Kleiner ist besser

Going subatomic mit Quarkus und Openshift

bei AVIATAR

>oc describe speaker thorsten -o yaml

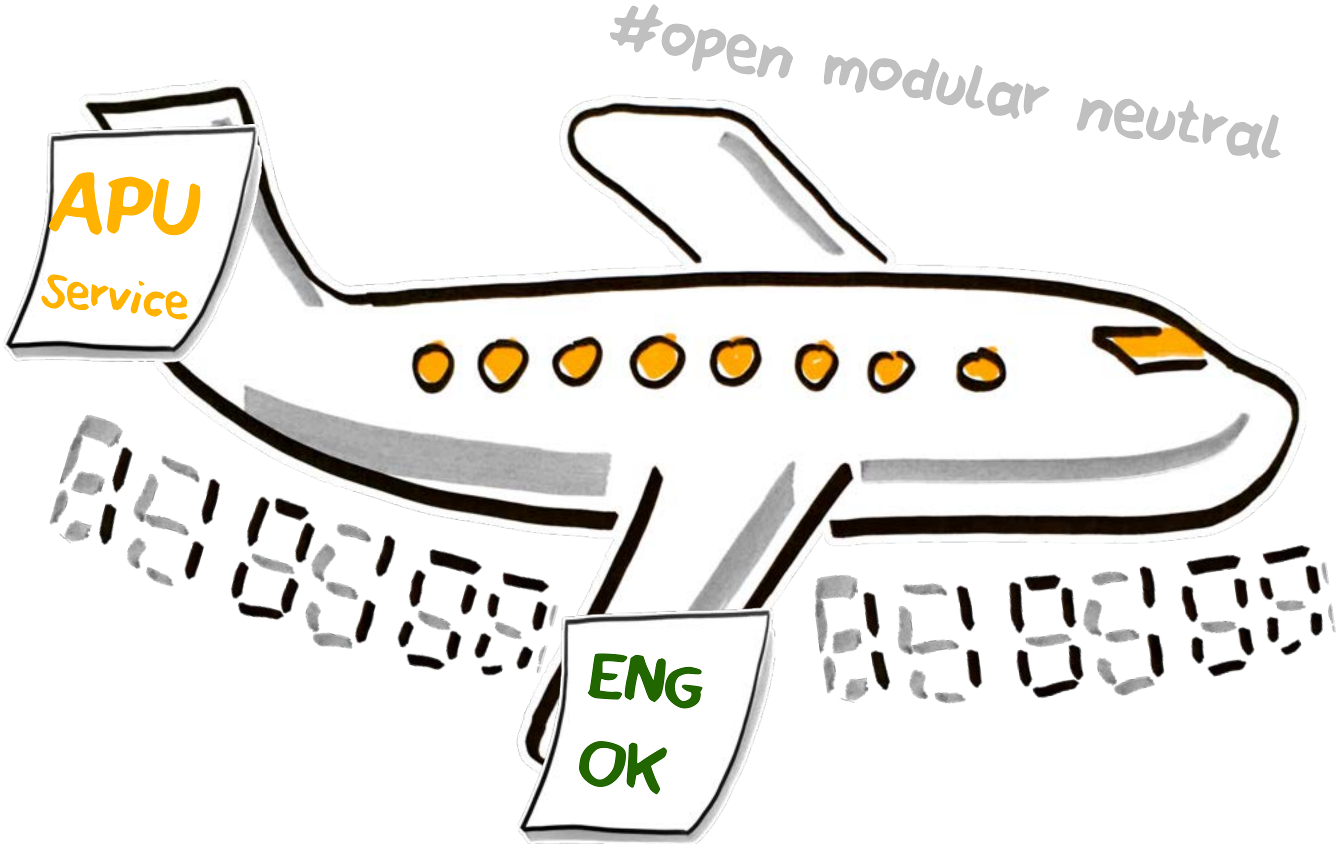
```
name: Thorsten Pohl
work:
  company: Lufthansa Technik
  position:
      - Architect AVIATAR Platform
      - Product Owner AVIATAR Core Services
contact:
  email: thorsten@aviatar.io
  twitter: @thorsten_pohl
  web: https://thorsten.pro
```

What is Lufthansa Technik?

VIP Aircraft welcome!

The world`s leading independent Provider of Aircraft Maintenance, Repair and Overhaul

AVIATAR

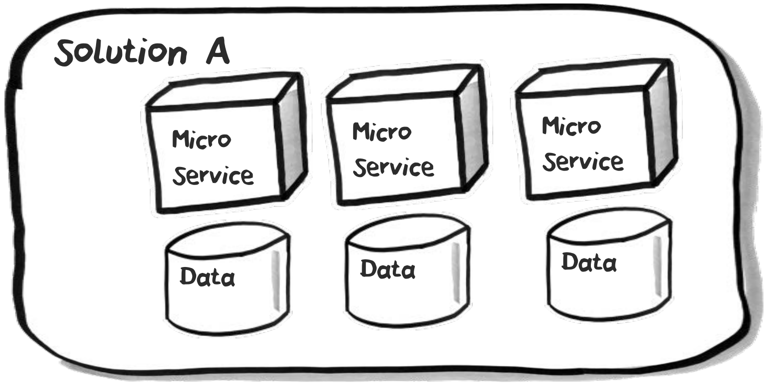


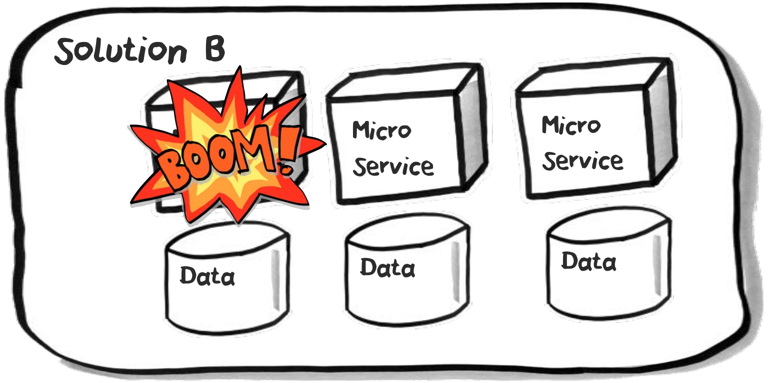
Innovation Award Winner 2018

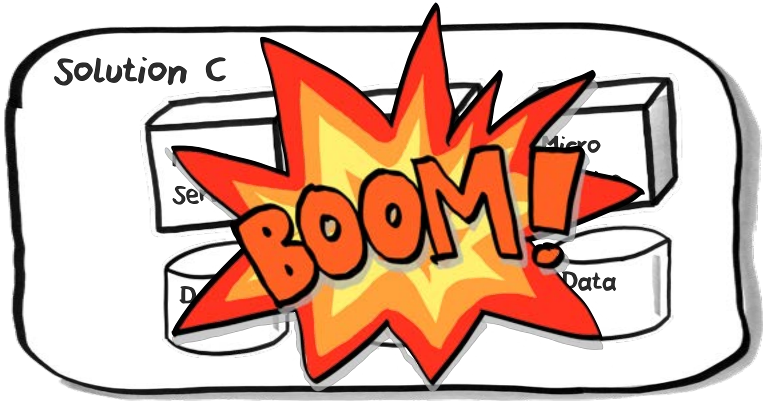
Our Challenge: How can we maintain our agile flexibility while growing?

Mastering Complexity:

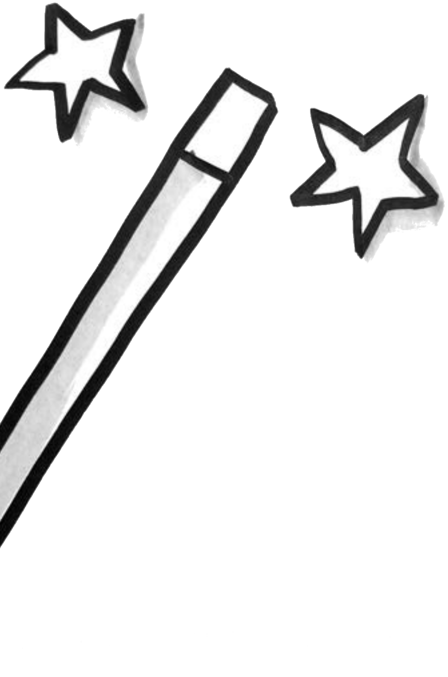
One Microservice per Functionality



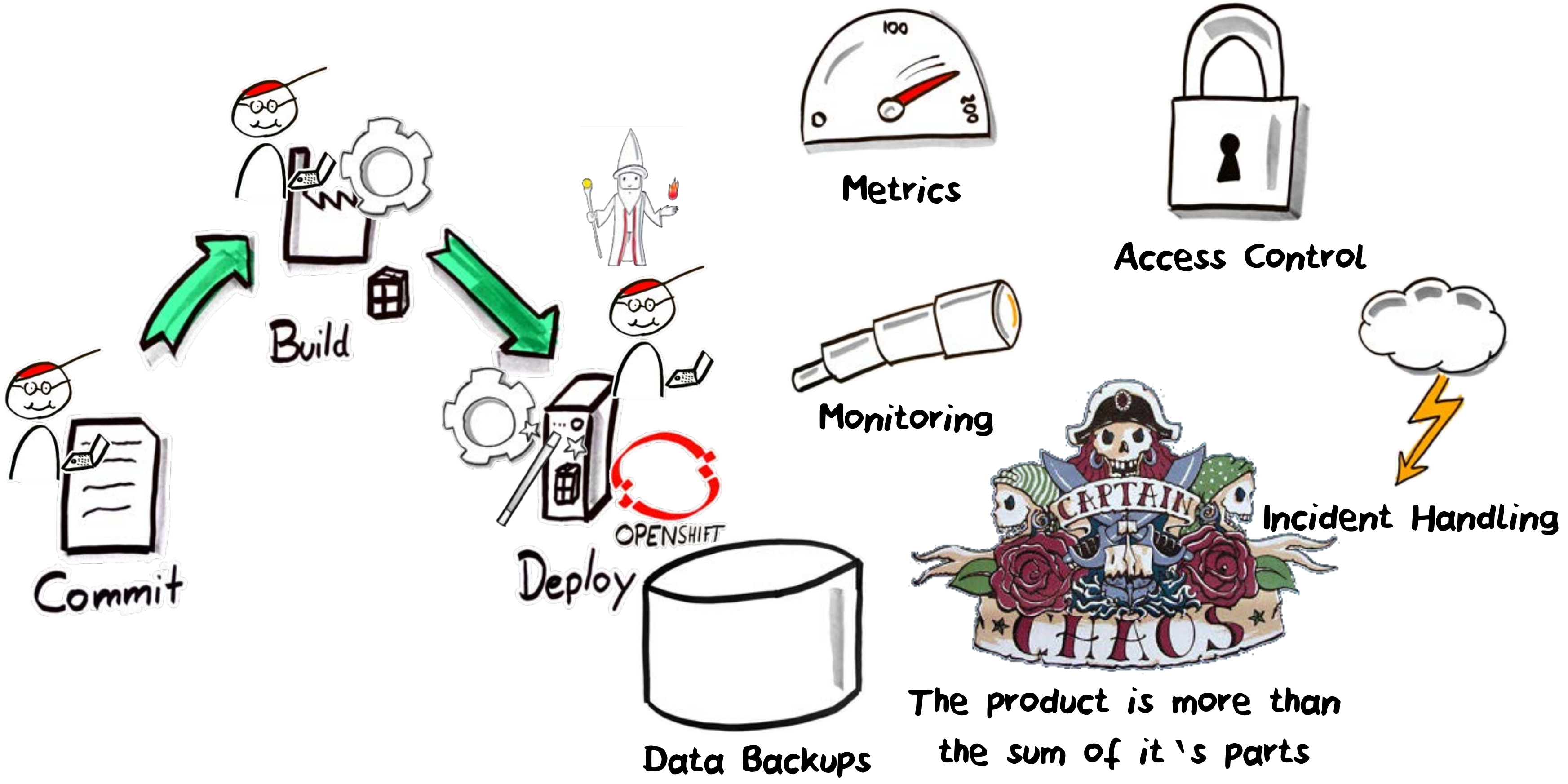




Microservices FTW!

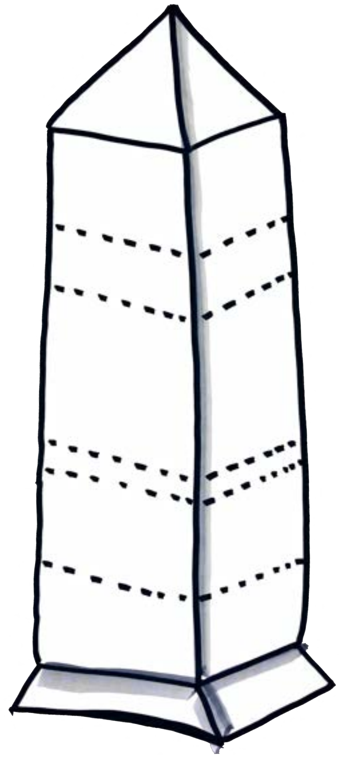


Running a service is work



The product is more than the sum of it's parts

I need a break, back to (majestic) monolith then?



https://m.signalvnoise.com/the-majestic-monolith/
https://jaxenter.de/microservices-alternative-majestic-modular-monoliths-85890

Not so fast, we have got some friend

Prometheus

Gatling
LOAD TESTING

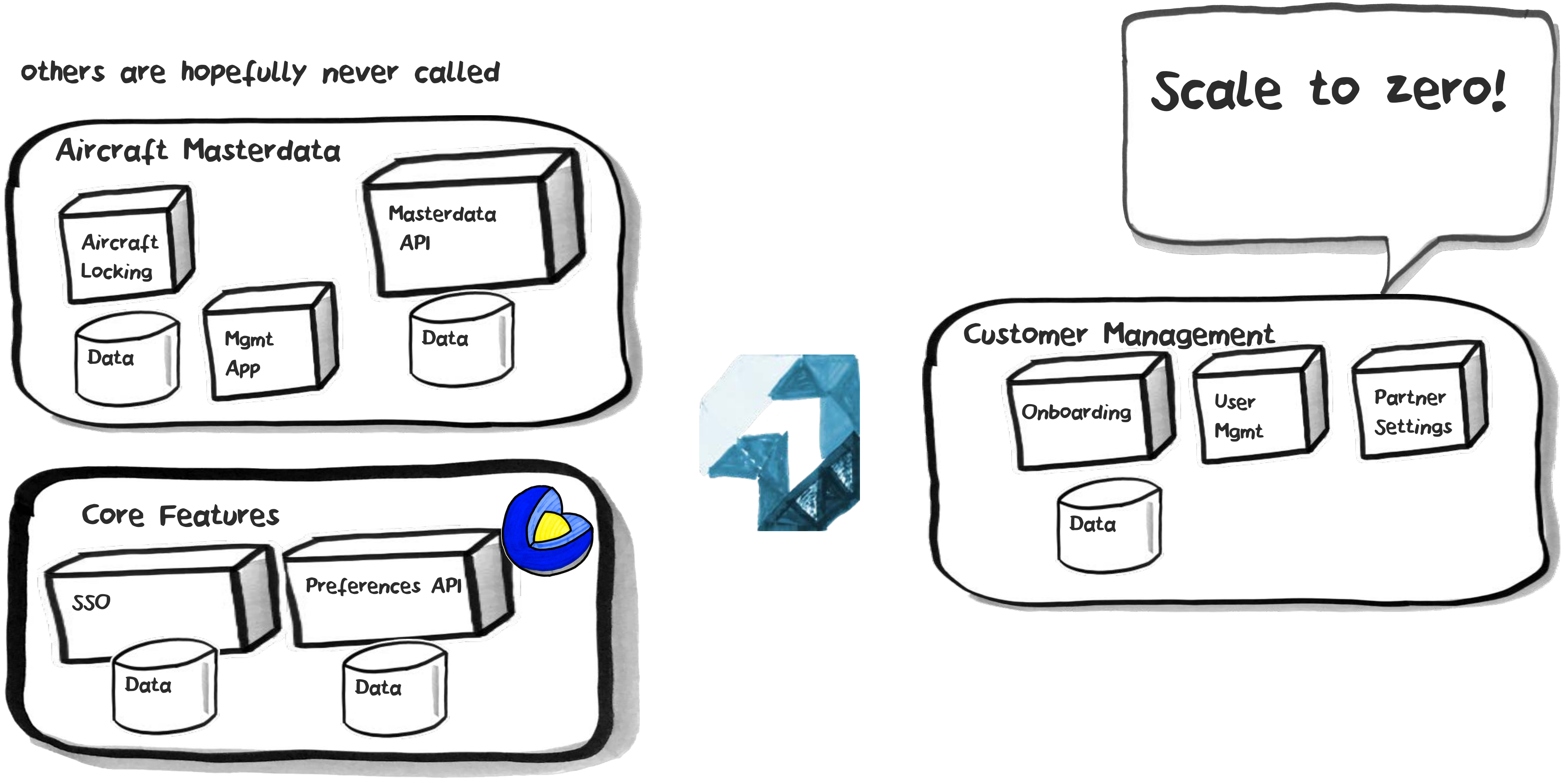
Grafana

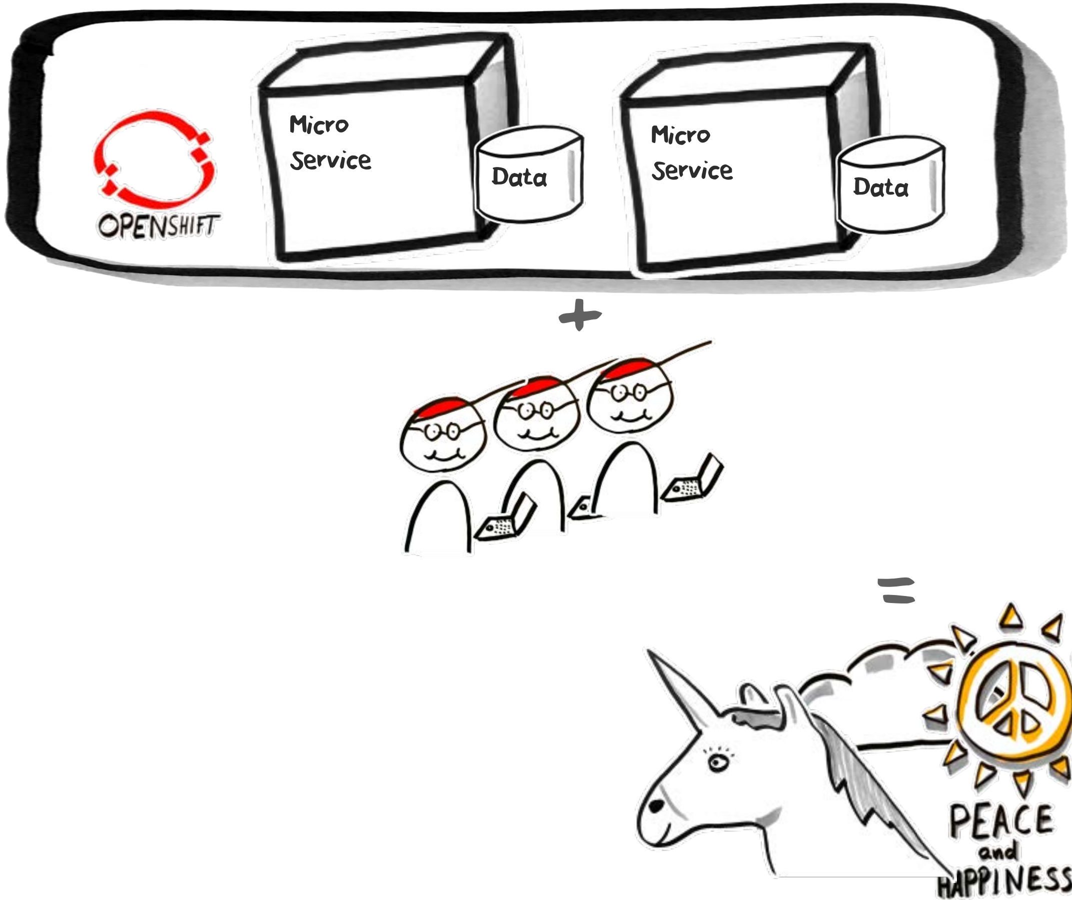
OPENSHIFT

Lenovo

Some Microservices are called seldomly

others are hopefully never called





Microservices

+

Autonomous Teams

Where has all my money gone?

Your balance

15000 CPU secs
8000 MiB RAM

1.598,75€

Your balance

15000 CPU secs
8000 MiB RAM

1.598,75€

Your balance

15000 CPU secs
8000 MiB RAM

1.598,75€

Your balance

15000 CPU secs
8000 MiB RAM

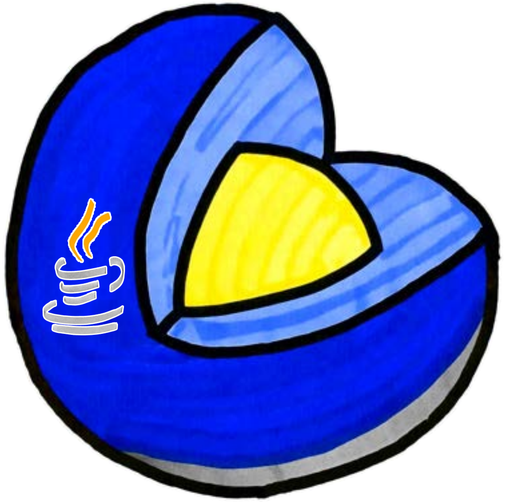
1.598,75€

Your balance

15000 CPU secs
8000 MiB RAM

1.598,75€

Let`s go subatomic



AVIATAR Compute Technology Stack

Spring Boot

+ ultra-versatile, performant, reliable, well-known, popular
- heavy-weight, ultra-versatile

JBoss EAP

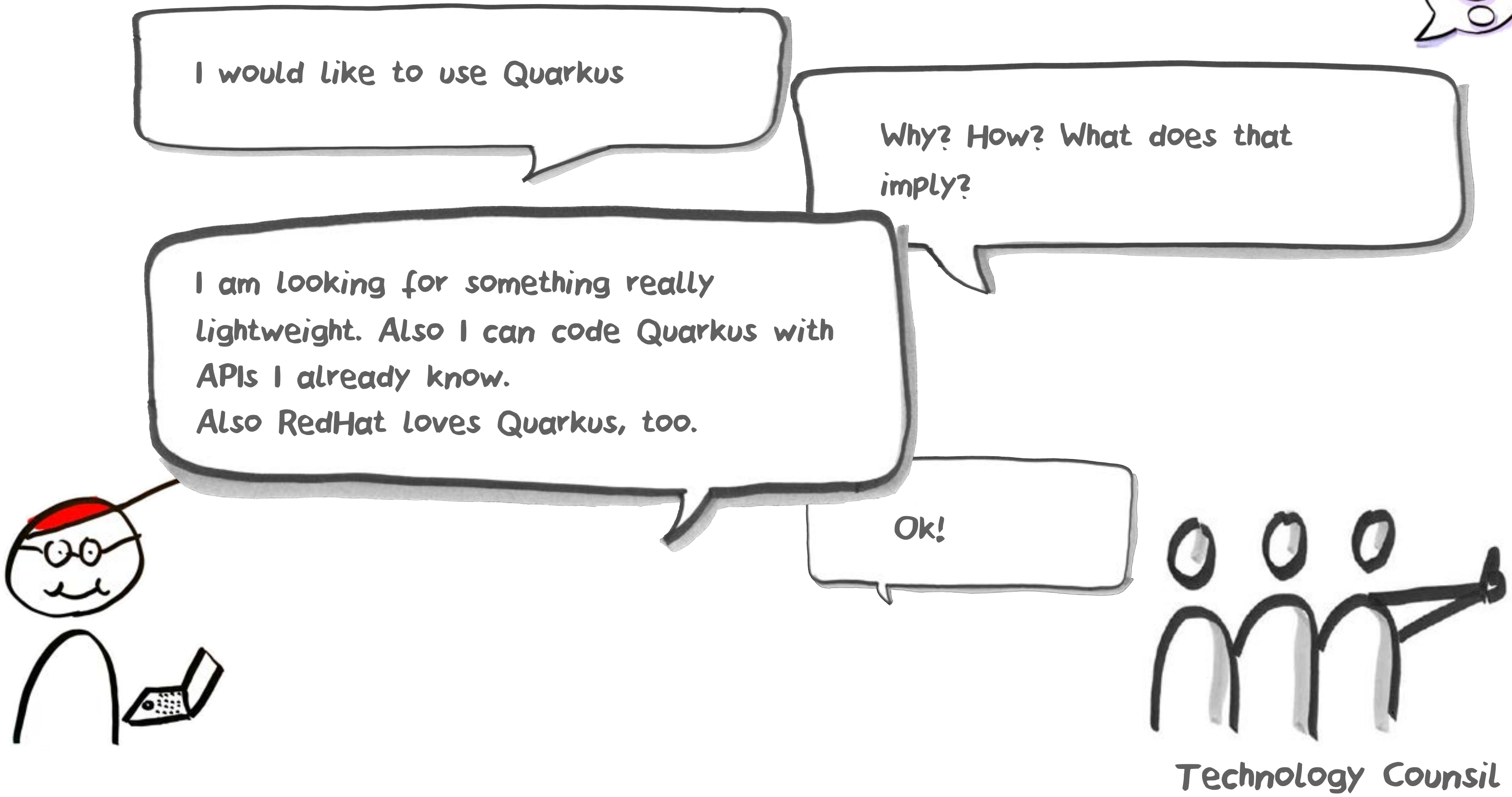
+ standards-based, performant, reliable
- heavy-weight, slower innovation, unpopular

Vert.X

+ light-weight, performant, reactive, openapi-3-router
- learning curve, easy to break, not widely known

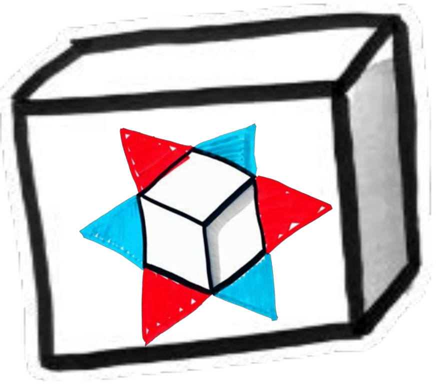
Looking for a small alternative

How do we change our stack?


I would like to use Quarkus
Why? How? What does that imply?
I am looking for something really lightweight. Also I can code Quarkus with APIs I already know. Also RedHat loves Quarkus, too.
Ok!
Technology Counsil

Quarkus at AVIATAR

When do we use it?



Functionality that is rather small

Functionality we want implemented and deployed independently

Functionality that might benefit from Serverless / Scale to zero

Tooling

Why do I like Quarkus?

Quarkus Applications are written in Java and Kotlin!

Why do I like Quarkus?

Quarkus has some great plugins

quarkus-rest-client

quarkus-oidc

quarkus-micrometer
+ micrometer-registry-prometheus

quarkus-smallrye-health

quarkus-schedule

Why do I like Quarkus?

It fits into our development process

API First

Jenkins

> mvn clean package

12 Factor App

Health Checks

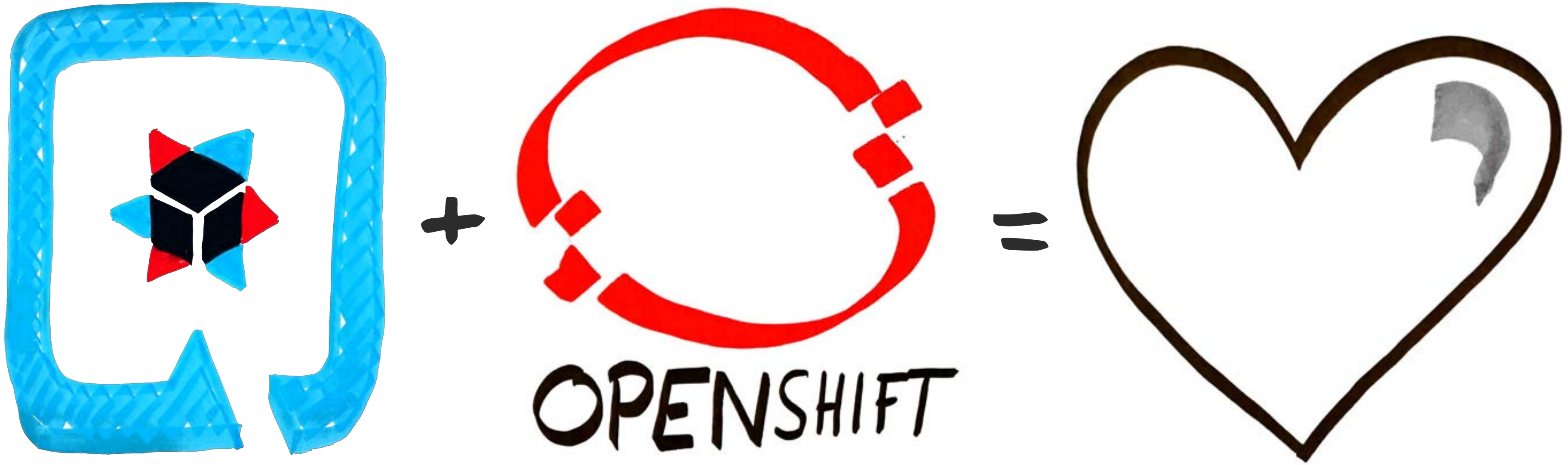
JWT Auth

JWT Auth

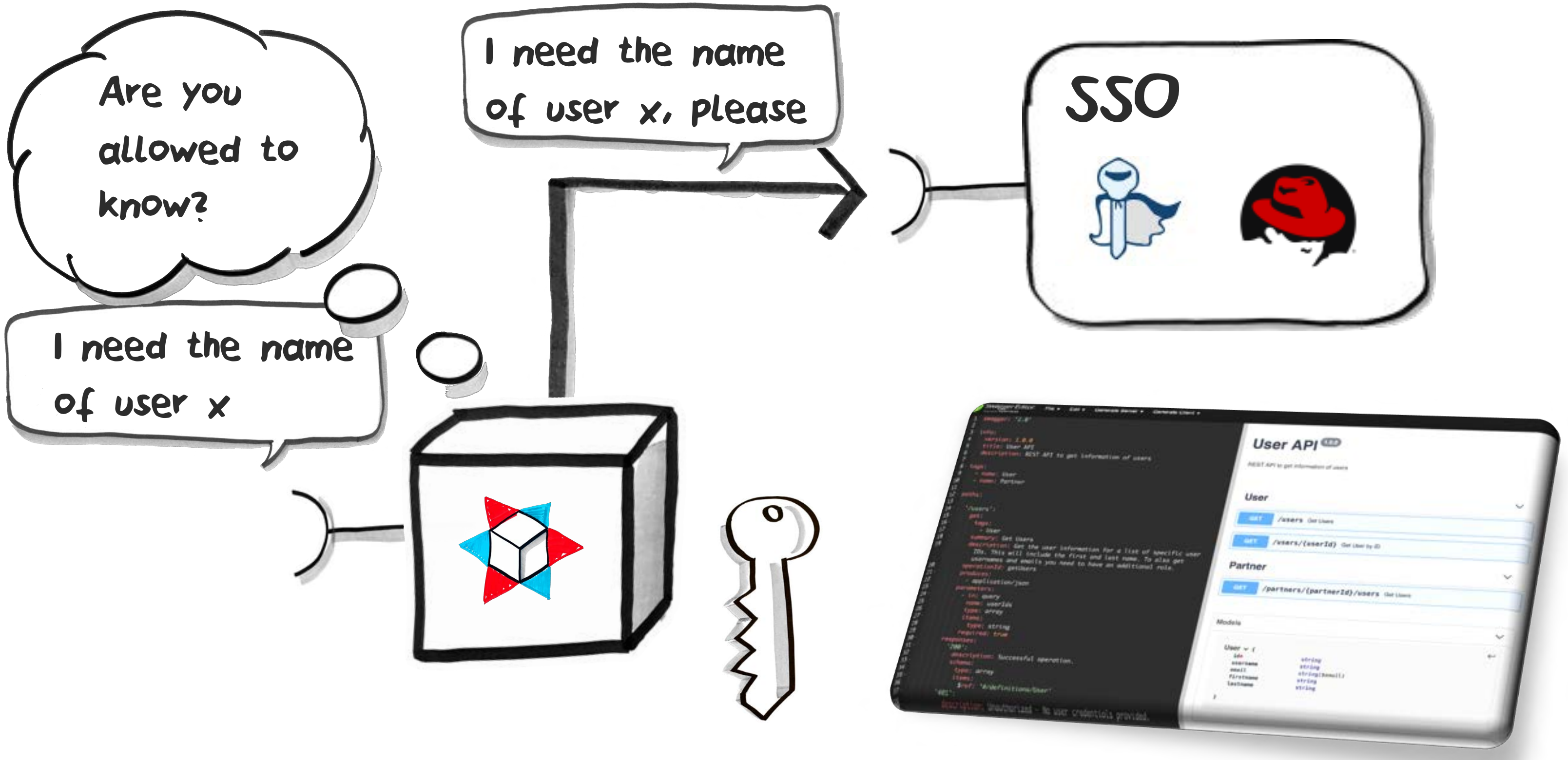
Source 2 Image

Prometheus / Grafana

Why do I like Quarkus?



Use Case Example: User API

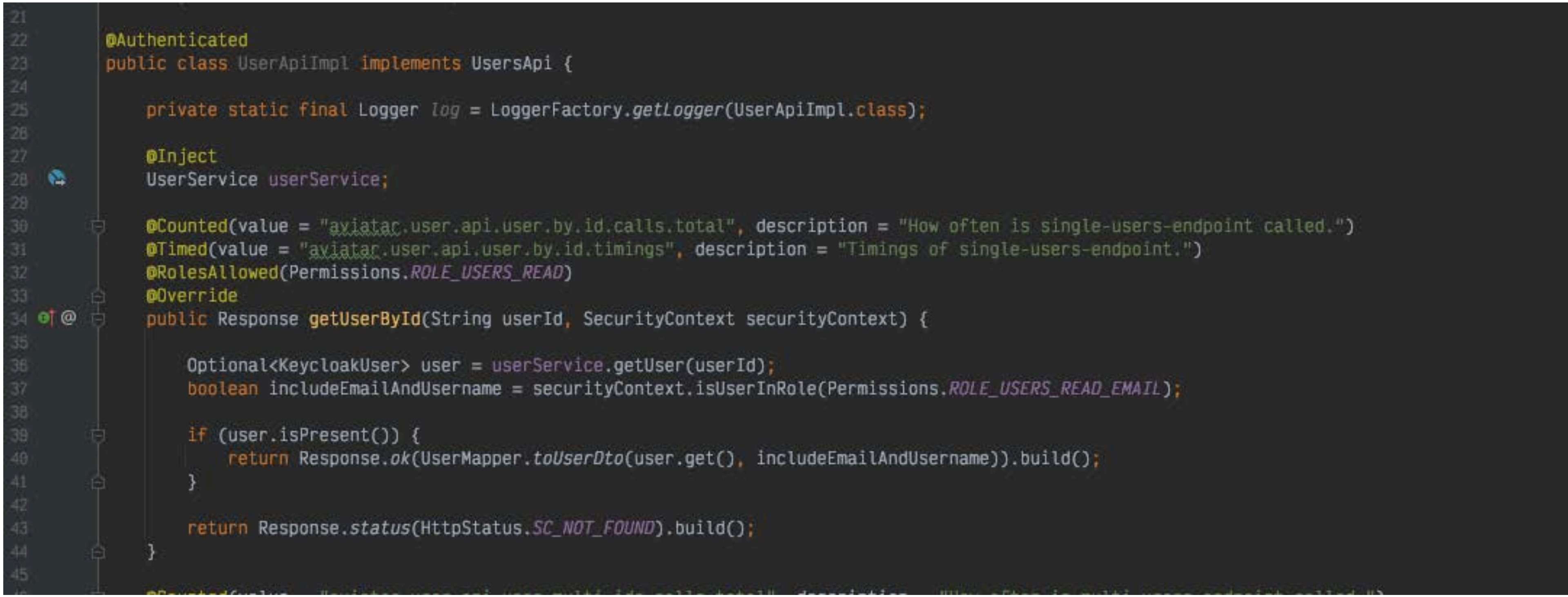

Are you allowed to know?
I need the name of user x
I need the name of user x, please
SSO
Swagger Editor
File
Edit
Generate Server
Generate Client
User API
REST API to get information of users
User
GET /users Get Users
GET /users/{userId} Get User by ID
Partner
GET /partners/{partnerId}/users Get Users
Models

```
swagger: "2.0"

info:
  version: 1.0.0
  title: User API
  description: REST API to get information of users

tags:
  - name: User
  - name: Partner

paths:

  '/users':
    get:
      tags:
        - User
      summary: Get Users
      description: Get the user information for a list of specific user
        IDs. This will include the first and last name. To also get
        usernames and emails you need to have an additional role.
      operationId: getUsers
      produces:
        - application/json
      parameters:
        - in: query
          name: userIds
          type: array
          items:
            type: string
          required: true
      responses:
        '200':
          description: Successful operation.
          schema:
            type: array
            items:
              $ref: '#/definitions/User'
        '401':
          description: Unauthorized - No user credentials provided.
```

```
User ∨ {
  ids          string
  username     string
  email        string($email)
  firstname    string
  lastname     string
}
```

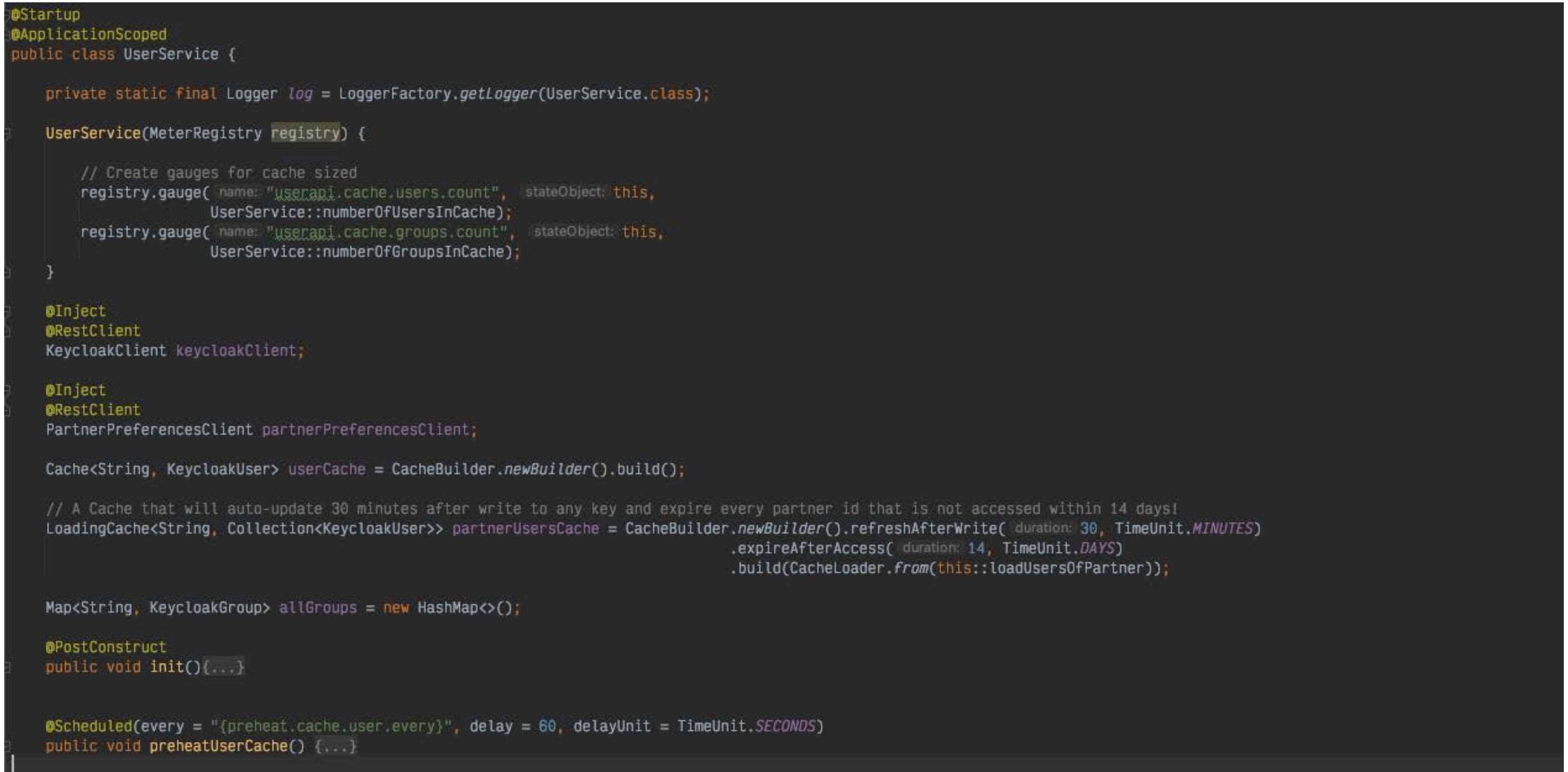
Code Example:

API-First, authenticated, metered and authorized REST API



```java
@Authenticated
public class UserApiImpl implements UsersApi {

    private static final Logger log = LoggerFactory.getLogger(UserApiImpl.class);

    @Inject
    UserService userService;

    @Counted(value = "aviatar.user.api.user.by.id.calls.total", description = "How often is single-users-endpoint called.")
    @Timed(value = "aviatar.user.api.user.by.id.timings", description = "Timings of single-users-endpoint.")
    @RolesAllowed(Permissions.ROLE_USERS_READ)
    @Override
    public Response getUserById(String userId, SecurityContext securityContext) {

        Optional<KeycloakUser> user = userService.getUser(userId);
        boolean includeEmailAndUsername = securityContext.isUserInRole(Permissions.ROLE_USERS_READ_EMAIL);

        if (user.isPresent()) {
            return Response.ok(UserMapper.toUserDto(user.get(), includeEmailAndUsername)).build();
        }

        return Response.status(HttpStatus.SC_NOT_FOUND).build();
    }

```

Cached, metric-laden, scheduled, service-consuming service with externalized configuration



```java
@Startup
@ApplicationScoped
public class UserService {

    private static final Logger log = LoggerFactory.getLogger(UserService.class);

    UserService(MeterRegistry registry) {

        // Create gauges for cache sized
        registry.gauge( name: "userapi.cache.users.count",  stateObject: this,
                       UserService::numberOfUsersInCache);
        registry.gauge( name: "userapi.cache.groups.count",  stateObject: this,
                       UserService::numberOfGroupsInCache);
    }

    @Inject
    @RestClient
    KeycloakClient keycloakClient;

    @Inject
    @RestClient
    PartnerPreferencesClient partnerPreferencesClient;

    Cache<String, KeycloakUser> userCache = CacheBuilder.newBuilder().build();

    // A Cache that will auto-update 30 minutes after write to any key and expire every partner id that is not accessed within 14 days!
    LoadingCache<String, Collection<KeycloakUser>> partnerUsersCache = CacheBuilder.newBuilder().refreshAfterWrite( duration: 30, TimeUnit.MINUTES)
                                                                            .expireAfterAccess( duration: 14, TimeUnit.DAYS)
                                                                            .build(CacheLoader.from(this::loadUsersOfPartner));

    Map<String, KeycloakGroup> allGroups = new HashMap<>();

    @PostConstruct
    public void init(){...}


    @Scheduled(every = "{preheat.cache.user.every}", delay = 60, delayUnit = TimeUnit.SECONDS)
    public void preheatUserCache() {...}
```

What is there not to like?

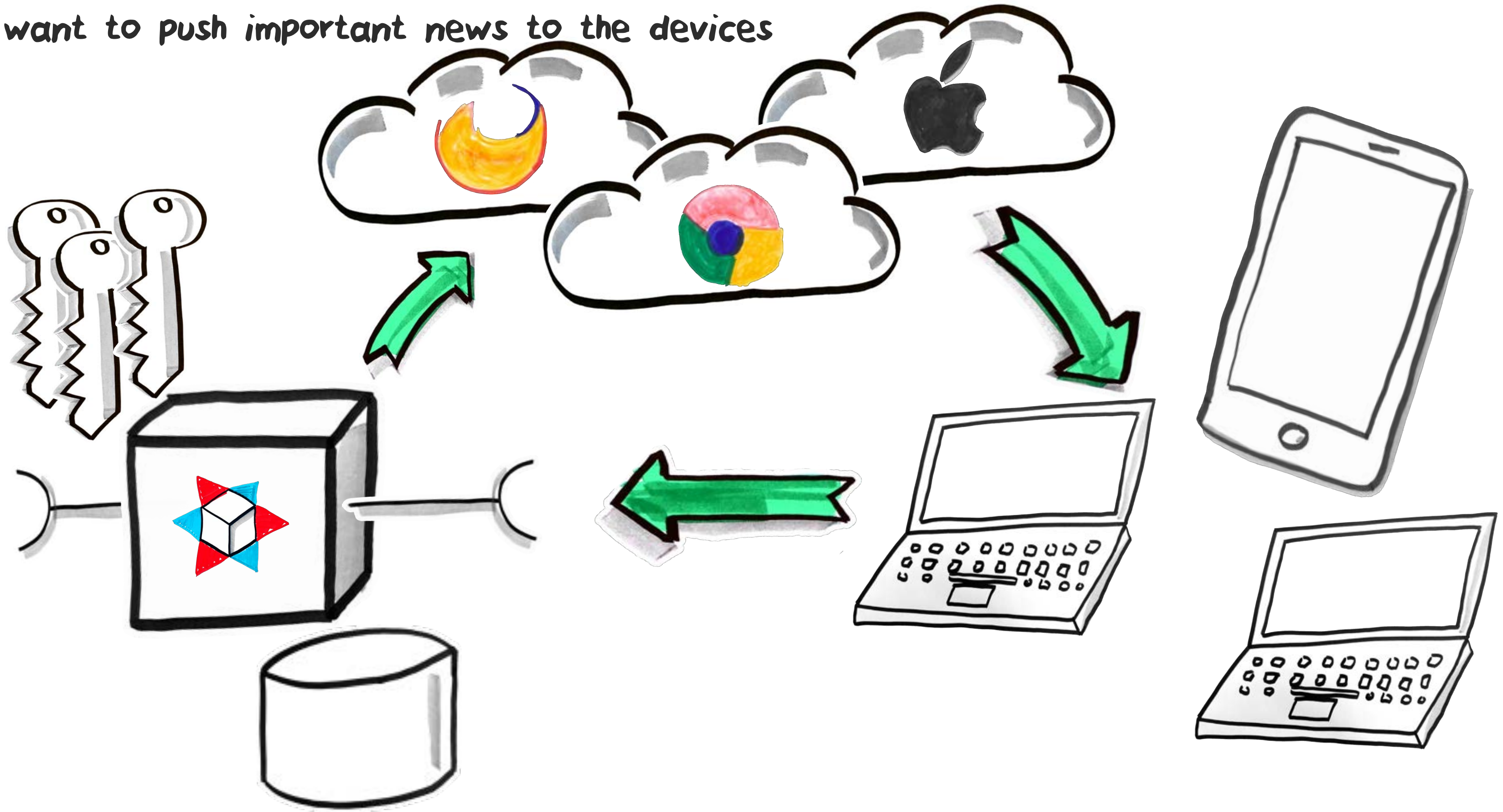
```
> oc describe limits
Name:        user-api-dev-limits
Namespace:   user-api-dev
Type         Resource  Min   Max      Default Request  Default Limit  Max Limit/Request Ratio
----         --------  ---   ---      ---------------  -------------  -----------------------
Pod          cpu       50m   1        -                -              -
Pod          memory    50Mi  8000Mi   -                -              -
Container    cpu       50m   1        50m              150m           -
Container    memory    50Mi  8000Mi   50Mi             250Mi          -
```

Better bring some RAM and Time

```
> oc get build
NAME            TYPE     FROM           STATUS     STARTED       DURATION
user-api-146    Source   Git@8afdb9b    Complete   7 weeks ago   29m16s
user-api-147    Source   Git@37e10fd    Complete   4 weeks ago   19m28s
```

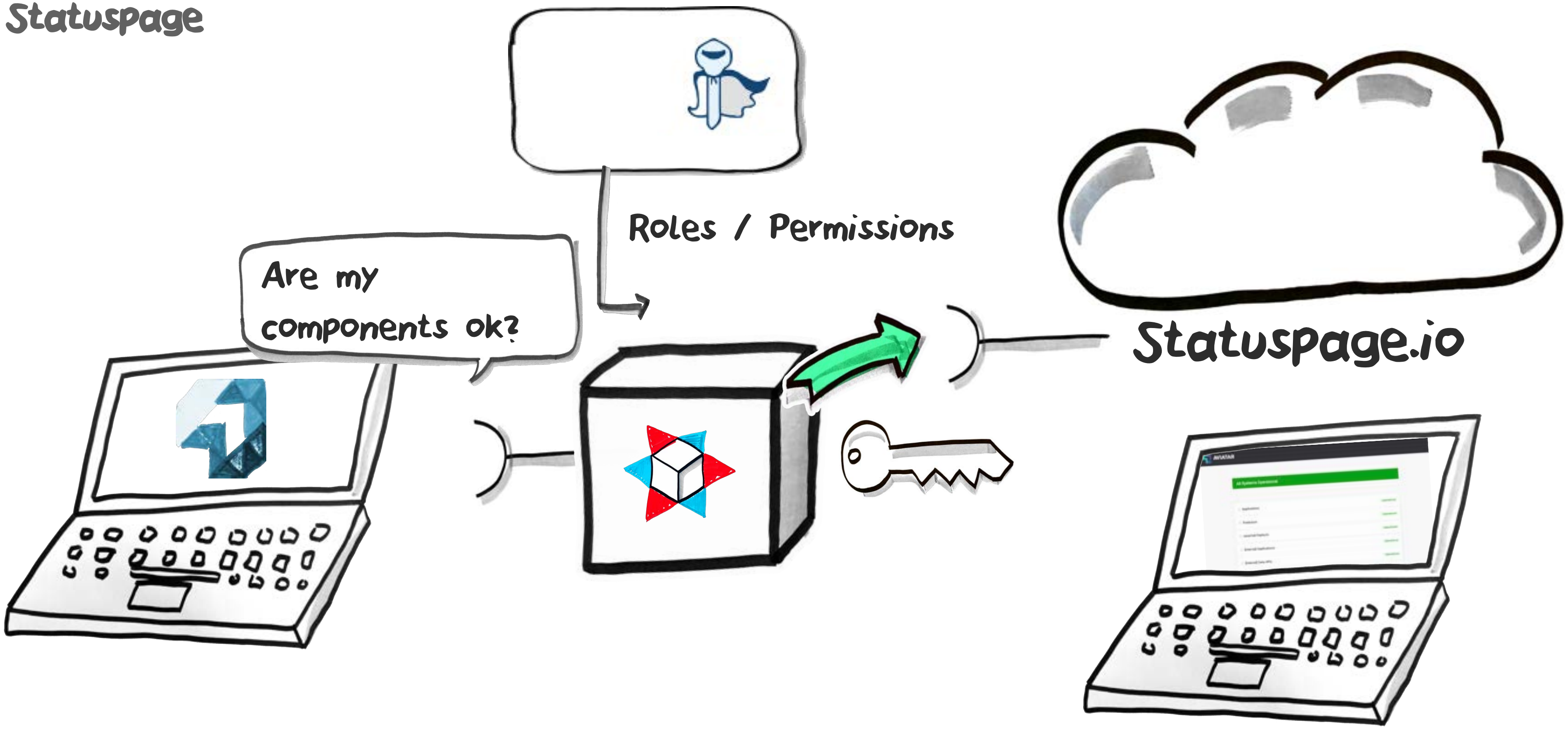
Use Case Example: Notification API

I just want to push important news to the devices



Use Case Example: Integrate a SaaS Offering

Statuspage



Resources: Do the math

10 Microservices

* 3 Stages

* 2 Instances

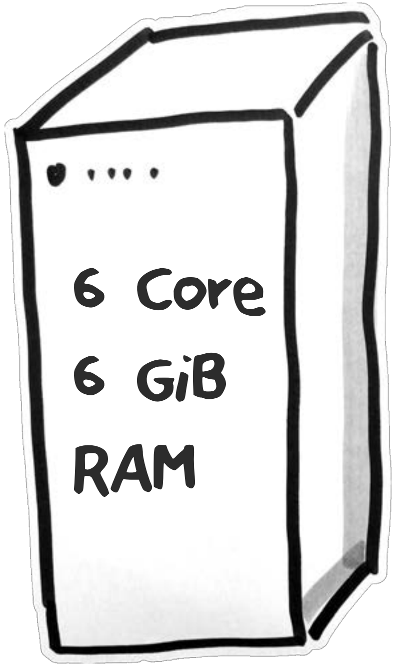
= 60 Instances

What if, each instance uses 100mCores instead of 250mCores

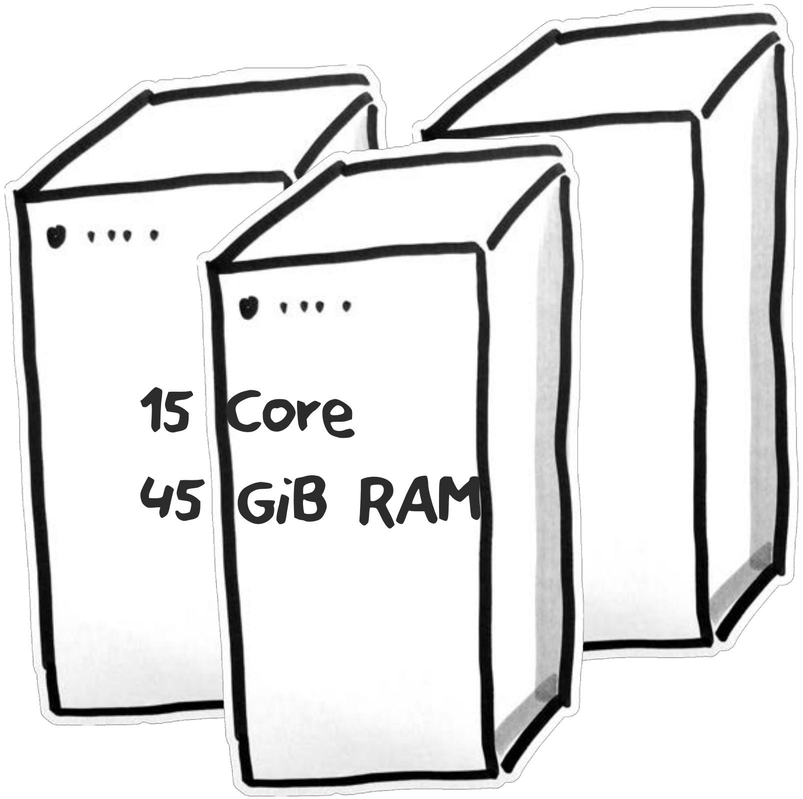
What if, each instance uses 100MiB of RAM instead of 750MiB or RAM

Resources: Do the math

We can overbook CPUs, but not RAM

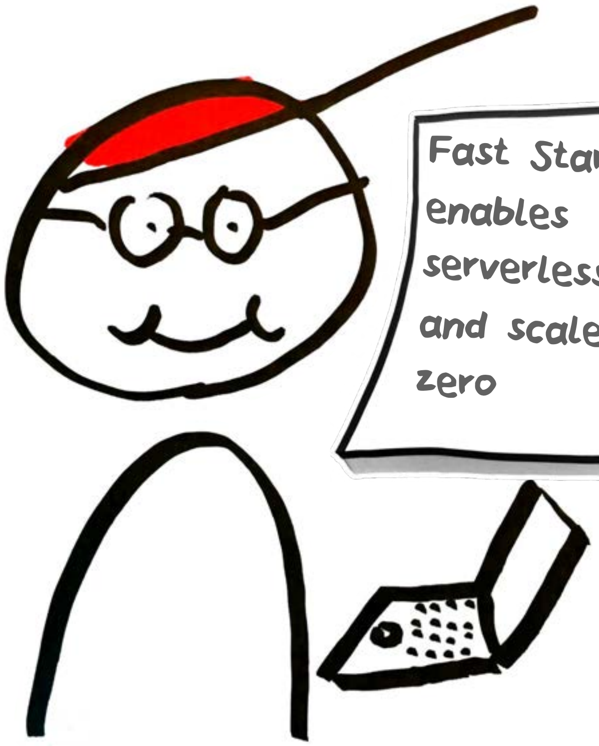


vs.



Summary

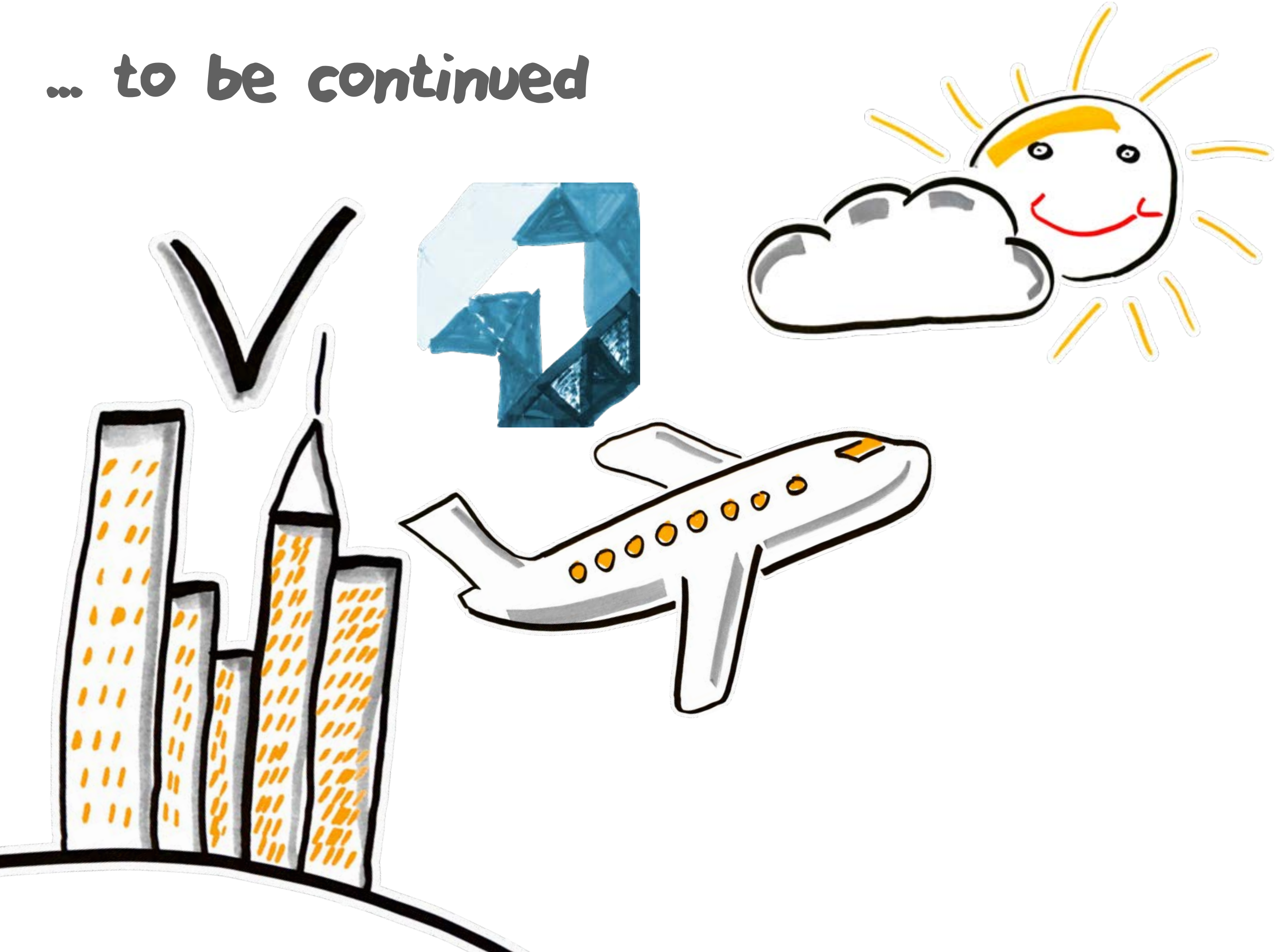
Small Footprint enables finer-grained Microservices



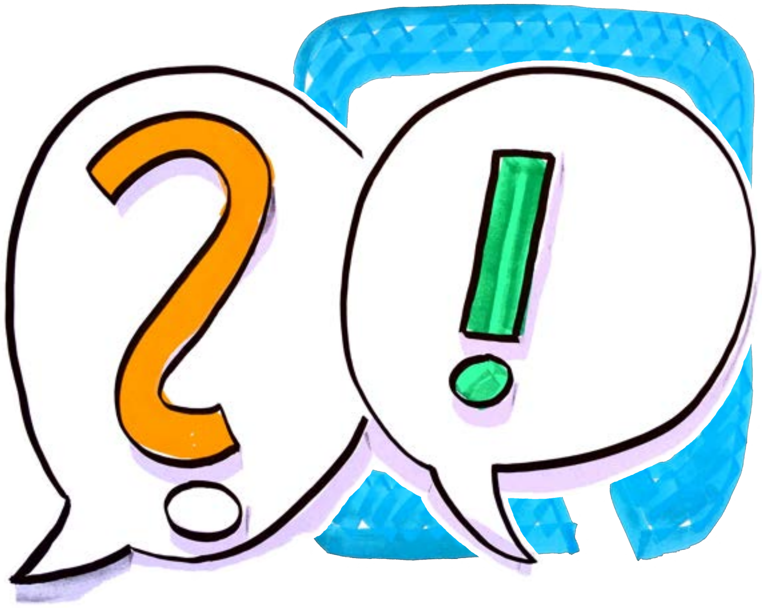
Fast Startup enables serverless and scale to zero

Container platform plus tooling enables microservices

… to be continued



Useful resources



- https://quarkus.io
- https://code.quarkus.io
- https://twitter.com/QuarkusIO
- https://github.com/quarkusio
- https://thorsten.pro

- Red Hat Summit 2020 Talk „Making Java Subatomic„

- Quarkus Blog about AVIATAR:
  https://quarkus.io/blog/aviatar-experiences-significant-savings/
- AVIATAR Innovation Award 2018:
  https://www.redhat.com/en/blog/announcing-winners-12th-annual-red-hat-innovation-awards
  https://www.redhat.com/en/success-stories/lufthansa-technik