adopt a strict roll-forward only strategy to production failures.

Tampereen yliopisto
Tampere University

Another

(https://www.infoq.com/articles/Continuous-Delivery-Maturity-Model/)

Base … one or more legacy systems of monolithic nature in terms of development, build and release. Many organizations at the base maturity level will have a diversified technology stack but have started to consolidate … to get best value from the effort spent on automation.

Beginner … the monolithic structure of the system is addressed by splitting the system into modules … this will also naturally drive an API managed approach to describe internal dependencies and also influence applying a structured approach to manage 3rd party libraries … importance of applying version control to database changes will also reveal itself.

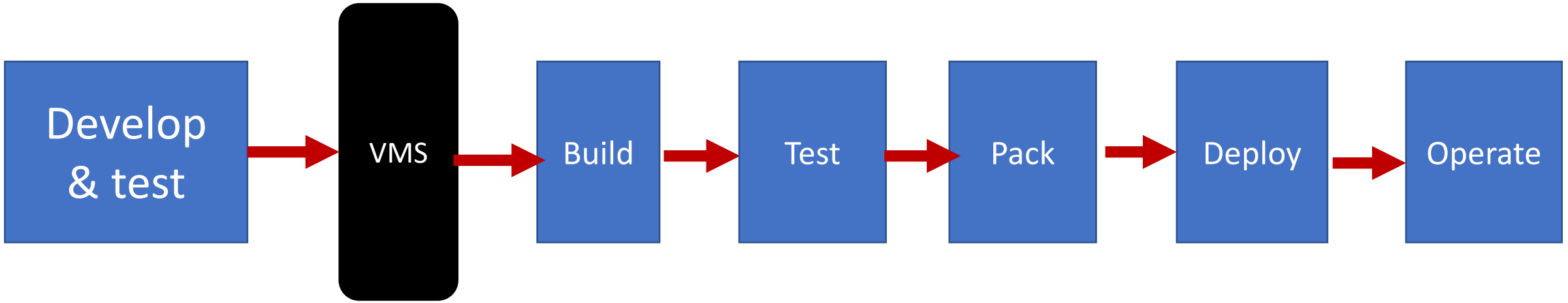
Intermediate. … a solid architectural base for continuous delivery … feature hiding for the purpose of minimizing repository branching to enable true continuous integration. … modularization will evolve into identifying and breaking out modules into components that are self-contained and separately deployed. … start migrating scattered and ad-hoc managed application and runtime configuration into version control and treat it as part of the application just like any other code.

Advanced. … split the entire system into self contained components and adopted a strict api-based approach to inter-communication so that each component can be deployed and released individually … every component is a self-contained releasable unit with business value, you can achieve small and frequent releases and extremely short release cycles..

Expert … some organizations will evolve the component based architecture further and value the perfection of reducing as much shared infrastructure as possible by also treating infrastructure as code and tie it to application components. The result is a system that is totally reproducible from source control, from the O/S and all the way up to application. …

Tampereen yliopisto
Tampere University

Simplified pipeline



C++

Python

Tampereen yliopisto
Tampere University

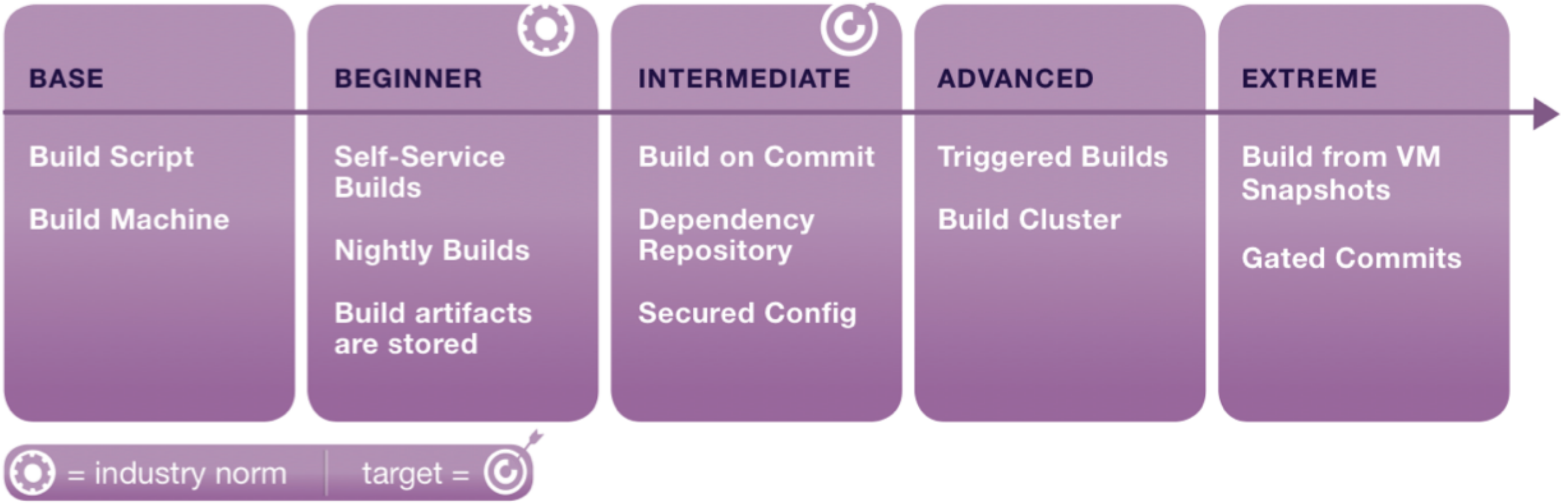
Build – which tools you know ?

- Make
  - Old
  - Declarative
  - Hard to debug
- Ant
  - Designed for Java
  - Based on XML-based configuration language
- Maven

Tampereen yliopisto
Tampere University

https://developer.ibm.com/urbancode/docs/continuous-delivery-maturity-model/

BUILDING



Tampereen yliopisto
Tampere University

Testing

- Automate, automate, automate
- Know any tools?

Business

AUTOMATED (functional acceptance tests)	MANUAL Showcase Usability testing Exploratory testing
Unit tests Integration tests System tests AUTOMATED	Nonfunctonal acceptance tests MANUAL/AUTOM.

Support coding

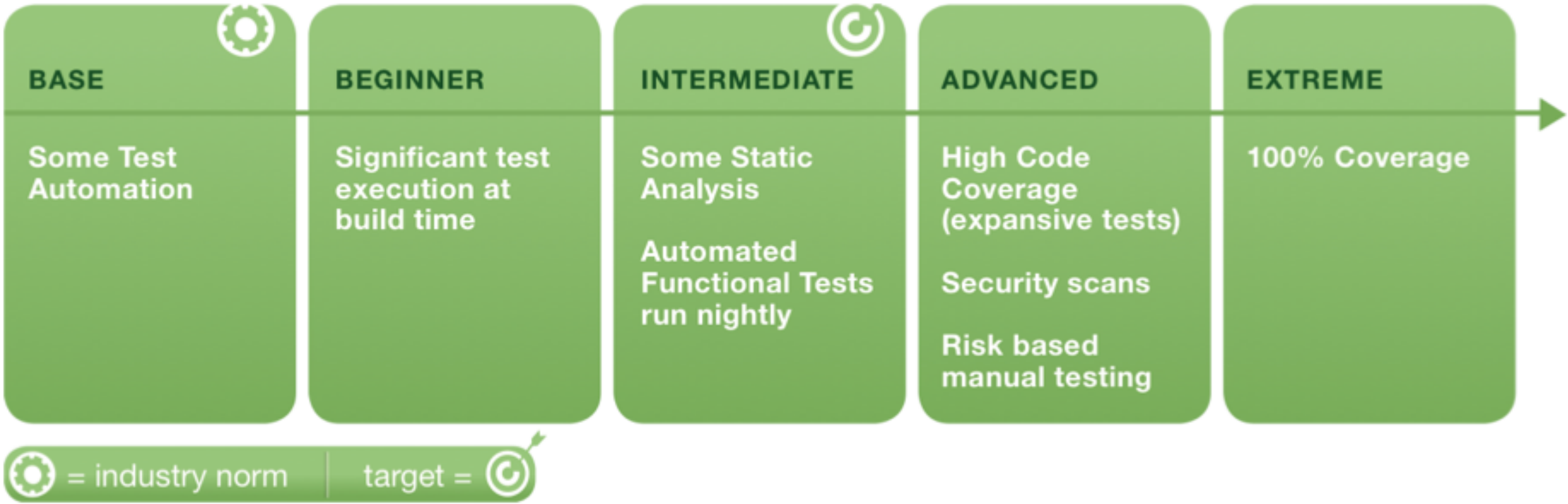
Critique project

Technology

Tampereen yliopisto
Tampere University

https://developer.ibm.com/urbancode/docs/continuous-delivery-maturity-model/

TESTING



Tampereen yliopisto
Tampere University

Pack

- Binaries
- Required Libraries
- Runtime (e.g. Python)
- Manifest file
- Help files
- Localization stuff
- …

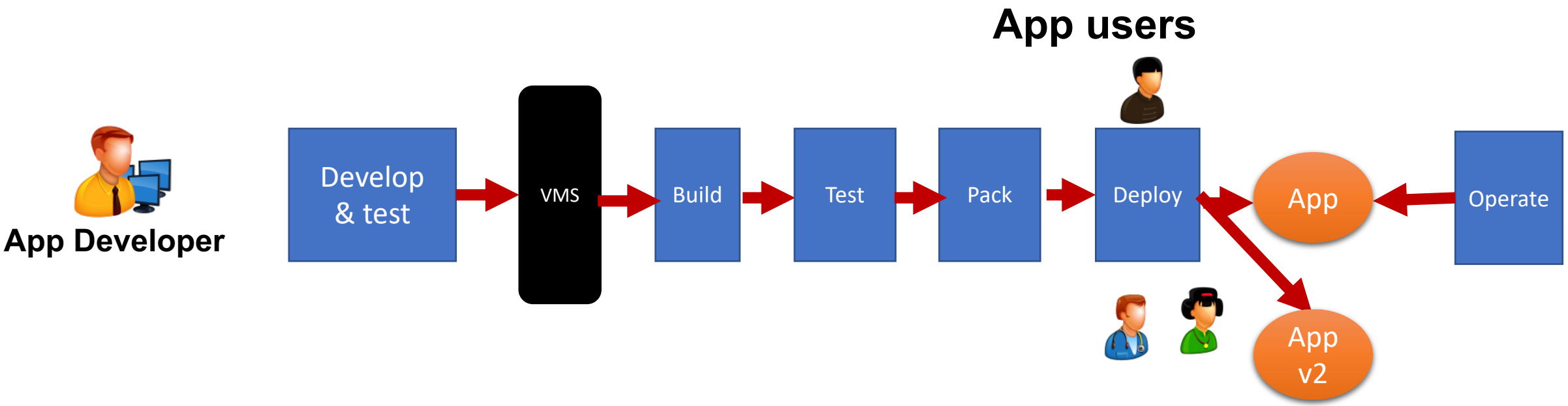
- Examples
  - Windows install shield
  - Java JAR
  - Android APK

What else comes to mind?

Deployment/Delivery

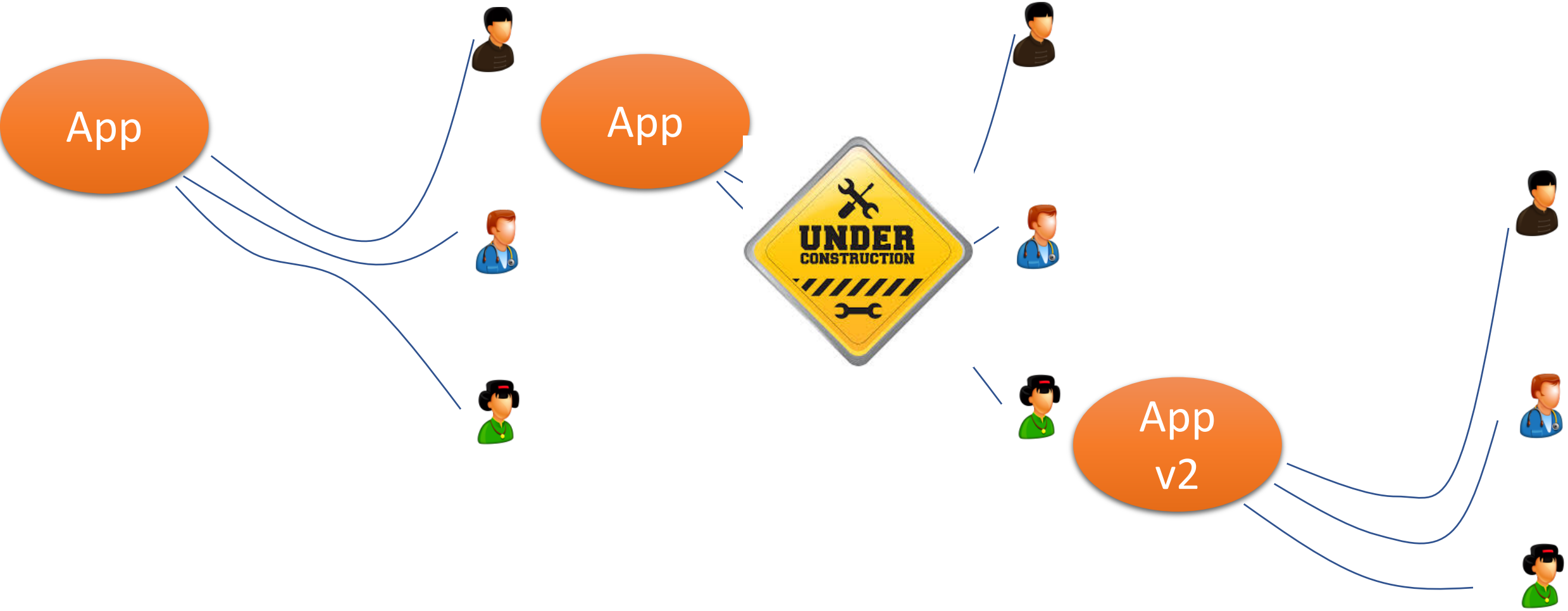
- Humble and Farley write
  - Creating the infrastructure (hardware, networking, middleware, …)
  - Installing correct version of the application
  - Configuring the application with its data

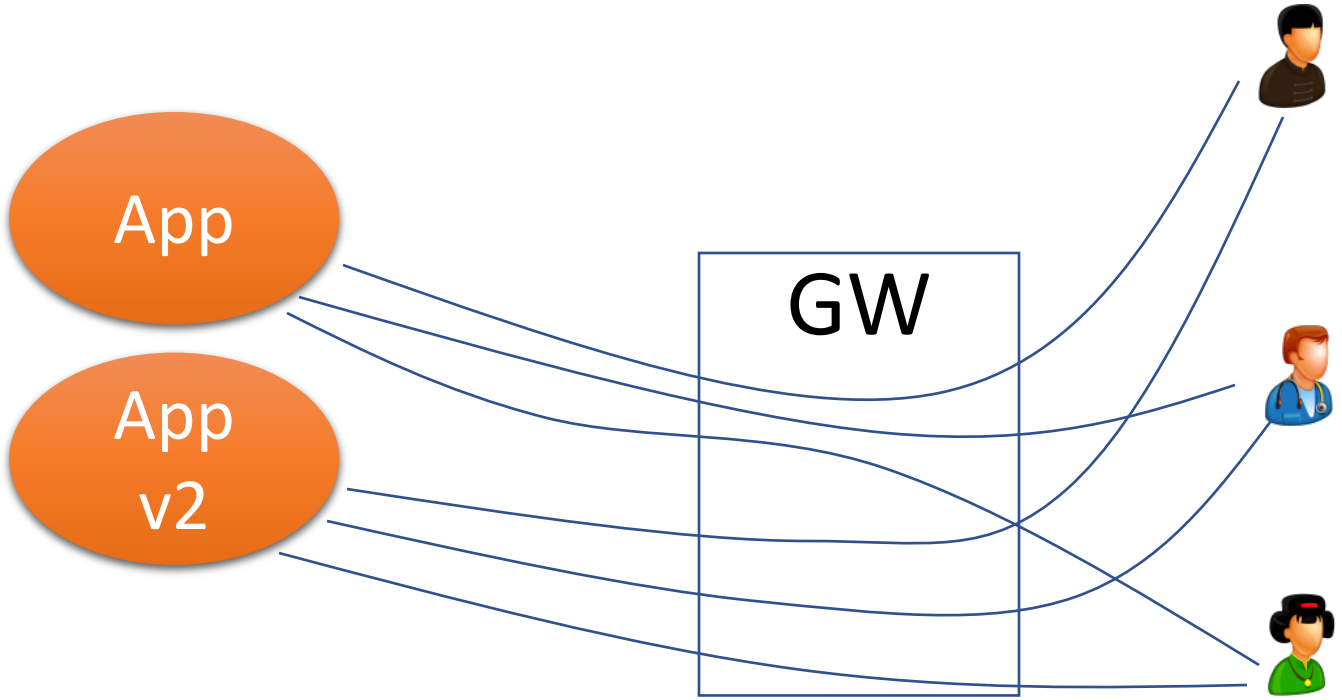
- Sounds a bit difficult?
- Text written before 2011
- First Docker release 2013


App users
App Developer
Develop & test
VMS
Build
Test
Pack
Deploy
App
Operate
App v2

Tampereen yliopisto
Tampere University

A possible strategy to deploy a new version?



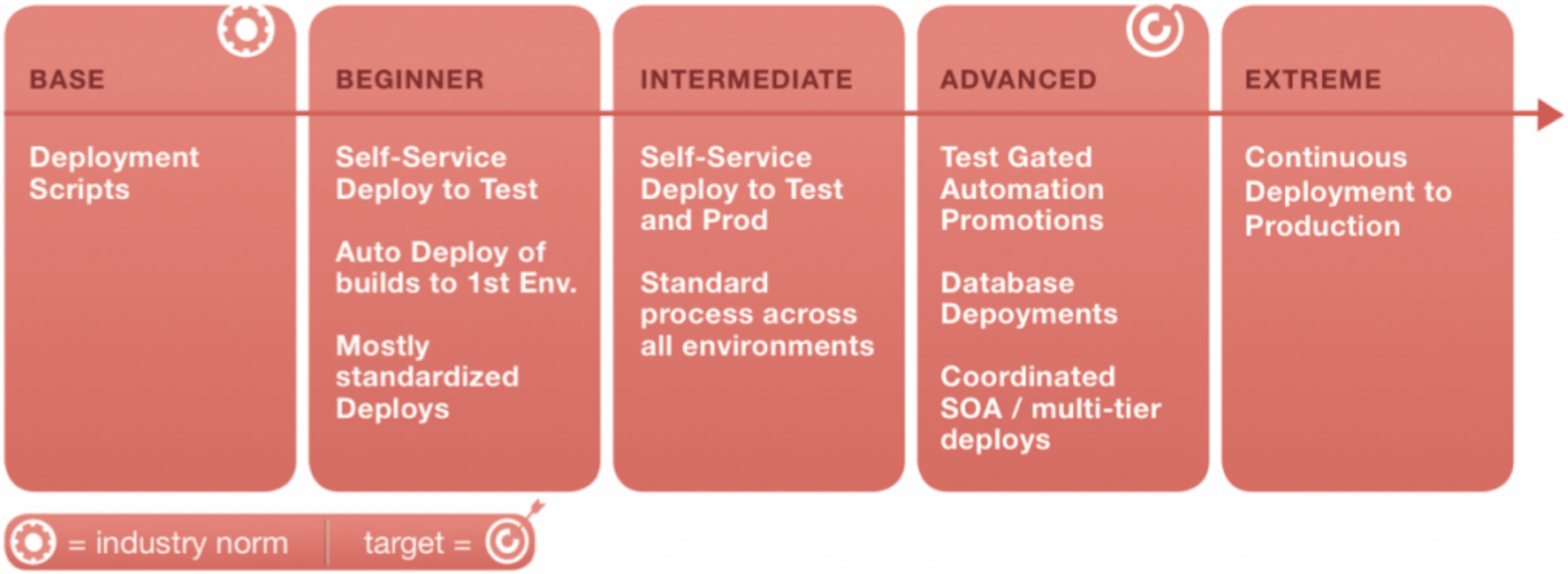


Problems & issues?

Tampereen yliopisto
Tampere University

https://developer.ibm.com/urbancode/docs/continuous-delivery-maturity-model/

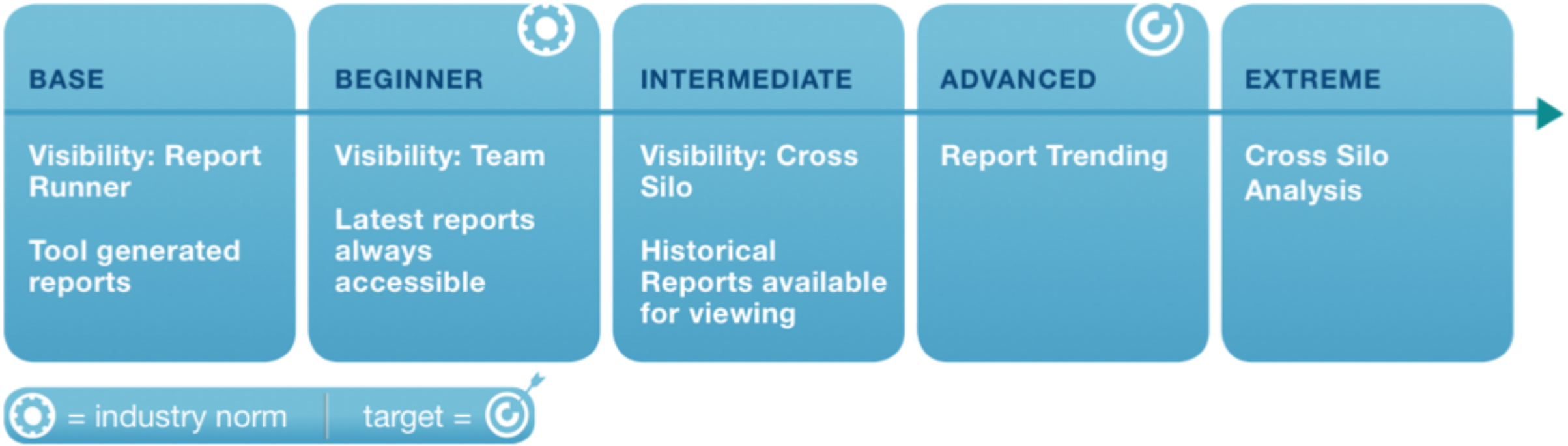
DEPLOYING



Tampereen yliopisto
Tampere University

https://developer.ibm.com/urbancode/docs/continuous-delivery-maturity-model/

REPORTING



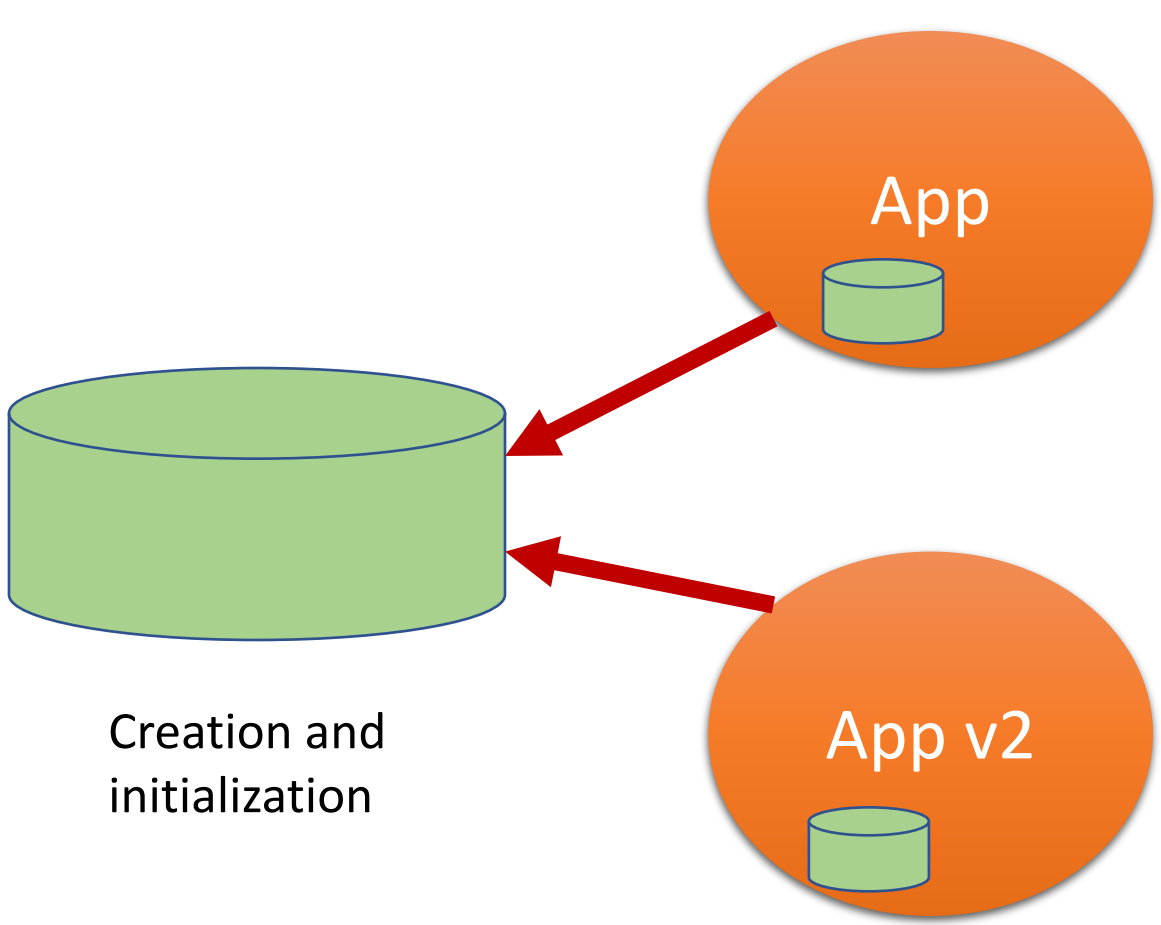
Tampereen yliopisto
Tampere University

Deployment strategies

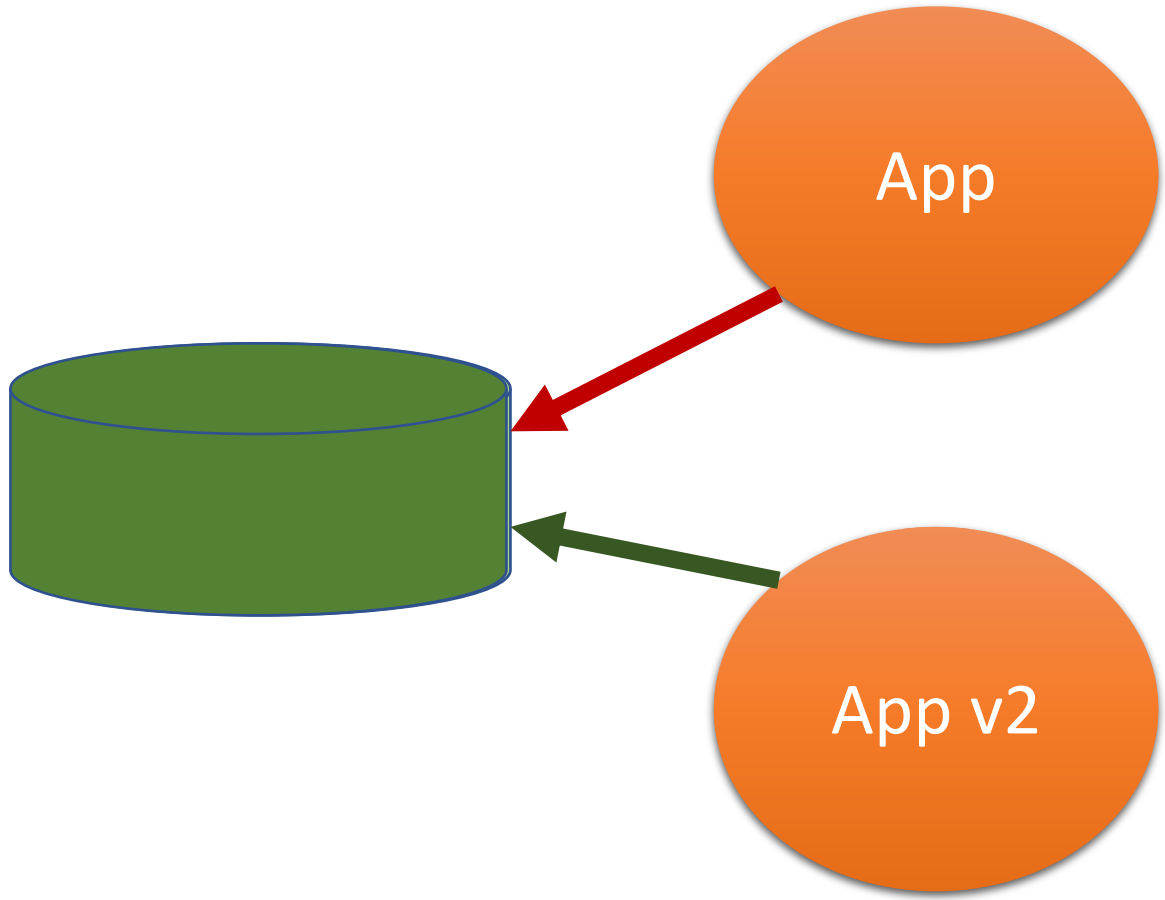
- Basic Deployment (aka Suicide) (https://harness.io/2018/02/deployment-strategies-continuous-delivery/) all nodes are updated at the same time
- Rolling Deployment (https://harness.io/2018/02/deployment-strategies-continuous-delivery/) nodes are updated incrementally,
- BlueGreenDeployment (http://martinfowler.com/bliki/BlueGreenDeployment.html) uses a router of incoming traffic as the tool. In this approach the new version (called green) is set up in parallel with the current (blue). When new (green) is ready, the router is switched to new (green) and blue is left as a backup. If something goes wrong with new, the router can be switched back to old - that means easy “rollback”.
- Canary Releases (http://martinfowler.com/bliki/CanaryRelease.html) implements the deployment incrementally. In this case the router first directs only part of the customers to the new version. If feedback is is good, the other customers are moved to new version, too

Tampereen yliopisto
Tampere University

How about the data?



Creation and initialization



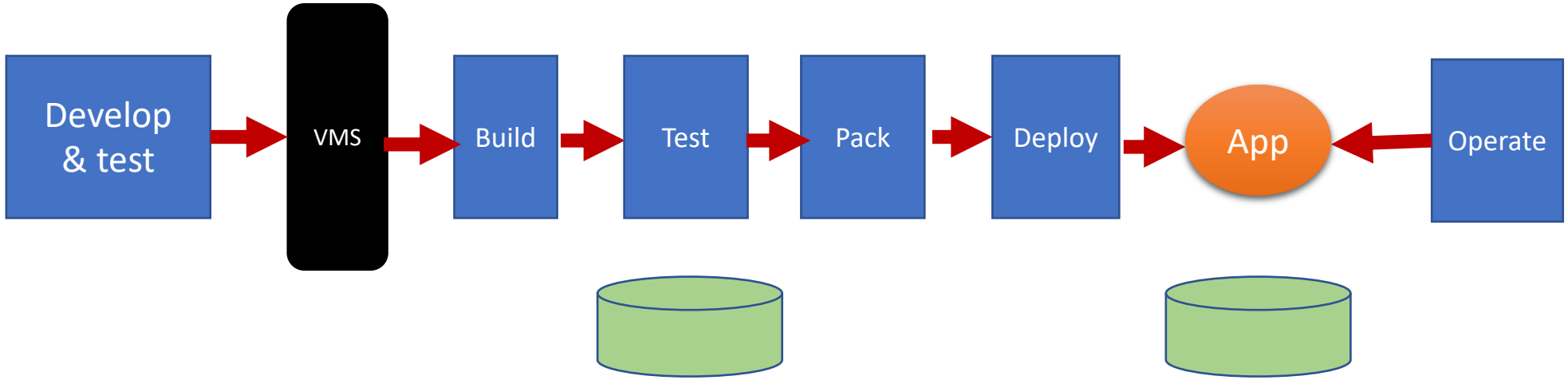
Tampereen yliopisto
Tampere University

Data migration

- Versions of the data bases
- Data migration scripts are needed.
- Rollback need to be possible

Tampereen yliopisto
Tampere University

How about test data base?



Tampereen yliopisto
Tampere University

Sidenote: DataOps

- *DataOps is a set of practices, processes and technologies that combines an integrated and process-oriented perspective on data with automation and methods from agile software engineering to improve quality, speed, and collaboration and promote a culture of continuous improvement.*

- DataOps includes the methods, tools, and will on the organizational level that have as a goal helping data and analytics team(s) better understand the data requirements of the whole organization. Achieving this leads to the needed data being provided when needed, in suitable format, and at quicker pace to those who need it.
- As a whole, DataOps seeks to make the analytics workflow more agile. For this, a will and methods are very likely needed, and quite likely using or developing new software tools. Where new software tools are required is likely automating and monitoring the data delivery pipeline as much as possible.

Tampereen yliopisto
Tampere University

DataOps

Roles and responsibilities (from DataOps Cookbook):

- Data Engineer - databases, programming
  - responsibilities: storing data, schemas
- DataOps Engineer - Agile development, DevOps, Statistical Process Control
  - responsibilities: orchestrating the analytics pipeline, automating quality
- Data Analyst - statistics, programming
  - responsibilities: reports, visualization
- Data Scientist - domain subject matter expert
  - responsibilities: models, algorithms

Tampereen yliopisto
Tampere University

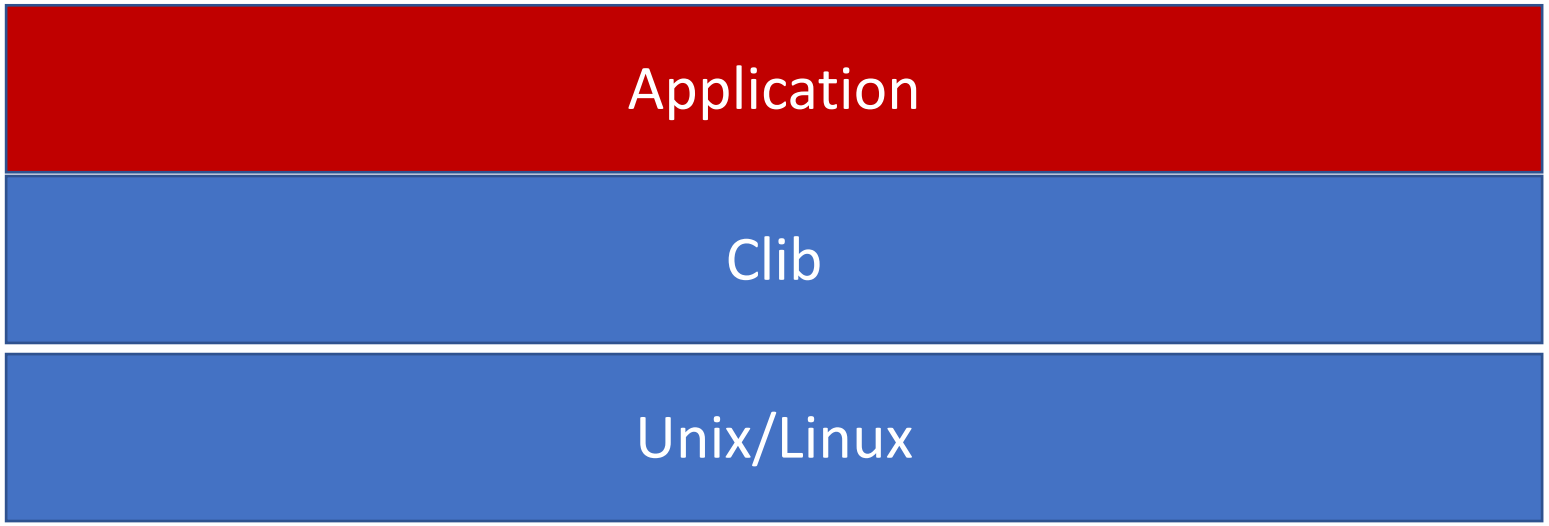
When testing becomes serious business

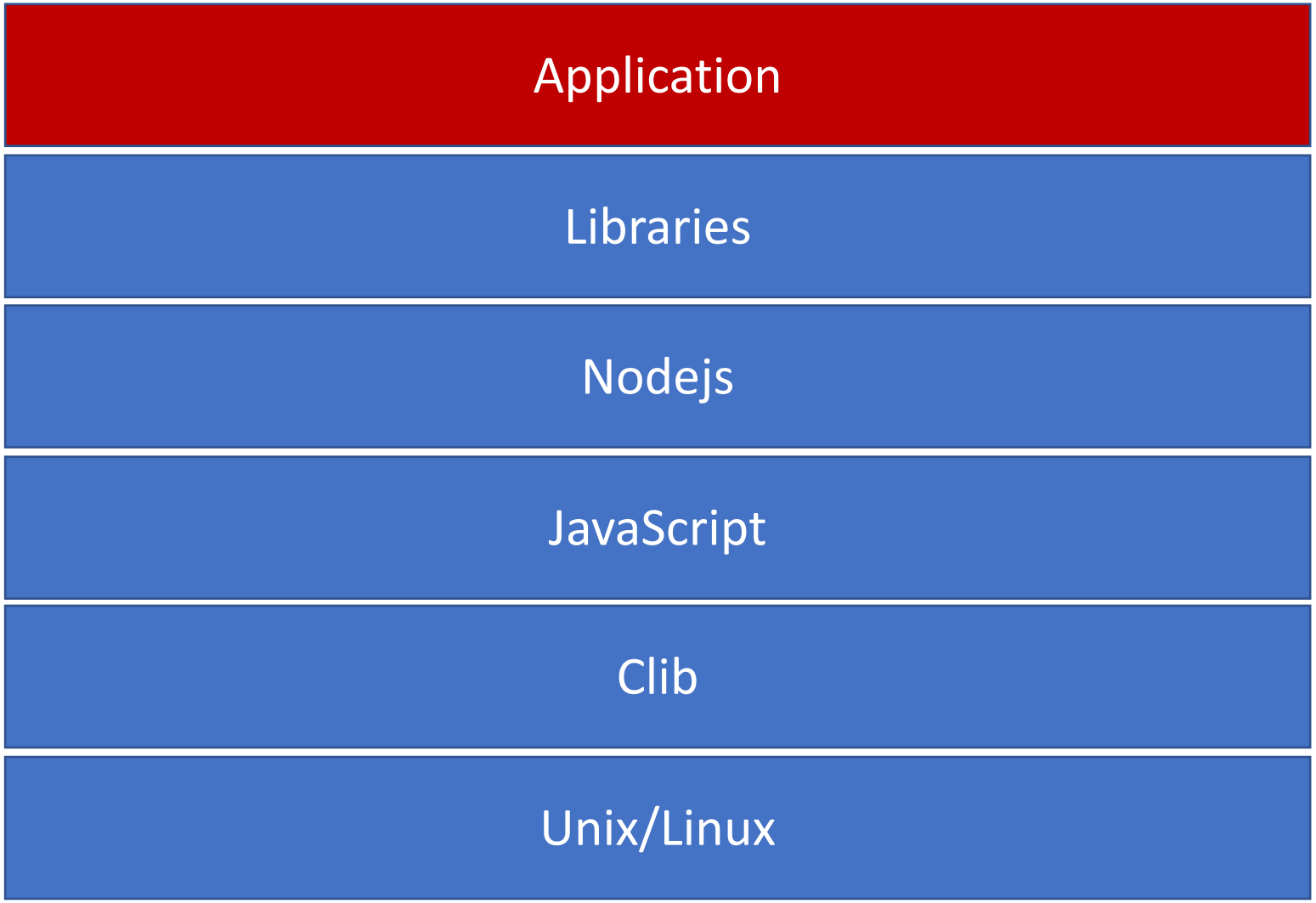
- Running tests start to take time

⇒Test case selection

⇒Test case prioritization

Tampereen yliopisto
Tampere University


Application
Clib
Unix/Linux


Application
Libraries
Nodejs
JavaScript
Clib
Unix/Linux

Tampereen yliopisto
Tampere University

Remember container use case example

- Your application needs
  - Certain version of nodejs
  - Set of libraries (certain versions)
  - Mongo database

- Your system has
  - Wrong version of nodejs
  - Mongo serving another application

- Solution
  - Create a docker image (container)
  - Install the image
  - Run the image

Tampereen yliopisto
Tampere University

2011: Base images used in exercise 4

- node:10 20
- node:10-alpine 3
- node:10.15.3-stretch 1
- node:10.16.3-alpine 1
- node:11 1
- node:11-alpine 1
- node:12 1
- node:12.2-alpine 1
- node:8.16.1-alpine 1
- node:8.16.1-jessie-slim 1
- node:alpine 1
- node:latest 2

- golang:alpine AS builder 1
- golang:latest 1
- python 1
- python:3 2
- python:3.6 2
- python:3.7-alpine 2
- python:latest 2
- ubuntu:latest 1

Tampereen yliopisto
Tampere University

2011: Base images used in exercise 4

- node:10 20
- node:10-alpi[illegible] 3
- node:10.15.3-stretch 1
- node:10.16.3-alpine
- node:11
- node:11-alpine
- node:12
- node:12.2-alpine
- node:8.16.1-alpine
- node:8.16.1-jessie-sli[illegible]
- node:alpine
- node:latest

- golang:alpine AS builder 1
- golang:latest 1
- python 1
- python:3 2

2
2
2
1

node:<version>
This is the defacto image. If you are unsure about what your needs are, you probably want to use this one. It is designed to be used both as a throw away container (mount your source code and start the container to start your app), as well as the base to build other images off of.

Tampereen yliopisto
Tampere University

- node:10 20
- node:10-alpine 3
- node:10.15.3-stretch 1
- node:10.16.3-alpine
- node:11
- node:11-alpine
- node:12
- node:12.2-alpine
- node:8.16.1-alpine
- node:8.16.1-jessie-slim
- node:alpine
- node:latest

- golang:alpine AS builder 1
- golang:latest 1
- python 1

node:<version>-slim

This image does not contain the common packages contained in the default tag and only contains the minimal packages needed to run node. Unless you are working in an environment where *only* the node image will be deployed and you have space constraints, we highly recommend using the default image of this repository.

- node:10
- node:10-alpine
- node:10.15.3
- node:10.16.3-alpine
- node:11
- node:11-alpine
- node:12
- node:12.2-alpine
- node:8.16.1-alpine
- node:8.16.1-jessie-slim
- node:alpine
- node:latest

node:<version>-alpine

This image is based on the popular Alpine Linux project, available in the alpine official image. Alpine Linux is much smaller than most distribution base images (~5MB), and thus leads to much slimmer images in general.

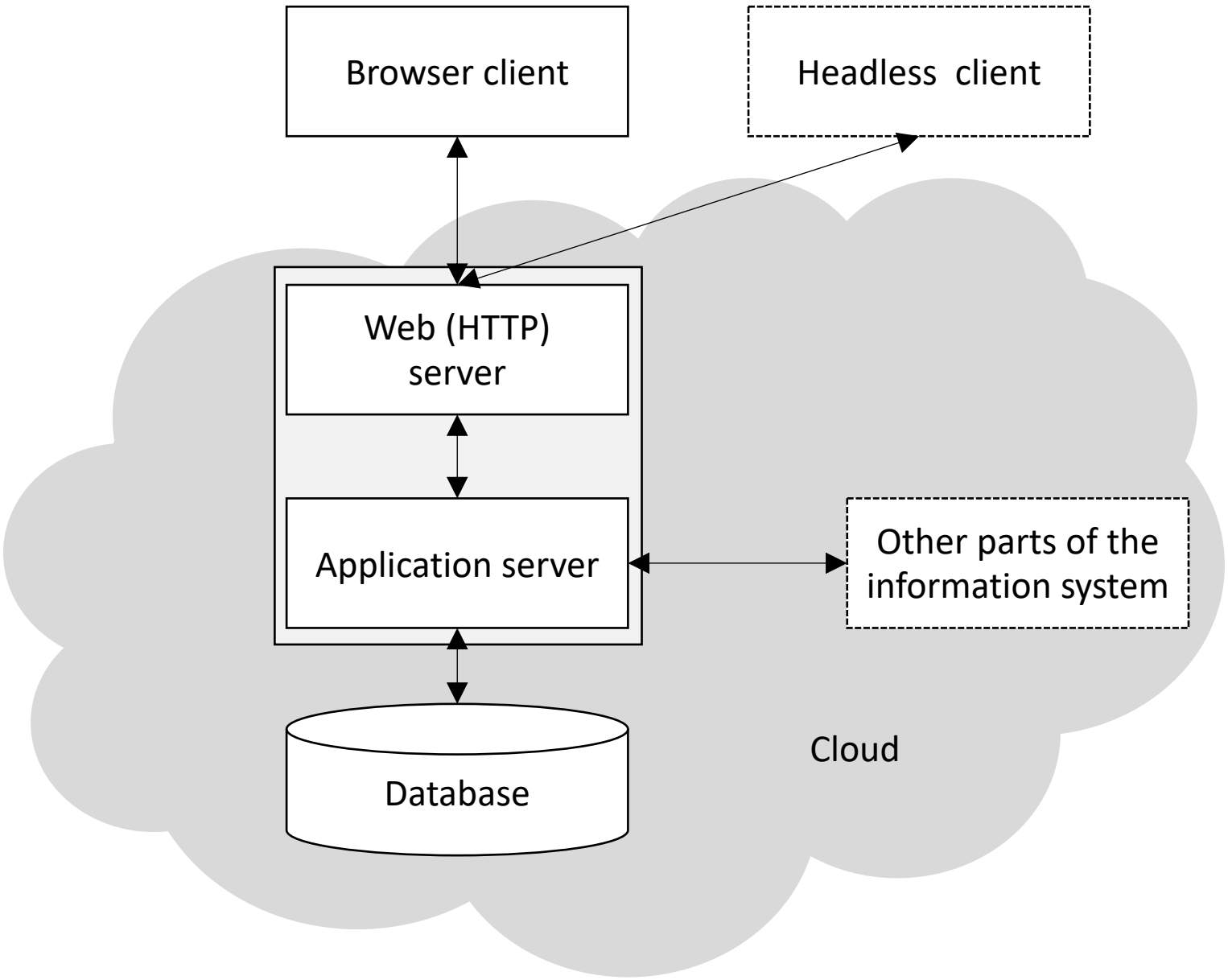
This variant is highly recommended when final image size being as small as possible is desired. The main caveat to note is that it does use musl libc instead of glibc and friends, so certain software might run into issues depending on the depth of their libc requirements. However, most software doesn't have an issue with this, so this variant is usually a very safe choice. See this Hacker News comment thread for more discussion of the issues that might arise and some pro/con comparisons of using Alpine-based images.

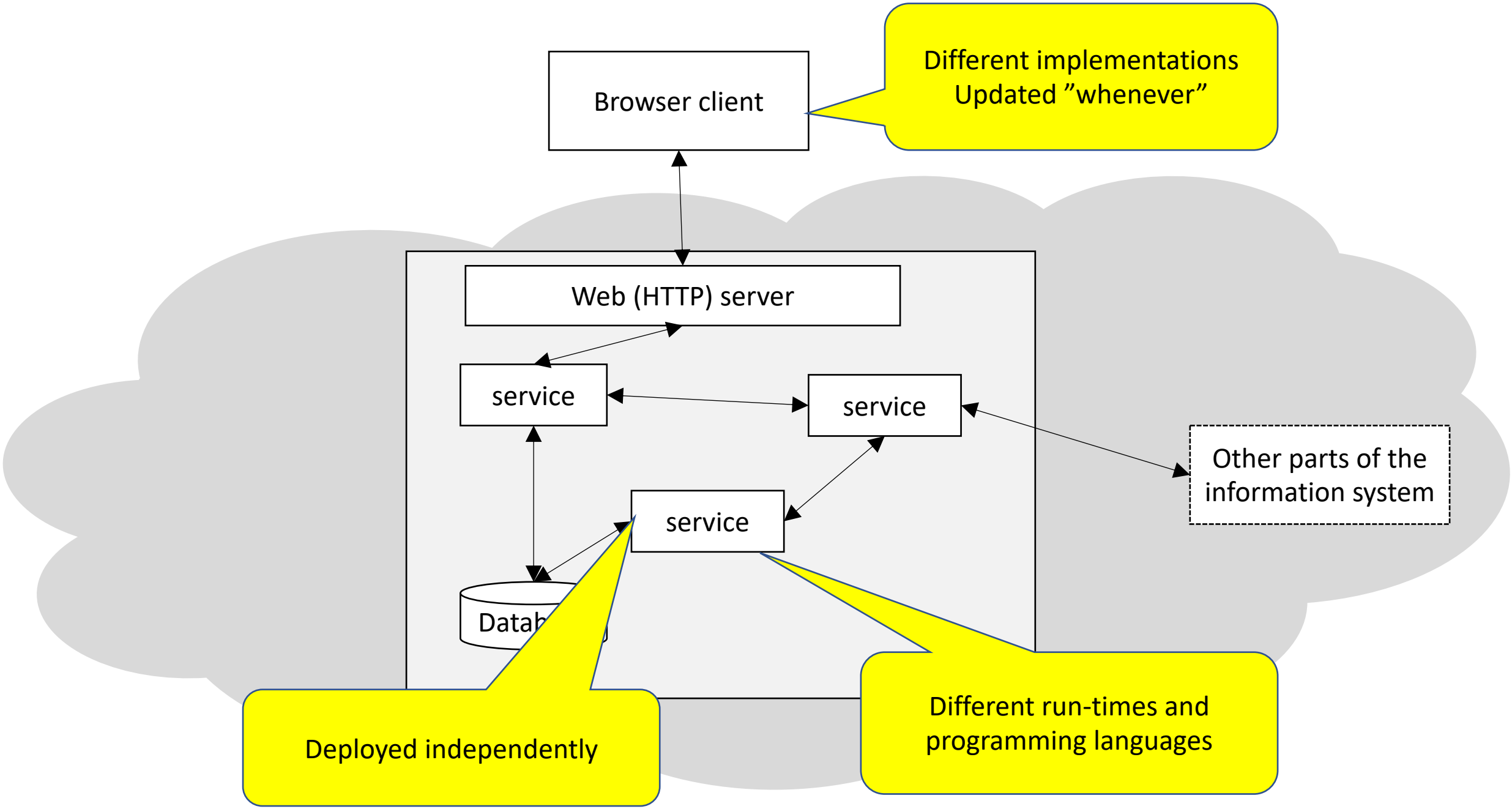
Tampereen yliopisto
Tampere University

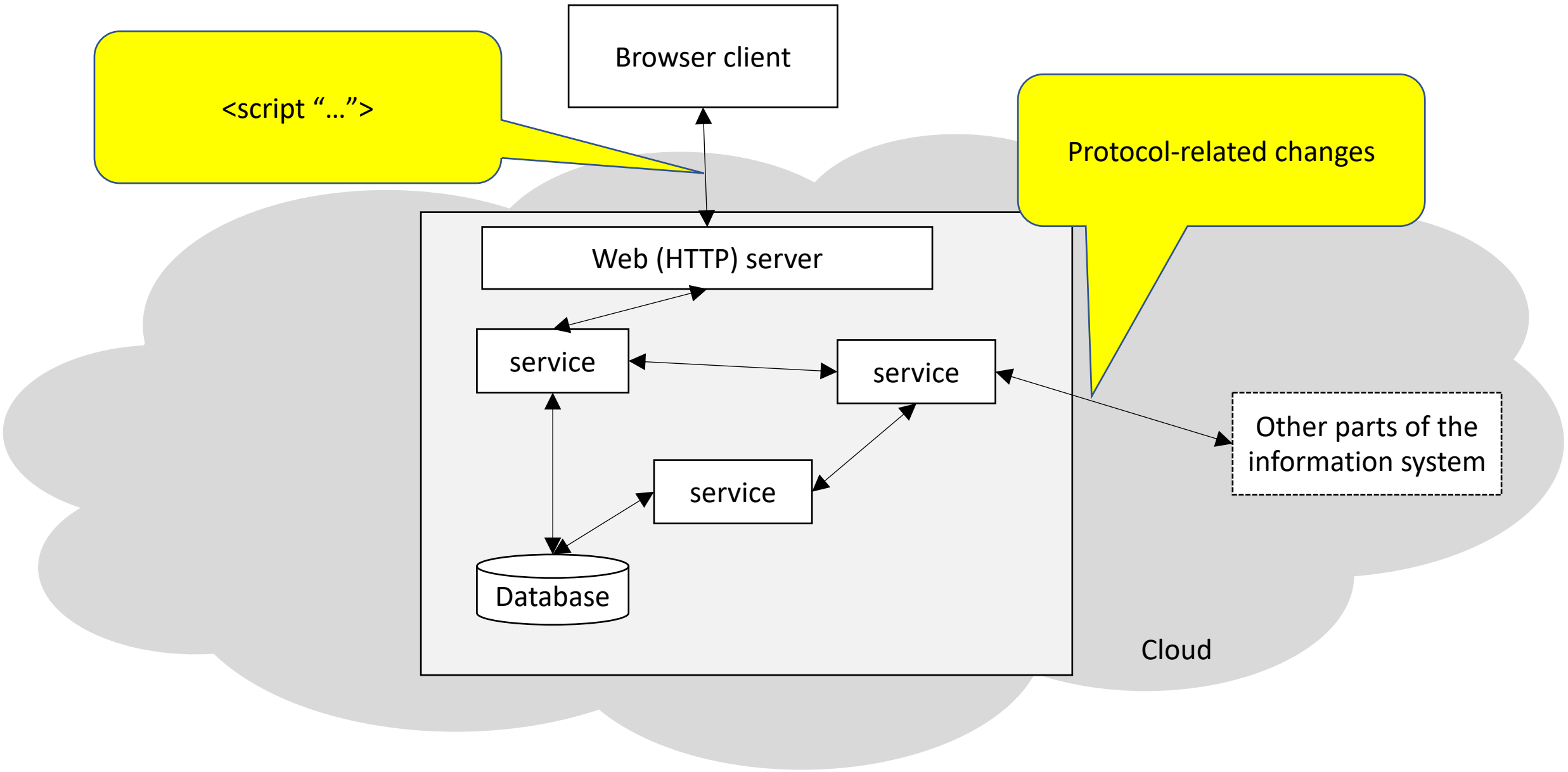
• node:10	20
• node:10-alpine	3
• node:10.15.3-stretch	1
• node:10.16.3-alpine	1
• node:11	2
• node:11-alpine	2
• node:12	2
• node:12.2-alpine	1
• node:8.16.1-alpine	
• node:8.16.1-jessie-slim	
• node:alpine	
• node:latest	

• golang:alpine AS builder	1
• golang:latest	1
• python	1
• python:3	2

Some of these tags may have names like buster, jessie, or stretch in them. These are the suite code names for releases of Debian and indicate which release the image is based on. If your image needs to install any additional packages beyond what comes with the image, you'll likely want to specify one of these explicitly to minimize breakage when there are new releases of Debian.


Browser client
Headless client
Web (HTTP) server
Application server
Other parts of the information system
Database
Cloud


Browser client
Different implementations
Updated ”whenever”
Web (HTTP) server
service
service
service
Other parts of the
information system
Deployed independently
Different run-times and
programming languages


<script "...">
Browser client
Protocol-related changes
Web (HTTP) server
service
service
service
Database
Other parts of the information system
Cloud