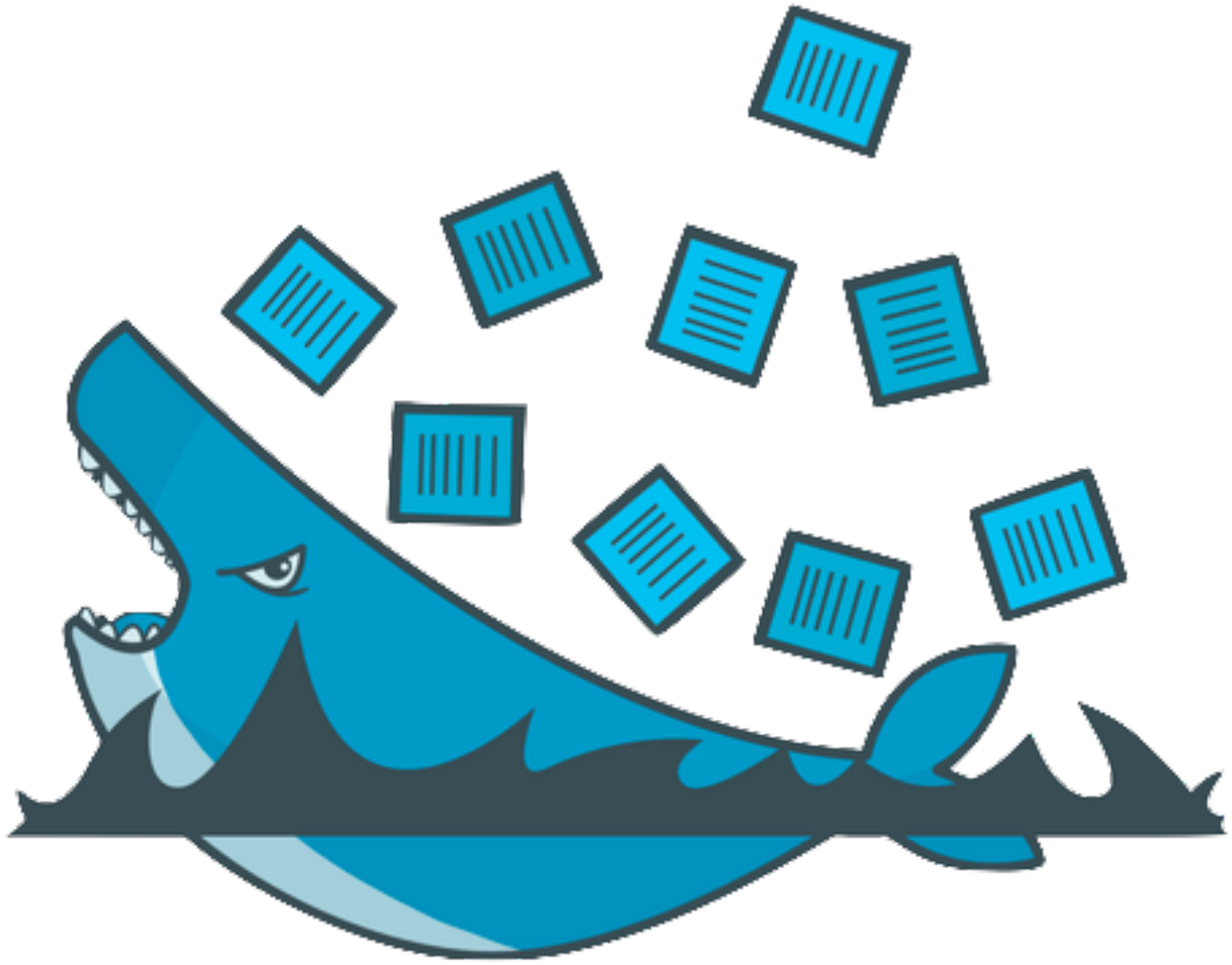


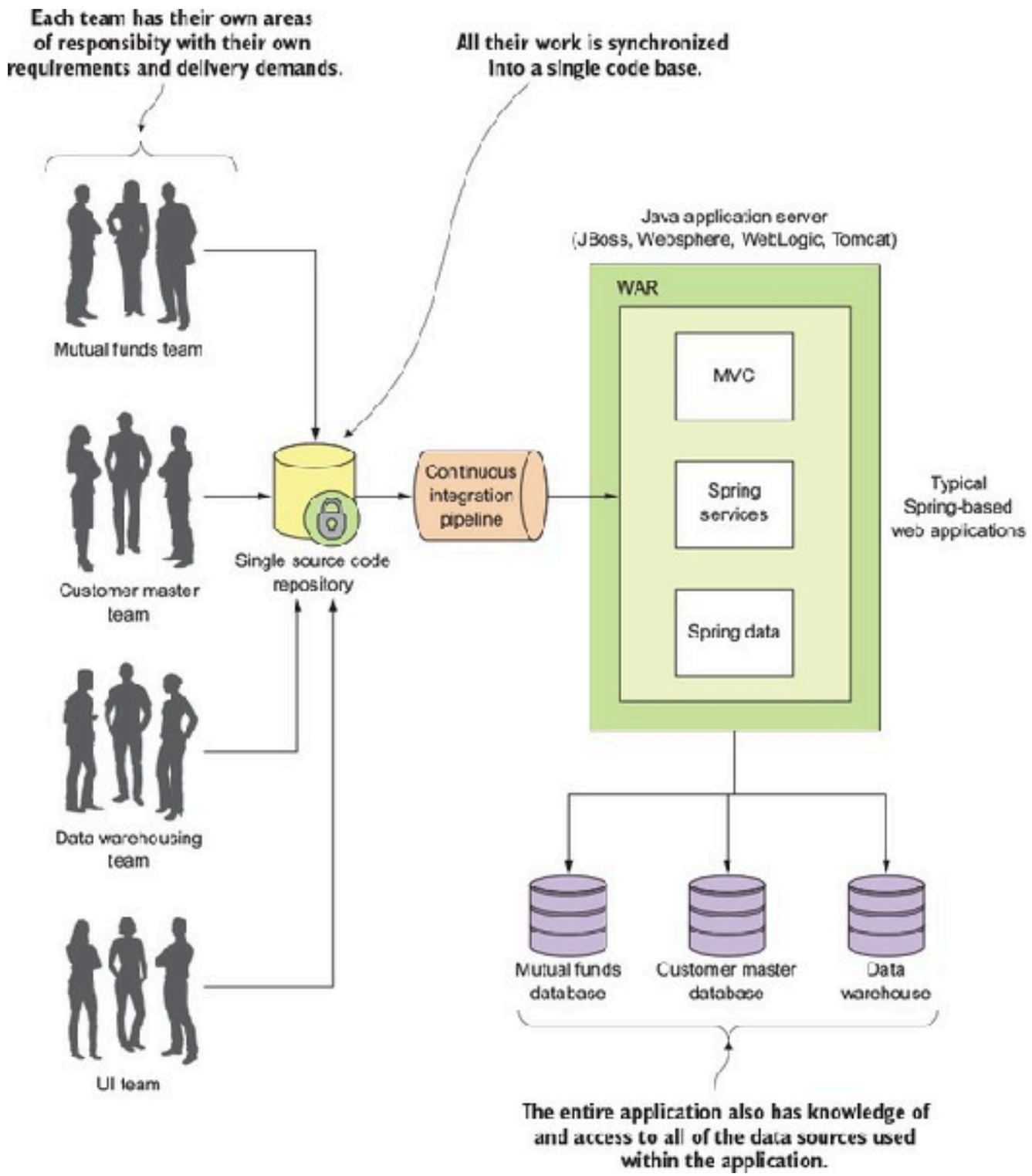
Docker workshop

Topics

Microservices
Spring Boot and Spring Cloud
Building Microservices with Spring boot
Testing Microservices
Deploying Micorservices

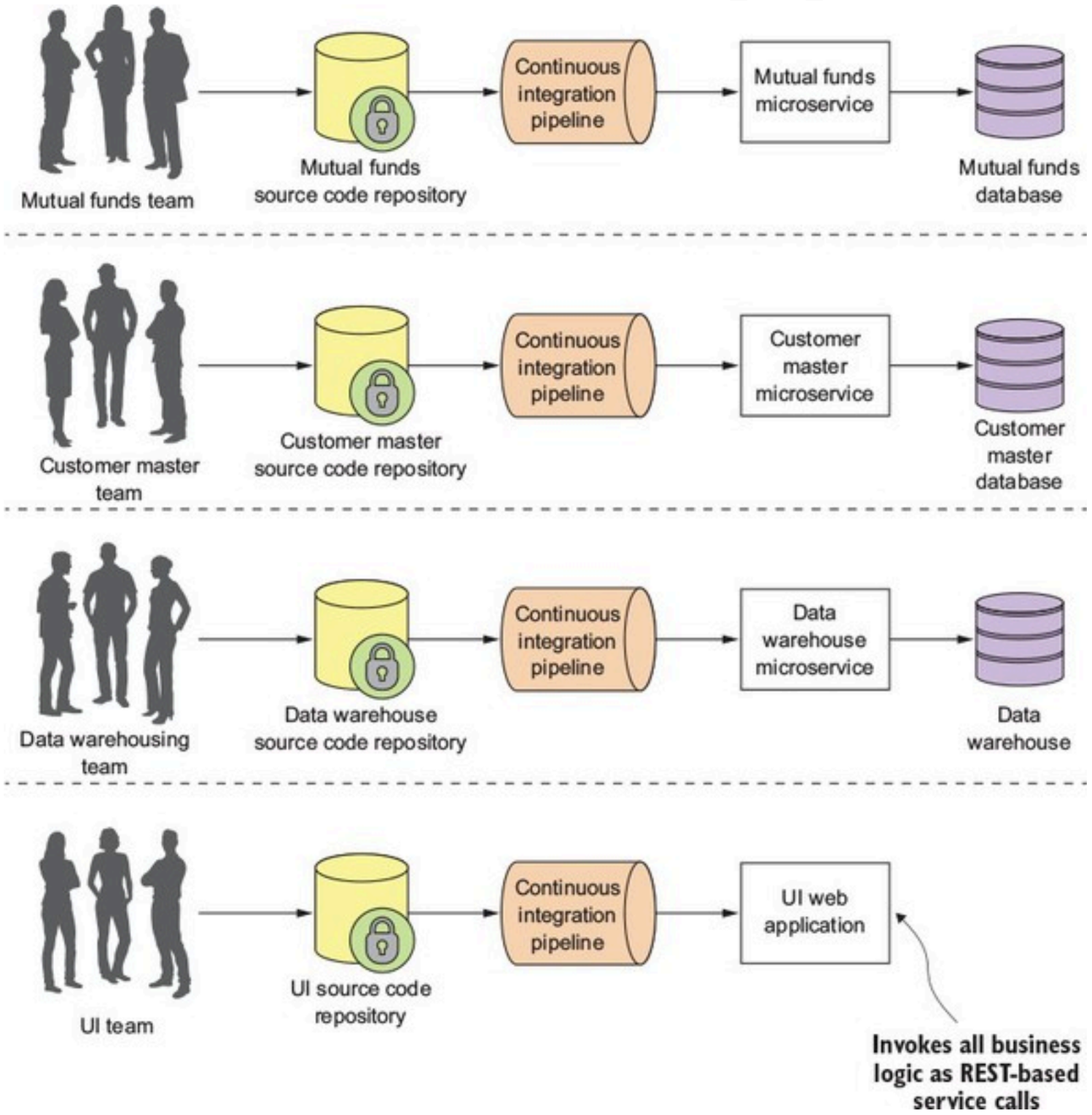
Microservices

Monolithic application

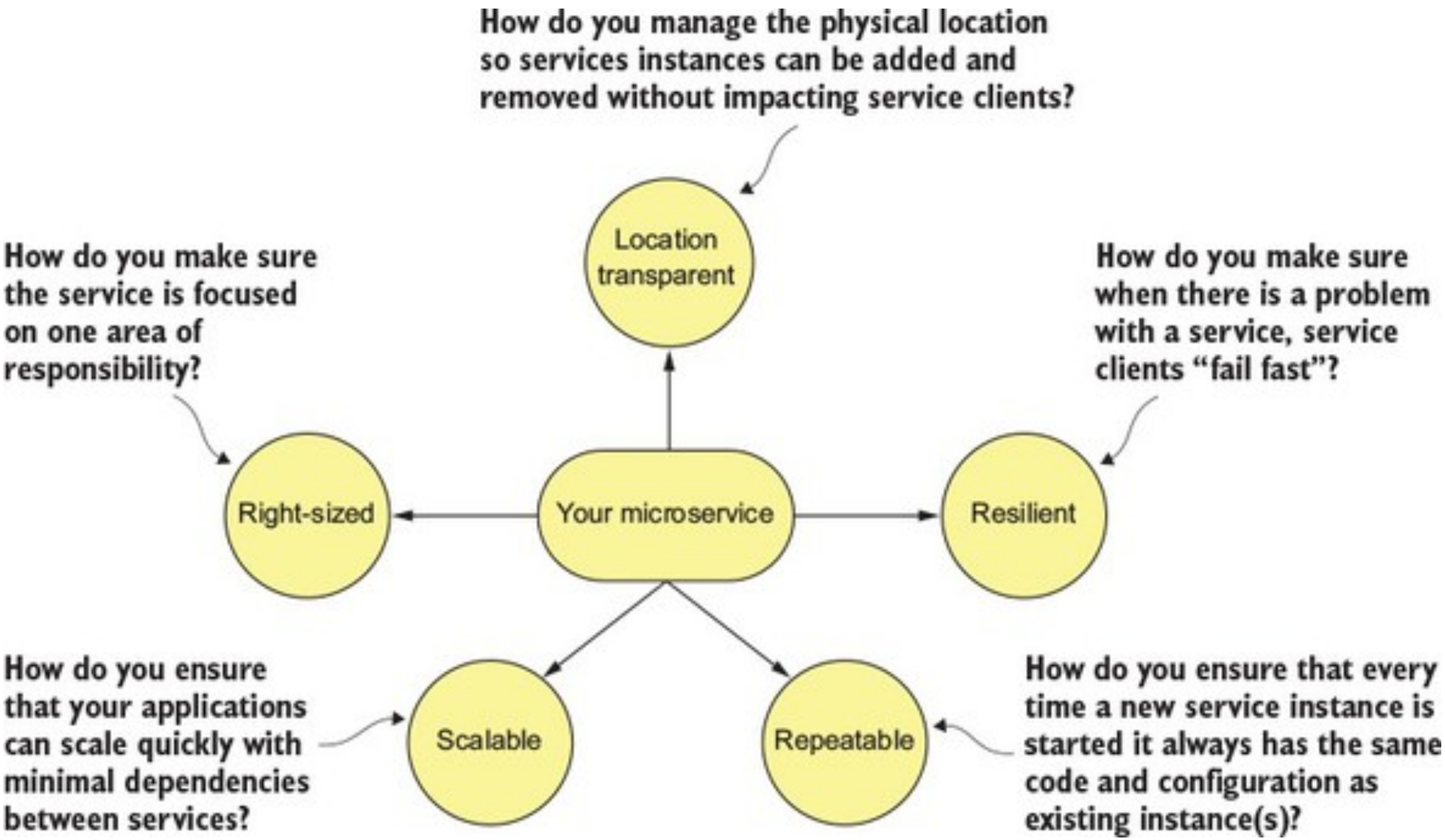


Decomposed into a set of service

Microservices application



Your Microservices



Microservices patterns

Core development patterns
Routing pattern
Client resiliency patterns
Security patterns
Logging and tracing patterns
Build and deployment patterns

Microservices perspectives

Architect

Software developer

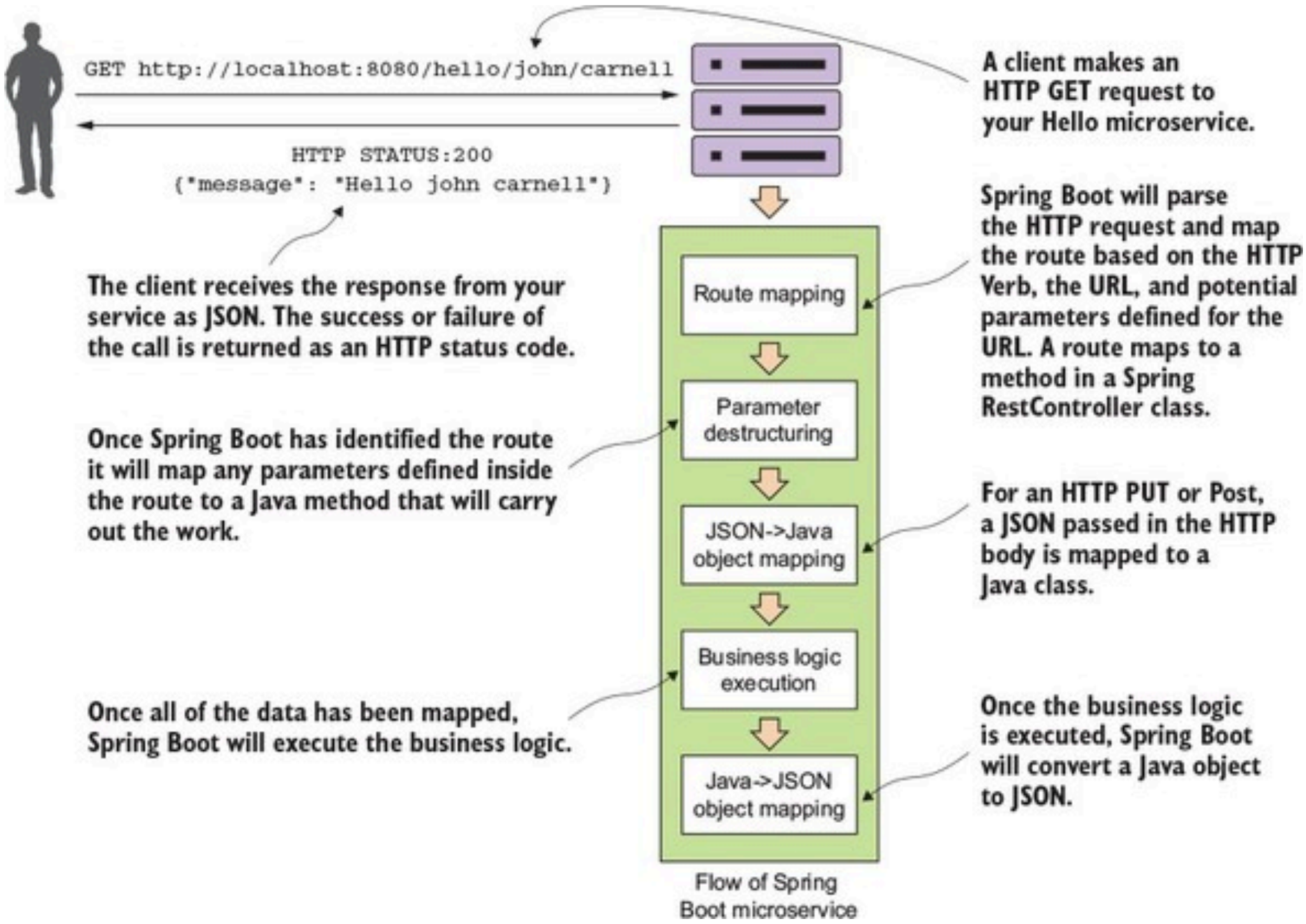
DevOps

Building with Spring Boot

Source code

https://github.com/up1/springboot-hello

Hello World



Flow of Spring Boot microservice

Microservices perspectives

Architect
Software developer
DevOps

Building the 12 factor

for microservices

https://12factor.net/

1. Codebase

All application code and server provisioning information should be in version control

Each service have own code repository

2. Dependencies

Explicitly declare the dependencies by build tool such as Maven (Java)

Third-party JAR should be specific version number

3. Config

Store configuration independently from code

Configuration should never in the same repository as your source code

4. Backing services

Easy to change the implement of your database

5. Build, release, run

Keep your build, release and run of deploy application completely separate

6. Processes

Service should always be stateless

Kill and replace at anytime without the fear

7. Port binding

host:container

8. Concurrency

When you need to scale,
launch more services and scale out

Scale out, not up

9. Disposability

Start and stop on demand

Startup time should be minimized

Process should shut down gracefully

10. Dev/Prod parity

Minimize the gaps that exist between all of the environments in which the service run

Include the developer’s machine

11. Logs

Logs are stream of events

Collect and write the logs to a central location

12. Admin processes

Developer will often have to do admin tasks

Tasks should be done via scripts, not ad hoc

Scripts should be repeatable and non-changing

4 principles

Service assembly

Service bootstrapping

Service registration/discovery

Service monitoring

1. Service assembly

How to package and deploy
Guarantee repeatability and consistency

2. Service bootstrapping

How to separate application and environment-specific configuration

Deploy without human intervention

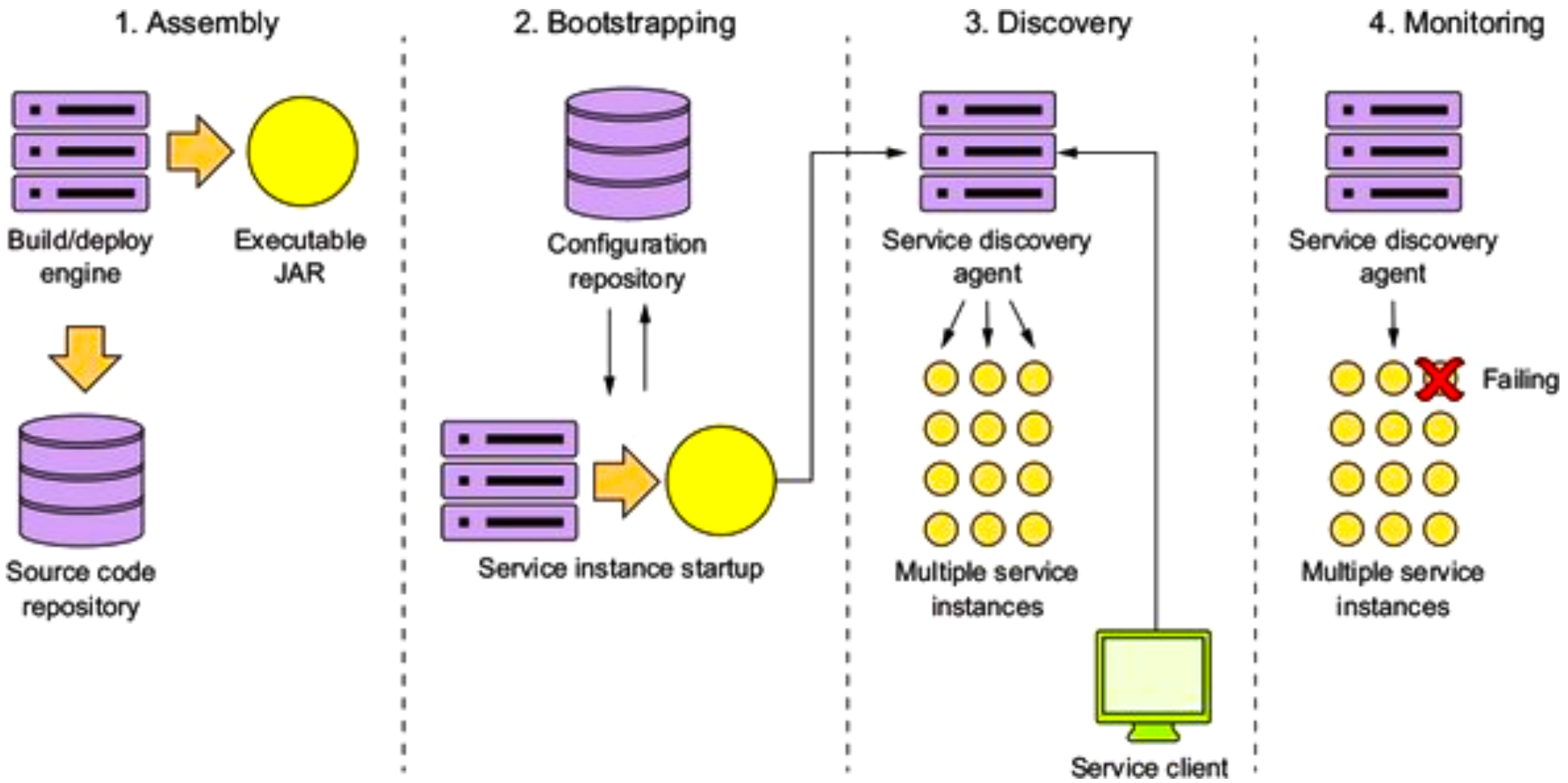
3. Service registration/discovery

When a new service is deployed,
how to make the new service discoverable ?

4. Service monitoring

How to monitoring many service ?

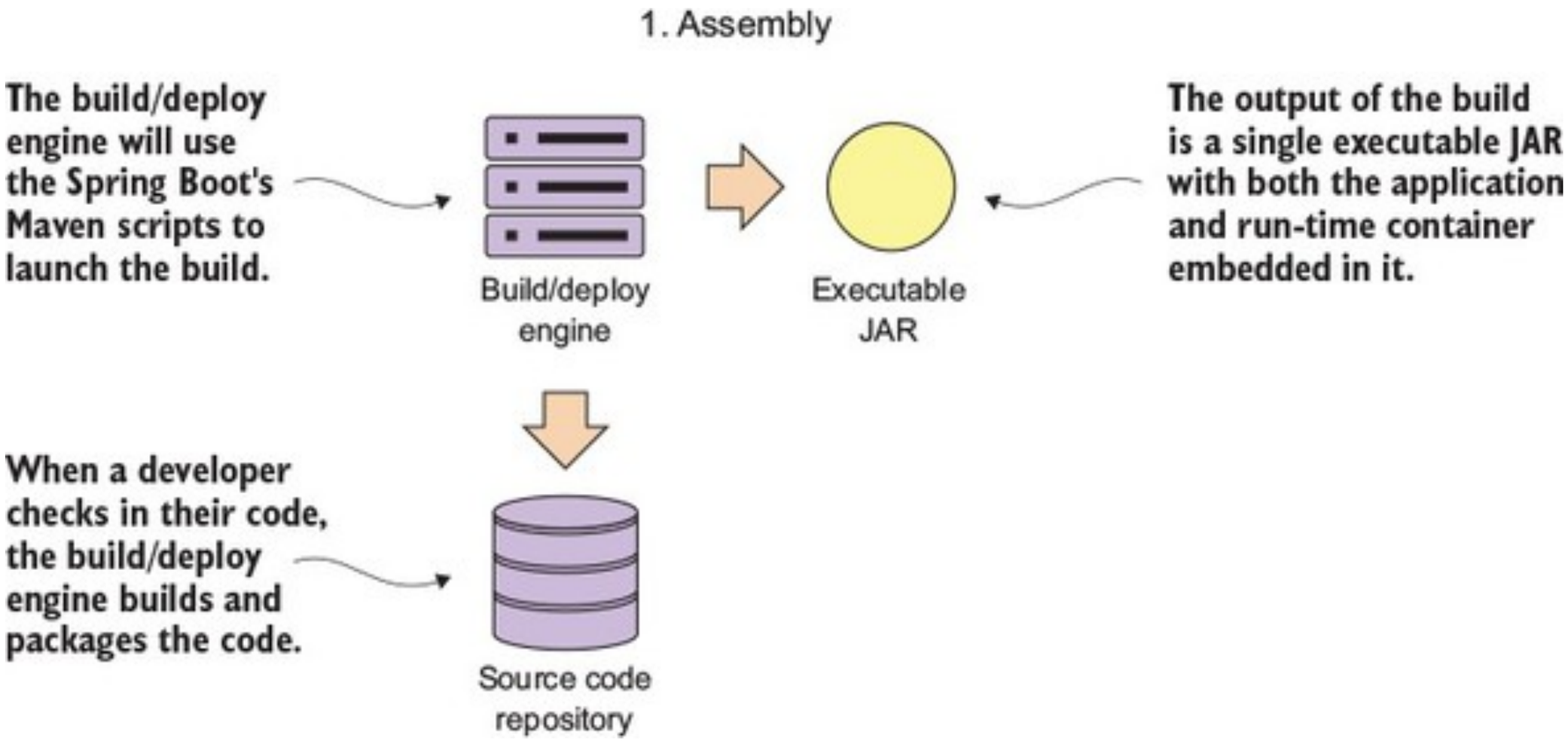
4 principles



Service assembly

package and deploy service

Service assembly



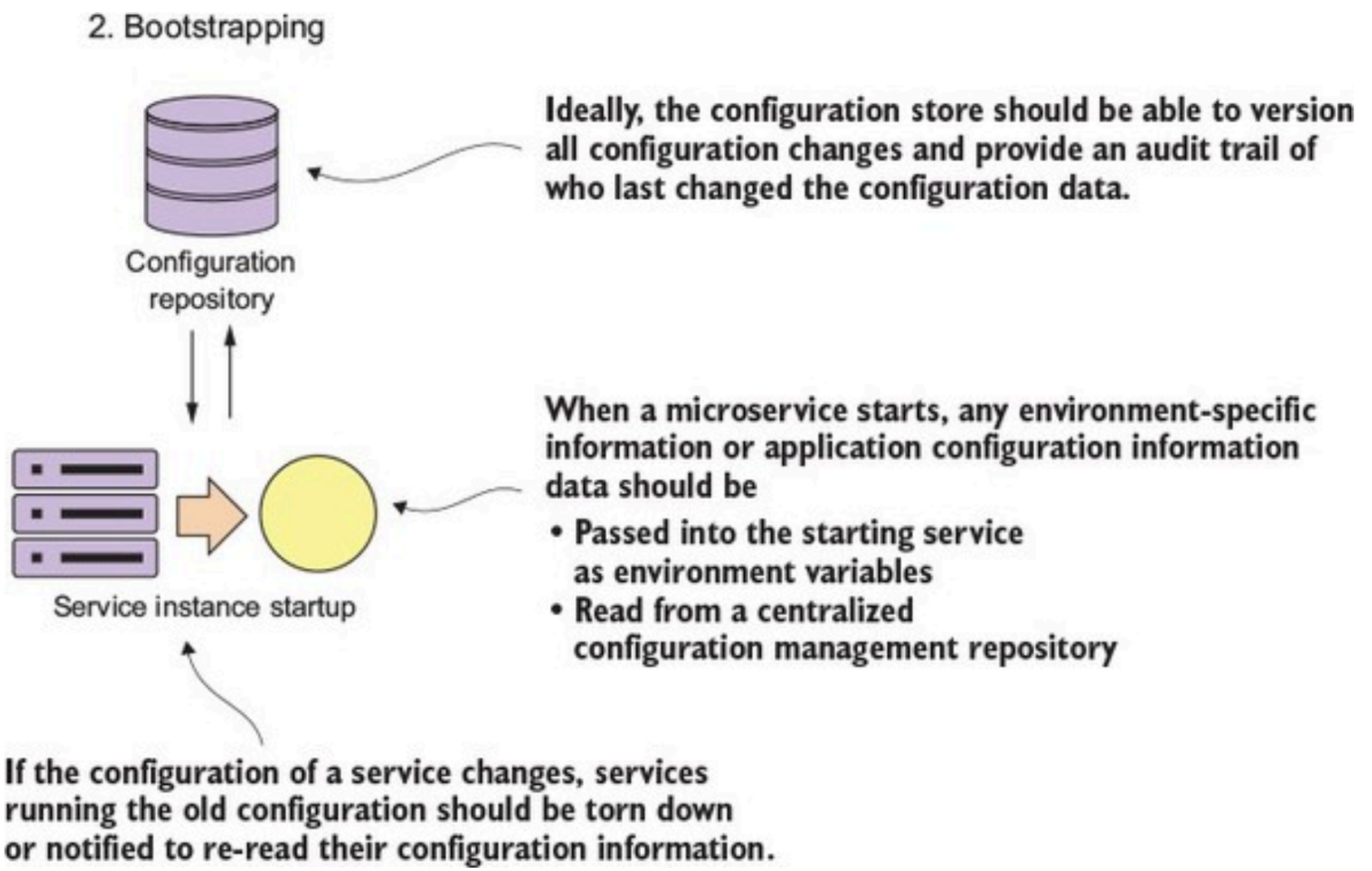
Spring boot

$mvn clean package && java -jar target/xxx.jar

Service bootstrapping

manage configuration of service

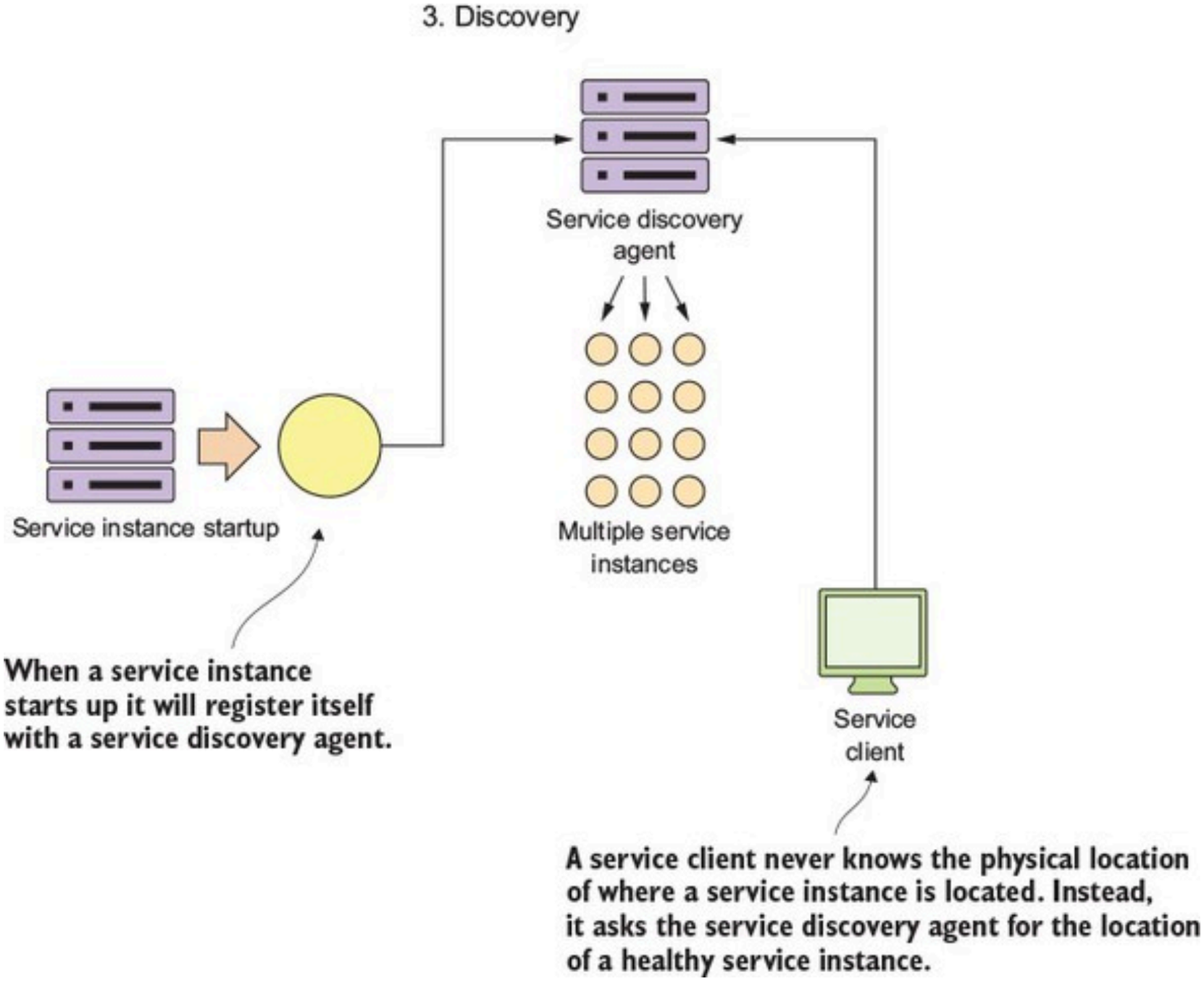
Service bootstrapping



Service registration/discovery

how clients communicate with service

Service registration/discovery

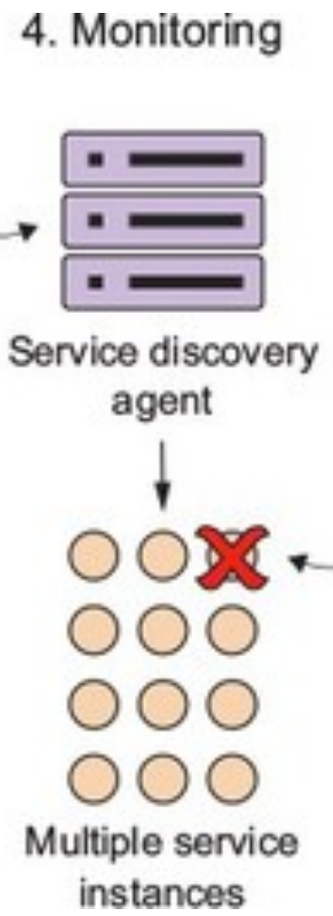


Service monitoring

health of service

Service monitoring

The service discovery agent monitors the health of a service instance. If the instance fails, the health check removes it from the pool of available instances.



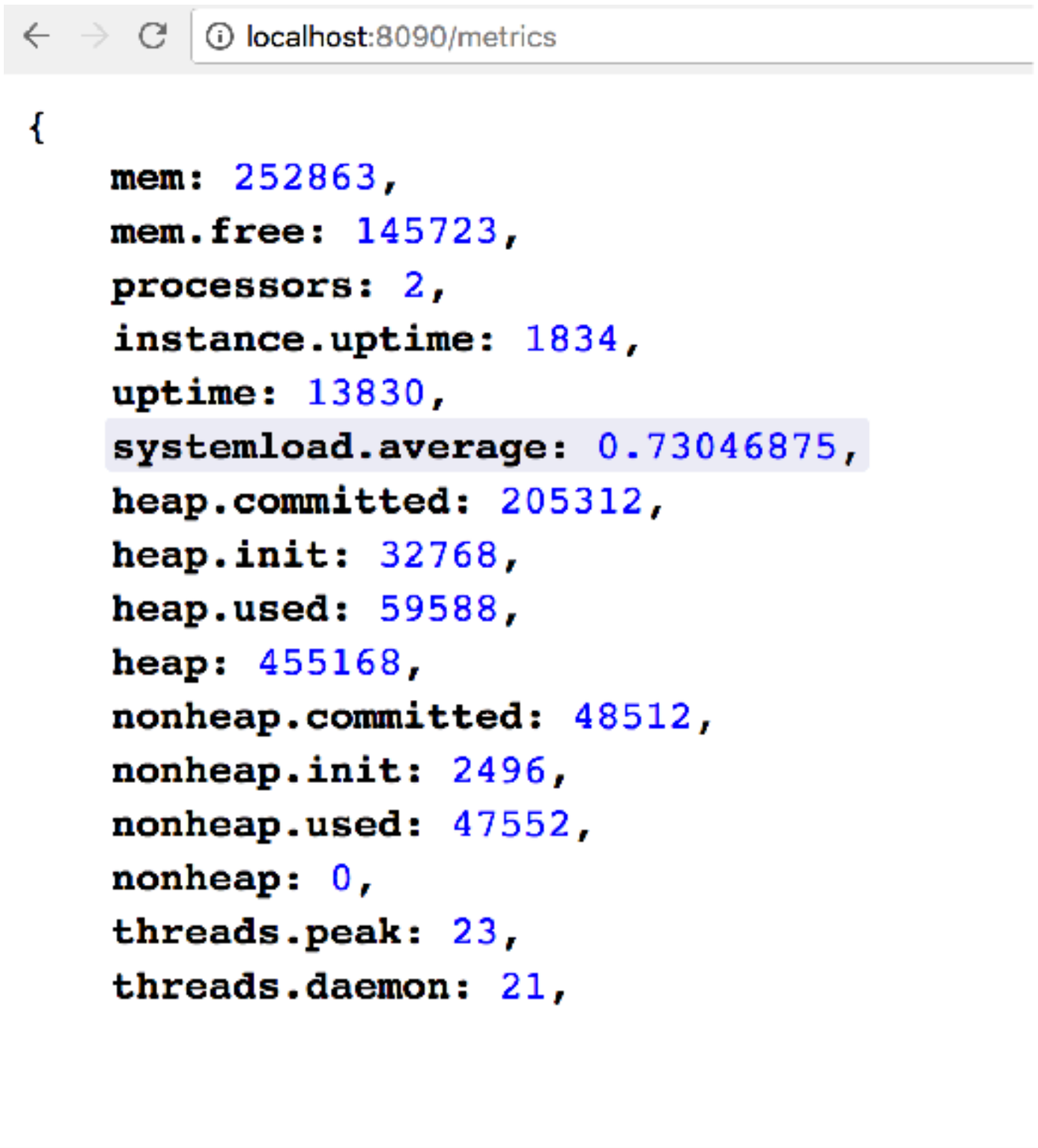
Most service instances will expose a health check URL that will be called by the service discovery agent. If the call returns an HTTP error or does not respond in a timely manner, the service discovery agent can shut down the instance or just not route traffic to it.

Spring actuator

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Health

Metrics


localhost:8090/metrics
{
mem: 252863,
mem.free: 145723,
processors: 2,
instance.uptime: 1834,
uptime: 13830,
systemload.average: 0.73046875,
heap.committed: 205312,
heap.init: 32768,
heap.used: 59588,
heap: 455168,
nonheap.committed: 48512,
nonheap.init: 2496,
nonheap.used: 47552,
nonheap: 0,
threads.peak: 23,
threads.daemon: 21,

Trace


```
[
  - {
        timestamp: 1515863045643,
      - info: {
            method: "GET",
            path: "/trace",
          - headers: {
              - request: {
                    host: "localhost:8090",
connection: "keep-alive",
upgrade-insecure-requests: "1",
user-agent: "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1
like Gecko) Chrome/64.0.3282.71 Safari/537.36",
accept:
"text/html,application/xhtml+xml,application/xml;q=0.9,imag
accept-encoding: "gzip, deflate, br",
accept-language: "en-US,en;q=0.9,th;q=0.8",
cookie: "Idea-70dc47e5=758587b0-a108-4247-975d-c403a454a0b2
offset=-25200000; lang=en-US; _ga=GA1.1.243173800.148038539
c8c5-4ce6-9690-34ddfade8a96; JSESSIONID.3a96feb5=node03cnw3
JSESSIONID.6cee443e=node01218k2twa1m51vxs8ihq0hhqp1.node0;
JSESSIONID.adc1ee8f=node05iwj5a1ustwr8s2ovultis60.node0;
```

สยามชำนาญกิจ

SIAM CHAMNAN KIT