Lecture 07

Rest of DevOps, CD, Deployment, Dependency Management

Kari Systä

06.10.2020

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

Tampereen yliopisto
Tampere University

Course practicalities

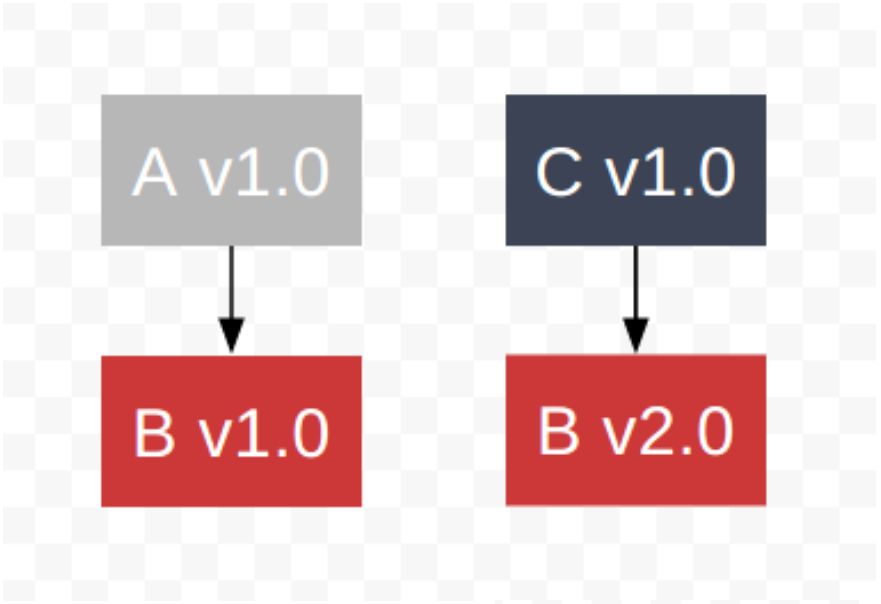
- Cloud exercise was returned by 70
- Compose execise was returned by 52+4
- Next exerise will be about message-queue communication

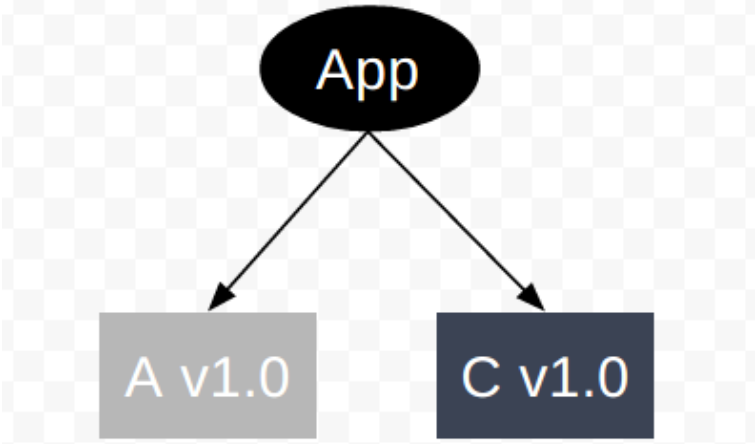
Tampereen yliopisto
Tampere University

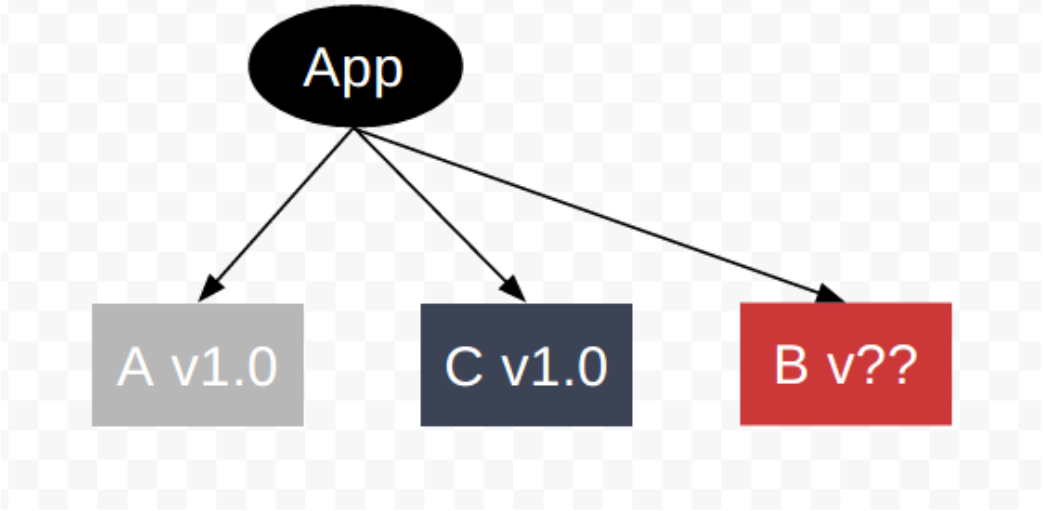
Dependency management

Tampereen yliopisto
Tampere University

https://npm.github.io/how-npm-works-docs/theory-and-design/dependency-hell.html







The old way

Static approach

- Libraries come with the compiler, or are intalled to the development tool
- Compiler integrates application with libraries
- The integrated system is deployed to users

The Web & Cloud way

Dynamic approach

- Libraries are downloaded in a dynamic manner
- Huge number of libraries available, use each other, and are frequently updates (continuous delivery)

- npm
- pip

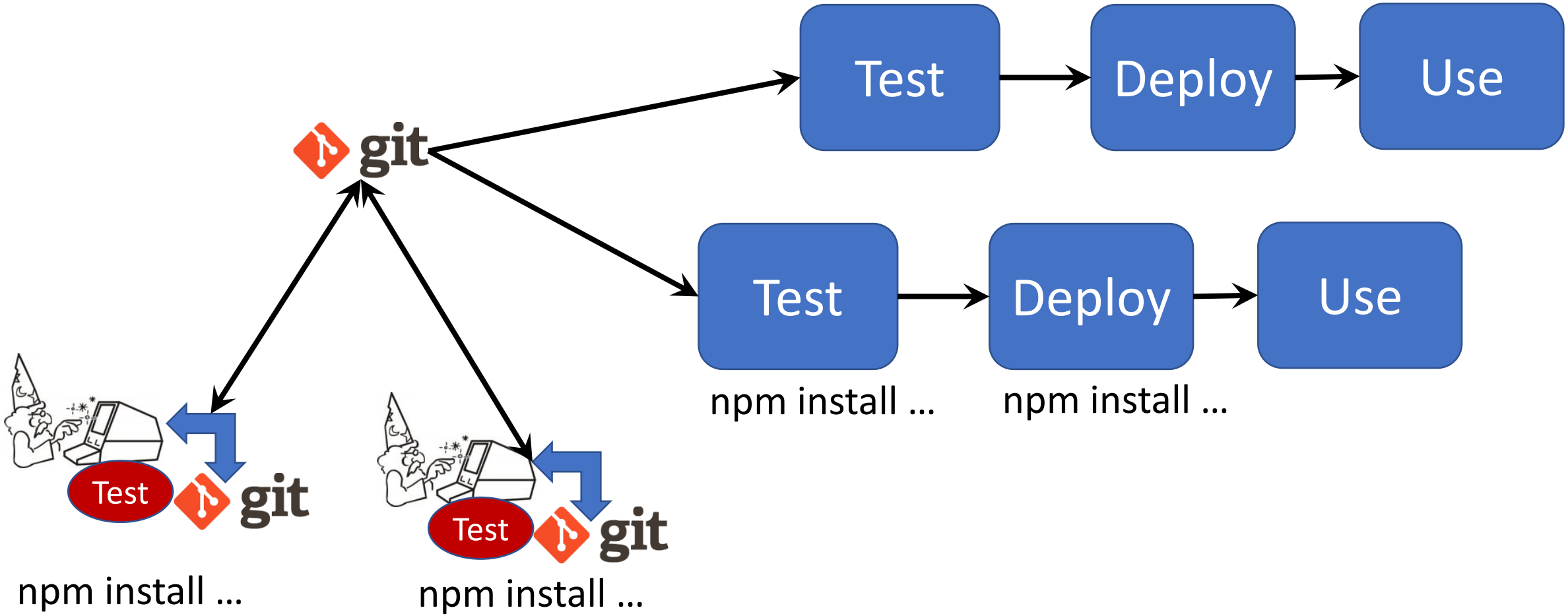
Tampereen yliopisto
Tampere University

Package.json

```
{
  "name": "service1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "4.17.1",
    "request": "2.88.0"
  }
}
```

Tampereen yliopisto
Tampere University

Development vs use



package-lock.json is automatically generated for any operations where npm modifies either the node_modules tree, or package.json. It describes the exact tree that was generated, such that subsequent installs are able to generate identical trees, regardless of intermediate dependency updates.

This file is intended to be committed into source repositories, and serves various purposes:

- Describe a single representation of a dependency tree such that teammates, deployments, and continuous integration are guaranteed to install exactly the same dependencies.
- Provide a facility for users to “time-travel” to previous states of node_modules without having to commit the directory itself.
- To facilitate greater visibility of tree changes through readable source control diffs.
- And optimize the installation process by allowing npm to skip repeated metadata resolutions for previously-installed packages.

Tampereen yliopisto
Tampere University

Package.json

```
{
  "name": "service1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "4.17.1",
    "request": "2.88.0"
  }
}
```

- Does not say anything about versions of packages used by express and request
- Allows use of later versions

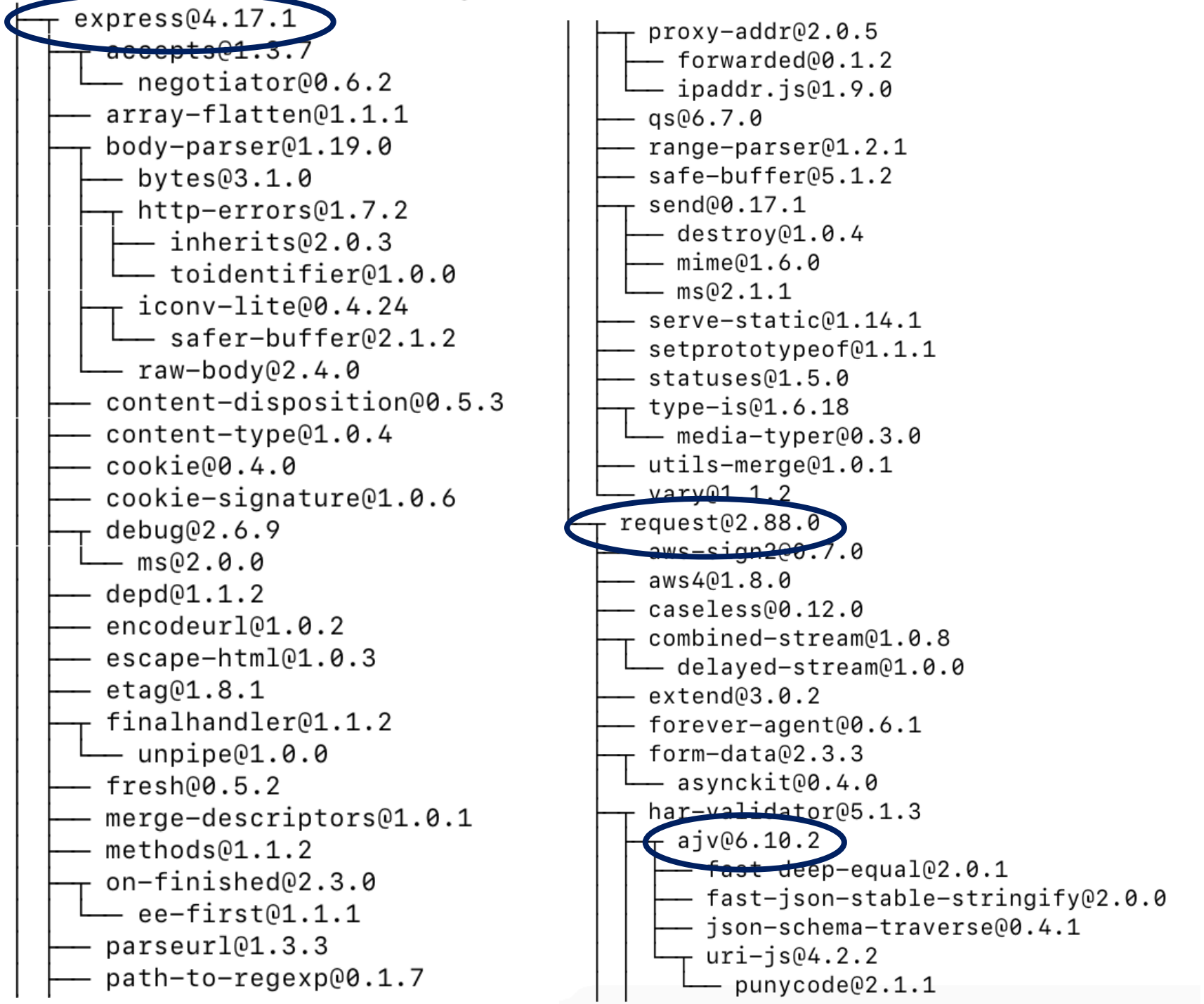
- **What is a possible problem with this?**

Package-lock.json


```
{
  "name": "service1",
  "version": "1.0.0",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "accepts": {
      "version": "1.3.7",
      "resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.7.tgz",
      "integrity": "sha512-Il80Qs2WjYlJIBNzNkK6KYqlVMTbZLXgHx2oT0pU/fjRHyEp+PEfEPY0R3WCwAGVOtauxh1hOxNgIf5bv7dQpA==",
      "requires": {
        "mime-types": "~2.1.24",
        "negotiator": "0.6.2"
      }
    },
    "ajv": {
      "version": "6.10.2",
      "resolved": "https://registry.npmjs.org/ajv/-/ajv-6.10.2.tgz",
      "integrity": "sha512-TXtUUEYHuaTEbLZWIKUr5pmBuhDLy+8KYtPYdcV8qC+pOZL+NKqYwvWSRrVXHn+ZmRRAu8vJTAznH7Oag6RVRw==",
      "requires": {
        "fast-deep-equal": "^2.0.1",
        "fast-json-stable-stringify": "^2.0.0",
        "json-schema-traverse": "^0.4.1",
        "uri-js": "^4.2.2"
      }
    },
```

Tampereen yliopisto
Tampere University

npm ls



Tampereen yliopisto
Tampere University

Nodejs has also changed (for example)

2018-10-23, Version 11.0.0 (Current), @jasnell

Notable Changes

Build

FreeBSD 10 is no longer supported. #22617

child_process

The default value of the windowsHide option has been changed to true. #21316

console

console.countReset() will emit a warning if the timer being reset does not exist. #21649

console.time() will no longer reset a timer if it already exists. #20442

Dependencies

V8 has been updated to 7.0. #22754

fs

The fs.read() method now requires a callback. #22146

The previously deprecated fs.SyncWriteStream utility has been removed.#20735

Tampereen yliopisto
Tampere University

Npm versions

- 6.12.0 — 15 days ago
- 6.12.0-next.0 — a month ago
- 6.11.3 — 2 months ago
- 6.11.2 — 2 months ago
- 6.11.1 — 2 months ago
- 6.11.0 — 2 months ago
- 6.10.3 — 3 months ago
- 6.10.2 — 3 months ago

- 6.14.8 — latest
- 6.14.7 — 2 months ago
- 6.14.6 — 3 months ago
- 6.14.5 — 5 months ago
- 6.14.4 — 6 months ago
- 6.14.3 — 6 months ago
- 6.14.2 — 7 months ago
- 6.14.1 — 7 months ago
- 6.14.0 — 7 months ago
- 6.13.7 — 8 months ago
- 6.13.6 — 9 months ago
- 6.13.5 — 9 months ago
- 6.13.4 — 10 months ago
- 6.13.3 — 10 months ago

Tampereen yliopisto
Tampere University

New in NPM 5

2. Lockfiles

With npm@5, lockfiles are the default (package-lock.json). This simply means that whatever files you get when you install a package will be the same every time you install that package after initial install. This eliminates the challenges developers had with having different files on different developer environments after installing the same package.

Tampereen yliopisto
Tampere University

And the language (ECMAScript / JavaScript)

ES5 (2009)

- This is the baseline version of JS which you can generally assume all run-times (except really old ones!) will support.

ES6 / ES2015

- Standard Modules — import and export
- Standardised Promises
- Classes & Inheritance
- Block-scoped variables — let and const
- Template Literals
- Object destructing into variables
- Generator functions
- Map and Set data structures
- Internationalisation for Strings, Numbers and Dates via Intl API

ES7 / ES2016

- Array.includes()
- Numeric exponent (power of) operator **

ES8 / ES2017

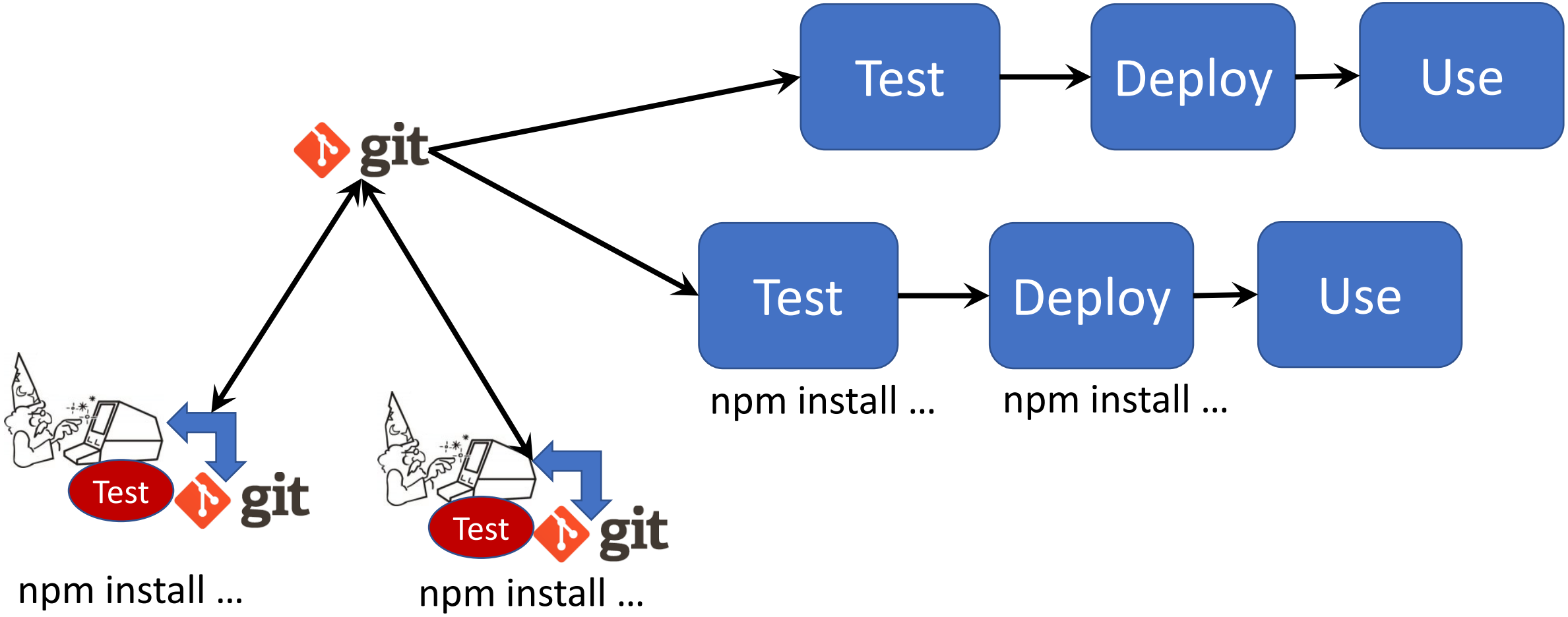
- Async Functions
- Object.entries
- String padding functions

ES9 / ES2018

- Object Rest/Spread const obj = { ...props };
- Asynchronous Iteration for await (...) {
- Promise finally() function
- Regular expression enhancements (lookbehind, named groups)

Tampereen yliopisto
Tampere University

Development vs use



Tampereen yliopisto
Tampere University

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

git clone ..
docker-compose up

Tampereen yliopisto
Tampere University

Towards solutions?

- Build a docker image and deploy that?

- Use package-lock.json and installation scripts?

Tampereen yliopisto
Tampere University

Other systems, like Pyhton and golang

- Python
  - Virtual environments
  - PIP
- Golang
  - go get ..

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 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

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

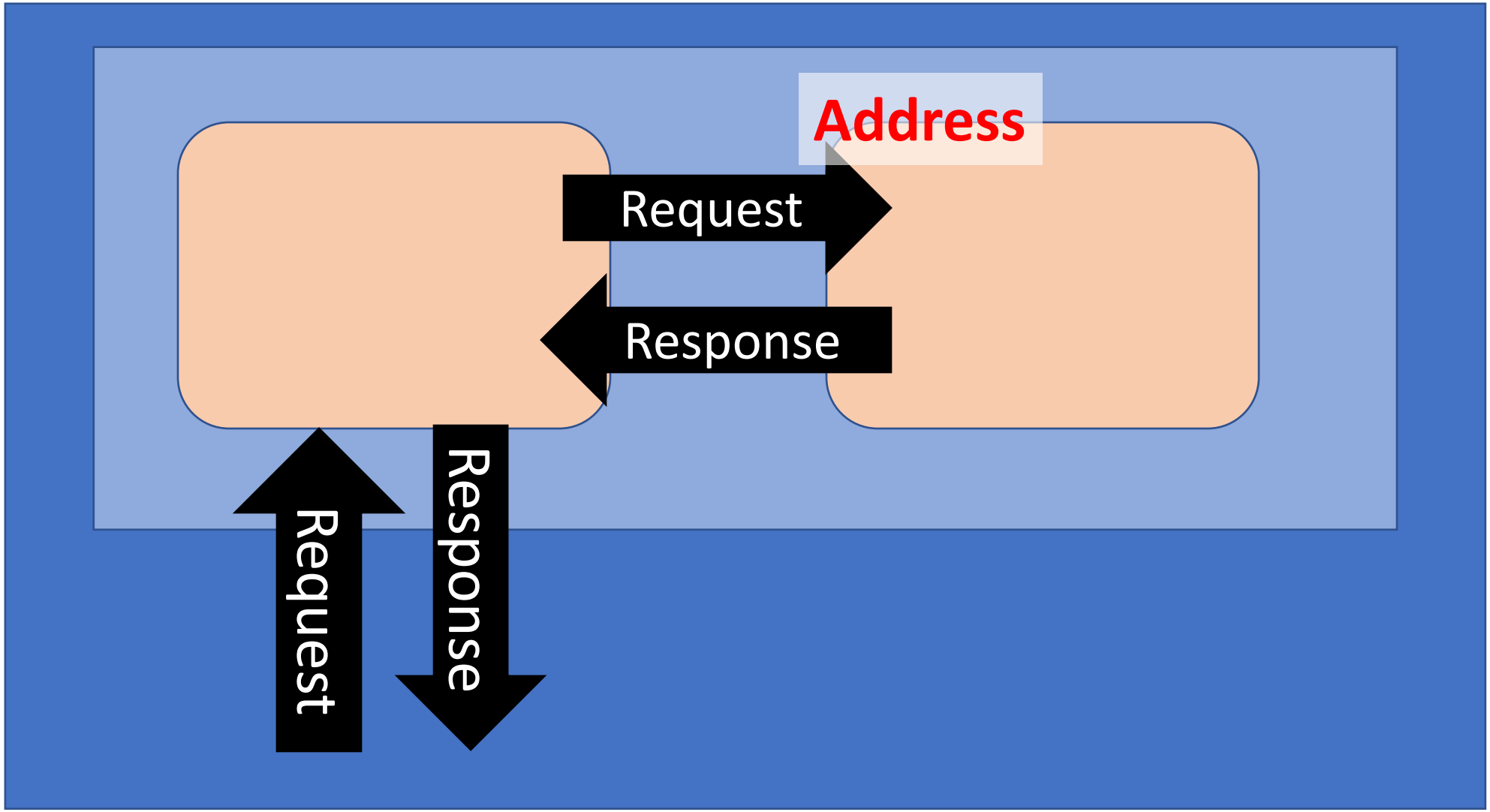
- **Performance**. You'll typically be able to access provide better performance than is possible with nonnative features. For example, you can deal with an input/output (I/O) system that works with autoscaling and loadbalancing features.
- **Efficiency**. Cloud-native applications' use of cloud-native 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.

More about that later in the course

Tampereen yliopisto
Tampere University

… instead now some bottom-up to help you in the next excersise

Back to old picture



Tampereen yliopisto
Tampere University

Corner-stones of REST

- Client-server architecture
  - Separation of concerns
- Statelessness
  - no client context being stored on the server between requests
- Cacheability

- Layered system
  - Client does not know if connected to other end directly
- Uniform interface

Do not call your design for previous exercise REST!

Uniform representation

- Resource identification in requests
  - URIs
  - Separated from representation (XML, JSON,…)
- Resource manipulation through representations
- Self-descriptive messages
- Hypermedia as the engine of application state (HATEOAS)
- Application to HTTP
  - URL’s
  - GET, PUT, POST, DELETE
  - MIME-types

Message-bus instead of HTTP

- Challenges: increased network operations, tight service coupling
- Message bus helps to define how services communicate, service discovery reduces operational complexity
- Asynchronous messaging leads to
  - loosed coupling
  - More complex logic (async is a cousin of parallelism)
- Actually, there are multiple options
  - RPC, REST, Asynchronous message, application-specific protocols

Tampereen yliopisto
Tampere University

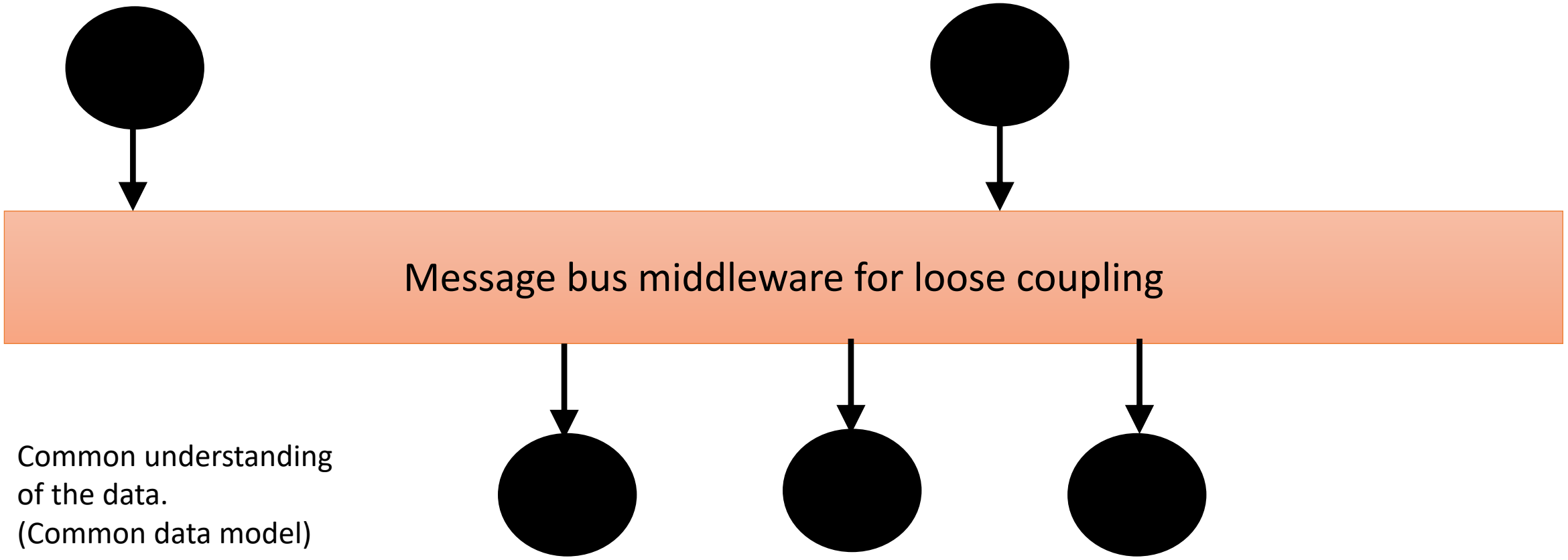
But the ”calls” can be laborous

```
let message = "Hello from " + req.client.remoteAddress + ":" +
req.client.remotePort + " to " + req.client.localAddress + ":" +
req.client.localPort;

request('http://server2:4000/getServer',  { json: true },
        (err, response, body) => {
                                    if (err) {
                                      return console.log(err);
                                    }
                                    res.send(message + " " + body); });
```

Tampereen yliopisto
Tampere University

The message bus approach



Common understanding
of the data.
(Common data model)

Tampereen yliopisto
Tampere University

RabbitMQ

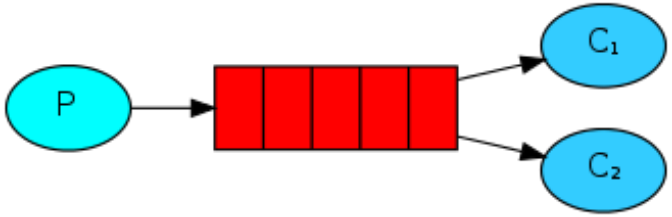
- An example of message queue technology
- Can be used to implement various architectures

Examples of RabbitMQ use

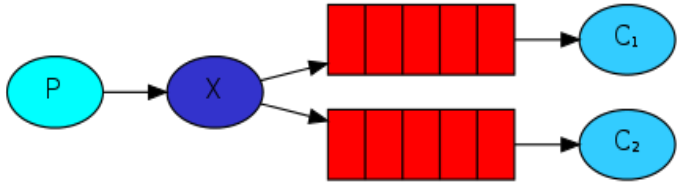
https://www.rabbitmq.com/getstarted.html

Simple queue

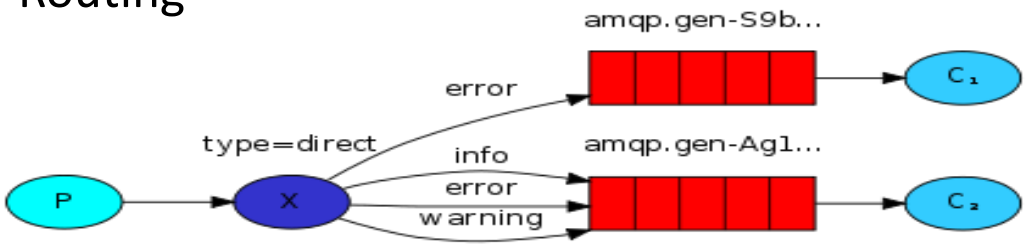
Task distribution



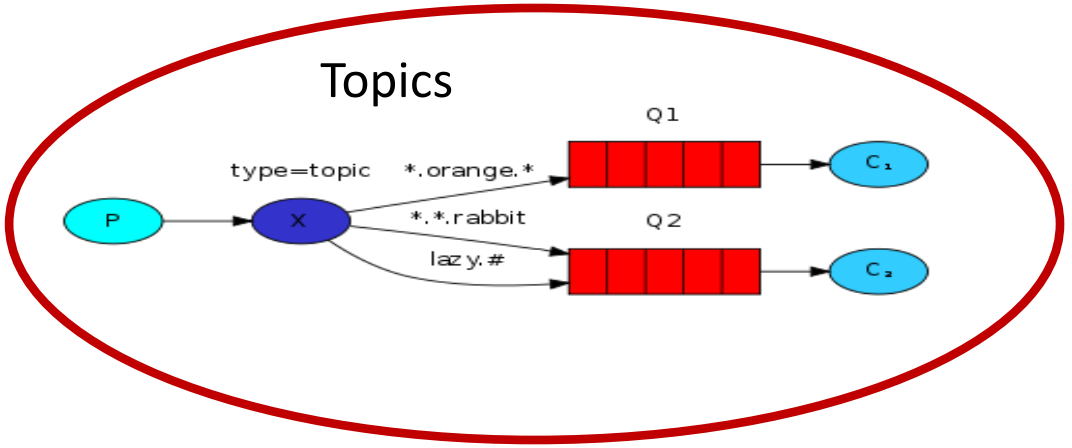
Publish/subscribe



Routing

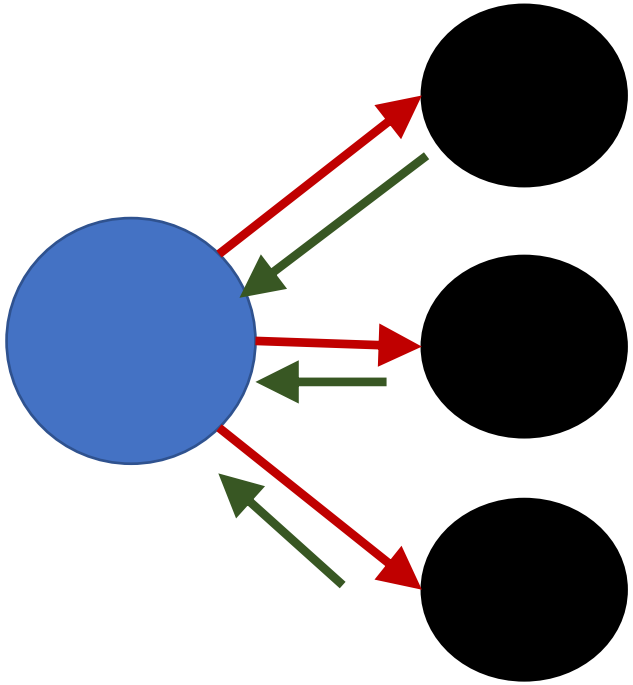


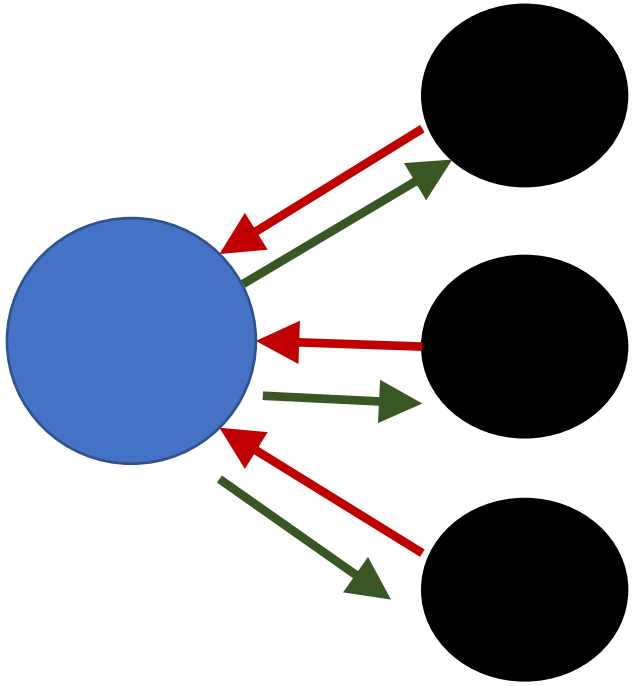
Topics



Tampereen yliopisto
Tampere University

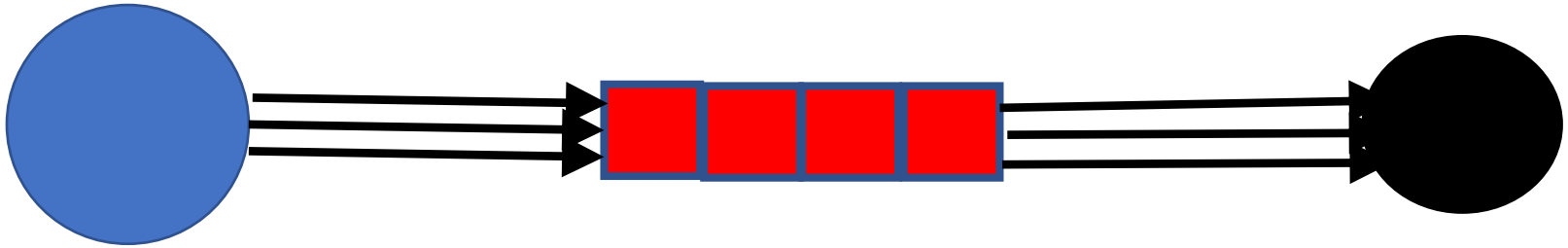
Publish-subscribe



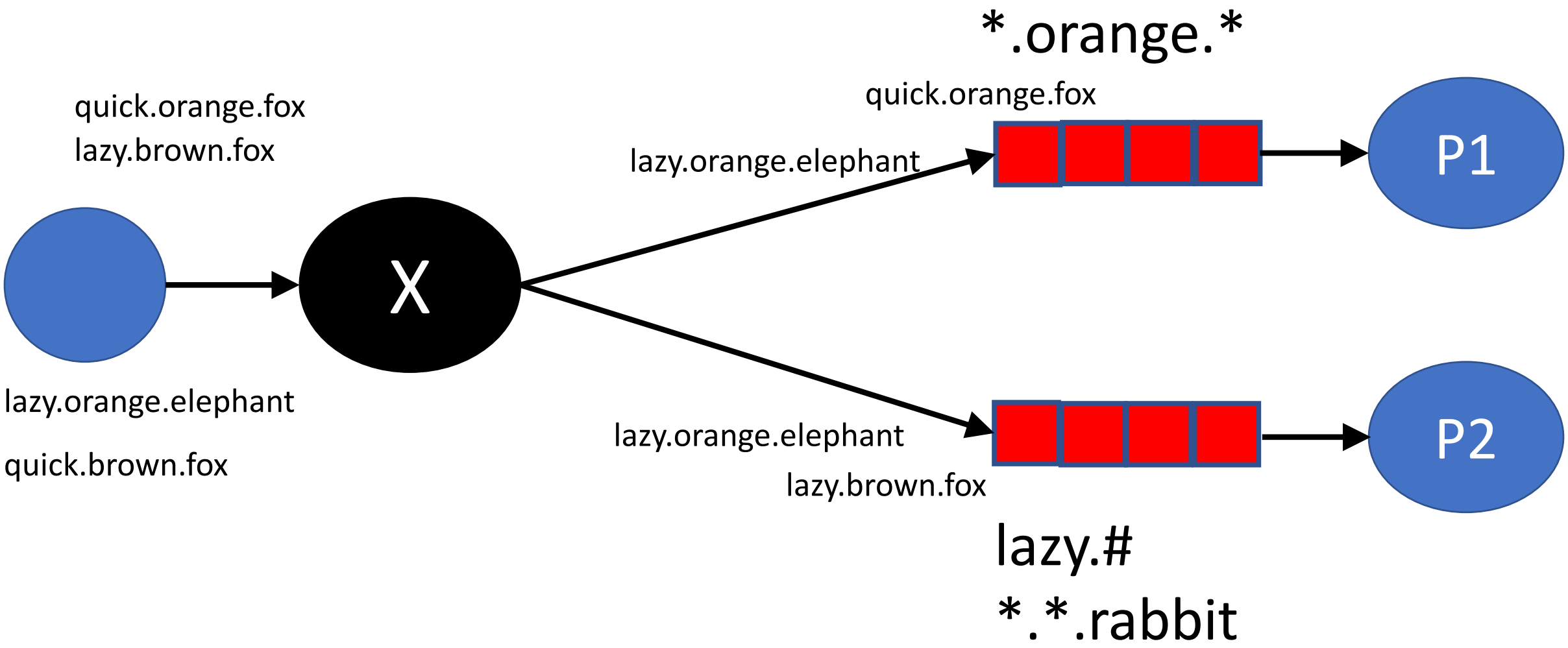


Tampereen yliopisto
Tampere University

Message queue

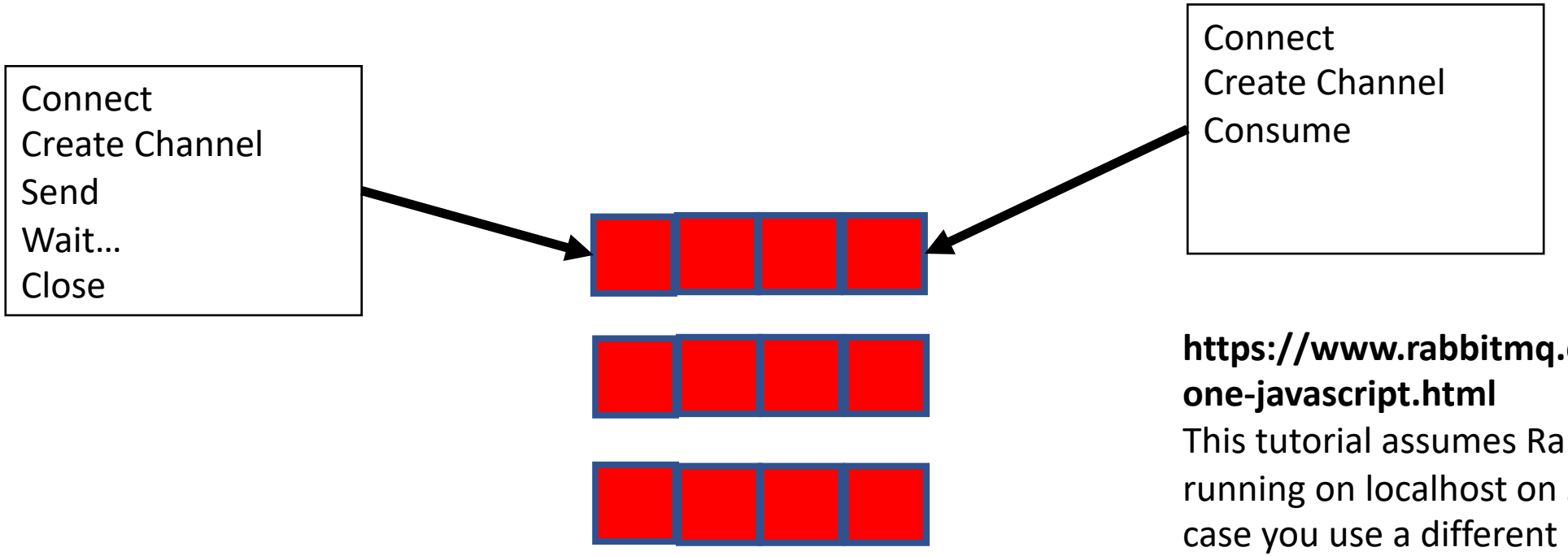


An example of topic-based communication (adopted from https://www.rabbitmq.com/tutorials/tutorial-five-python.html)



Tampereen yliopisto
Tampere University

RabbitMQ – steps in practice



https://www.rabbitmq.com/tutorials/tutorial-one-javascript.html

This tutorial assumes RabbitMQ is installed and running on localhost on standard port (5672). In case you use a different host, port or credentials, connections settings would require adjusting.

Tampereen yliopisto
Tampere University

Next exercise

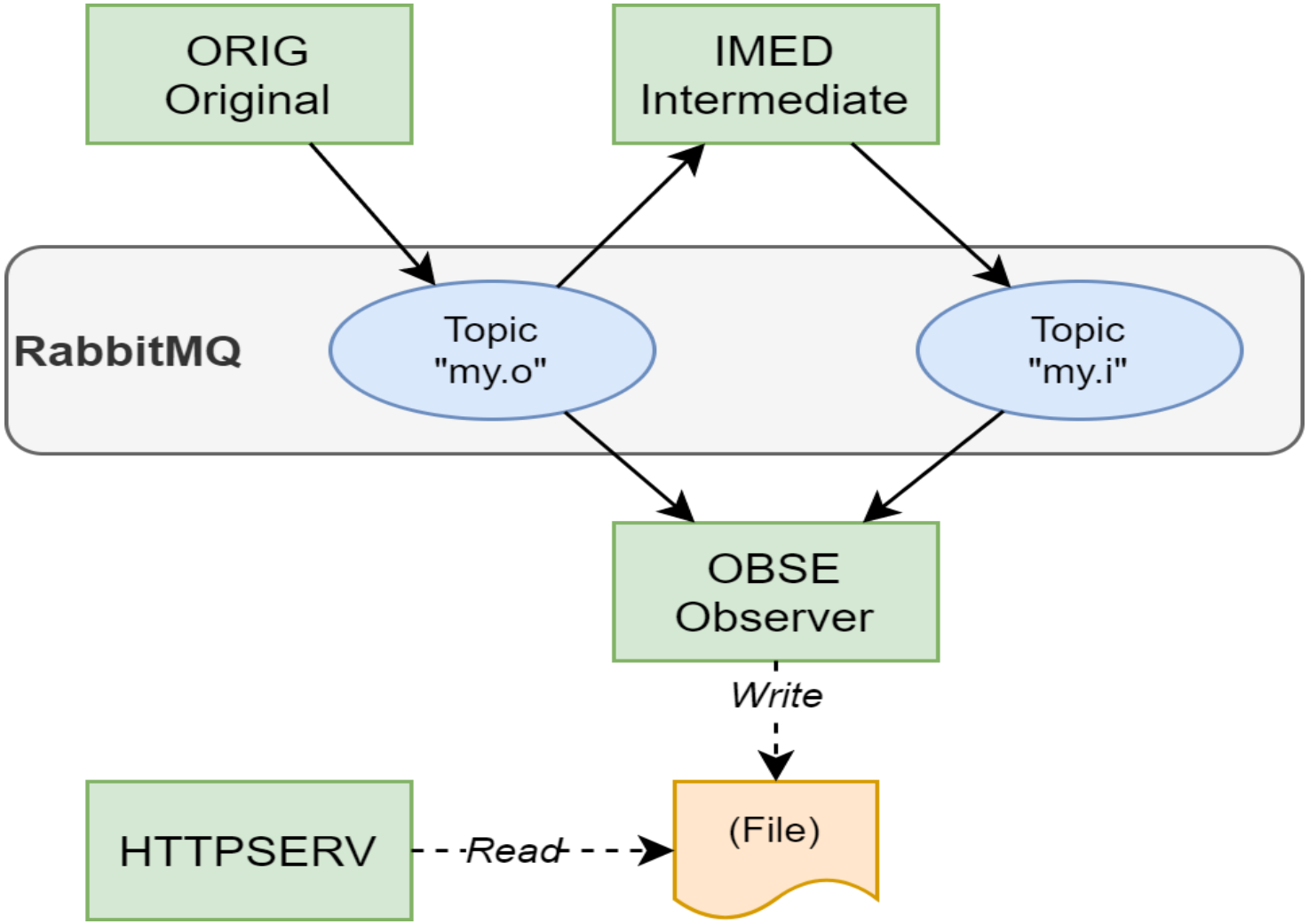
You create a bigger system of several processes and message queue infrastructure

Grading policy:

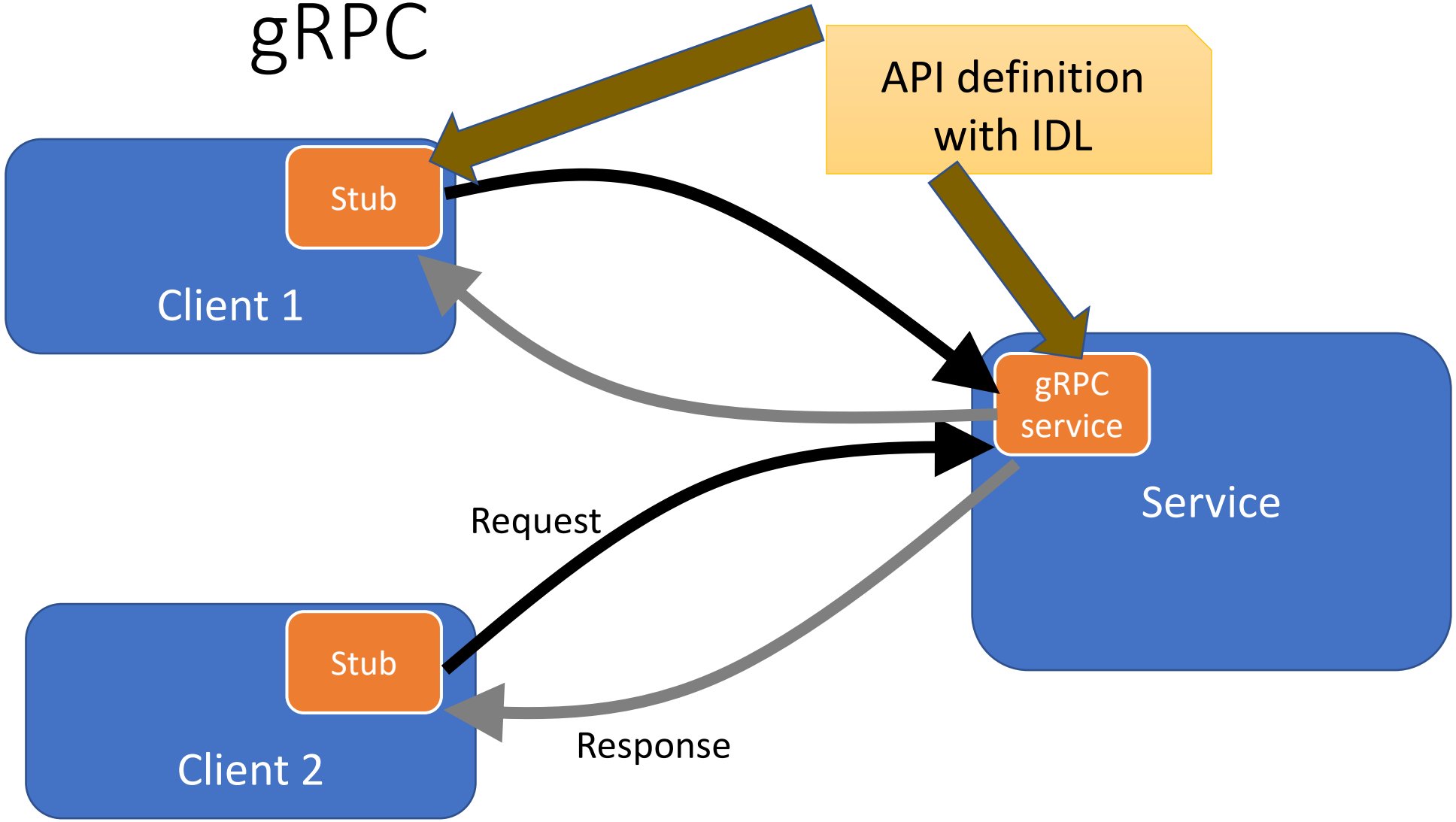
- maximum 6 points are given (total of the course will be about 50)
- missing the deadline: points reduced by 0.5 points / day
- how well the requirements are met: 2p
- following the good programming and docker practices: 2p
- quality of the document: 2p

Deadline:

- Mon, Oct 5 2020, 6 p.m. – Mon, Oct 19 2020, noon
  Late submissions are allowed until Sat, Oct 31 2020, 11:59 p.m. but points are only worth 70%.


ORIG
Original
IMED
Intermediate
RabbitMQ
Topic
"my.o"
Topic
"my.i"
OBSE
Observer
Write
HTTPSERV
Read
(File)

gRPC


API definition
with IDL
Stub
Client 1
gRPC
service
Service
Request
Stub
Client 2
Response

Tampereen yliopisto
Tampere University

Example API description

```
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
  // Sends another greeting
  rpc SayHelloAgain (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest { string name = 1; }

// The response message containing the greetings message
HelloReply { string message = 1; }
```

Tampereen yliopisto
Tampere University

Call in JavaScript and Python

```
function main() {
  var client = new hello_proto.Greeter('localhost:50051',
                                       grpc.credentials.createInsecure());
  client.sayHello({name: 'you'}, function(err, response) {
    console.log('Greeting:', response.message);
  });
  client.sayHelloAgain({name: 'you'}, function(err, response) {
    console.log('Greeting:', response.message);
  });
}
```

```
def run():
    channel = grpc.insecure_channel('localhost:50051')
    stub = helloworld_pb2_grpc.GreeterStub(channel)
    response = stub.SayHello(helloworld_pb2.HelloRequest(name='you'))
    print("Greeter client received: " + response.message)
    response = stub.SayHelloAgain(helloworld_pb2.HelloRequest(name='you'))
    print("Greeter client received: " + response.message)
```

Tampereen yliopisto
Tampere University

And C++

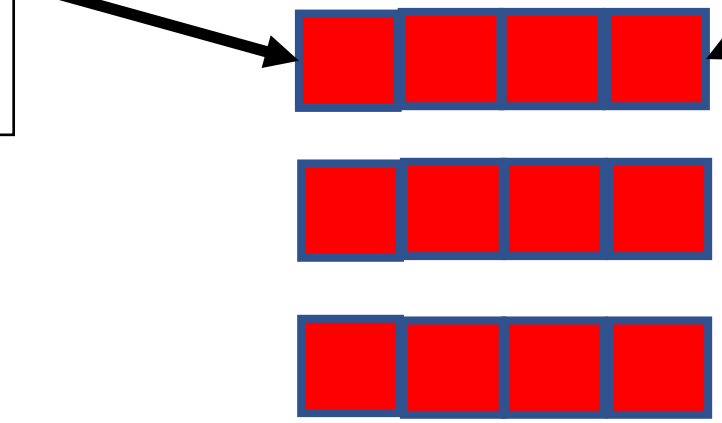
```
std::string SayHelloAgain(const std::string& user) {
  // Follows the same pattern as SayHello.
  HelloRequest request;
  request.set_name(user);
  HelloReply reply;
  ClientContext context;

  // Here we can use the stub's newly available method we just added.
  Status status = stub_->SayHelloAgain(&context, request, &reply);
  if (status.ok()) {
    return reply.message();
  } else {
    std::cout << status.error_code() << ": " << status.error_message()
              << std::endl;
    return "RPC failed";
  }
}
```

Tampereen yliopisto
Tampere University

RapidMQ

Connect
Create Channel
Send
Wait…
Close



Connect
Create Channel
Consume

https://www.rabbitmq.com/tutorials/tutorial-one-javascript.html
This tutorial assumes RabbitMQ is installed and running on localhost on standard port (5672). In case you use a different host, port or credentials, connections settings would require adjusting.