



JAVA
USER
GROUP
CH

Thriving in the Cloud:

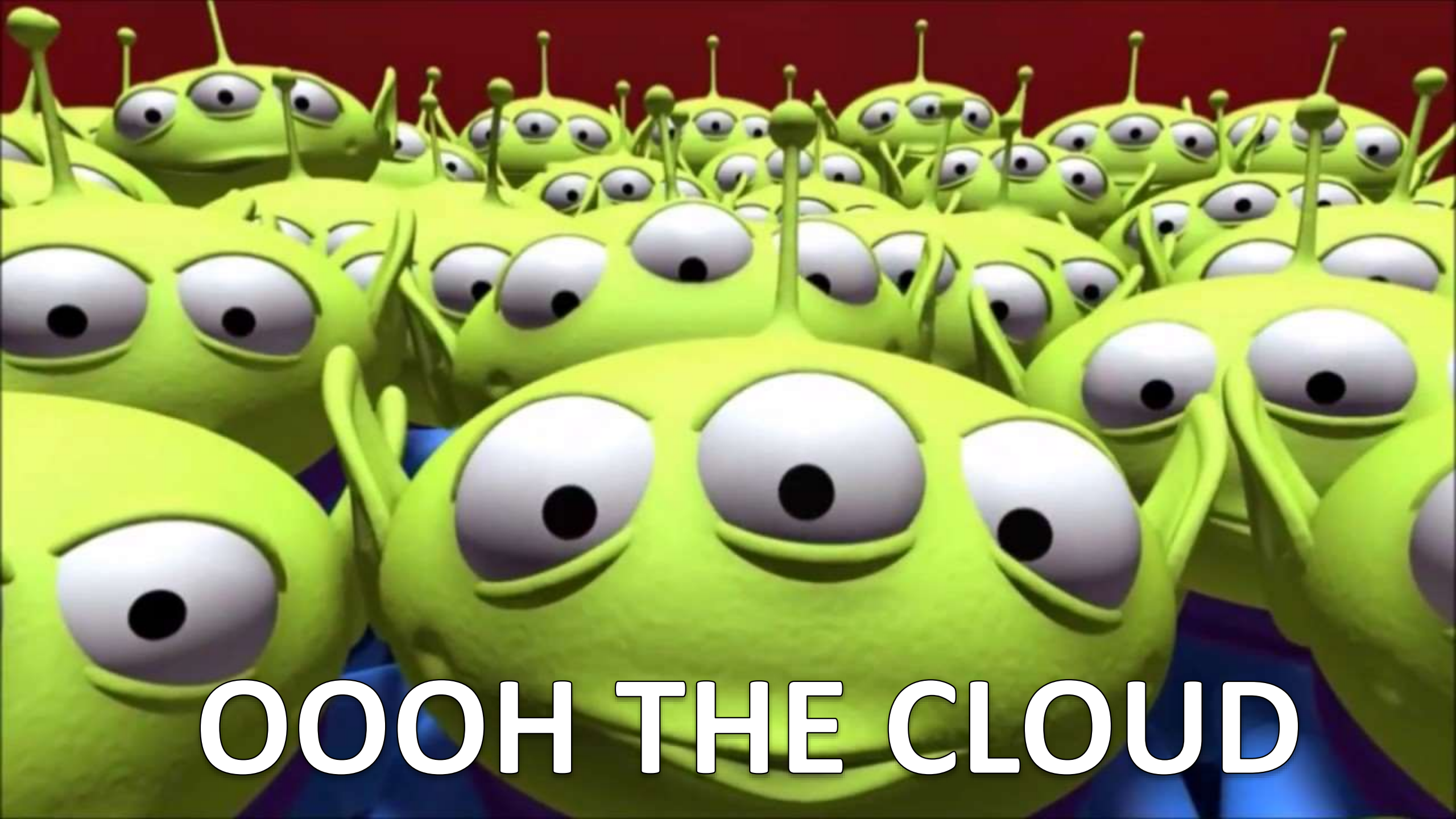
Going Beyond the 12
Factors



GRACE, JANSEN

DEVELOPER ADVOCATE, IBM

@gracejansen27



OOOH THE CLOUD

What does the Cloud offer?

Co\$t

Speed

Scalability

Resiliency

Fashion

Flexibility

Innovation





12 Factor Apps



12 Factor App Methodology



THE TWELVE-FACTOR APP

INTRODUCTION

In the modern era, software is commonly delivered as a service: called *web apps*, or *software-as-a-service*. The twelve-factor app is a methodology for building software-as-a-service apps that:

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.

The twelve-factor methodology can be applied to apps written in any programming language, and which use any combination of backing services (database, queue, memory cache, etc).

BACKGROUND

The contributors to this document have been directly involved in the development and deployment of hundreds of apps, and indirectly witnessed the development, operation, and scaling of hundreds of thousands of apps via our work on the Heroku platform.

This document synthesizes all of our experience and observations on a wide variety of software-as-a-service apps in the wild. It is a triangulation on ideal practices for app development, paying particular attention to the dynamics of the organic growth of an app over time, the dynamics of collaboration between developers working on the app's codebase, and avoiding the cost of software erosion.

Our motivation is to raise awareness of some systemic problems we've seen in modern application development, to provide a shared vocabulary for discussing those problems, and to offer a set of broad conceptual solutions to those problems with accompanying terminology. The format is inspired by Martin Fowler's books *Patterns of Enterprise Application Architecture* and *Refactoring*.

<https://12factor.net/>

The original 12 factors:

1. Codebase
2. Dependencies
3. Configuration
4. Backing Services
5. Build, release, run
6. Processes
7. Port Binding
8. Concurrency
9. Disposability
10. Dev/prod parity
11. Logs
12. Admin Processes

Revised 15 factors

1. One Codebase, one application
2. API first
3. Dependency management
4. Design, build, release, and run
5. Configuration, credentials and code
6. Logs
7. Disposability
8. Backing services
9. Environment parity
10. Administrative processes
11. Port binding
12. Stateless processes
13. Concurrency
14. Telemetry
15. Authentication and authorization

Revised 15 factors

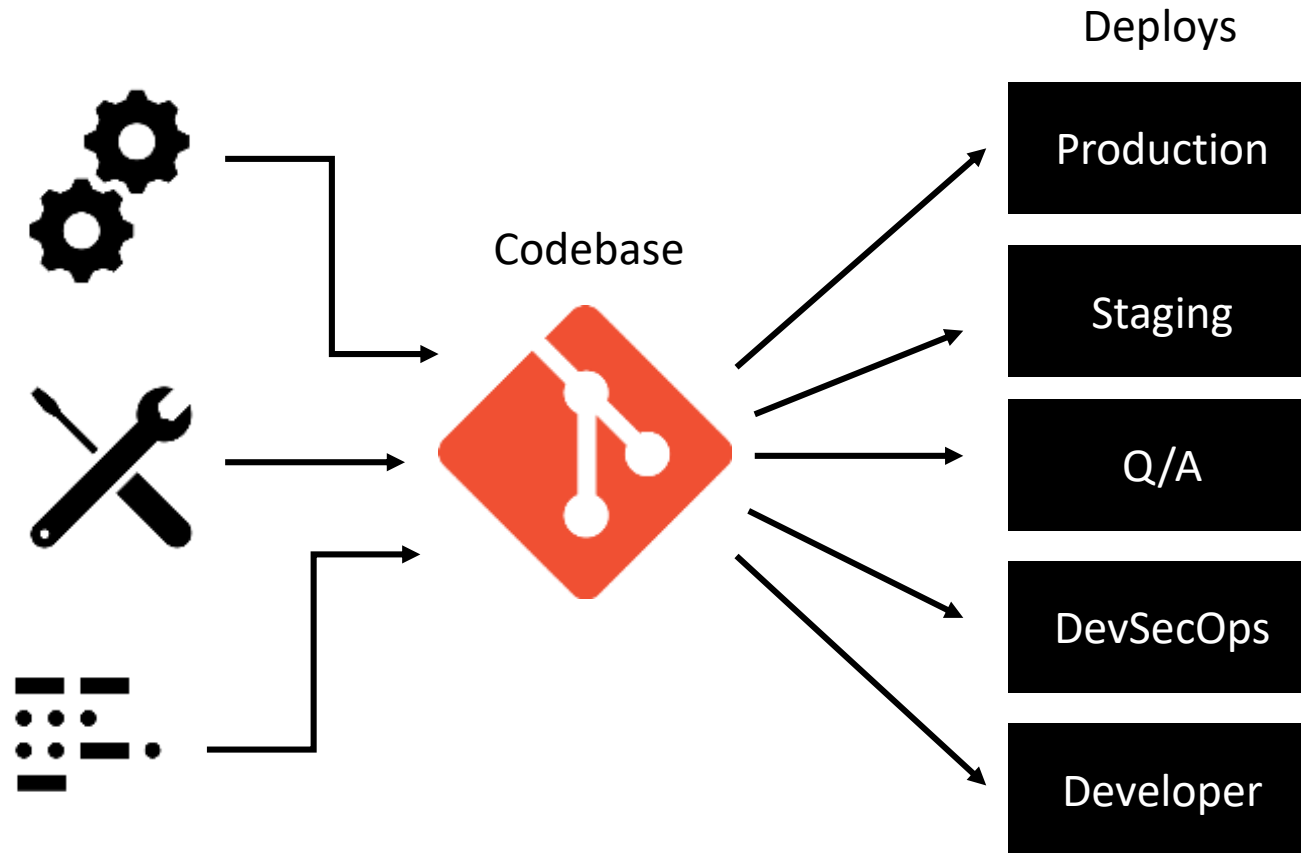
1. One Codebase, one application
2. API first
3. Dependency management
4. Design, build, release, and run
5. Configuration, credentials and code
6. Logs
7. Disposability
8. Backing services
9. Environment parity
10. Administrative processes
11. Port binding
12. Stateless processes
13. Concurrency
14. Telemetry
15. Authentication and authorization



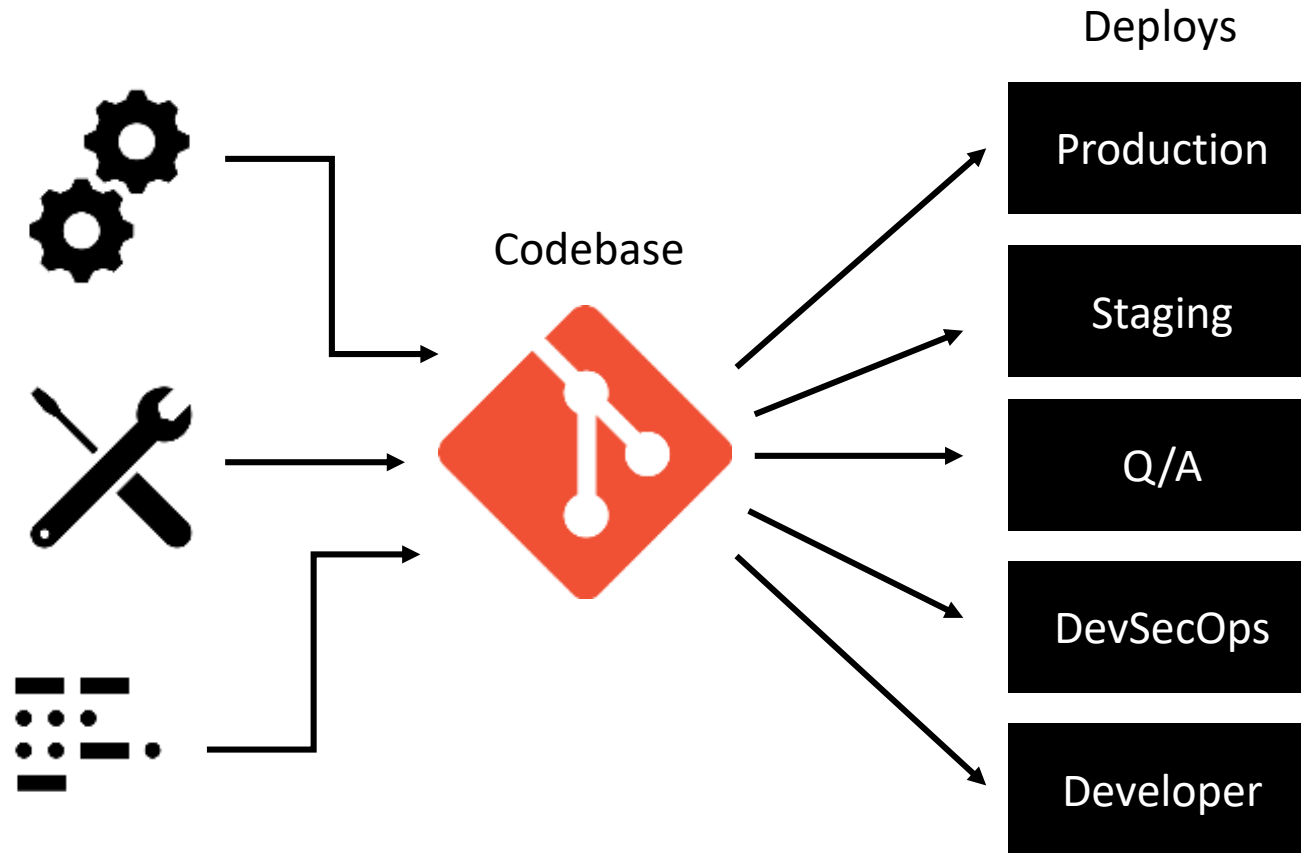
The Original 12 Factors



1. One codebase, one application



1. One codebase, one application



3. Dependency management

Application Source Code

Makes use of...
→

```
21
22 ✓ <dependencies>
23   <!-- Provided dependencies -->
24   <dependency>
25     <groupId>jakarta.platform</groupId>
26     <artifactId>jakarta.jakartaee-api</artifactId>
27     <version>8.0.0</version>
28     <scope>provided</scope>
29   </dependency>
30   <dependency>
31     <groupId>org.eclipse.microprofile</groupId>
32     <artifactId>microprofile</artifactId>
33     <version>4.0.1</version>
34     <type>pom</type>
35     <scope>provided</scope>
36   </dependency>
37   <!-- For tests -->
38   <dependency>
39     <groupId>org.junit.jupiter</groupId>
40     <artifactId>junit-jupiter</artifactId>
41     <version>5.7.1</version>
42     <scope>test</scope>
43   </dependency>
```


3. Dependency management

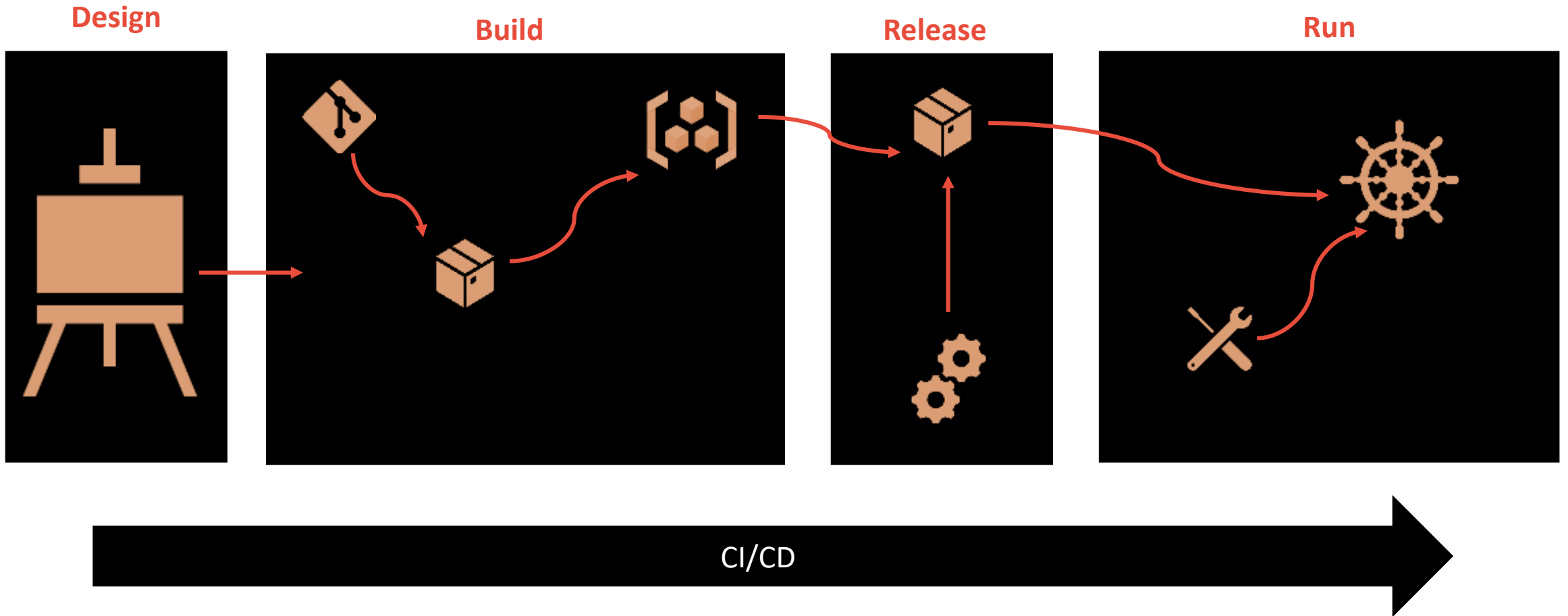
Application Source Code

Makes use of...
→

```
21
22 <dependencies>
23   <!-- Provided dependencies -->
24   <dependency>
25     <groupId>jakarta.platform</groupId>
26     <artifactId>jakarta.jakartaee-api</artifactId>
27     <version>8.0.0</version>
28     <scope>provided</scope>
29   </dependency>
30   <dependency>
31     <groupId>org.eclipse.microprofile</groupId>
32     <artifactId>microprofile</artifactId>
33     <version>4.0.1</version>
34     <type>pom</type>
35     <scope>provided</scope>
36   </dependency>
37   <!-- For tests -->
38   <dependency>
39     <groupId>org.junit.jupiter</groupId>
40     <artifactId>junit-jupiter</artifactId>
41     <version>5.7.1</version>
42     <scope>test</scope>
43   </dependency>
```

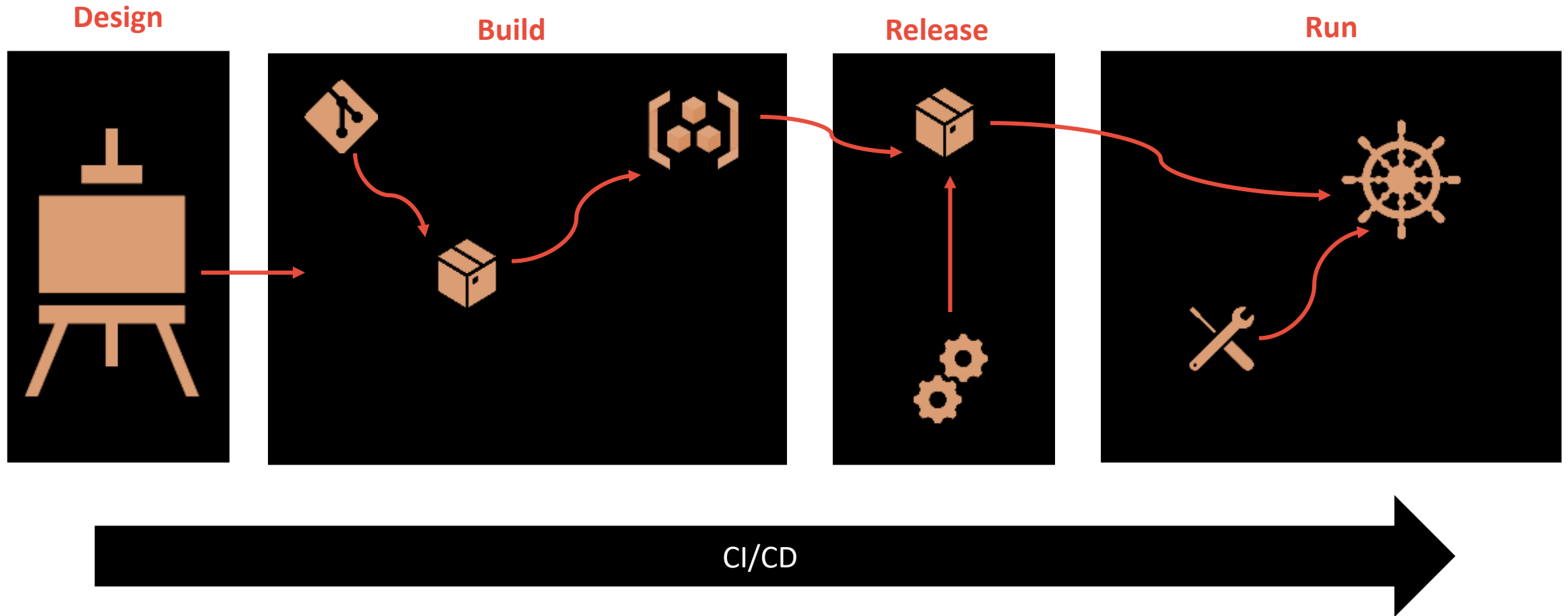


4. Design, build, release, run





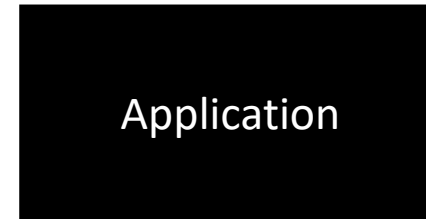
4. Design, build, release, run



5. Configuration, credentials and code

```
15 <variable name="default.http.port" defaultValue="9088"/>
16 <variable name="default.https.port" defaultValue="9443"/>
17
18 <webApplication location="guide-getting-started.war" contextRoot="/dev" />
19
20 <mpMetrics authentication="false"/>
21
22 <!-- tag::logging[] -->
23 <logging traceSpecification="com.ibm.ws.microprofile.health,=*all" />
24 <!-- end::logging[] -->
25
26 <httpEndpoint host="*" httpPort="${default.http.port}"
27   httpsPort="${default.https.port}" id="defaultHttpEndpoint"/>
28
```

Uses...
←



5. Configuration, credentials and code

```
15 <variable name="default.http.port" defaultValue="9088"/>
16 <variable name="default.https.port" defaultValue="9443"/>
17
18 <webApplication location="guide-getting-started.war" contextRoot="/dev" />
19
20 <mpMetrics authentication="false"/>
21
22 <!-- tag::logging[] -->
23 <logging traceSpecification="com.ibm.ws.microprofile.health,=*all" />
24 <!-- end::logging[] -->
25
26 <httpEndpoint host="*" httpPort="${default.http.port}"
27     httpsPort="${default.https.port}" id="defaultHttpEndpoint"/>
28
```

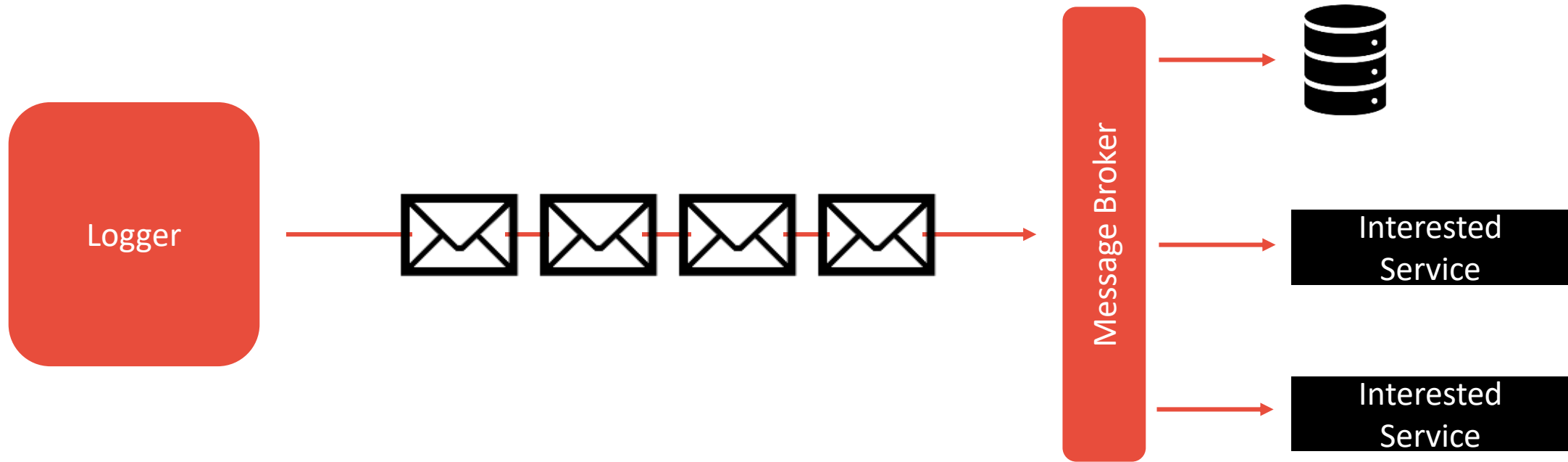
Uses...
←

Application

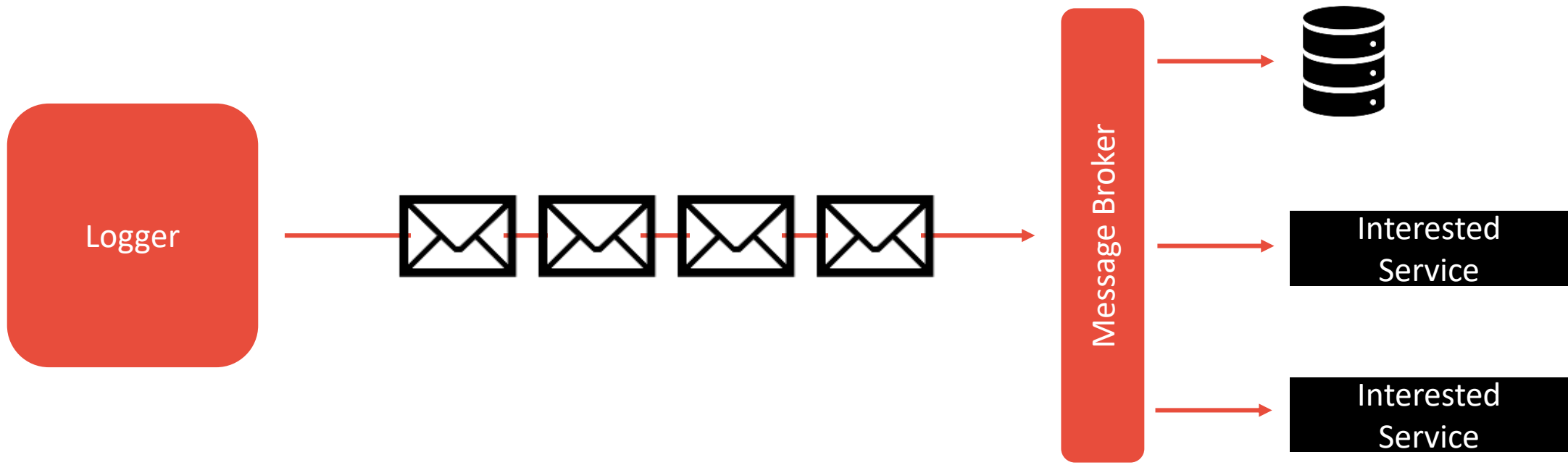


<https://openliberty.io/guides/microprofile-config.html>

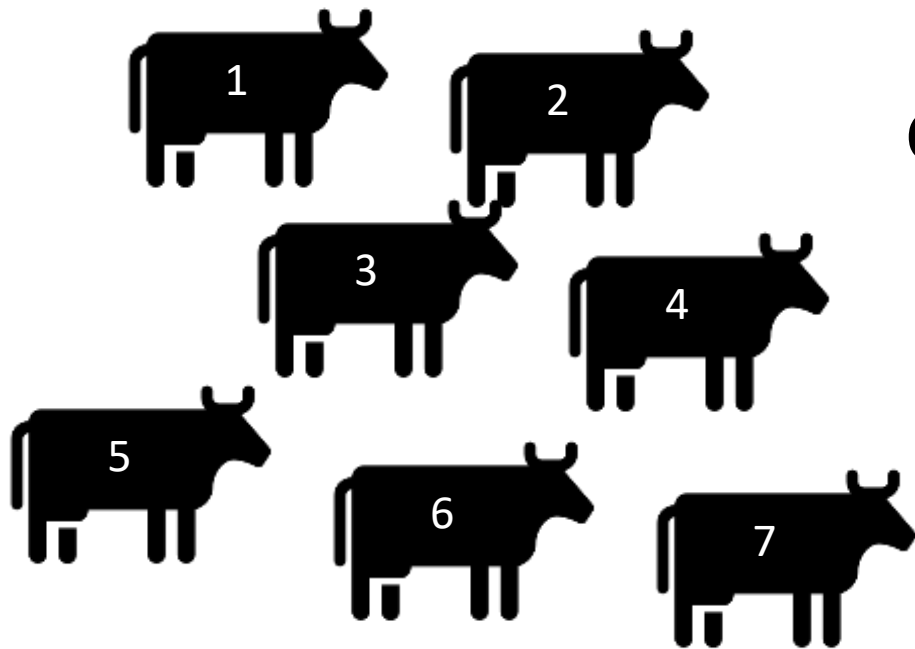
6. Logs



6. Logs



7. Disposability



Cattle

VS

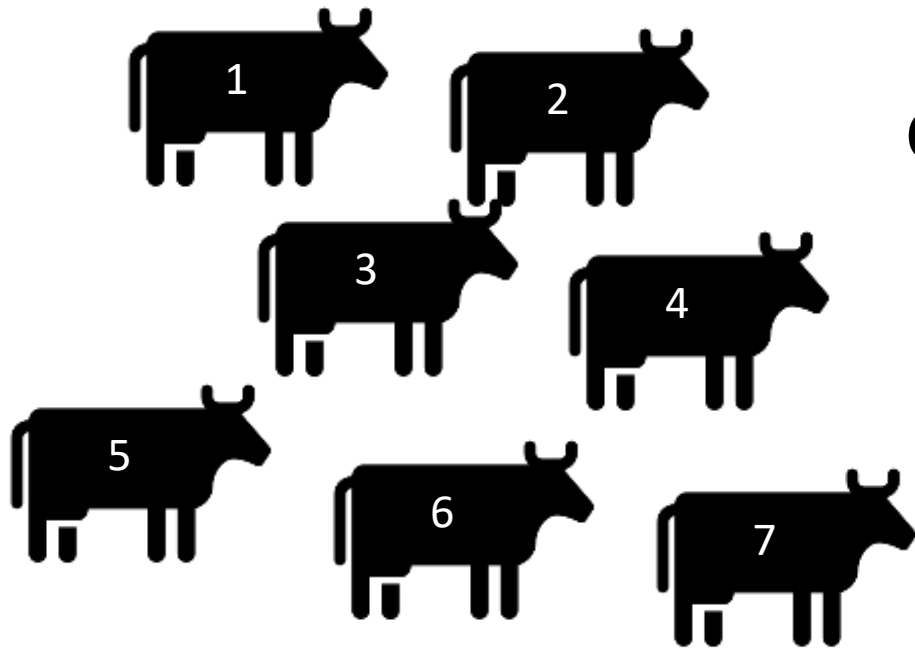
Pet



Carl the Cat



7. Disposability



Cattle

VS

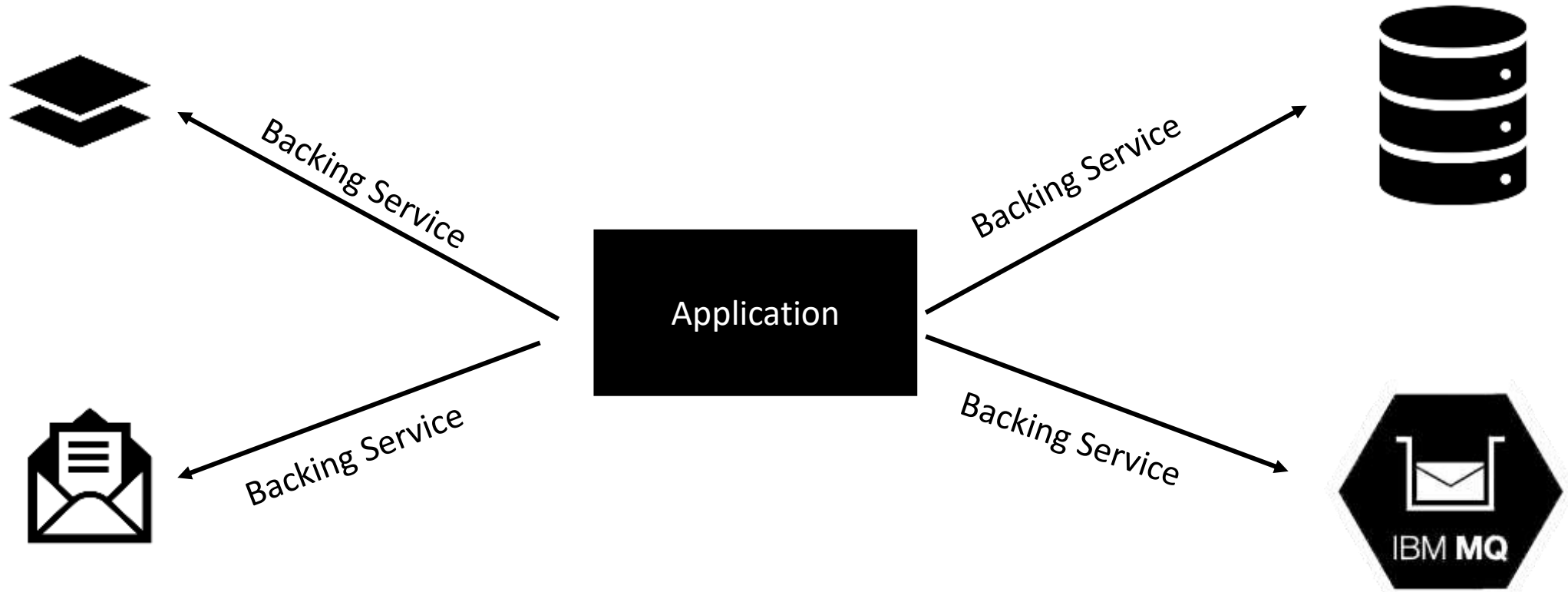
Pet



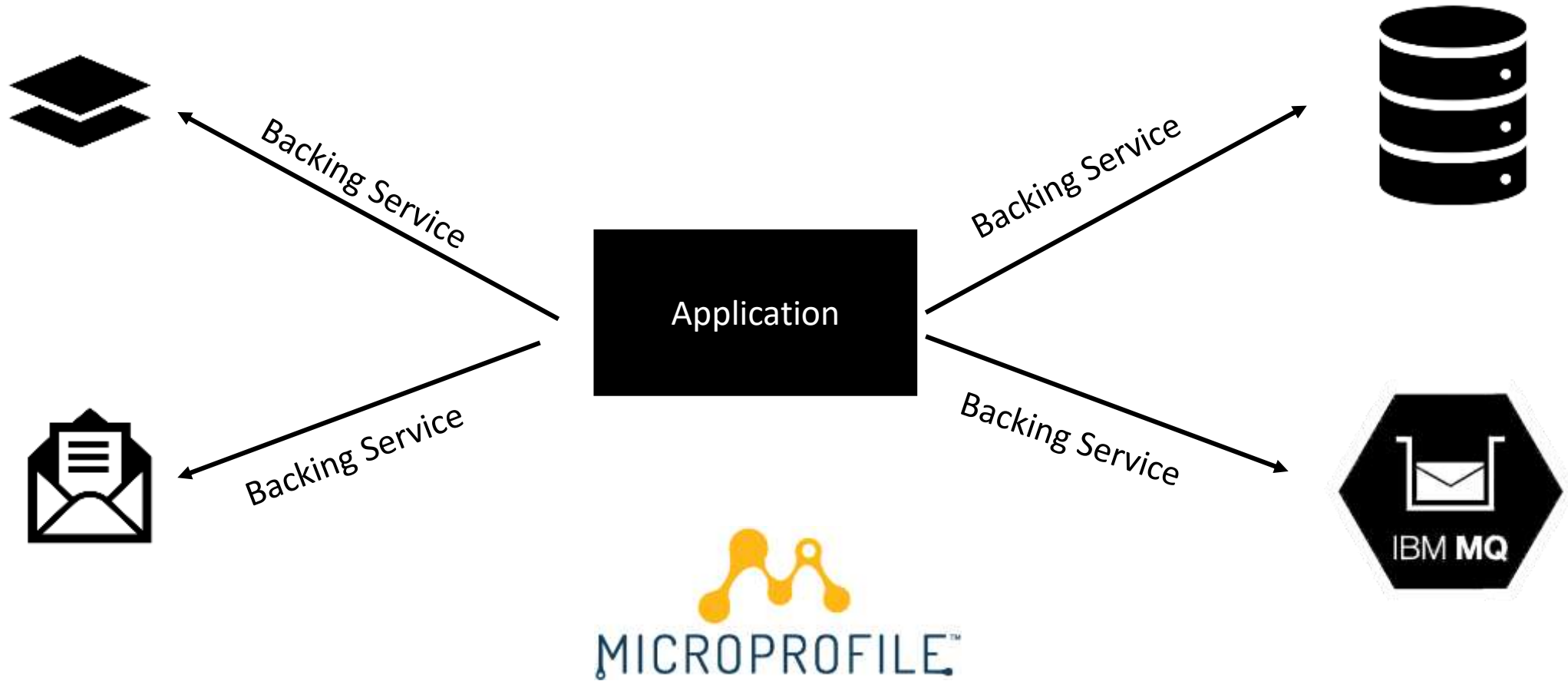
Carl the Cat

<https://openliberty.io/guides/microprofile-fallback.html>

8. Backing Services



8. Backing Services



9. Environmental parity

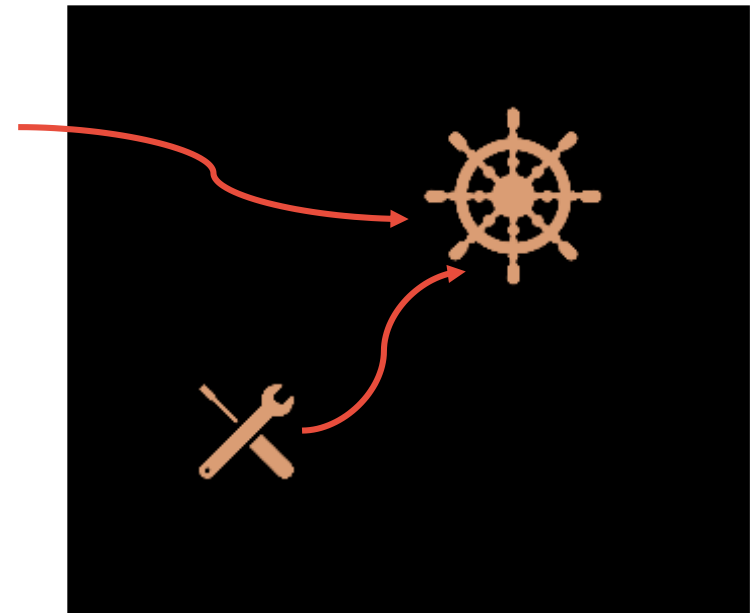
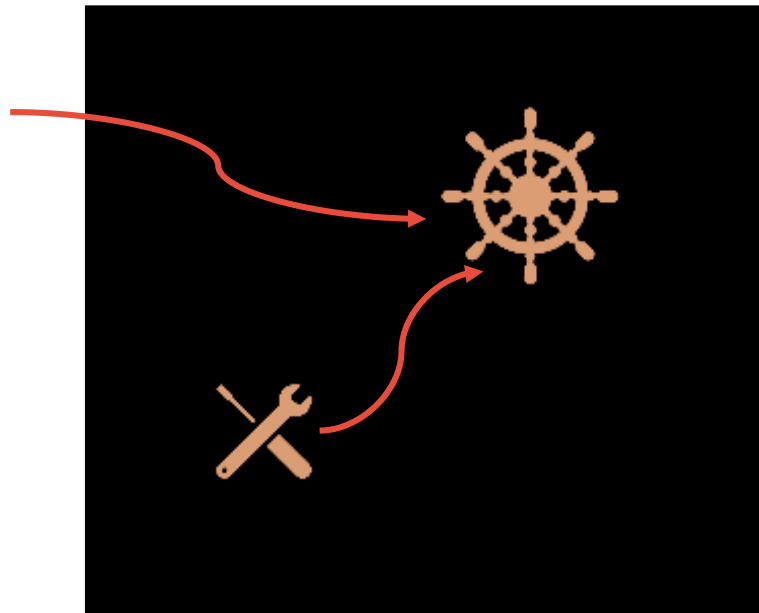
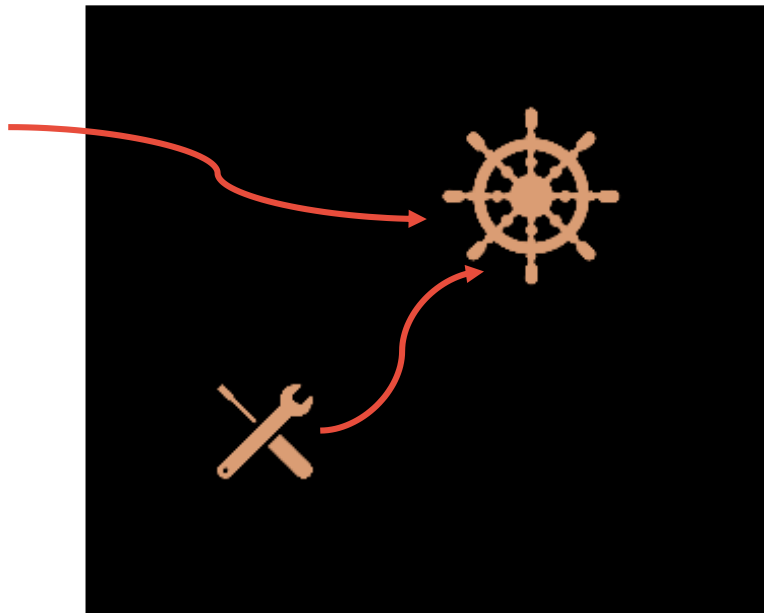
Development

=

Q/A

=

Production



9. Environmental parity



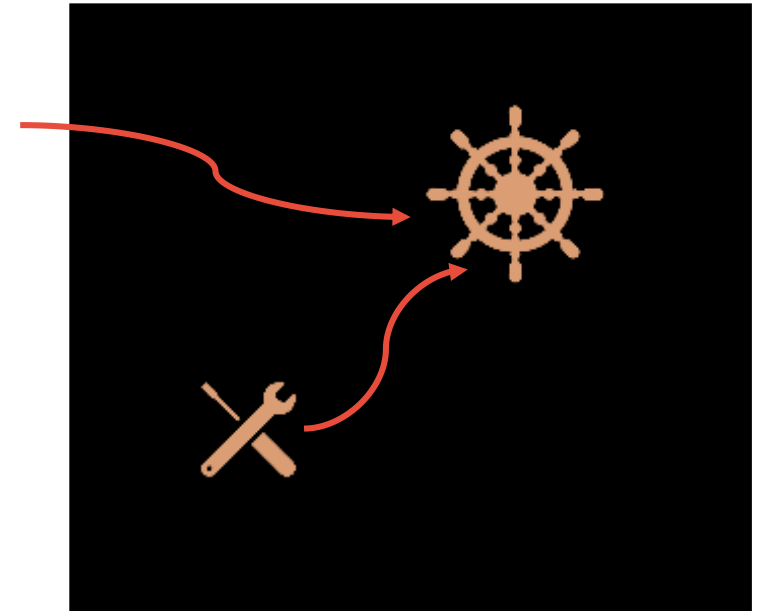
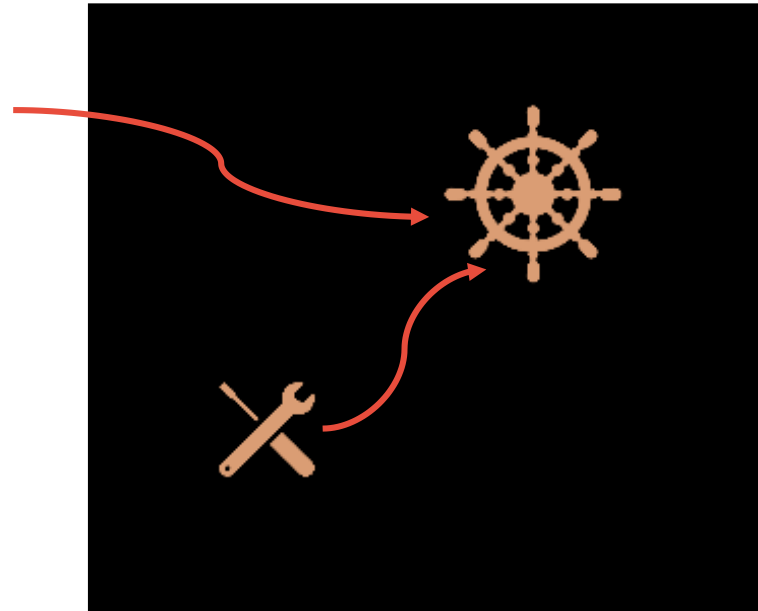
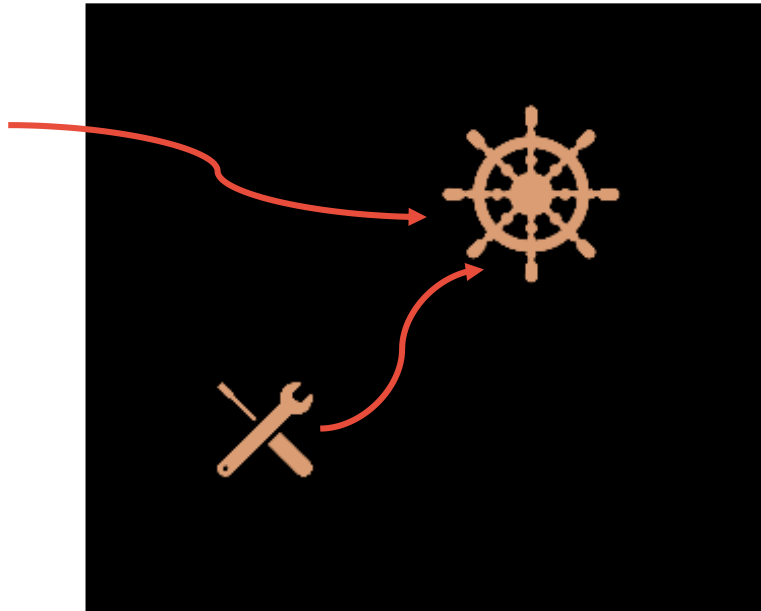
Development

=

Q/A

=

Production



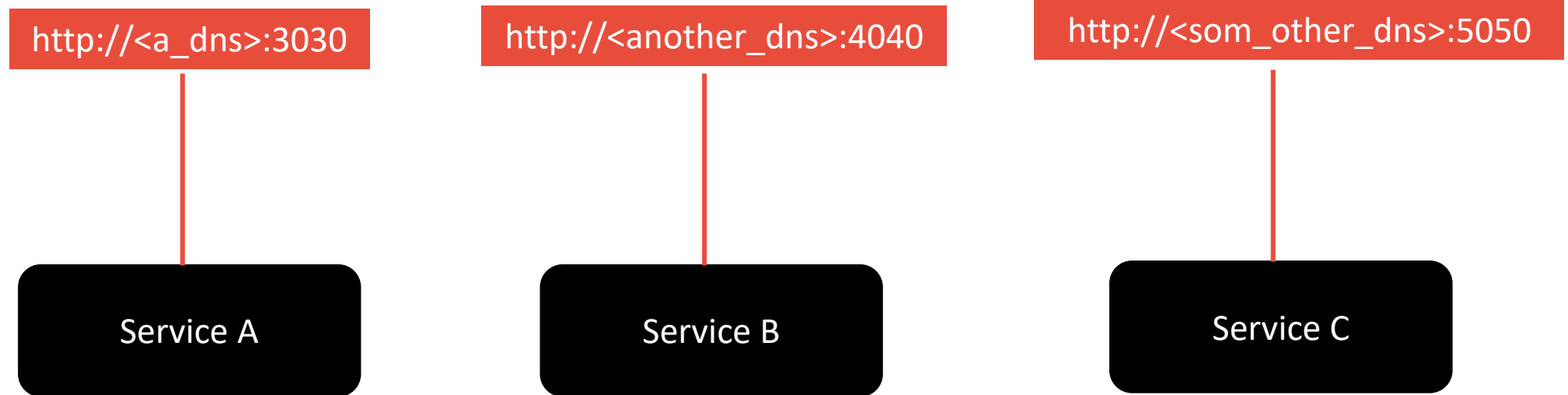
<https://openliberty.io/guides/microshed-testing.html>

10. Administrative processes



<https://openliberty.io/guides/kubernetes-intro.html>

11. Port Binding



11. Port Binding



http://<a_dns>:3030

Service A

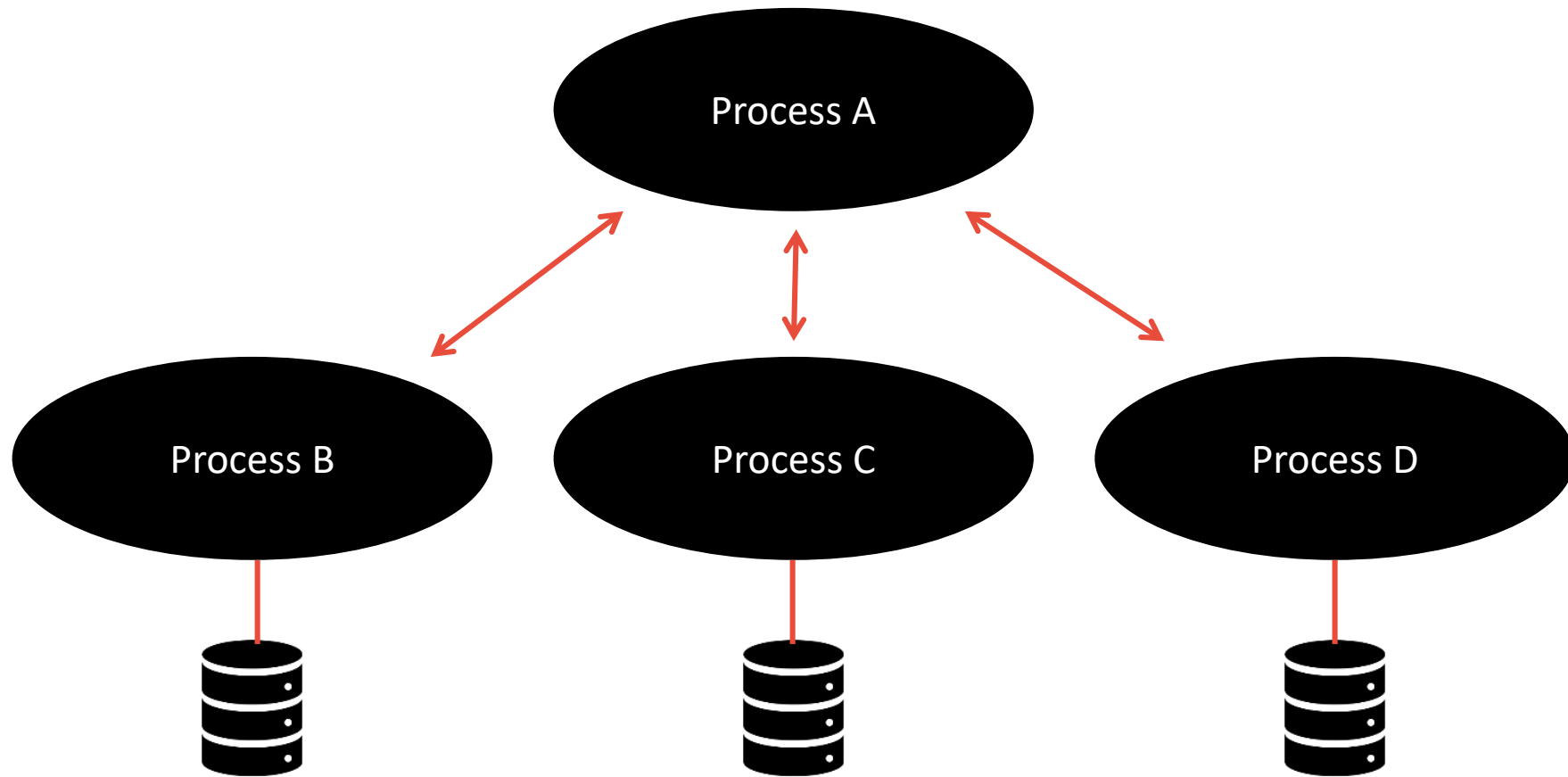
http://<another_dns>:4040

Service B

http://<som_other_dns>:5050

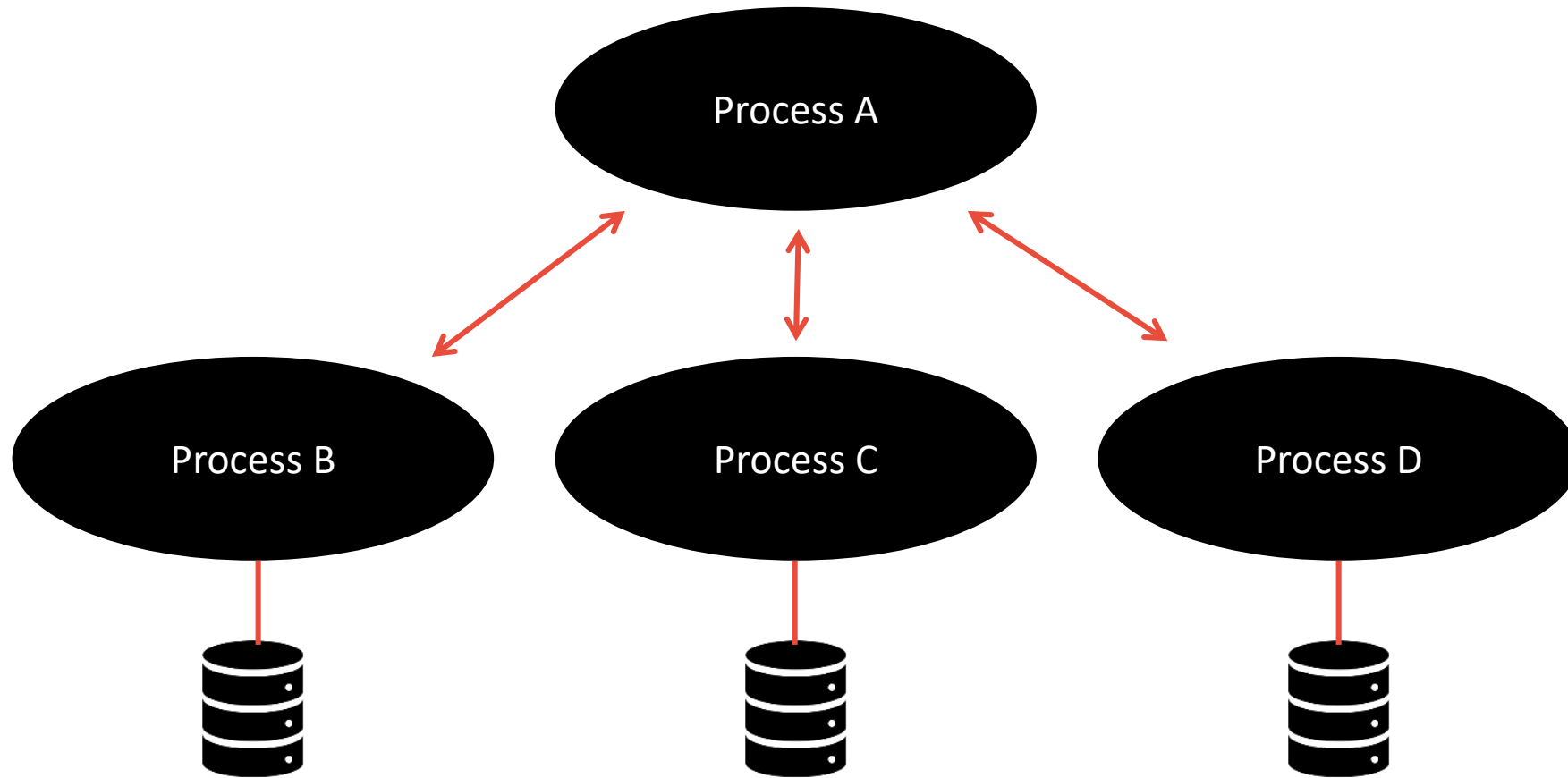
Service C

12. Stateless processes



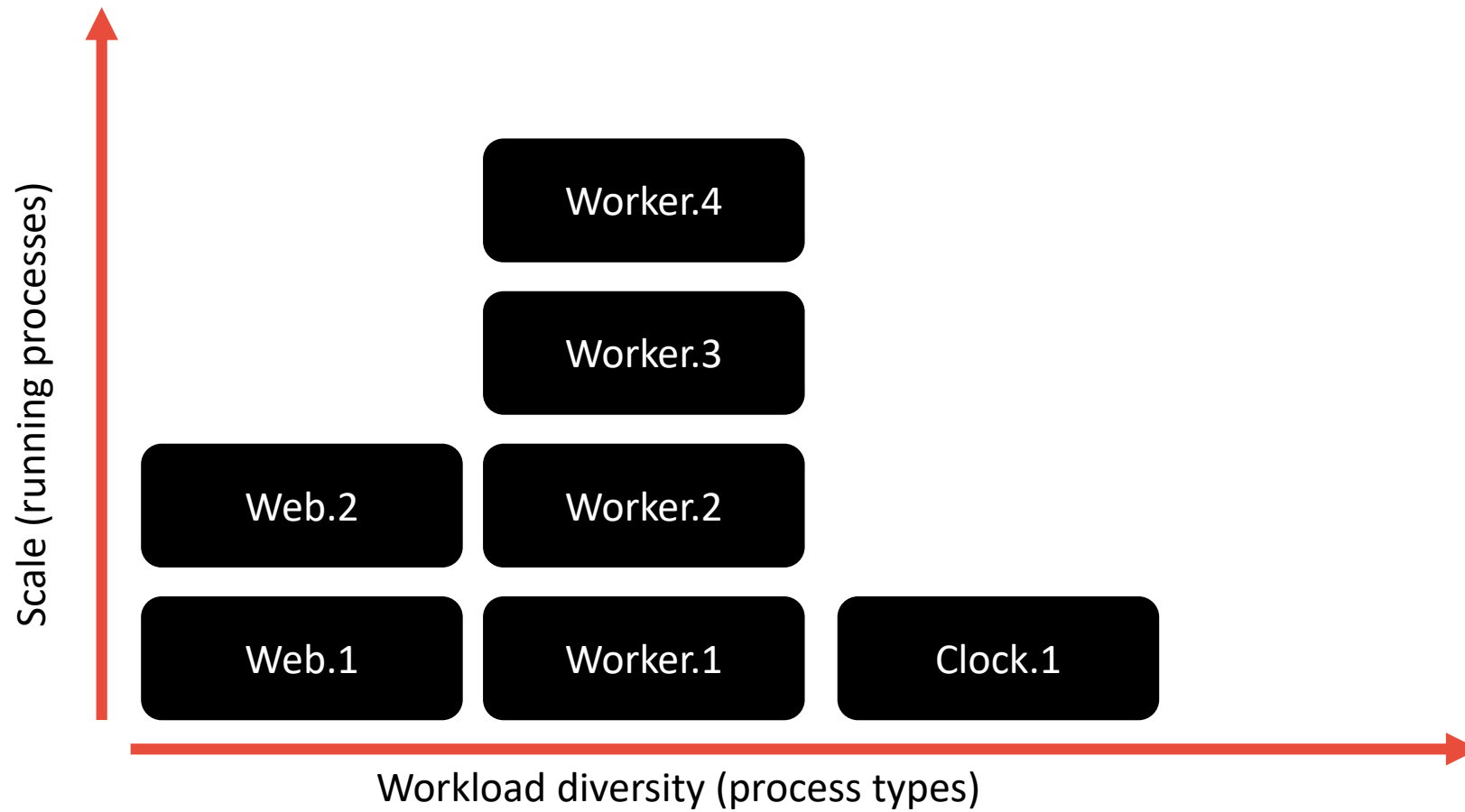


12. Stateless processes



<https://openliberty.io/guides/rest-intro.html>

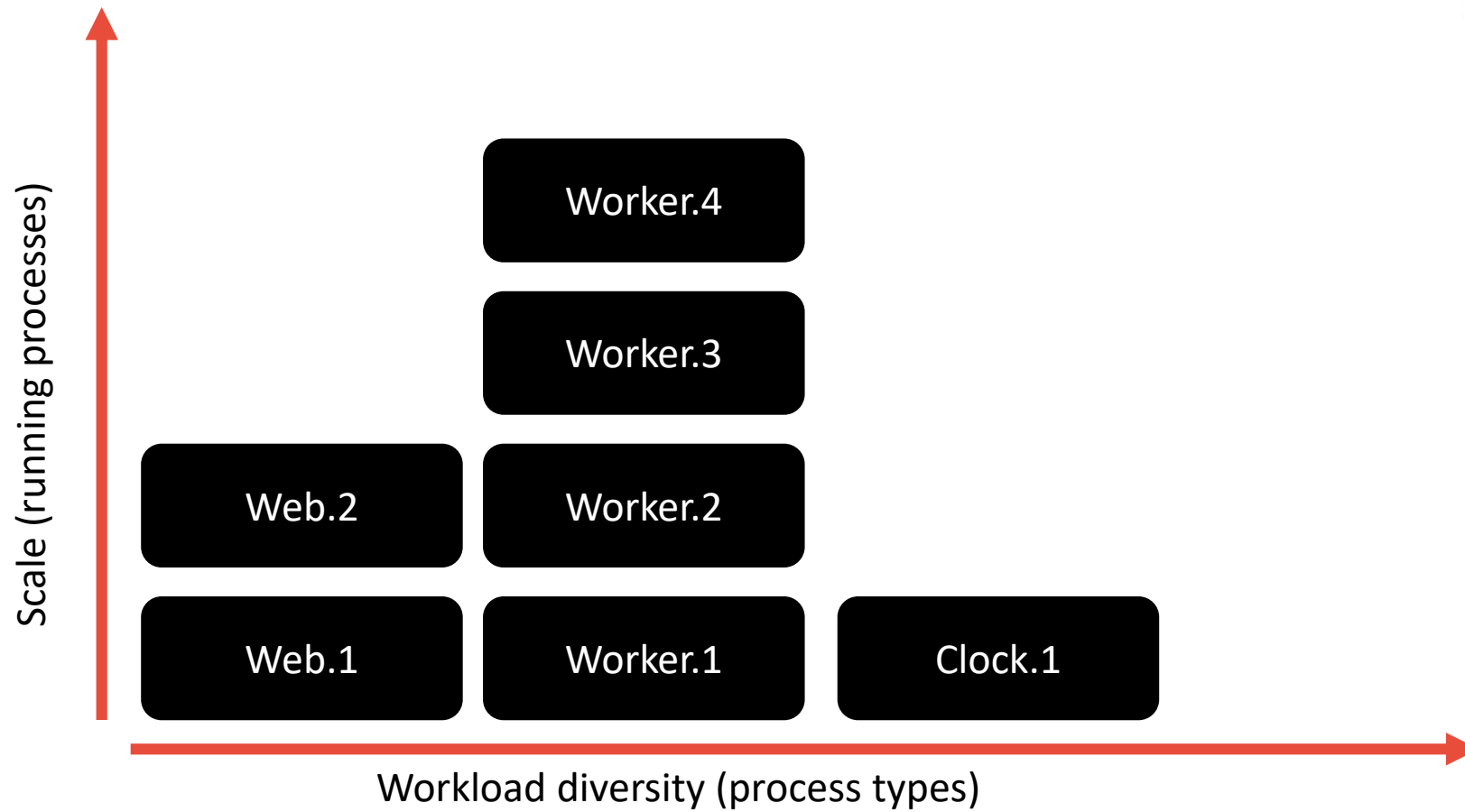
13. Concurrency





kubernetes

13. Concurrency

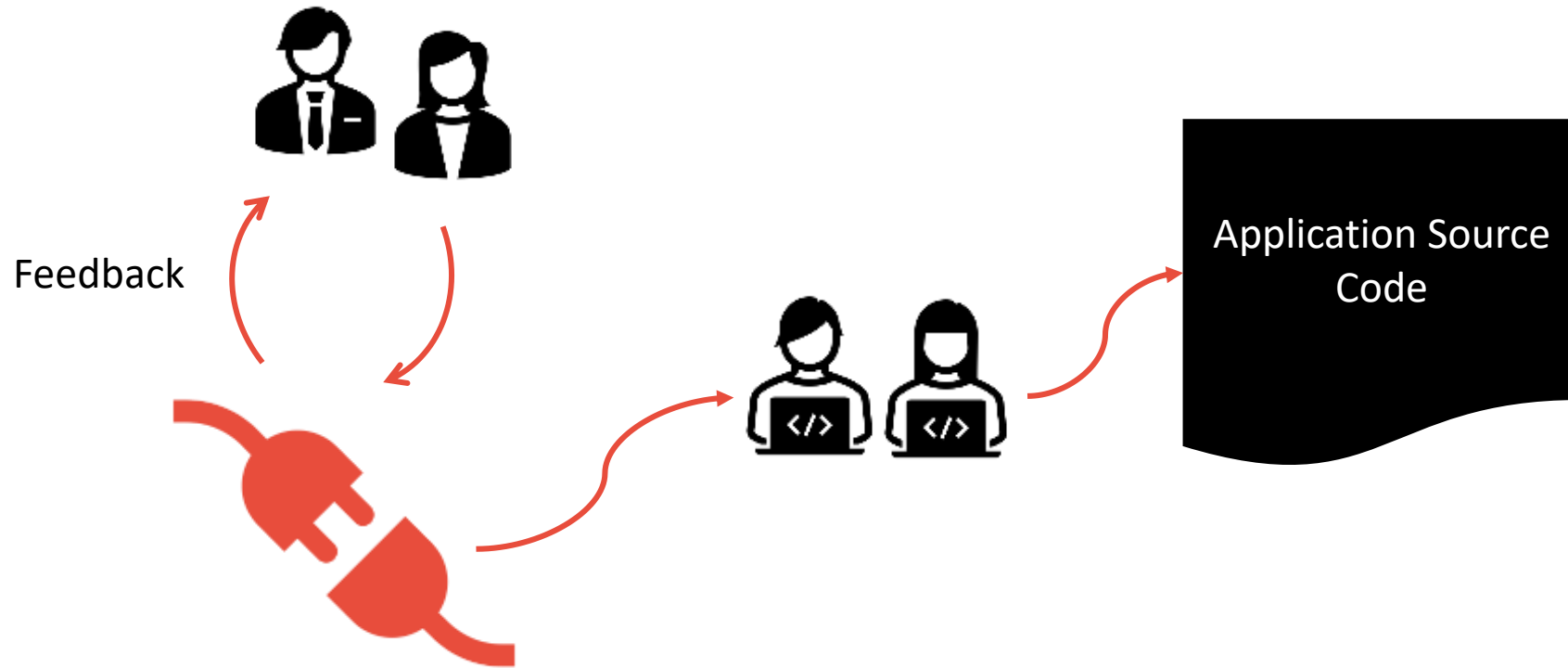




Thriving in the cloud through the revised 15 factors

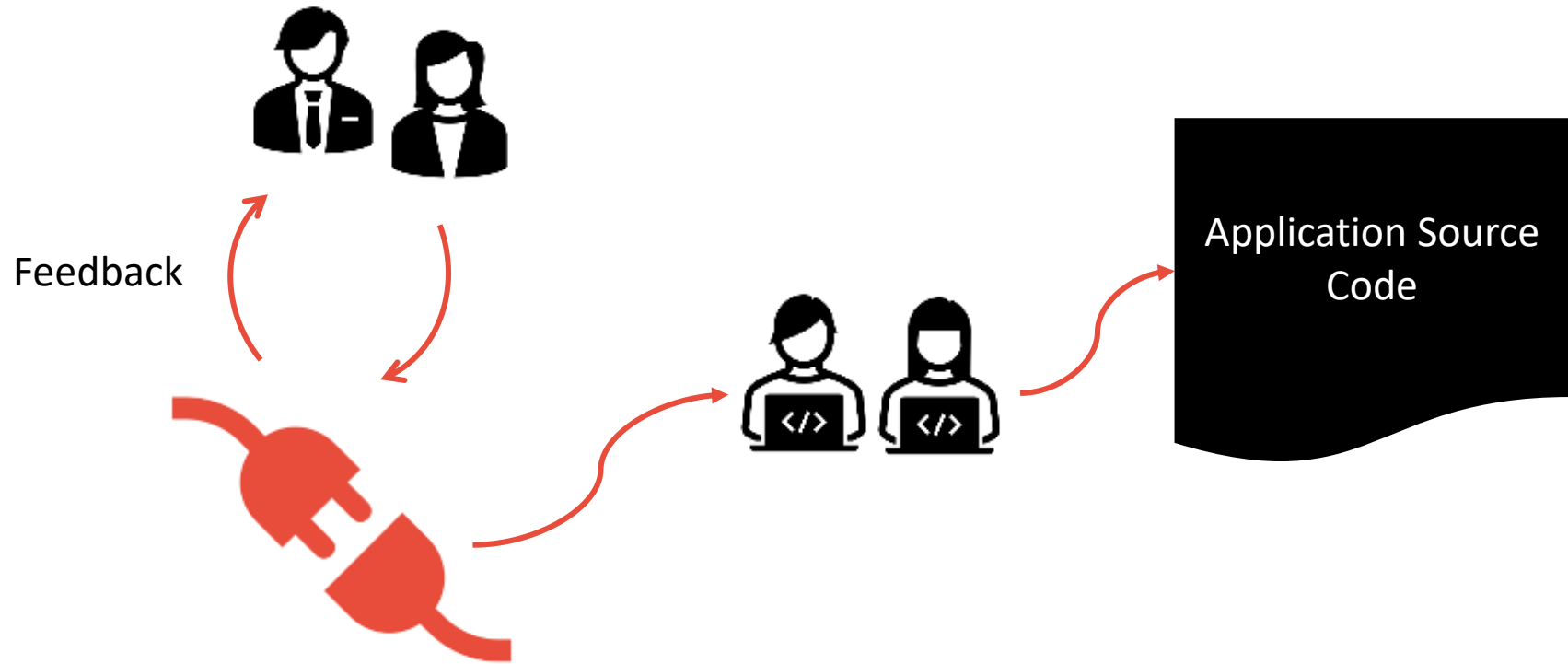


2. API first

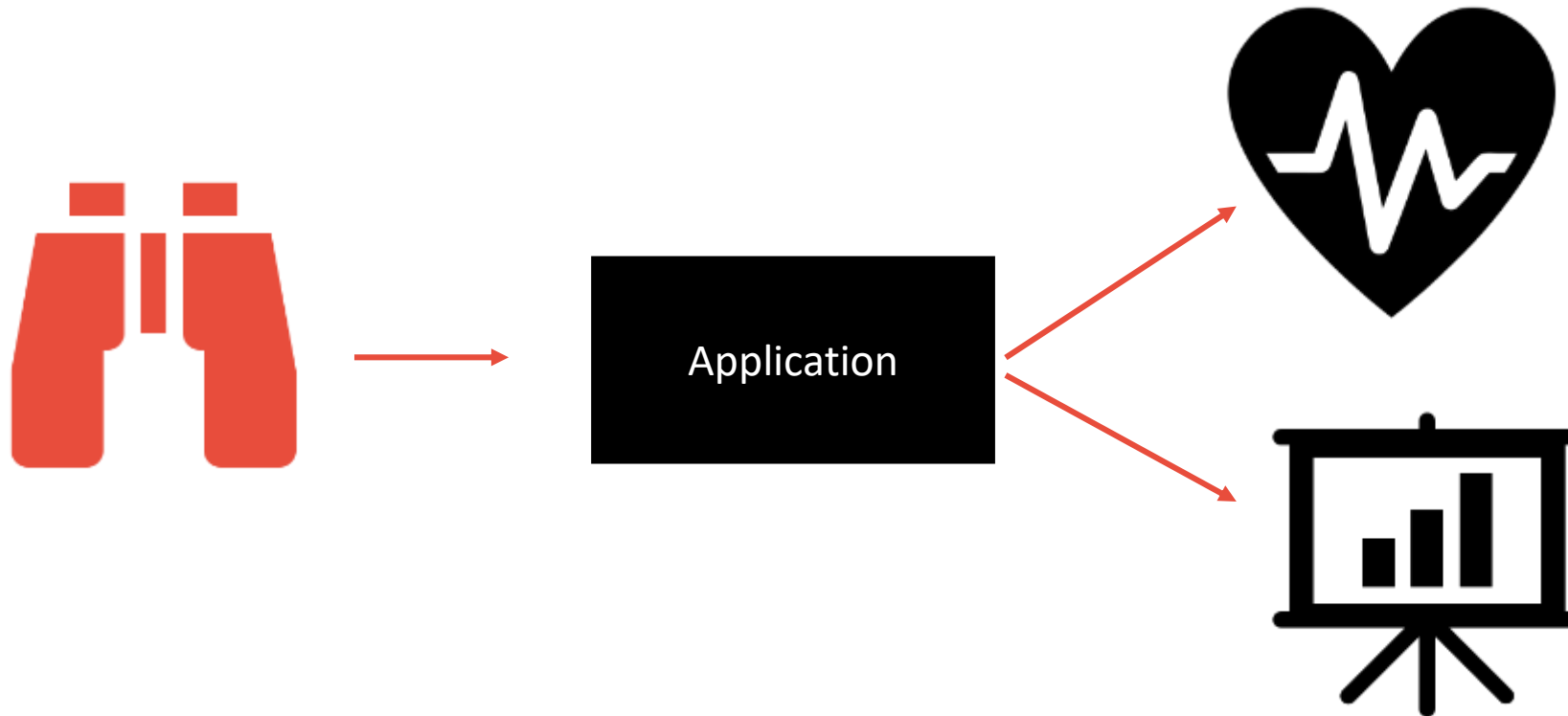


<https://openliberty.io/guides/microprofile-openapi.html>

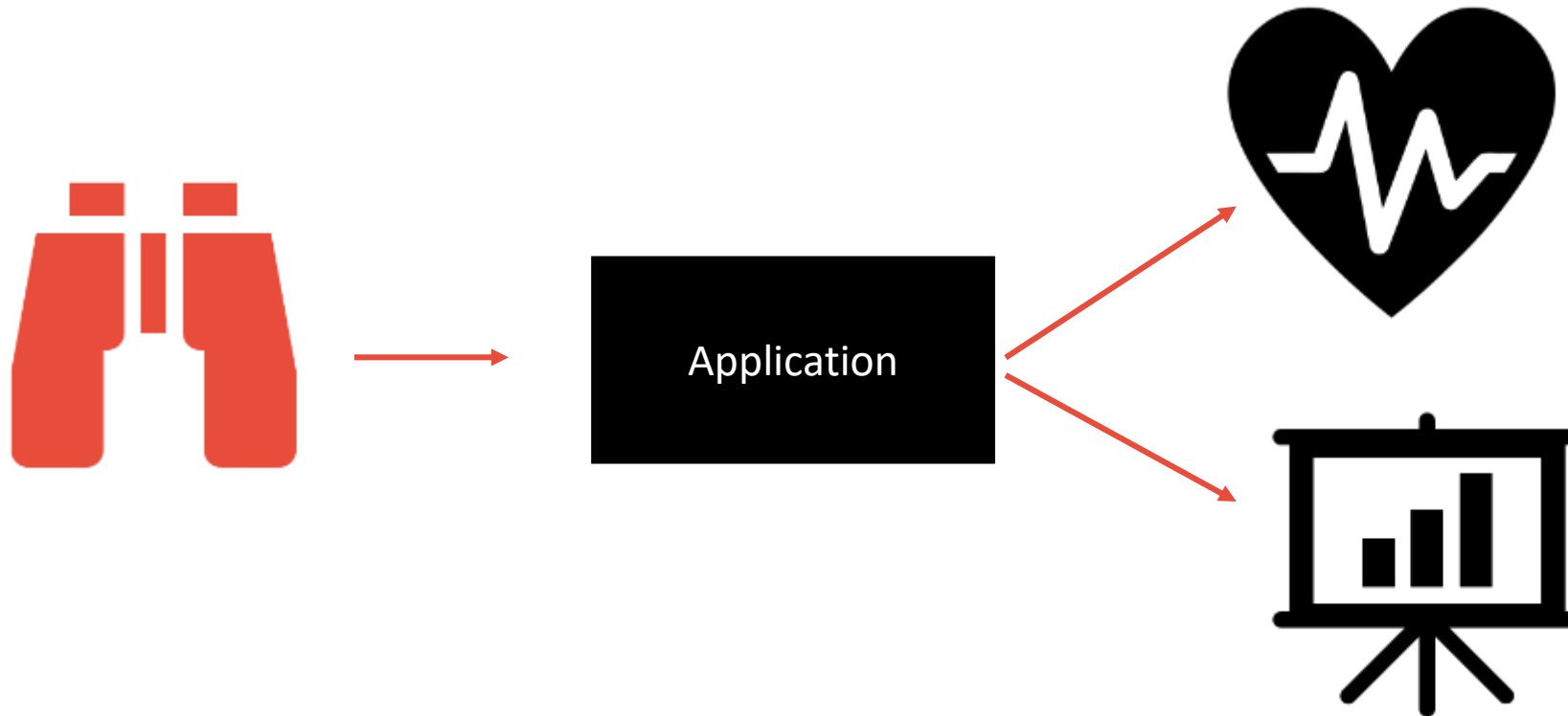
2. API first



14. Telemetry



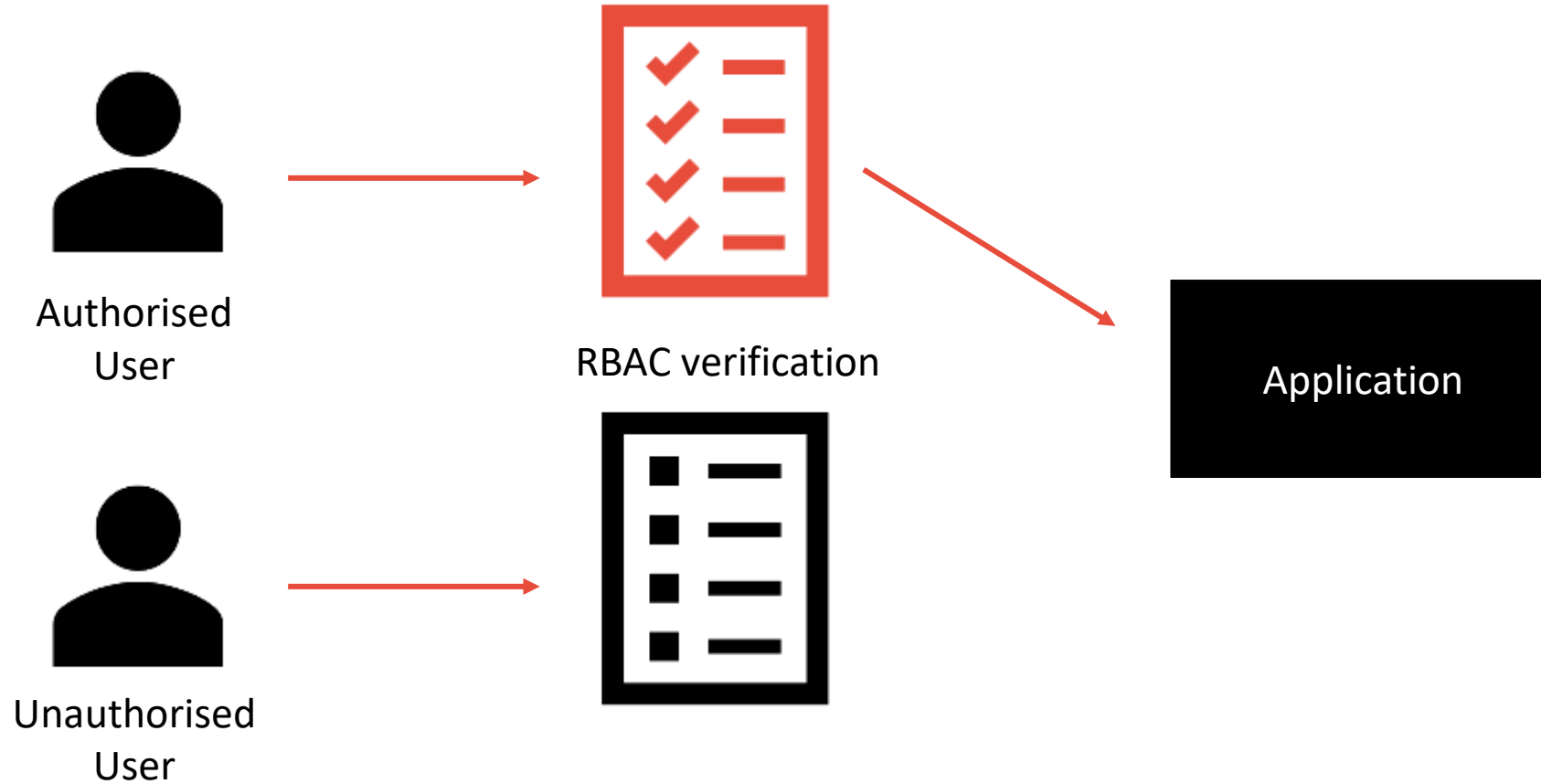
14. Telemetry



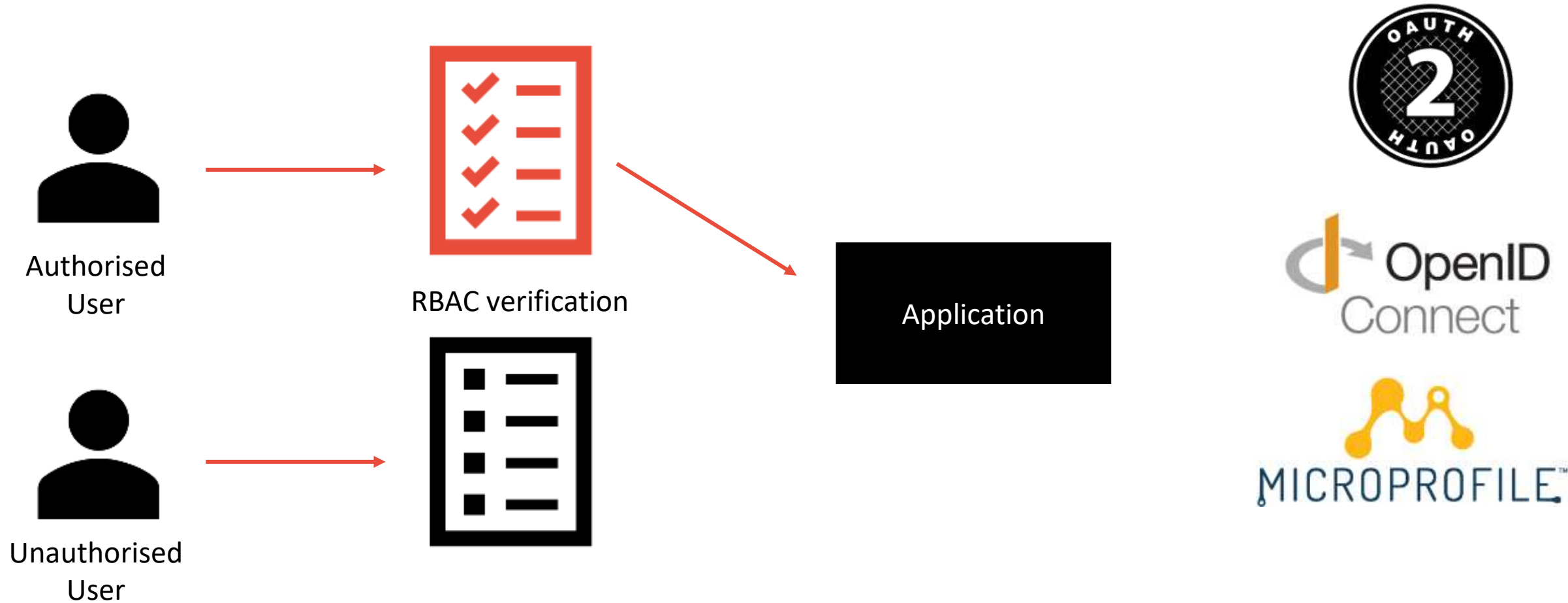
<https://openliberty.io/guides/microprofile-metrics.html>

<https://openliberty.io/guides/microprofile-health.html>

15. Authentication and authorization

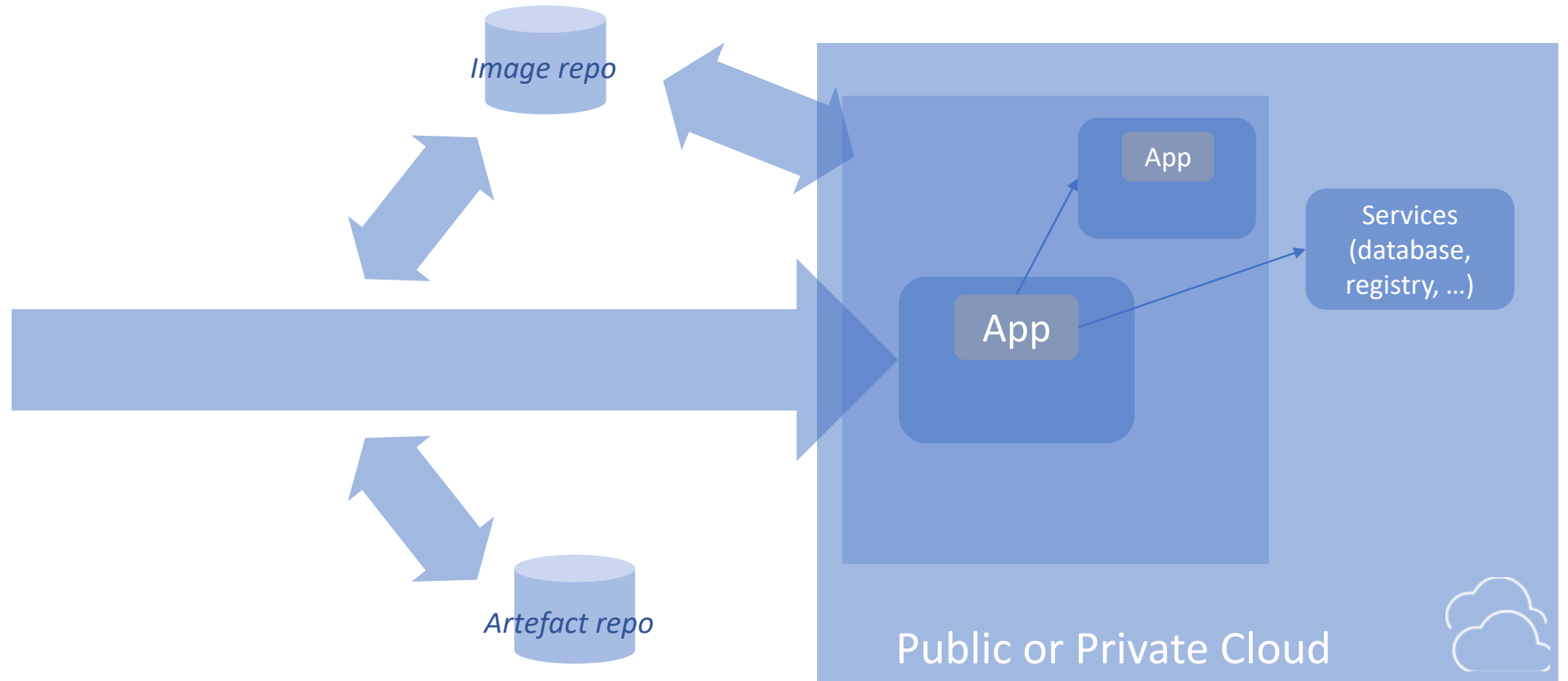


15. Authentication and authorization

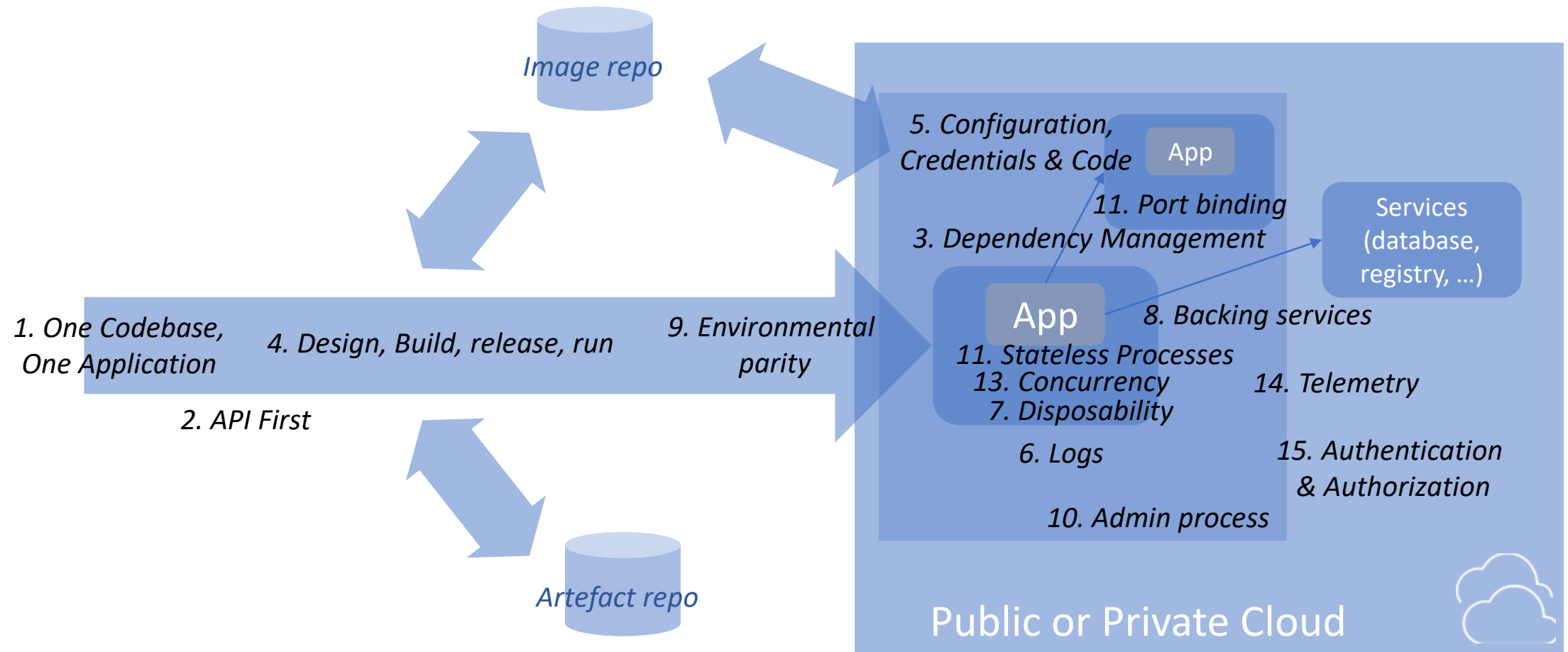


<https://openliberty.io/guides/microprofile-jwt.html>

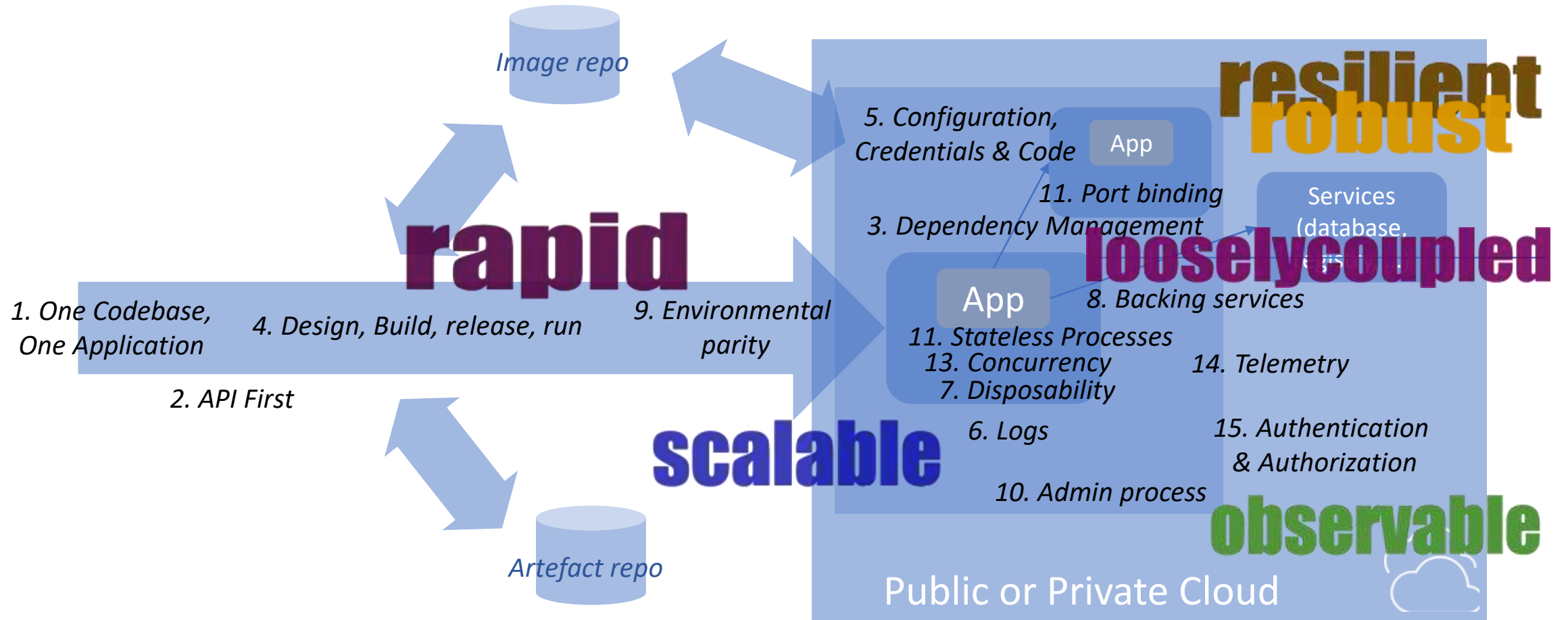
15-Factor App



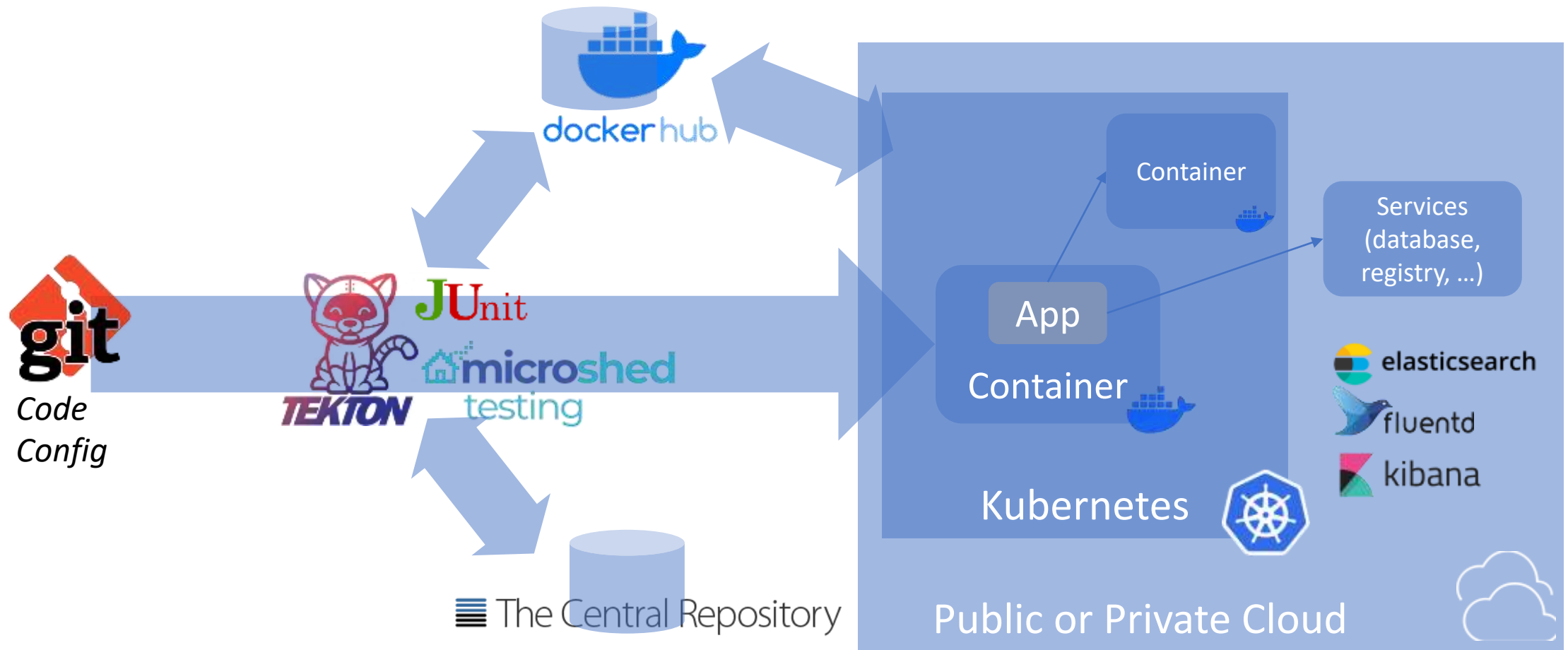
15-Factor App



15-Factor App



A Cloud-native instantiation





Hands on experience

Interactive cloud-native labs

The screenshot displays an interactive lab environment with the following components:

- Instructions Panel (Left):** Titled "Getting Started", it provides step-by-step guidance. It includes terminal commands for cloning a repository, navigating to the project directory, and running the application. A "Support" button is visible on the right side of this panel.
- Code Editor (Center):** Shows the `pom.xml` file for a Maven project. The XML content is as follows:

```
<?xml version='1.0' encoding='utf-8'?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>io.openliberty.guides</groupId>
  <artifactId>guide-rest-intro</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <liberty.var.default.http.port>9080</liberty.var.default.http.port>
    <liberty.var.default.https.port>9443</liberty.var.default.https.port>
    <liberty.var.app.context.root>LibertyProject</liberty.var.app.context.root>
  </properties>

  <dependencies>
    <!-- Provided dependencies -->
    <dependency>
      <groupId>jakarta.platform</groupId>
      <artifactId>jakarta.jakartaee-api</artifactId>
      <version>8.0.0</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.eclipse.microprofile</groupId>
      <artifactId>microprofile</artifactId>
      <version>3.3</version>
      <type>pom</type>
      <scope>provided</scope>
    </dependency>
  </dependencies>
```
- Terminal (Bottom):** Shows the execution of the following commands:

```
thelia@theia-docker-jicoleman:/home/project$ git clone https://github.com/openliberty/guide-rest-intro.git
Cloning into 'guide-rest-intro'...
remote: Enumerating objects: 131, done.
remote: Counting objects: 100% (131/131), done.
remote: Compressing objects: 100% (88/88), done.
remote: Total 1494 (delta 46), reused 89 (delta 23), pack-reused 1363
Receiving objects: 100% (1494/1494), 288.39 KiB | 6.55 MiB/s, done.
Resolving deltas: 100% (567/567), done.
thelia@theia-docker-jicoleman:/home/project$ cd guide-rest-intro
thelia@theia-docker-jicoleman:/home/project/guide-rest-intro$
```

Open Liberty Guides

<https://openliberty.io/guides/>

The screenshot shows the Open Liberty Guides website. At the top, there is a dark blue navigation bar with the Open Liberty logo and links for 'Get Started', 'Guides', 'Docs', 'Support', and 'Blog'. Below the navigation bar, the word 'Guides' is prominently displayed in white, followed by the tagline 'The quickest way to learn all things Open Liberty, and beyond!'. A search bar with the text 'Filter guides' and a close button 'X' is located in the top right corner. The main content area is divided into sections. The first section is 'Developing your cloud-native application', which includes a sub-section 'Getting started'. Under 'Getting started', there are two guide cards: 'Getting started with Open Liberty' (25 minutes) and 'Injecting dependencies into microservices' (15 minutes). Below this, there is a section for 'RESTful service', which includes four guide cards: 'Creating a RESTful web service', 'Consuming RESTful services with template interfaces', 'Consuming a RESTful web service', and 'Documenting RESTful APIs'. On the left side, there is a vertical navigation menu with categories: 'DEVELOP (37 guides)', 'BUILD AND TEST (10 guides)', and 'DEPLOY (9 guides)'. Each category has a list of sub-topics.

DEVELOP (37 guides)

- Getting started
- RESTful service
- Reactive service
- Configuration
- Fault tolerance
- Observability
- Security
- Persistence
- Client side

BUILD AND TEST (10 guides)

- Build
- Test
- Containerize

DEPLOY (9 guides)

- Kubernetes
- Cloud deployment

Developing your cloud-native application

Getting started

Getting started with Open Liberty

Learn how to develop a Java application on Open Liberty with Maven and Docker.

🕒 25 minutes

Injecting dependencies into microservices

Learn how to use Contexts and Dependency Injection (CDI) to manage and inject dependencies into microservices.

🕒 15 minutes

RESTful service

Creating a RESTful web service

Learn how to create a REST service with JAX-RS, JSON-B, and Open Liberty.

Consuming RESTful services with template interfaces

Learn how to use MicroProfile Rest Client to invoke RESTful services over HTTP in a type-

Consuming a RESTful web service

Explore how to access a simple RESTful web service and consume its resources in Java using JSON-B and JSON-P.

Documenting RESTful APIs

Explore how to document and filter RESTful APIs from code or static files by using MicroProfile OpenAPI.



Summary

Summary:

- Twelve-factor applications = great start
- But... to thrive in the cloud, we need to look beyond the 12 factors
- No excuses!
 - Lots of open-source tools and technologies available to help
- Action: Evaluate your own applications against these 15 factors and consider what you could do to enable them to truly thrive in the cloud

Useful Resources

- General:
 - <https://openliberty.io/blog/2019/09/05/12-factor-microprofile-kubernetes.html>
 - [https://www.cdta.org/sites/default/files/awards/beyond the 12-factor app pivotal.pdf](https://www.cdta.org/sites/default/files/awards/beyond_the_12-factor_app_pivotal.pdf)
 - <https://developer.ibm.com/articles/creating-a-12-factor-application-with-open-liberty/>
- Design, build, release, run:
 - <https://developer.ibm.com/devpractices/devops/patterns/make-continuous-delivery-easier-with-tekton-dashboards/>
 - <https://dzone.com/articles/deploying-microprofile-microservices-with-tekton>
- Logging:
 - <https://developer.ibm.com/videos/use-json-logging-in-open-liberty/>
 - <https://developer.ibm.com/videos/send-open-liberty-logs-to-elastic-stack/>
 - <https://openliberty.io/blog/2021/02/10/ocp-log-forwarding.html>
 - <https://community.ibm.com/community/user/communities/community-home/librarydocuments/viewdocument?DocumentKey=65596910-8d01-48d2-a99e-d94794d022af>

Useful Resources

- Stateless Processes:
 - <https://openliberty.io/guides/sessions.html>
- Concurrency:
 - <https://developer.ibm.com/technologies/containers/tutorials/autoscale-application-on-kubernetes-cluster/>
- Authentication and Authorisation:
 - <https://openliberty.io/blog/2021/03/26/MP-JWT-1.2.html>
 - <https://openliberty.io/blog/2020/09/04/securing-open-liberty-azure.html>
- Open Liberty Tools:
 - <https://openliberty.io/blog/2021/04/21/admin-center-21004.html>

Connect with us



Jobs ▾

Open Liberty

Worldwide



Join now

Sign



Open Liberty

IT Services and IT Consulting

Open source implementation of Eclipse MicroProfile and Jakarta EE for developing fast cloud-native Java microservices

[Follow](#)

About us

A lightweight open framework for building fast and efficient cloud-native Java microservices.

Build cloud-native apps and microservices while running only what you need. Open Liberty is the most flexible server runtime available to Java™ developers in this solar system.

Develop cloud-native Java microservices

Open Liberty is fast to start up with a low memory footprint and live reload for quick iteration. Simple to add and remove features from the latest versions of MicroProfile and Jakarta EE. Zero migration lets you focus on what's important, not the APIs changing under you.

Similar pages

-  **MicroProfile**
Software Development
-  **Java Developer Zone**
IT Services and IT Consulting
Gandhinagar, Gujarat
-  **Quarkusio**
Software Development
-  **Payara Services Ltd**
IT Services and IT Consulting
Malvern, Worcestershire

[Show more similar pages](#)

Browse jobs

- Director Of Support jobs**
102,832 open jobs
- Software Engineer jobs**
464,597 open jobs
- Vice President Operations jobs**
10,334 open jobs



Open Liberty

[@OpenLibertyIO](#)

Open Liberty is an open source implementation of Eclipse MicroProfile, Jakarta EE, and Java EE for developing fast cloud-native microservices.

Alpha Centauri [openliberty.io](#) Joined September 2017

502 Following 3,625 Followers

Tweets Tweets & replies Media Likes

Pinned Tweet

openlibertyio @OpenLibertyIO · Sep 7, 2018
What's Open Liberty like? Have a go at deploying and packaging a RESTful web service with Maven in this guide:



openliberty.io
Getting started with Open Liberty
A getting started tutorial with examples on how to rapidly develop, build, and package a Java ...

18 Retweets 39 Likes

openlibertyio @OpenLibertyIO · 1h
Our latest #OpenLiberty release (22.0.0.3) introduces the ability for SQL Operations to be retried in transaction recovery logs and includes several significant bug fixes.

<https://www.linkedin.com/company/openlibertyio/>

<https://twitter.com/OpenLibertyIO>



THANK YOU

