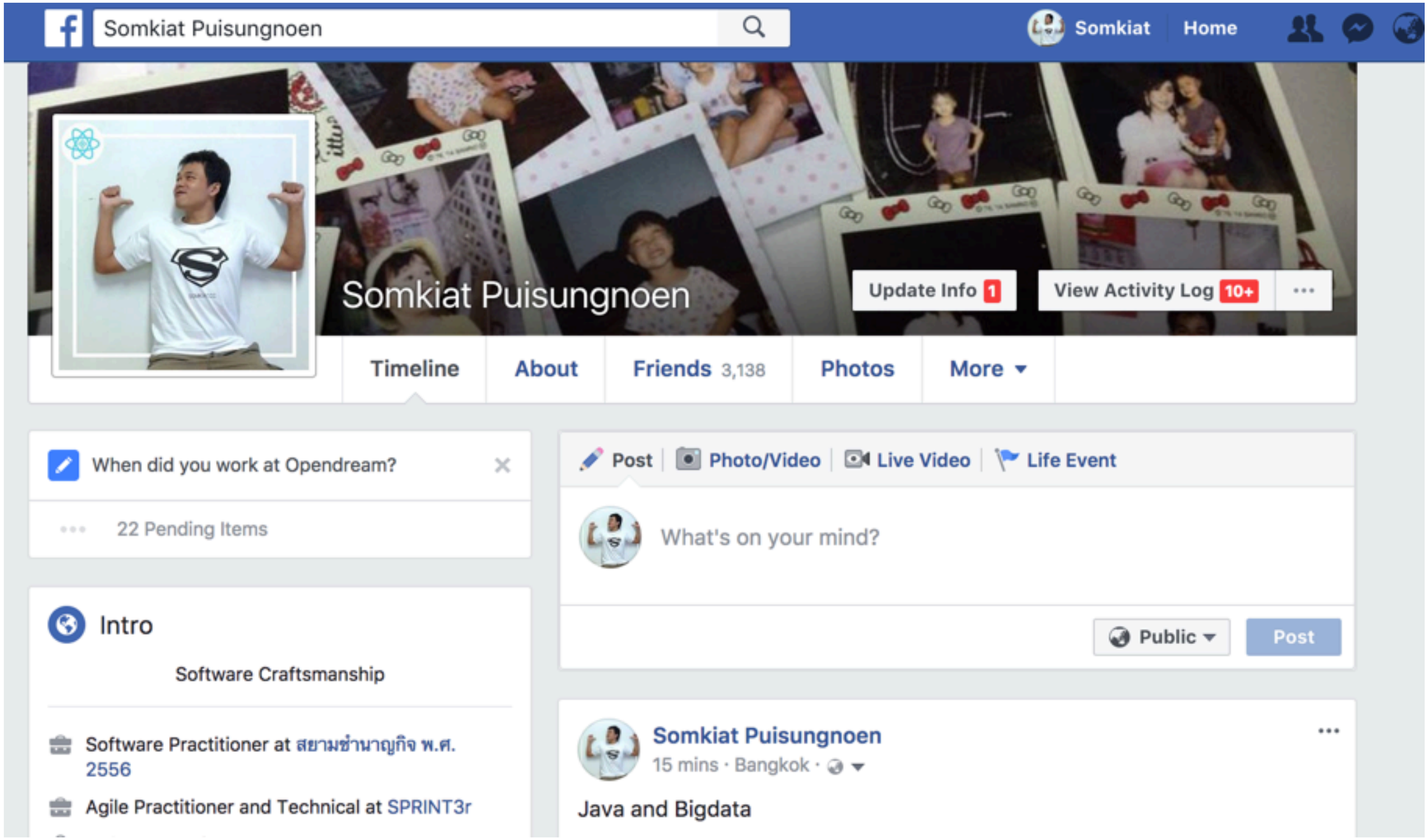
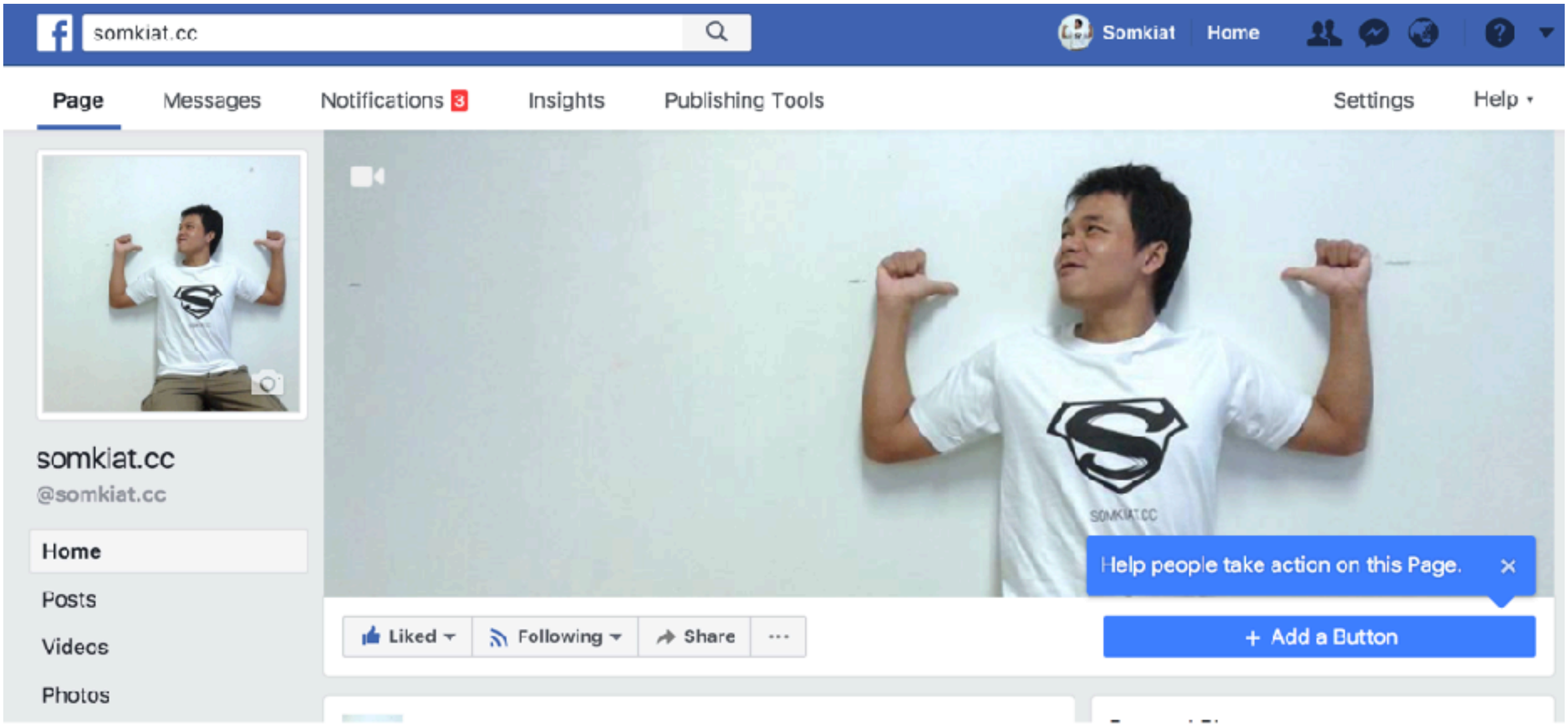
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Java Framework Day 1





Agenda

- RESTful API
- Java frameworks
- Building RESTful API with Spring Boot
- Spring Boot with Spring Data (JDBC/JPA)
- Spring Boot with Spring Data (MongoDB)
- Testing with Spring Boot
- Q/A

REST

REST

REpresentation **S**tate **T**ransfer

The style of software architecture behind RESTful services

Defined in 2000 by Roy Fielding

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Goals

Scalability
Generality of interfaces
Independent deployment of components

RESTful service

REST Request Messages

RESTful request is typically in form of **U**niform **R**esource **I**dentifiers (URI)

REST Request Messages

RESTful request is typically in form of **U**niform **R**esource **I**dentifiers (URI)

Structure of URI depend on specific service

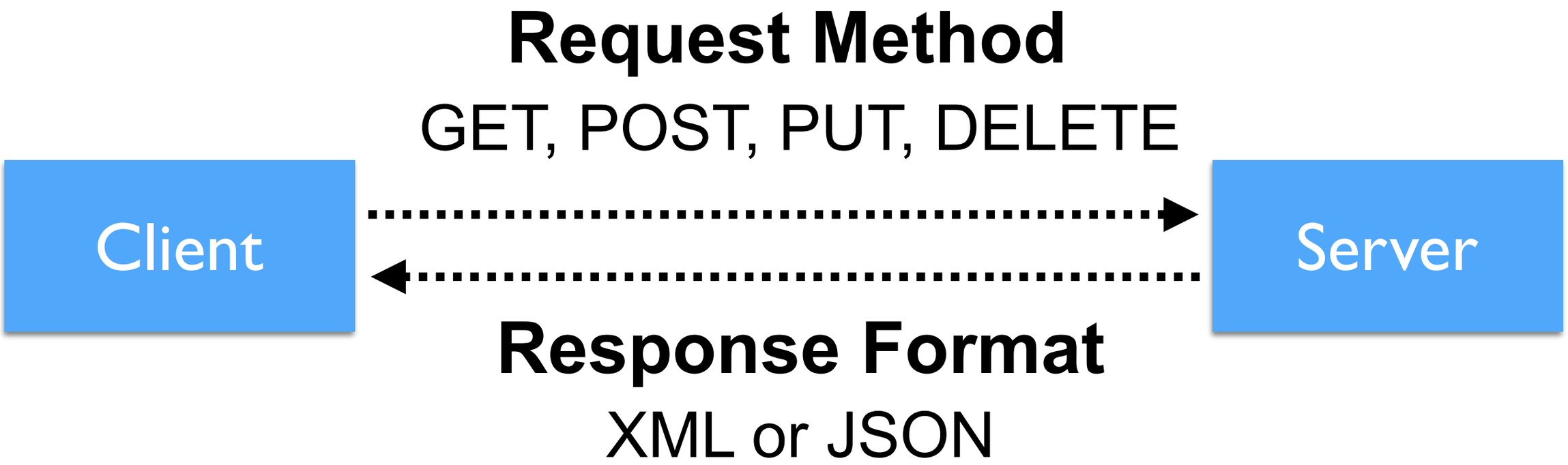
REST Request Messages

RESTful request is typically in form of **U**niform **R**esource **I**dentifiers (URI)

Structure of URI depend on specific service

Request can include parameter and data in body of request as XML, JSON etc.

REST Request & Response

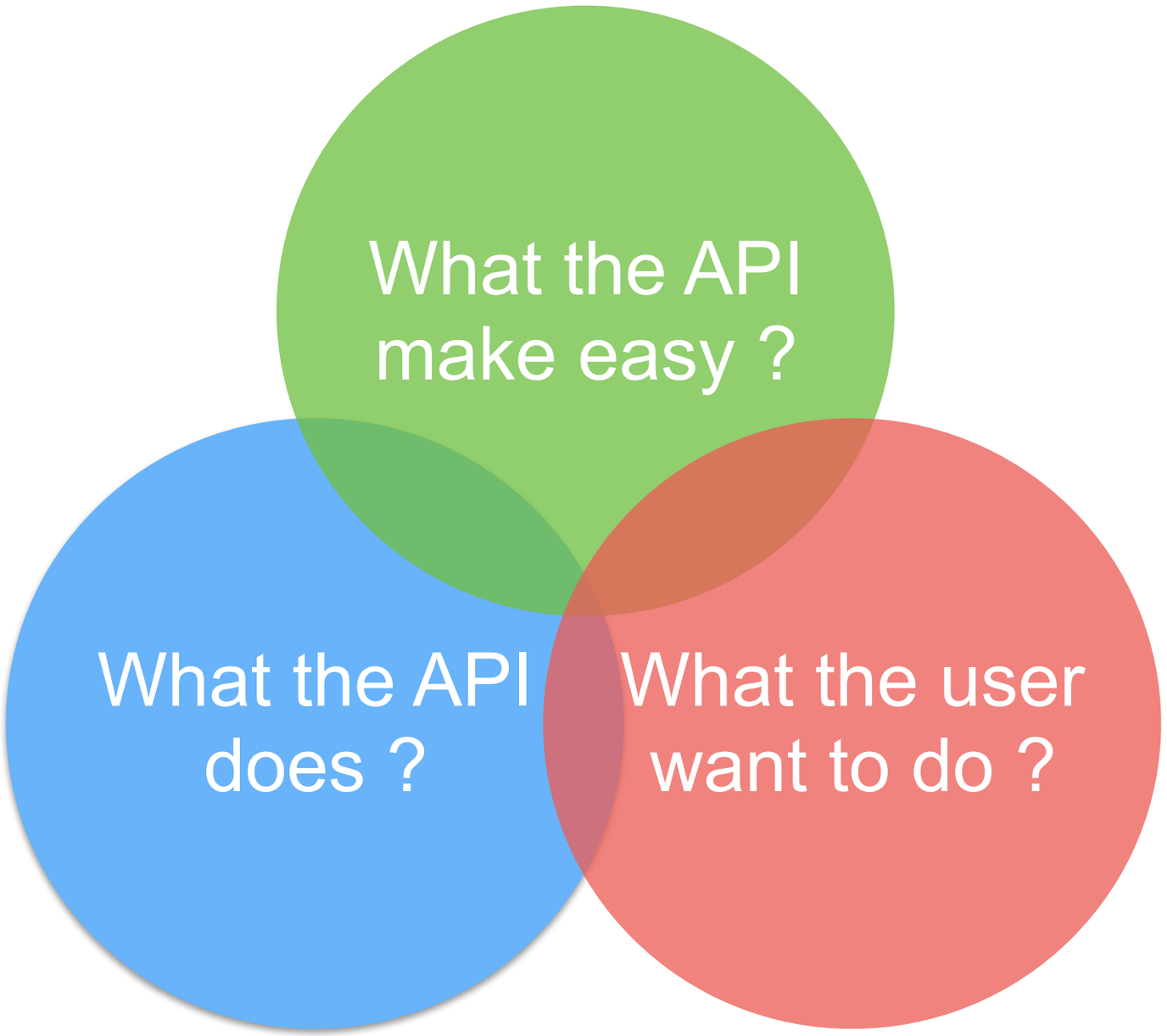


HTTP Methods meaning

Method	Meaning
GET	Read data
POST	Create/Insert new data
PUT/PATCH	Update data or insert if a new id
DELETE	Delete data

Response format ?

Good APIs ?



Java Framework for RESTful

undertow

Jersey
RESTful Web Services in Java

Java
J2EE

WildFly SWARM

JAX-RS

Ratpack
A toolkit for JVM web applications

REST Easy

spring
by Pivotal

Spark

Restlet

akka

Apache CXF

STRUTS

play

unirest
Lightweight HTTP Request Client Libraries

Dropwizard

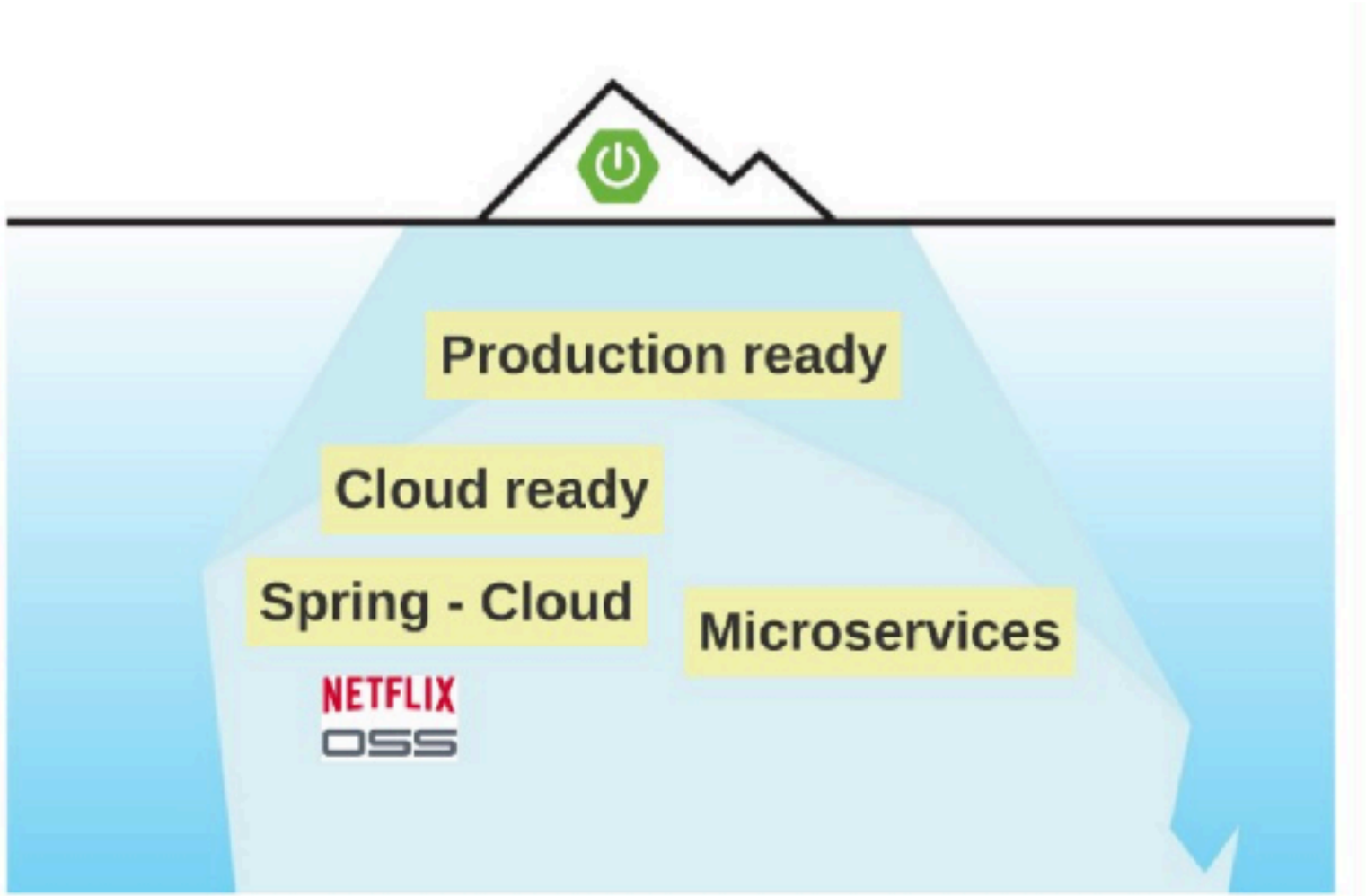
NINJA
web framework

ACT.framework

Hello Spring Boot 2.x

Why ?

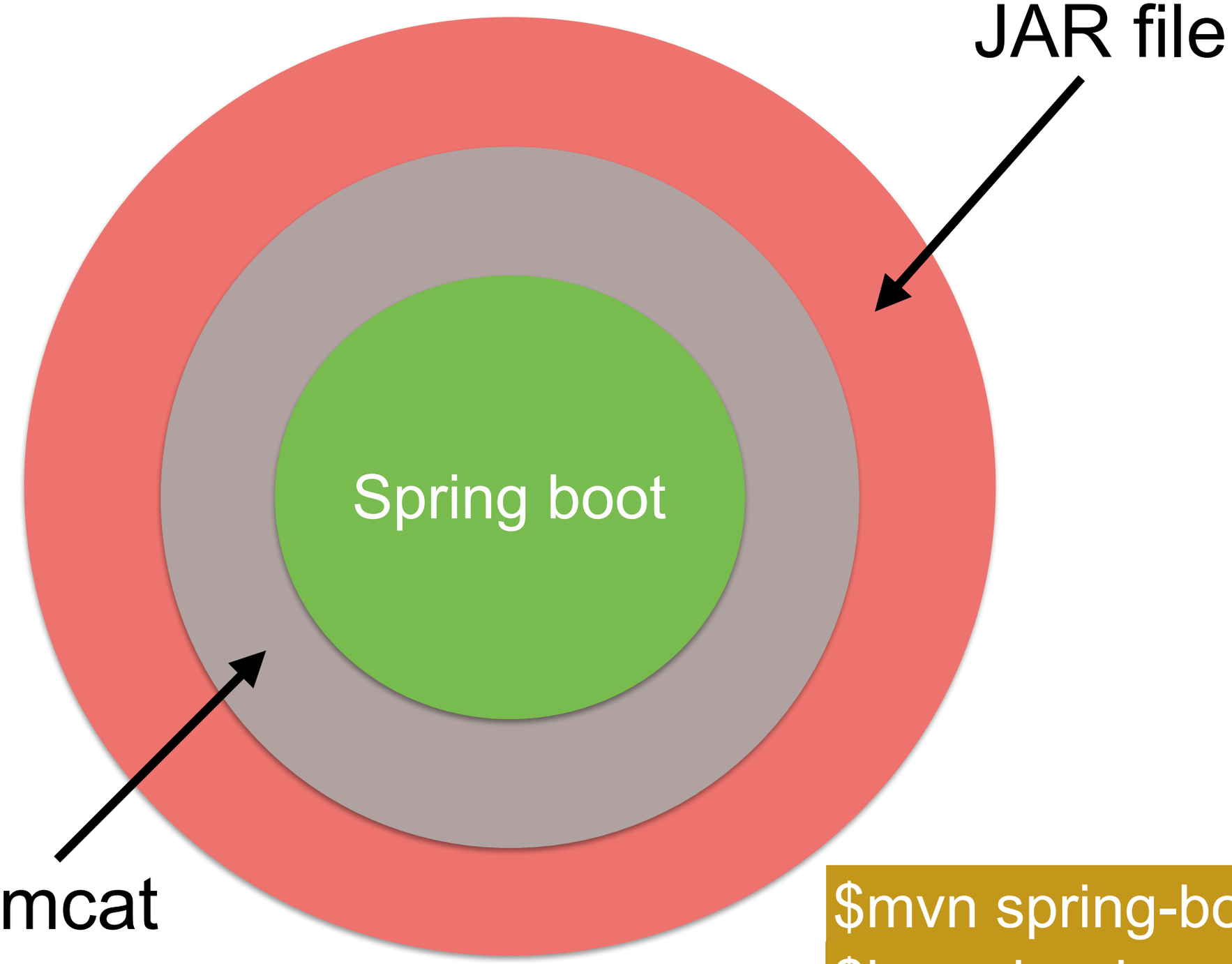
Application skeleton generator
Reduce effort to add new technologies



What ?

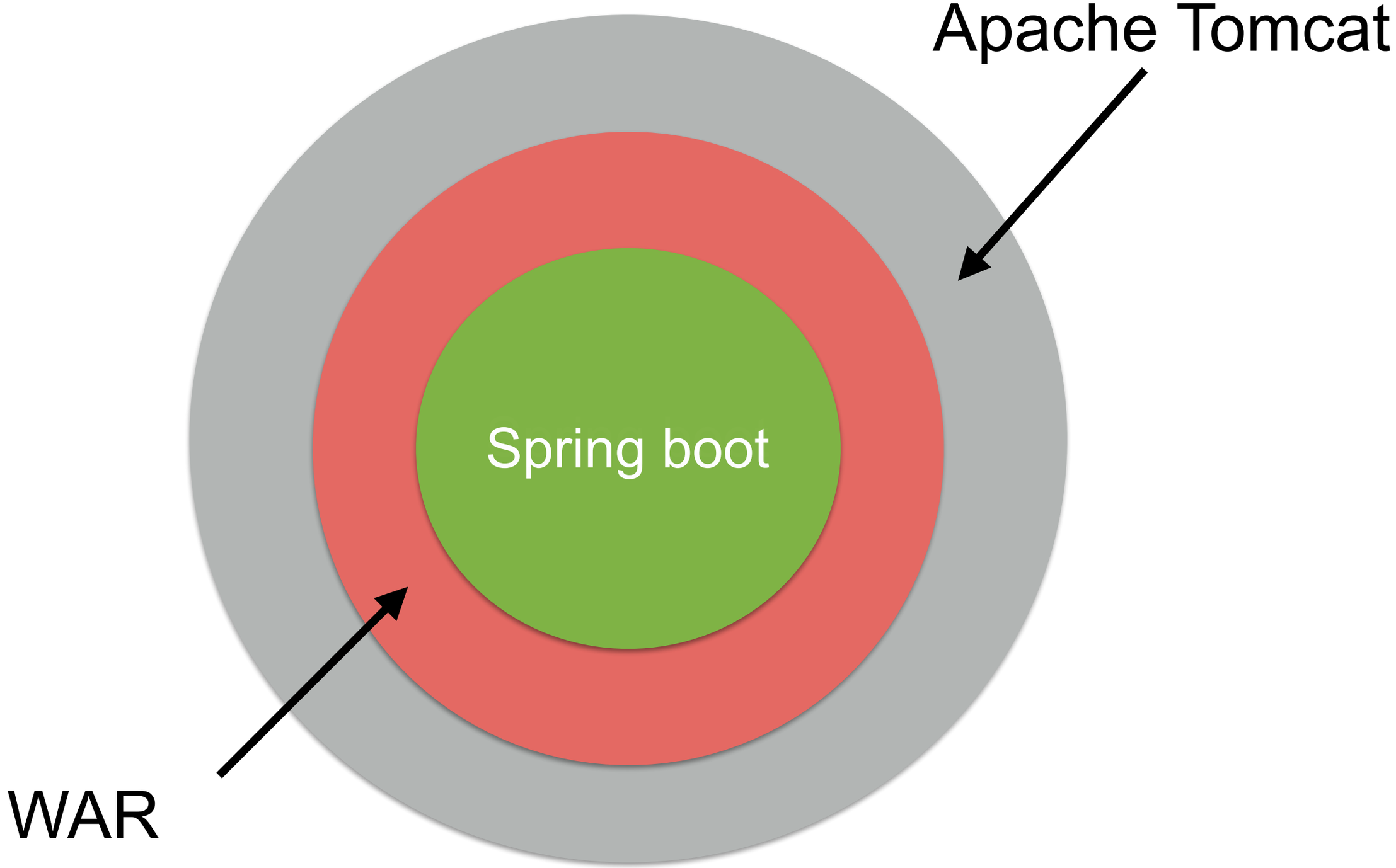
Embedded application server
Integration with tools/technologies (starter)
Production tools (monitoring, health check)
Configuration management
Dev tools
No source code generation, no XML

How ?



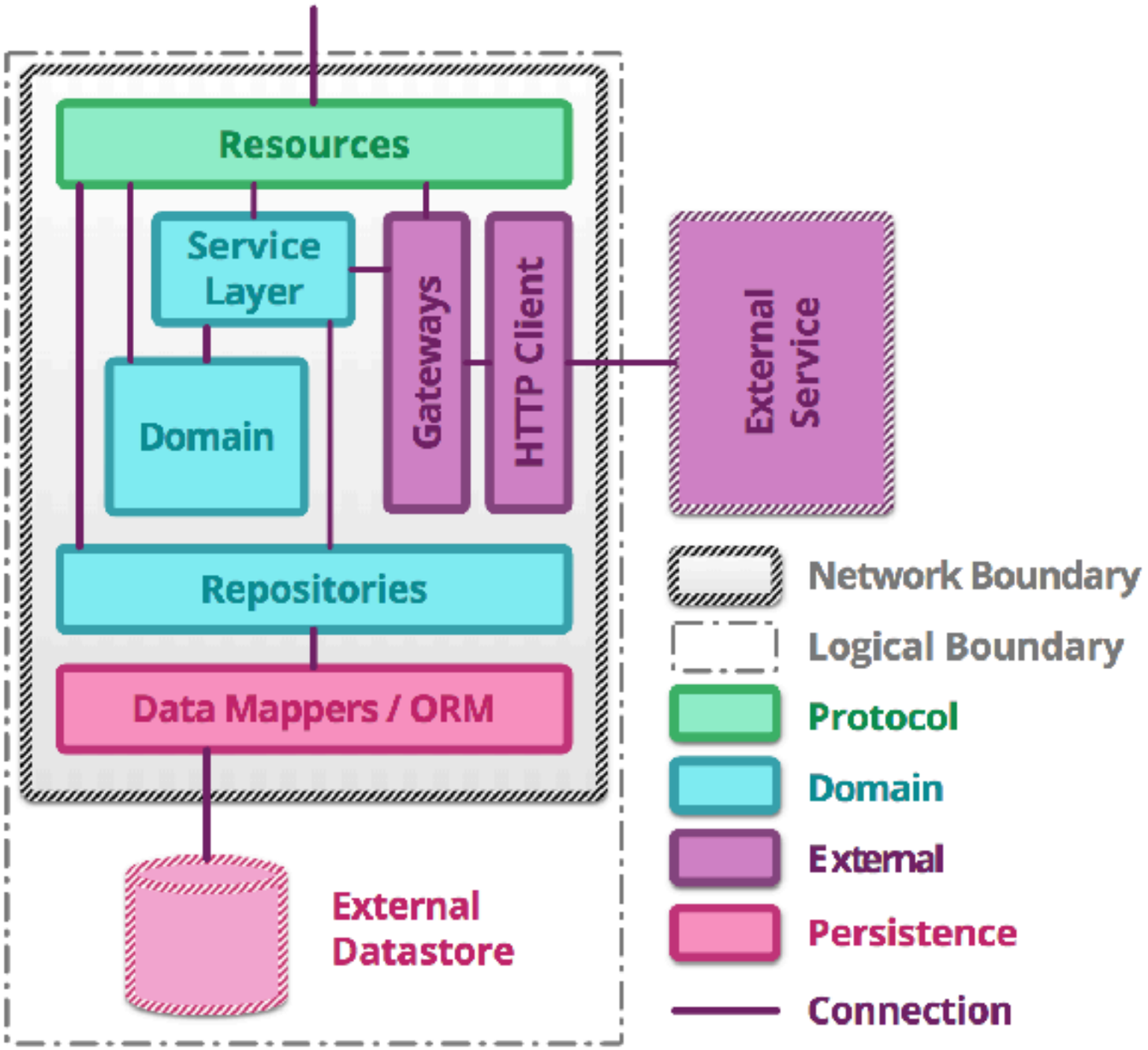

```
$mvn spring-boot:run
$java -jar demo.jar
```

How ?

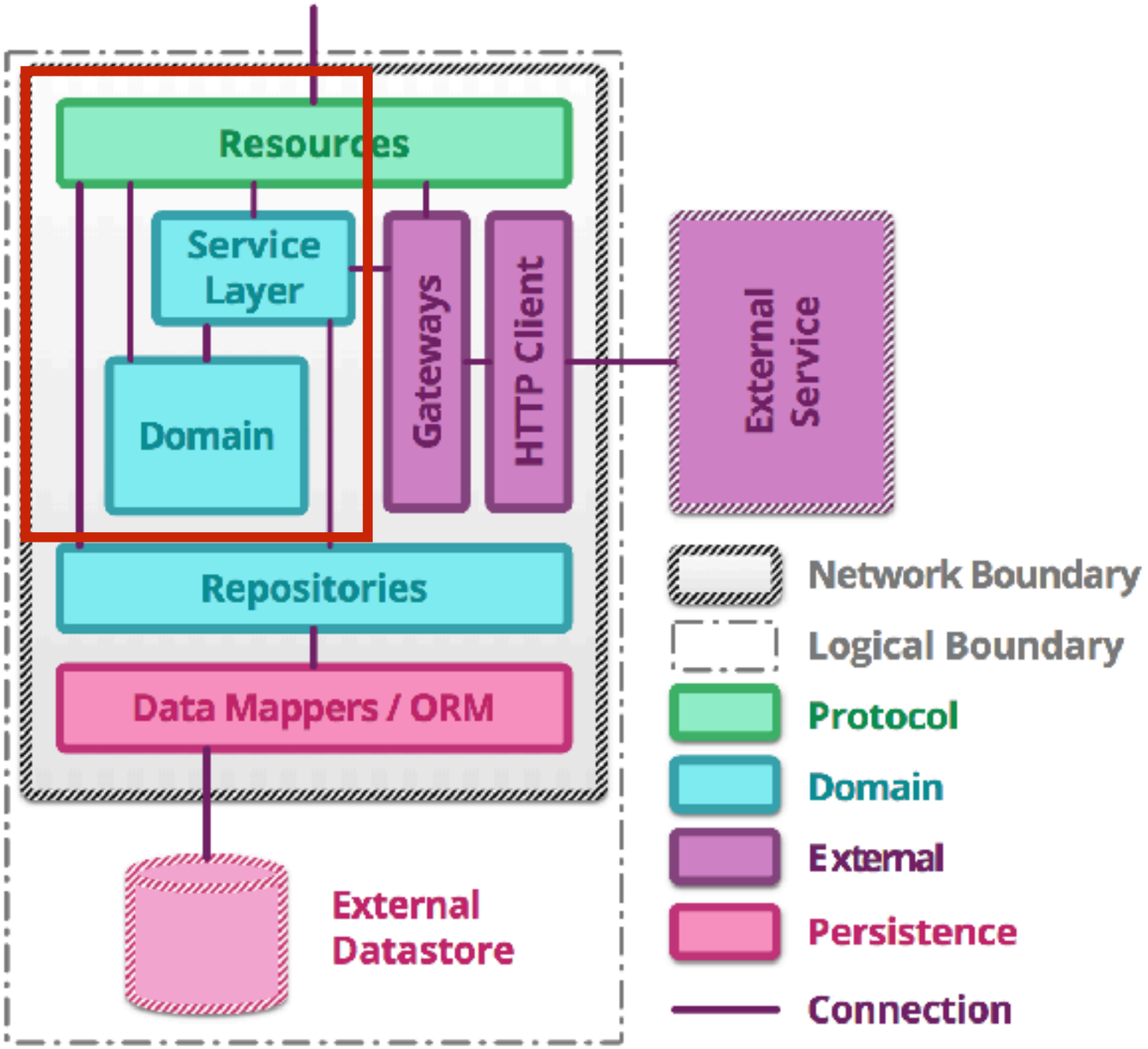


Building RESTful API with Spring Boot

Service Structure



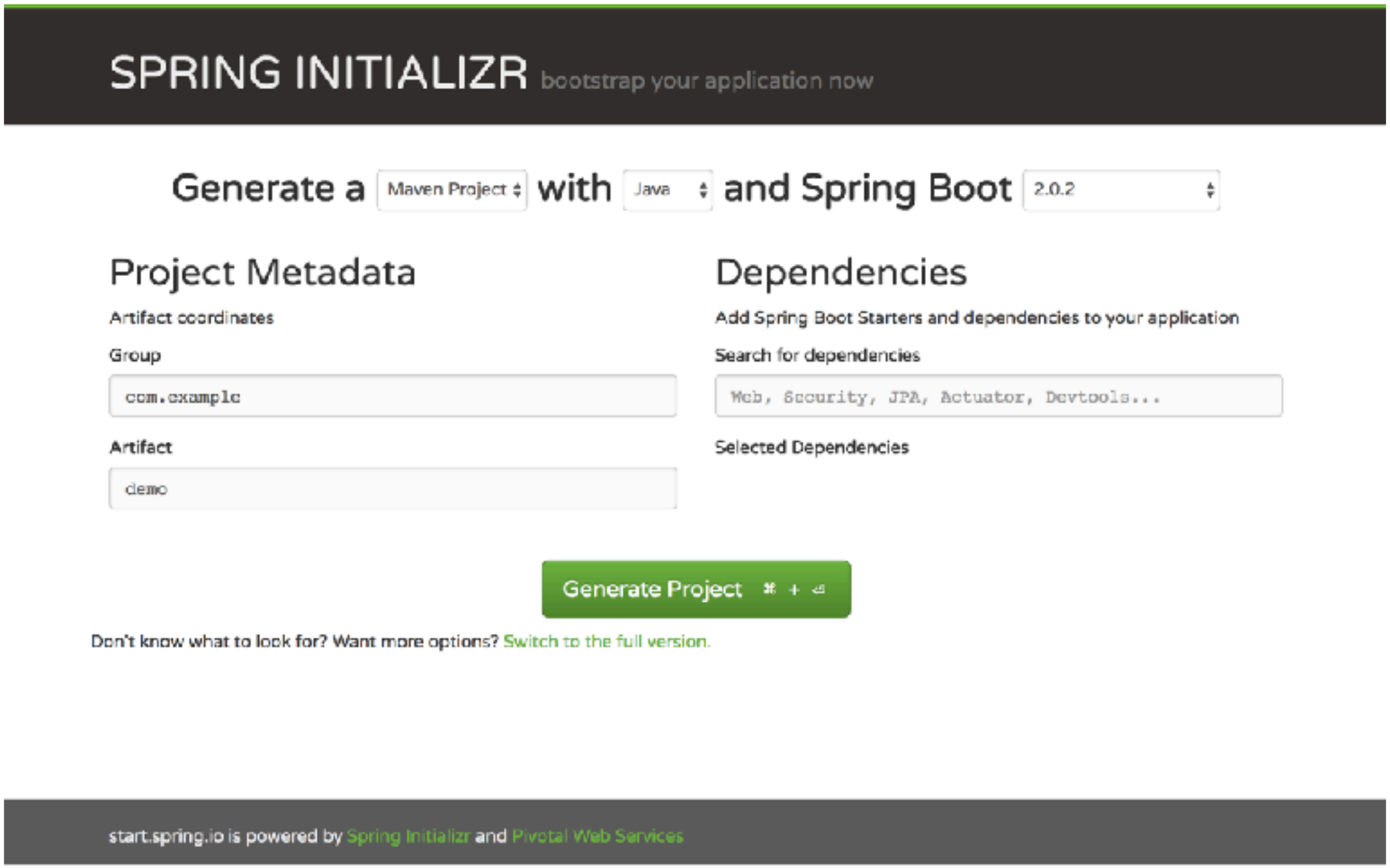
Service Structure



Create project with Spring Initializr

Spring Initializr

https://start.spring.io/



Structure of Spring Boot

Spring Boot main class

```java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

Run project

$./mvnw package
$java -jar target/<file name>.jar

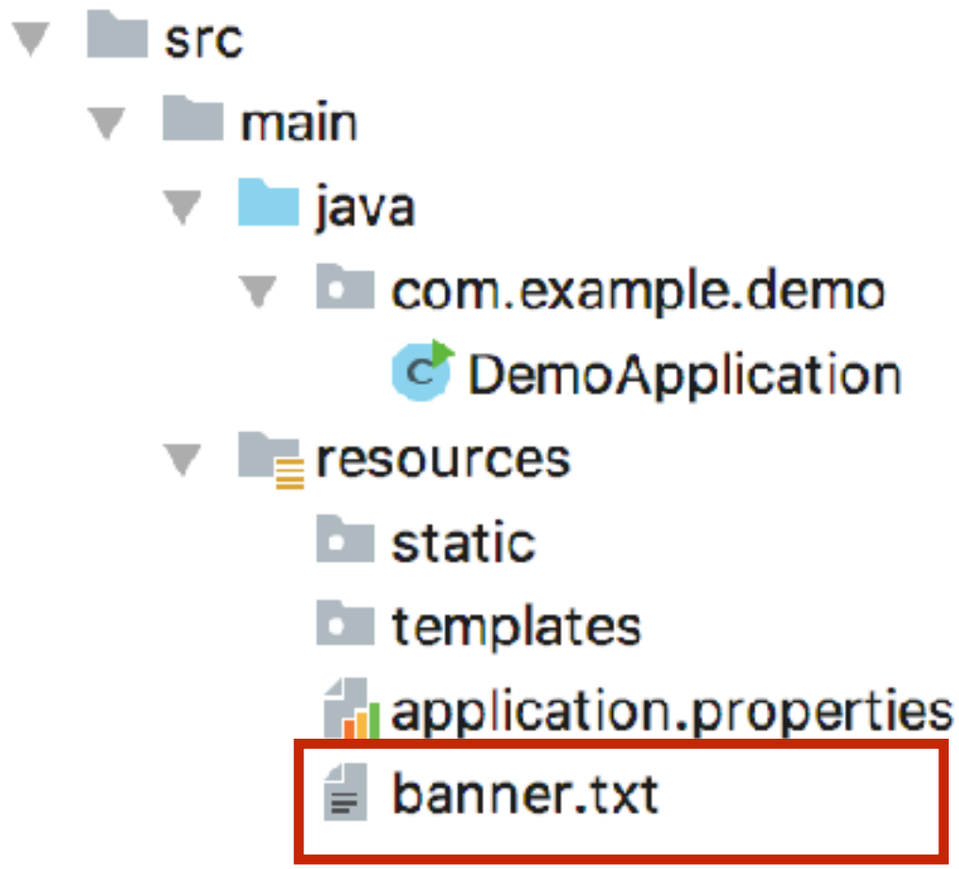
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '   |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.2.RELEASE)

2018-06-07 13:03:30.412  INFO 12828 --- [
oApplication               : Starting DemoApplication on
D 12828 (started by somkiat in /Users/somkiat/Down
2018-06-07 13:03:30.418  INFO 12828 --- [
oApplication               : No active profile set, fall
```

Custom banner


```
2018-06-07 13:03:30.412  INFO 12828 --- [
oApplication               : Starting DemoApplication on
D 12828 (started by somkiat in /Users/somkiat/Down
2018-06-07 13:03:30.418  INFO 12828 --- [
oApplication               : No active profile set, fall
```

Custom banner (1)

Create file banner.txt or banner.png in resources folder



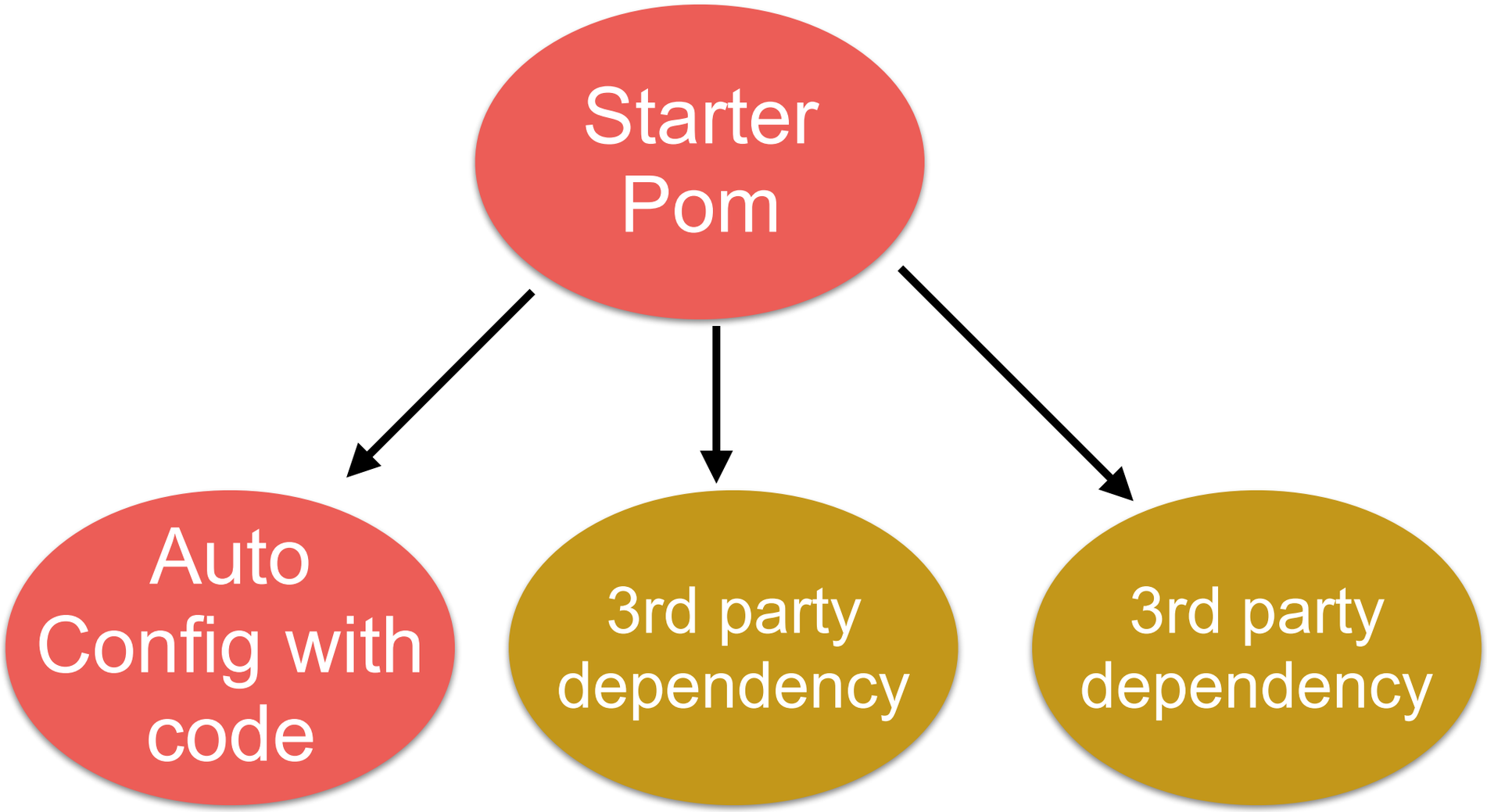
Custom banner (2)

```
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        ImageBanner banner = new ImageBanner(
                new ClassPathResource("try.png"));

        new SpringApplicationBuilder()
                .sources(DemoApplication.class)
                .banner(banner)
                .run();
    }
}
```

Disable banner

```java
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        new SpringApplicationBuilder()
        .sources(DemoApplication.class)
        .bannerMode(Banner.Mode.OFF)
        .run(args);
    }
}
```

Anatomy of Starter



Configuration management

Properties/XML/YAML

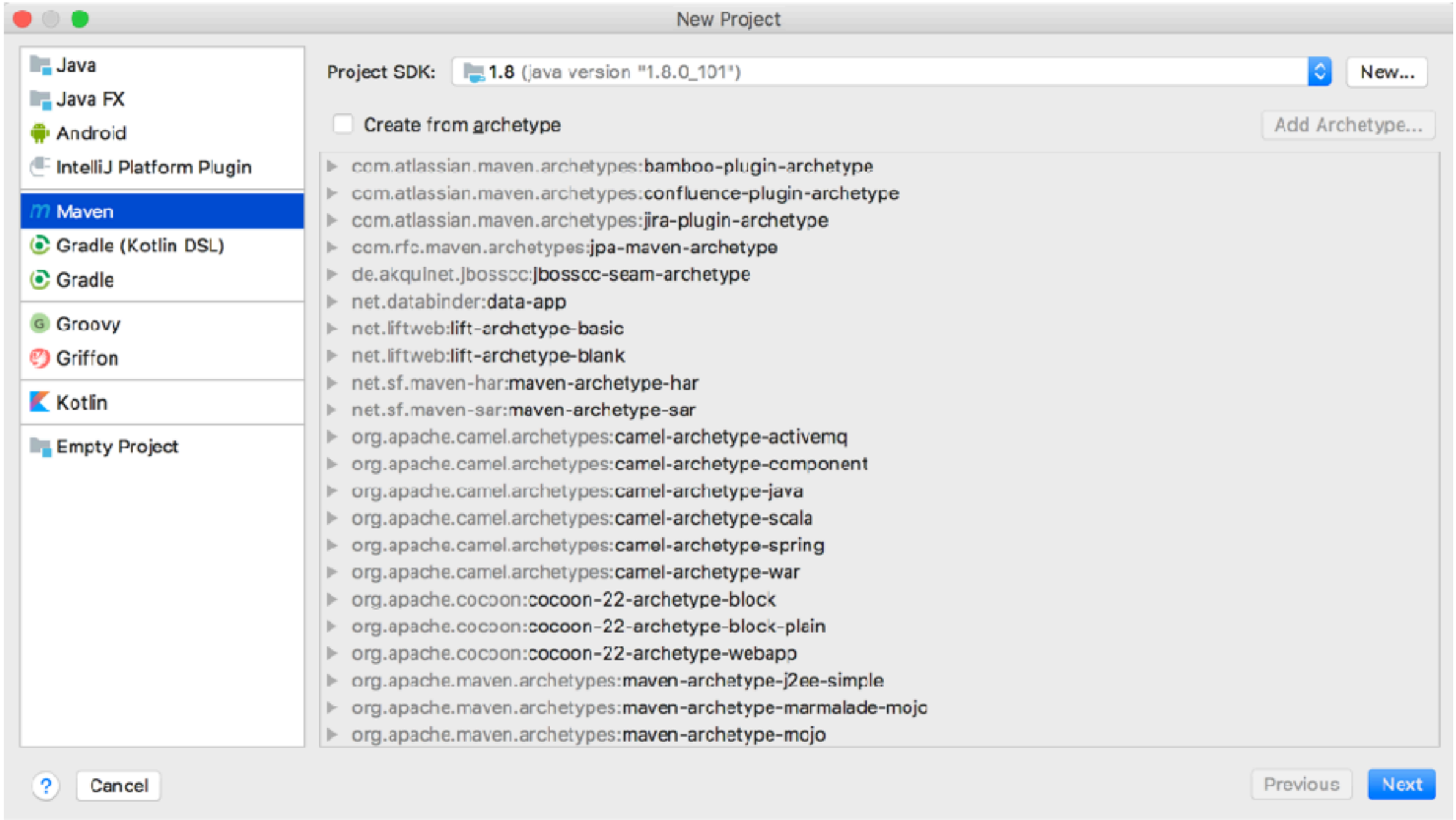
Config server (App, Spring cloud config, git)

17 ways !!!

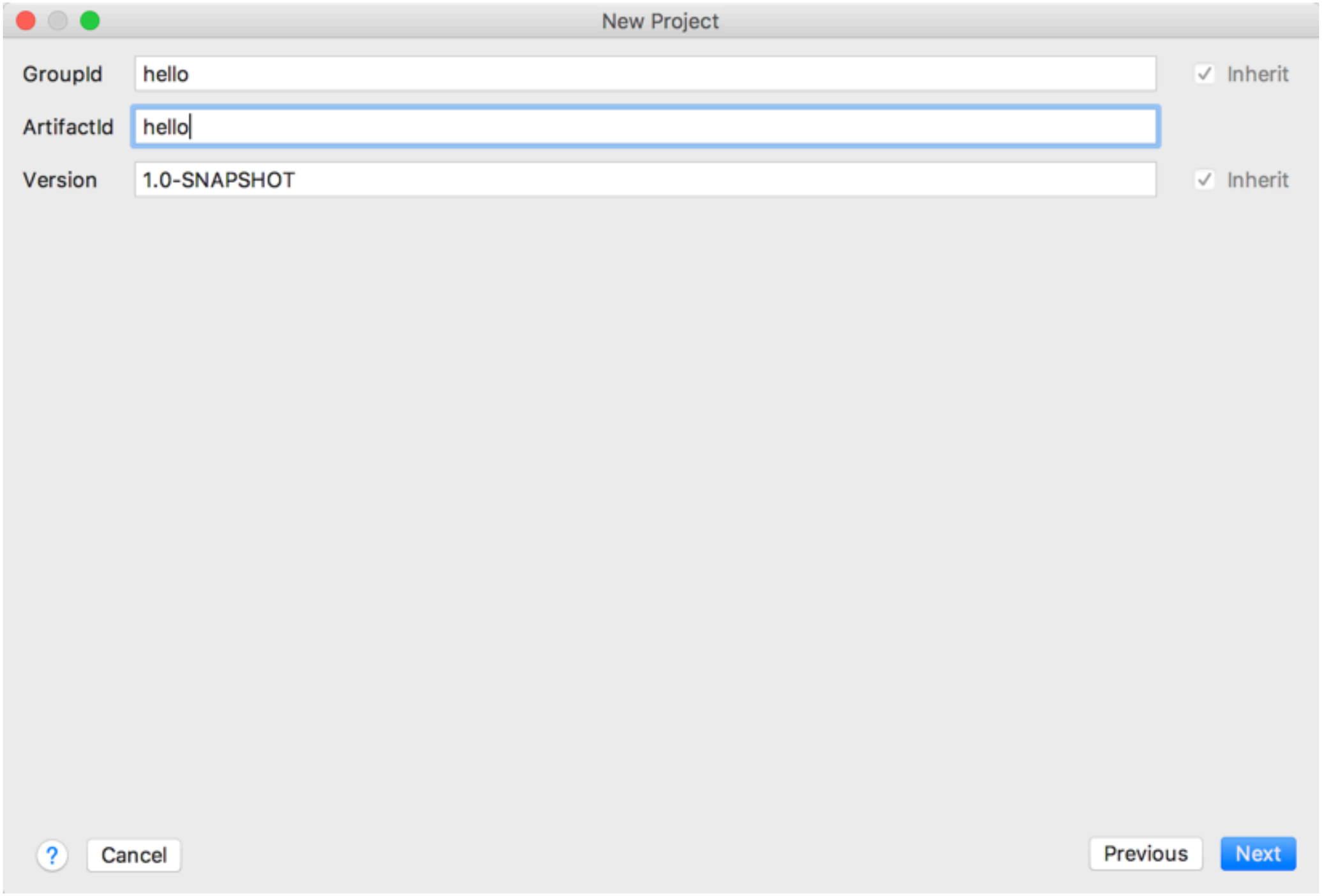
https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html

Create project with IDE

1. Create Maven Project



2. Project Name



3. Modify pom.xml (1)

```xml
<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.o
    <modelVersion>4.0.0</modelVersion>

    <groupId>hello</groupId>
    <artifactId>hello</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.0.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
    </properties>
```

3. Modify pom.xml (2)

```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <finalName>hello</finalName>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

4. Create Spring boot application

hello.HelloApplication.java

```
package hello;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class HelloApplication {

    public static void main(String[] args) {
        SpringApplication.run(HelloApplication.class, args);
    }

}
```

5. Create model class

hello.model.Hello.java

```
package hello.domain;

public class Hello {

    private String message;

    public Hello(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

6. Create REST Controller

hello.controller.HelloController.java

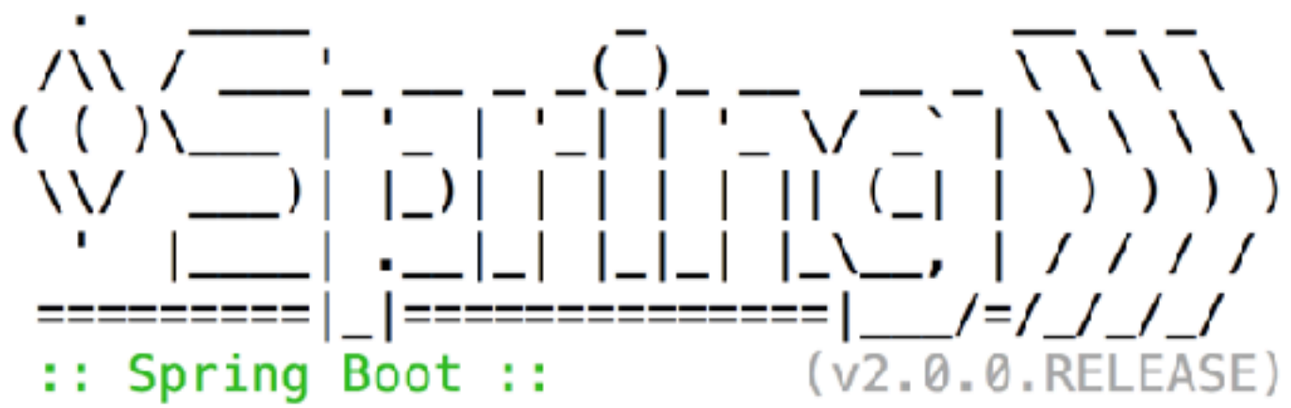
```java
package hello.controller;

import hello.domain.Hello;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @GetMapping("/hello/{name}")
    public Hello sayHi(@PathVariable String name) {
        return new Hello("Hello " + name);
    }

}
```

7. Compile and Packaging

$mvn clean package

8. Run

$java -jar target/hello.jar

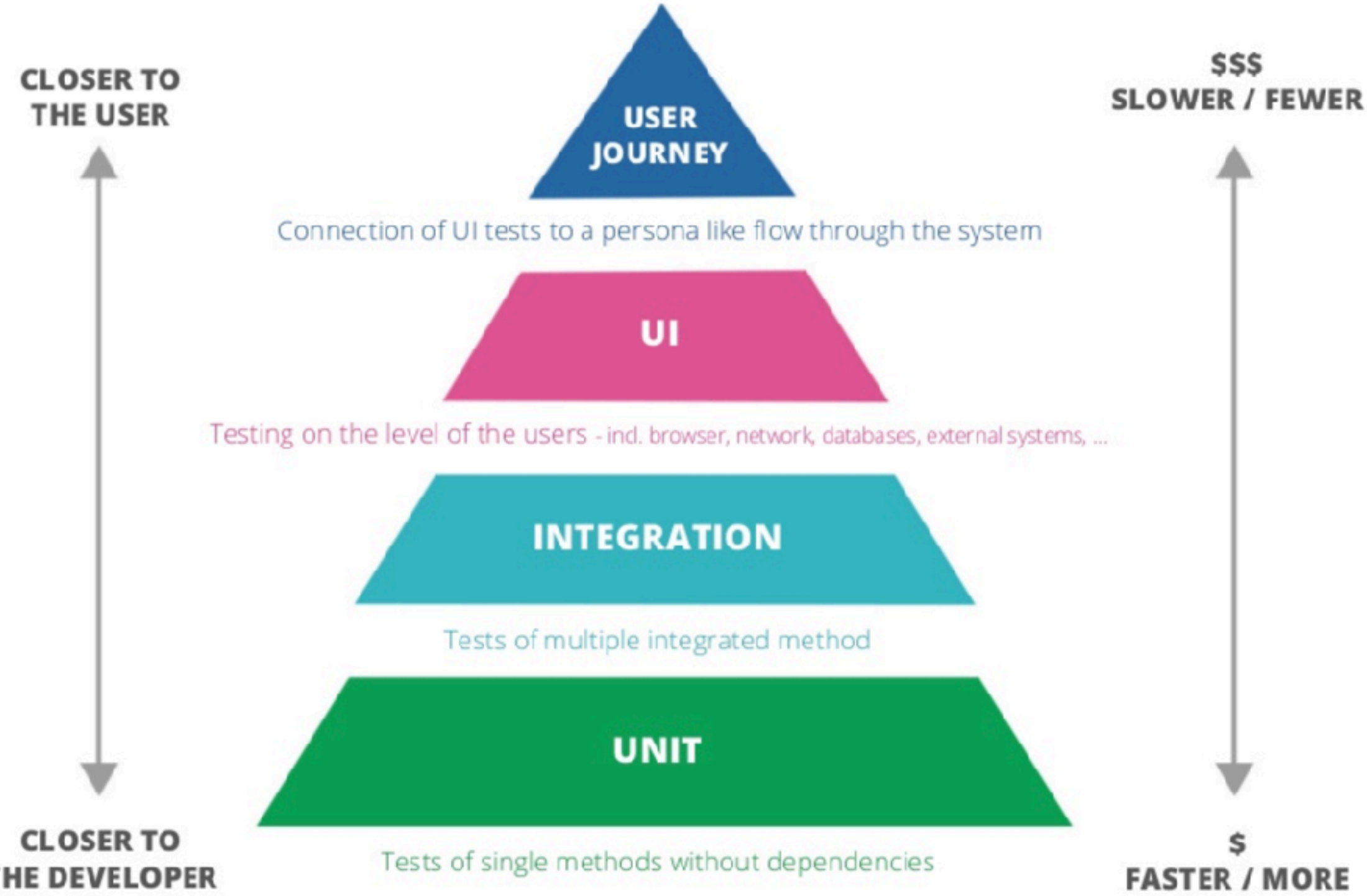



```
2018-03-05 23:37:18.018  INFO 30560 --- [           mair
                    : Starting HelloApplication v1.0-SNAPSH(
th PID 30560 (/Users/somkiat/data/slide/microservice/sl:
op/course-microservice/slide/4days-workshop/workshop/he
 by somkiat in /Users/somkiat/data/slide/microservice/s
hop/course-microservice/slide/4days-workshop/workshop/he
2018-03-05 23:37:18.023  INFO 30560 --- [           mair
                    : No active profile set, falling back to
2018-03-05 23:37:18.138  INFO 30560 --- [           mair
```

9. Open in browser

http://localhost:8080/hello/somkiat

How to test the Hello service ?



Unit tests

How to use model ?

```
public class HelloTest {

    @Test
    public void success_to_create_model_with_constructor() {
        Hello hello = new Hello("Somkiat");
        assertEquals( expected: "Somkiat", hello.getMessage());
    }

}
```

API/Controller tests

How to use controller ?

Spring boot provides MockMvc to test Controller

```java
package hello.controller;

import hello.domain.Hello;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @GetMapping("/hello/{name}")
    public Hello sayHi(@PathVariable String name) {
        return new Hello("Hello " + name);
    }

}
```

API/Controller tests

```java
@RunWith(SpringRunner.class)
@WebMvcTest(controllers = HelloController.class)
public class HelloControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Test
    public void shouldReturnHelloSomkiat() throws Exception {
        mockMvc.perform(get( urlTemplate: "/hello/somkiat"))
                .andExpect(
                        jsonPath( expression: "$.message")
                                .value( expectedValue: "Hello somkiat"))
                .andExpect(status().is2xxSuccessful());
    }

}
```

Compile with testing

$mvn clean package

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
```

% of Code/Test coverage

Add coverage to pom.xml (1)

```xml
<build>
    <finalName>hello</finalName>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>

        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.5.1</version>
            <configuration>
                <source>${java.version}</source>
                <target>${java.version}</target>
                <encoding>${project.build.sourceEncoding}</encoding>
            </configuration>
        </plugin>
```

Add coverage to pom.xml (2)

```xml
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>cobertura-maven-plugin</artifactId>
    <version>2.7</version>
    <configuration>
        <formats>
            <format>html</format>
            <format>xml</format>
        </formats>
    </configuration>
    <executions>
        <execution>
            <phase>package</phase>
            <goals>
                <goal>cobertura</goal>
            </goals>
        </execution>
    </executions>
    <dependencies>
        <dependency>
            <groupId>org.ow2.asm</groupId>
            <artifactId>asm</artifactId>
            <version>5.0.3</version>
        </dependency>
    </dependencies>
</plugin>
```

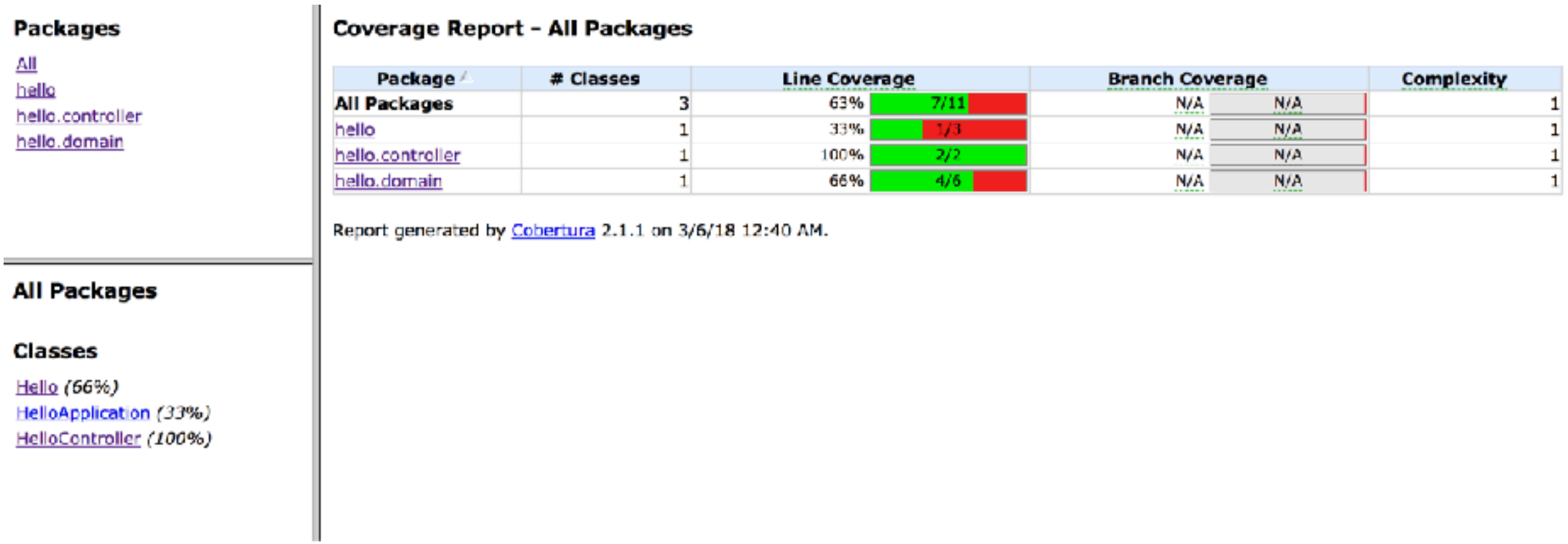
Run test again

$mvn clean package

```
Cobertura Report generation was successful.
Cobertura 2.1.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file
Cobertura: Loaded information on 3 classes.
time: 125ms

Cobertura Report generation was successful.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

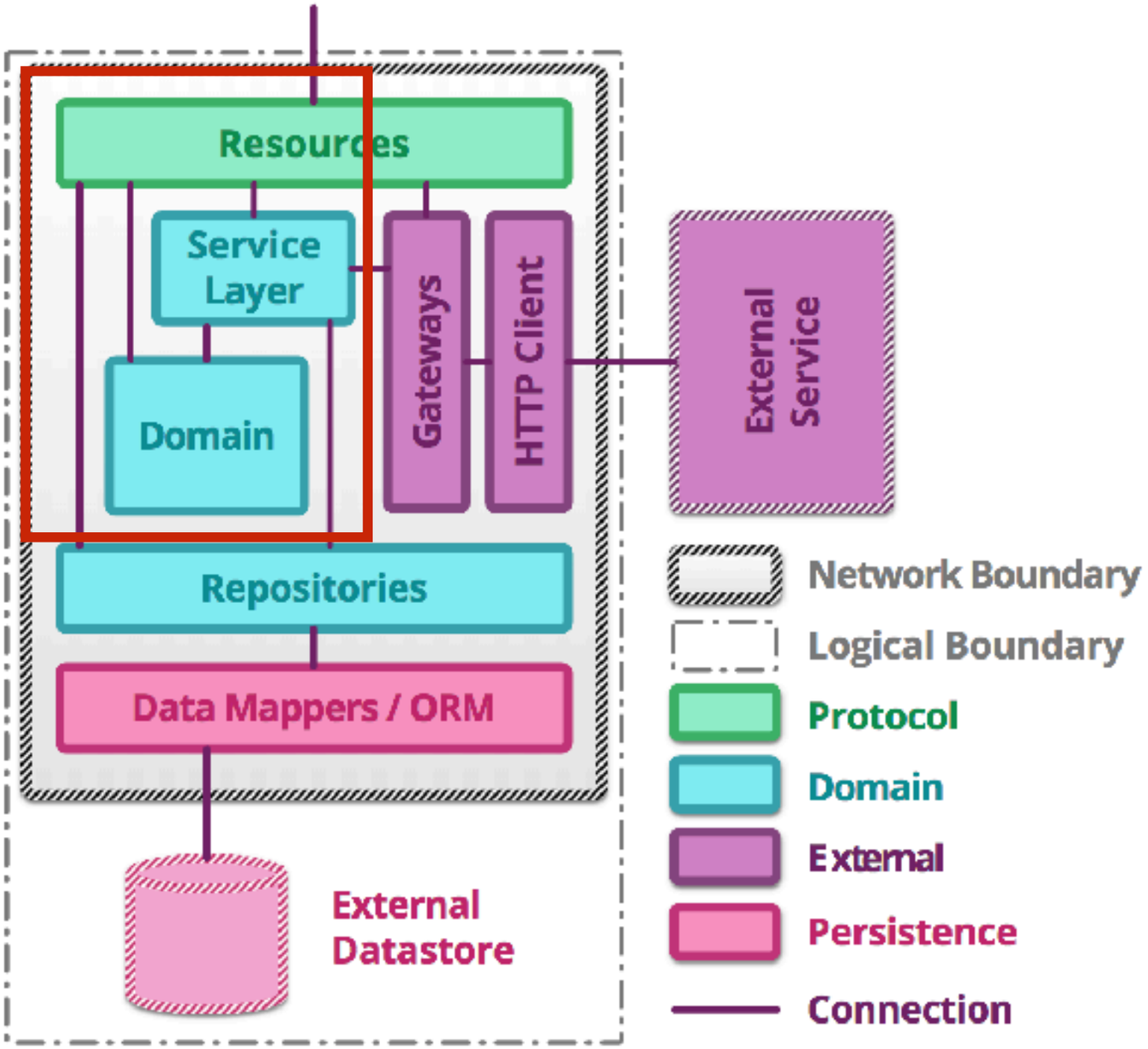
Coverage report

open target/site/cobertura/index.html

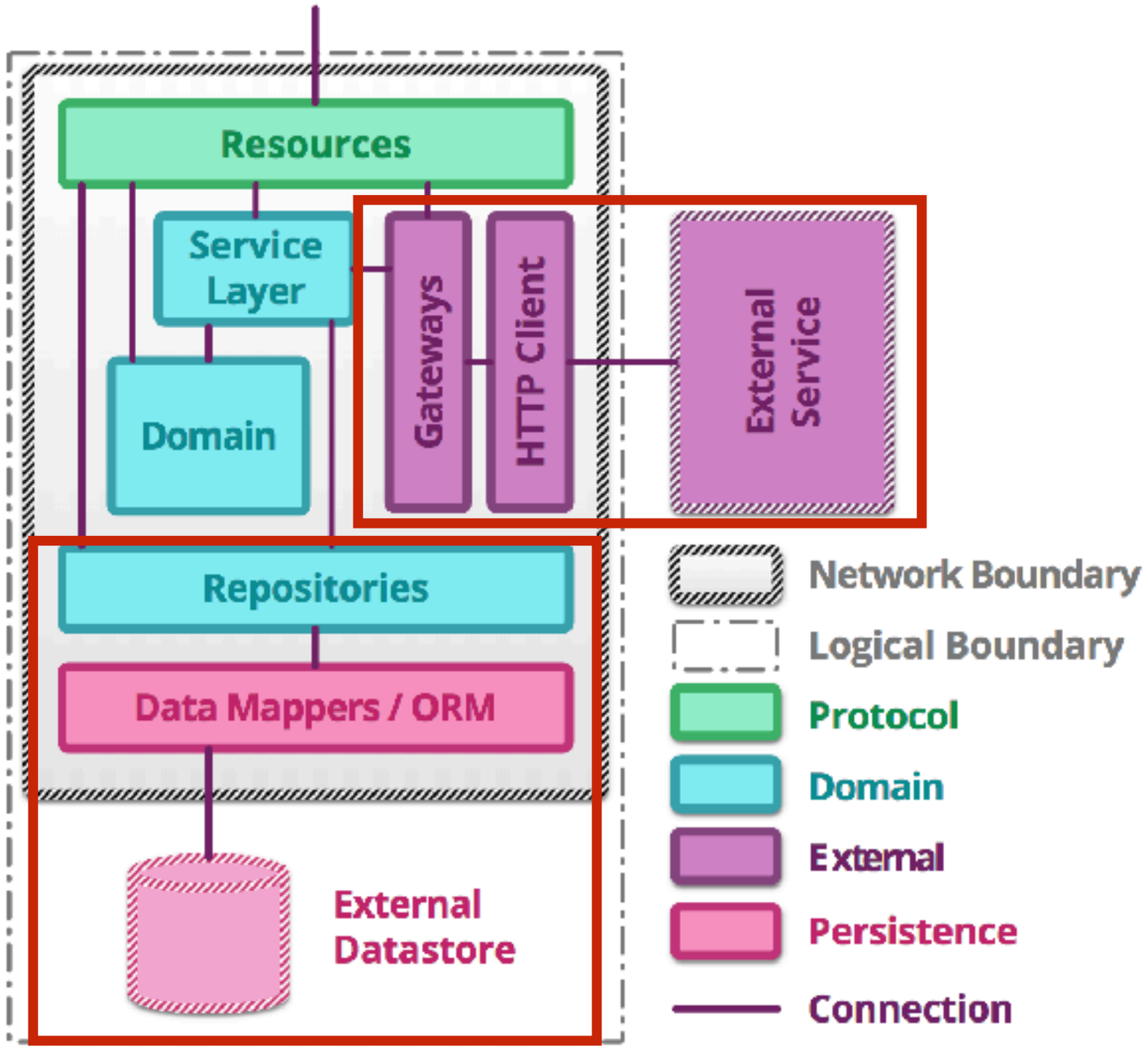


How to improve % of coverage ?

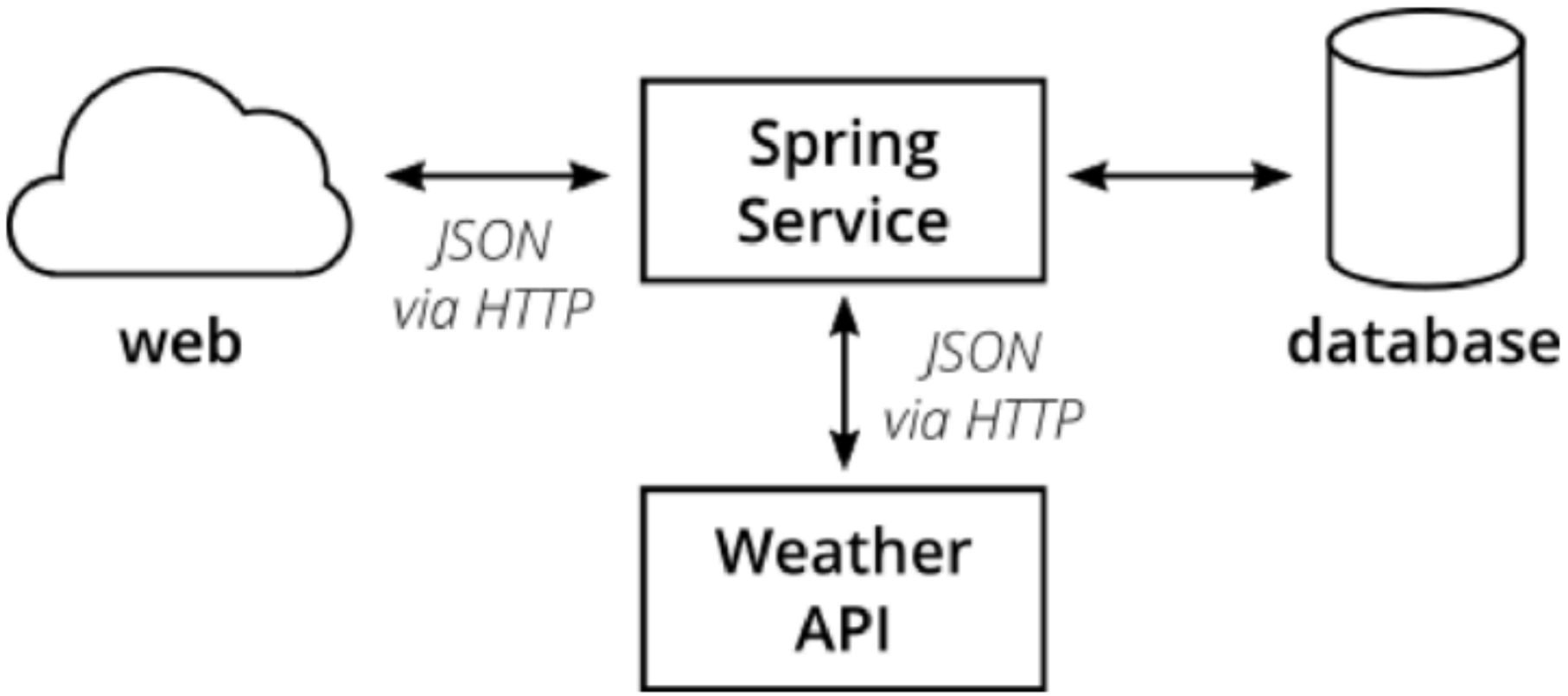
Service Structure



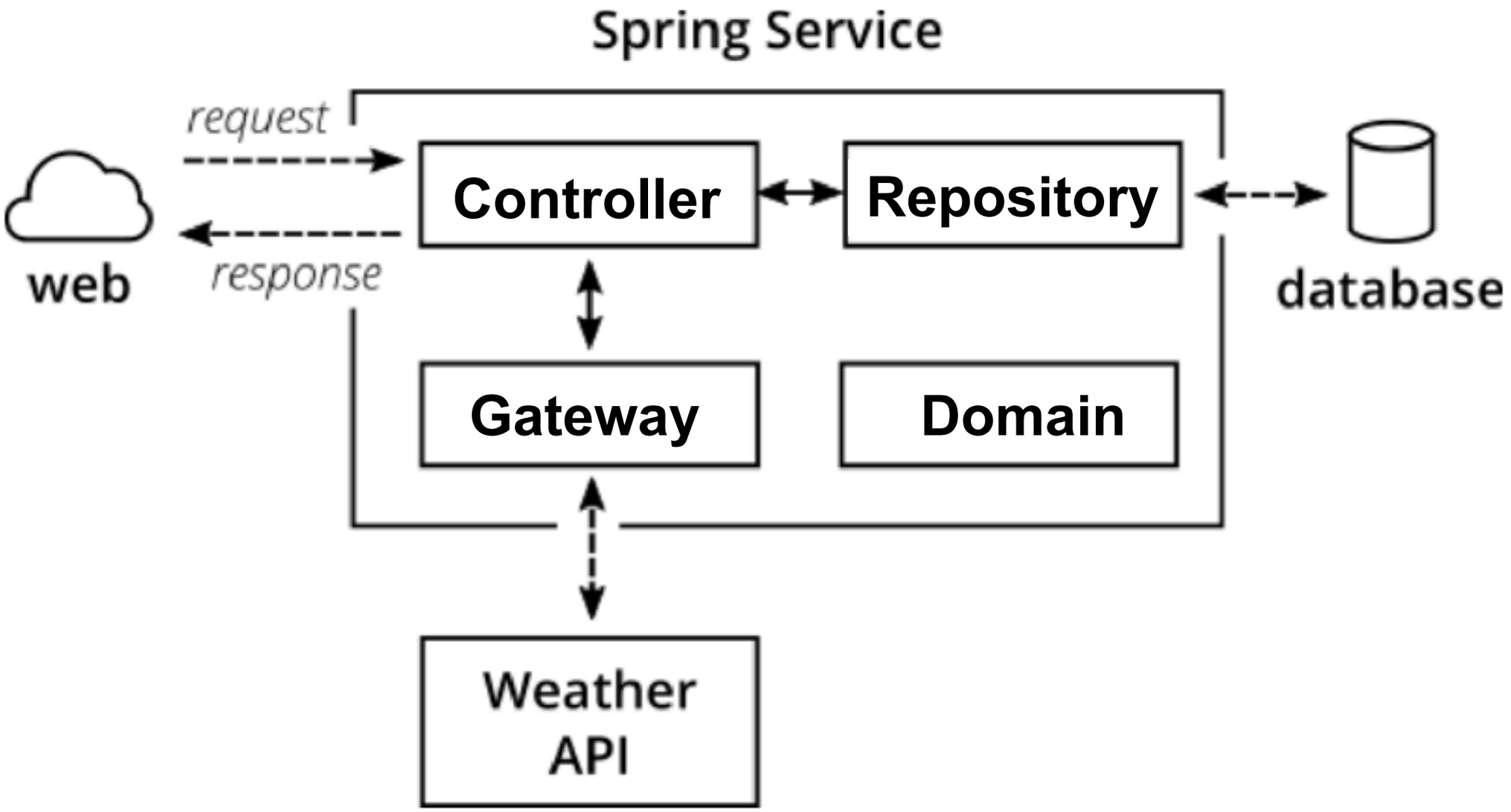
Service Structure



Sample application

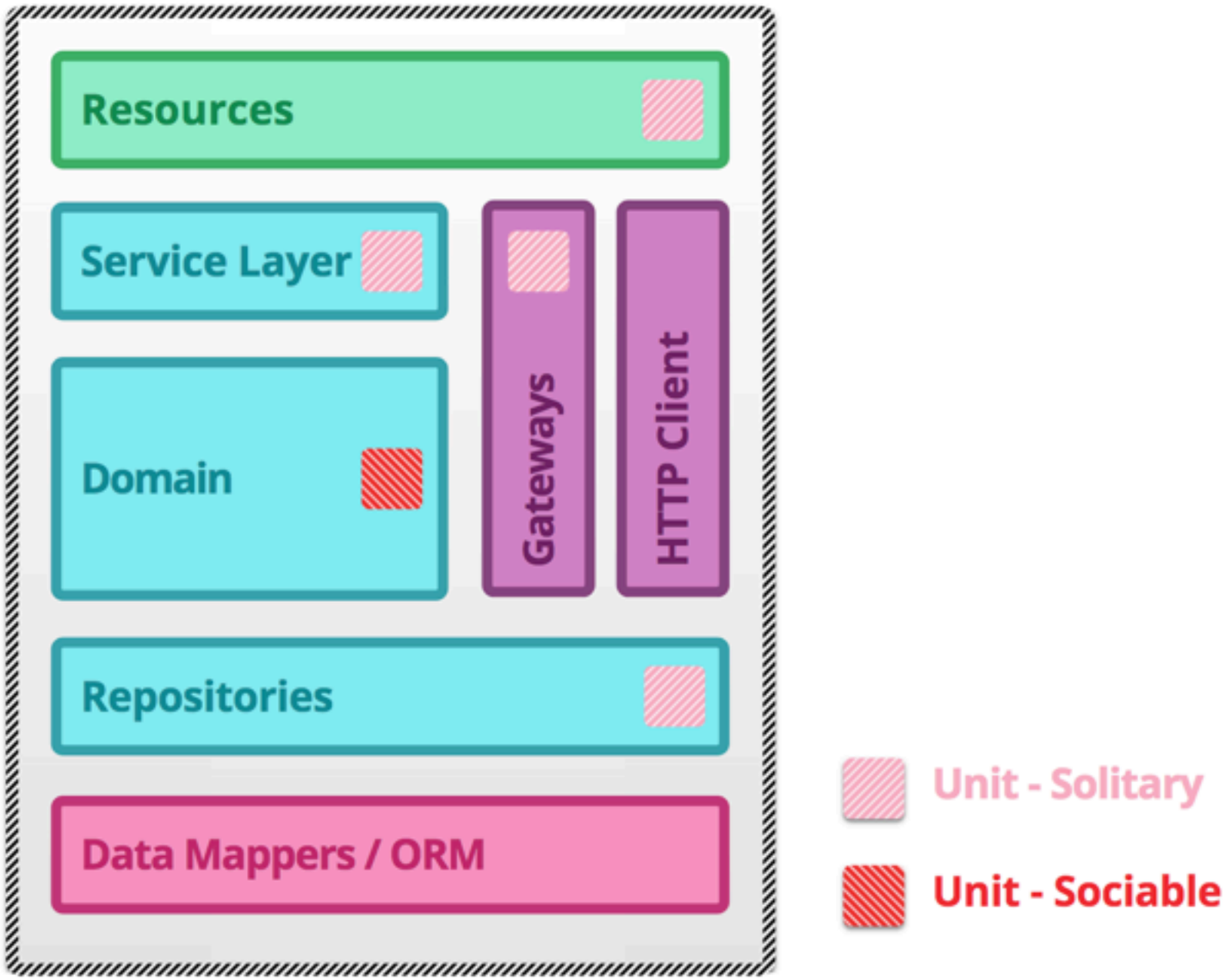


Project structure

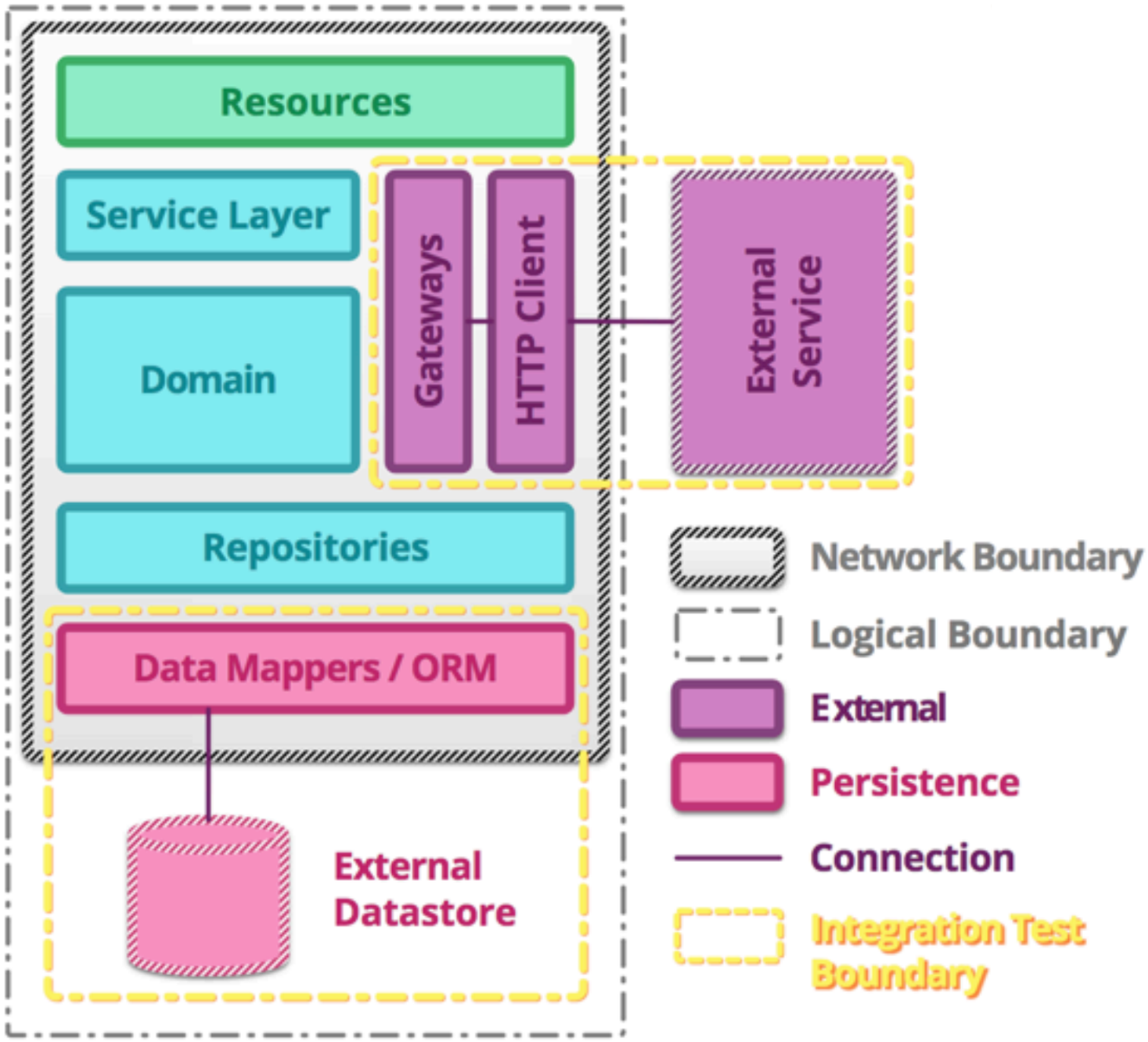


How to test ?

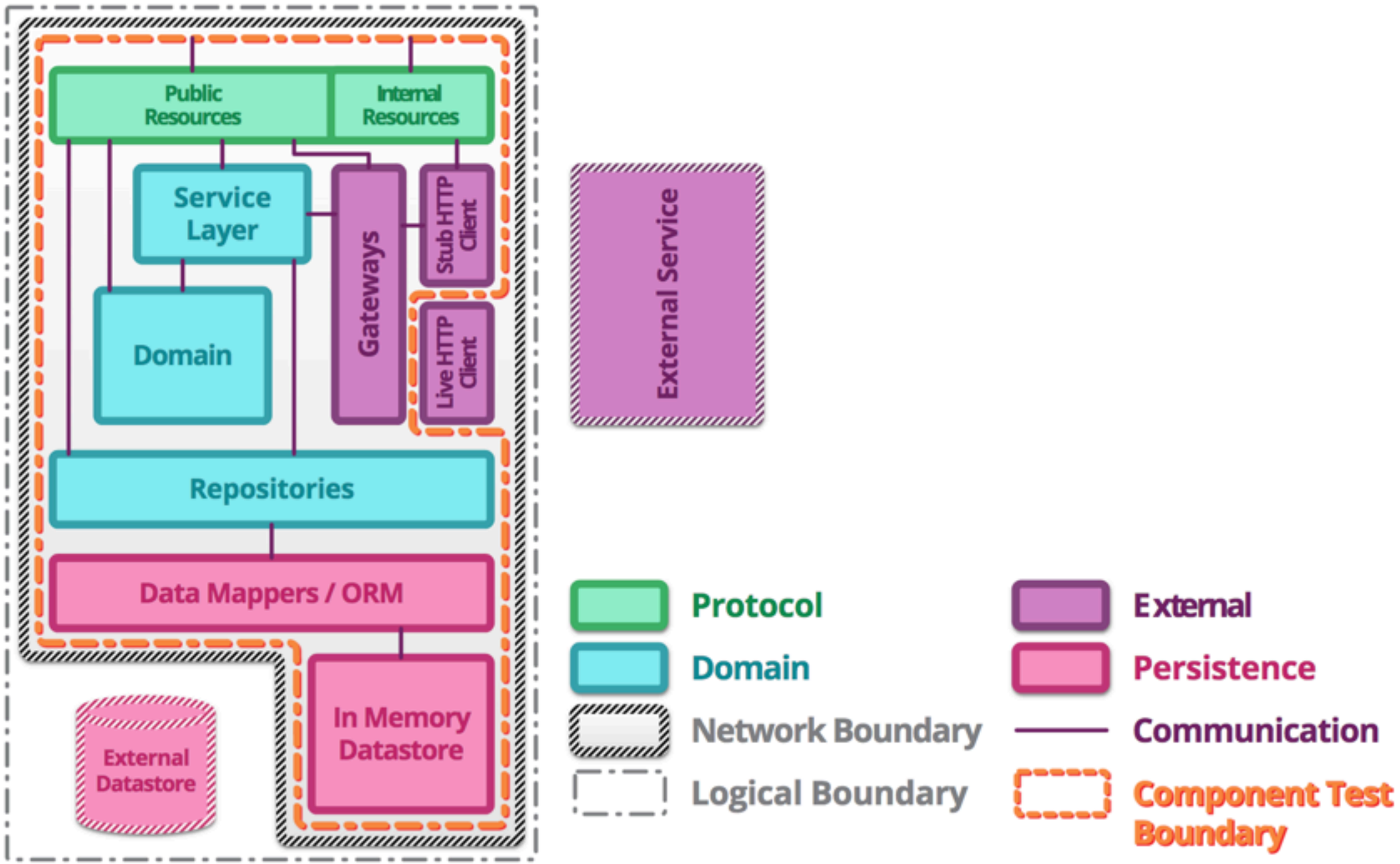
Unit testing



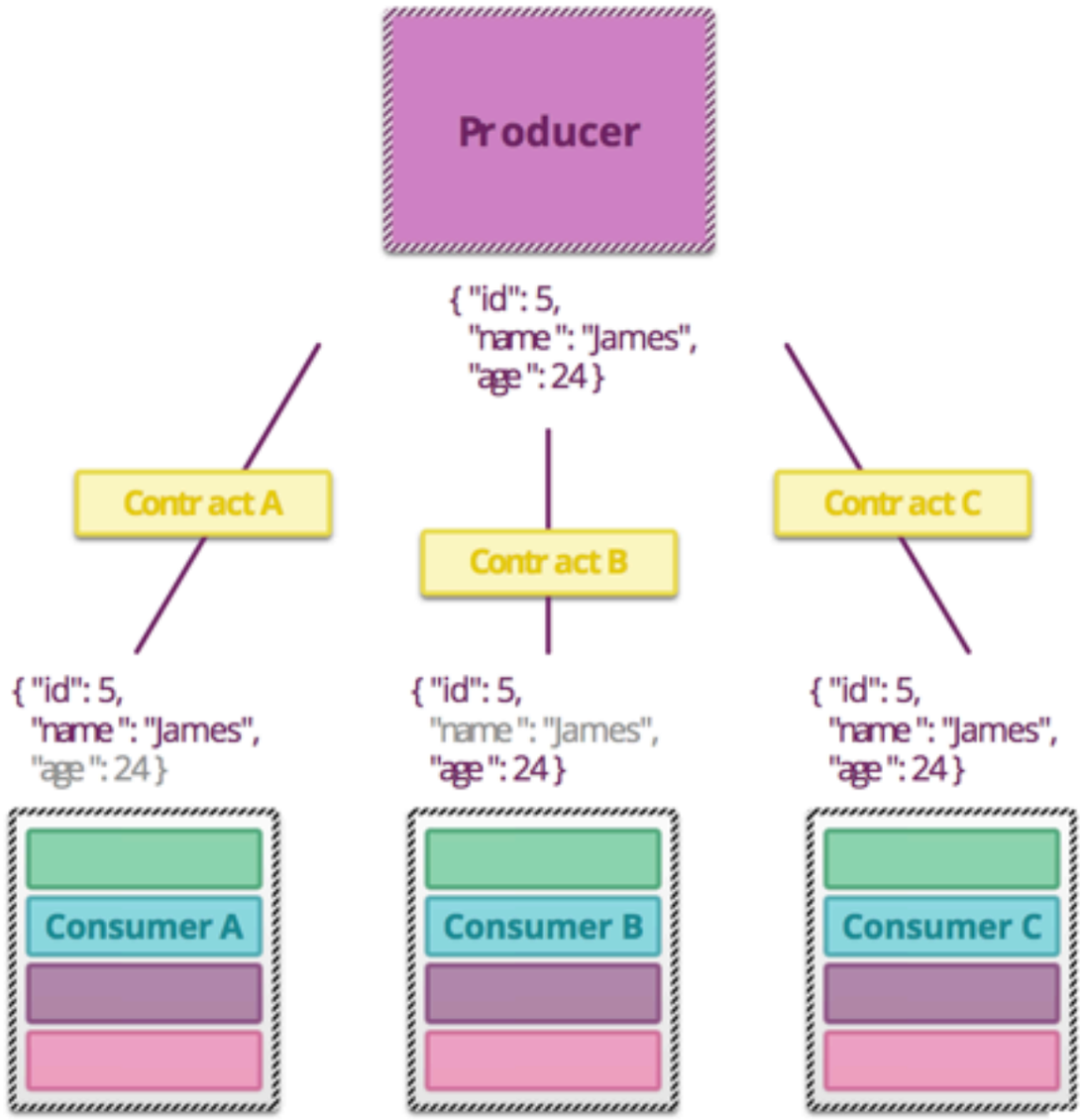
Integration testing



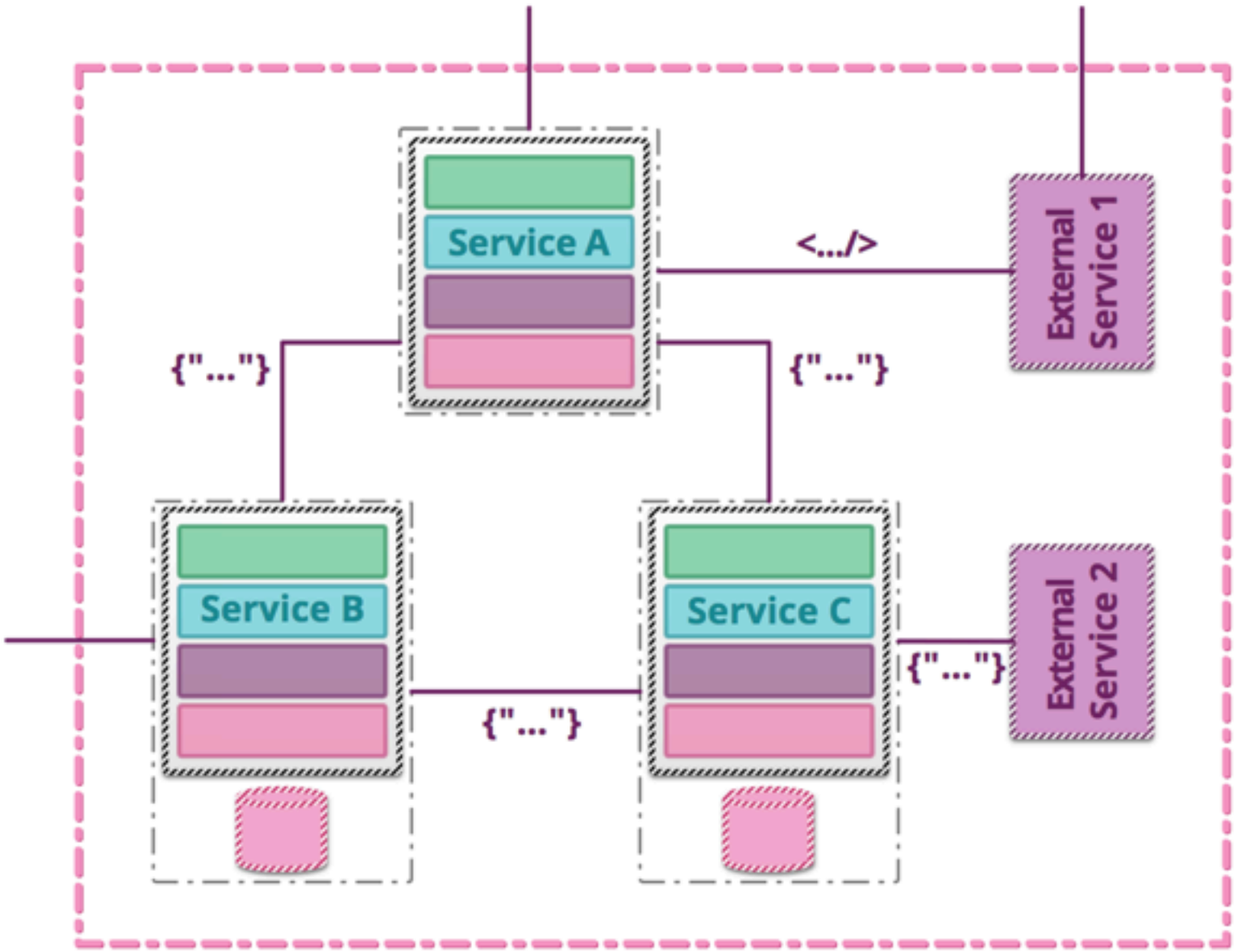
Component testing



Contract testing



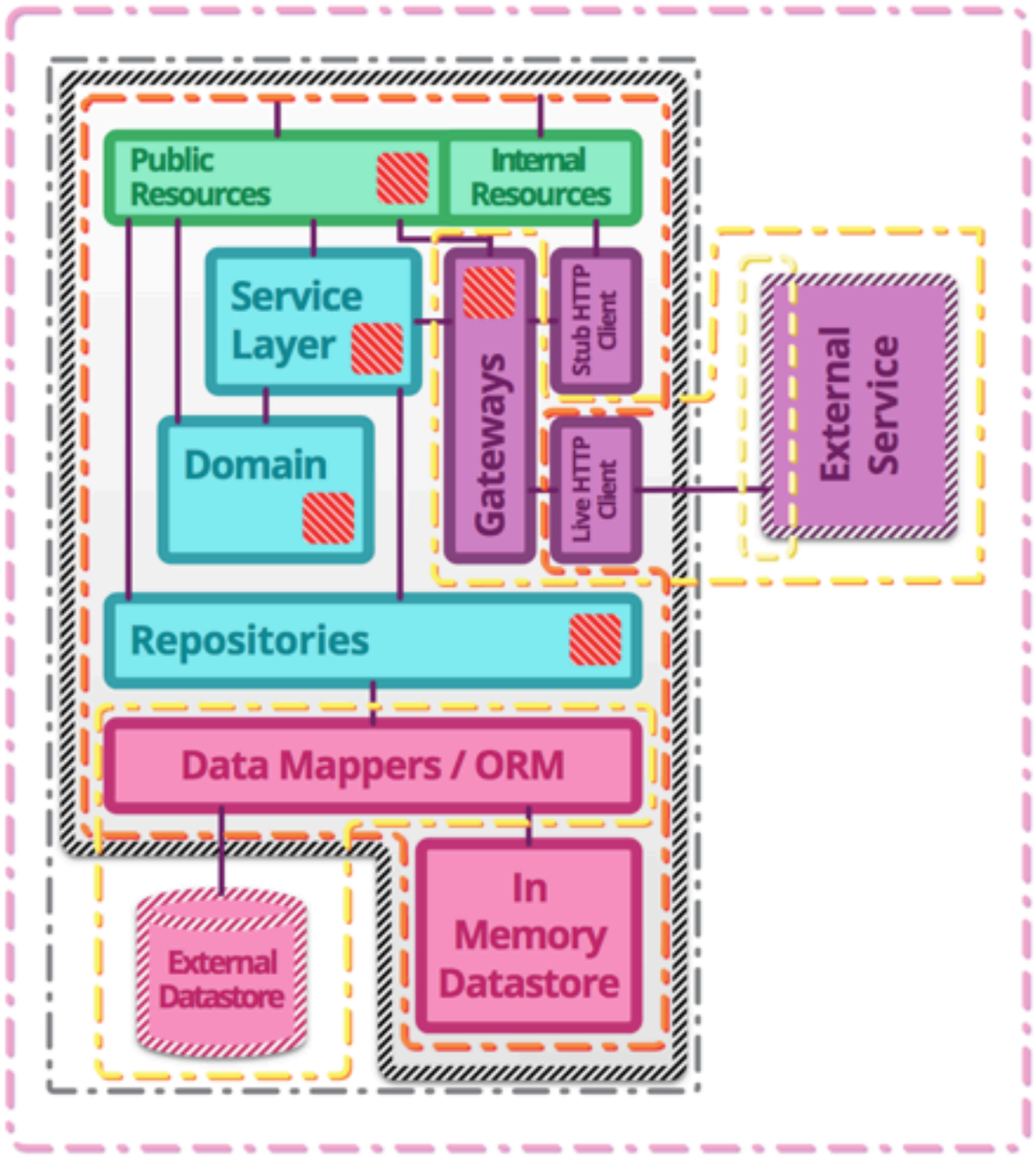
End-to-End testing



Summary

Unit tests : exercise the smallest pieces of testable software in the application to determine whether they behave as expected.

Integration tests : verify the communication paths and interactions between components to detect interface defects.



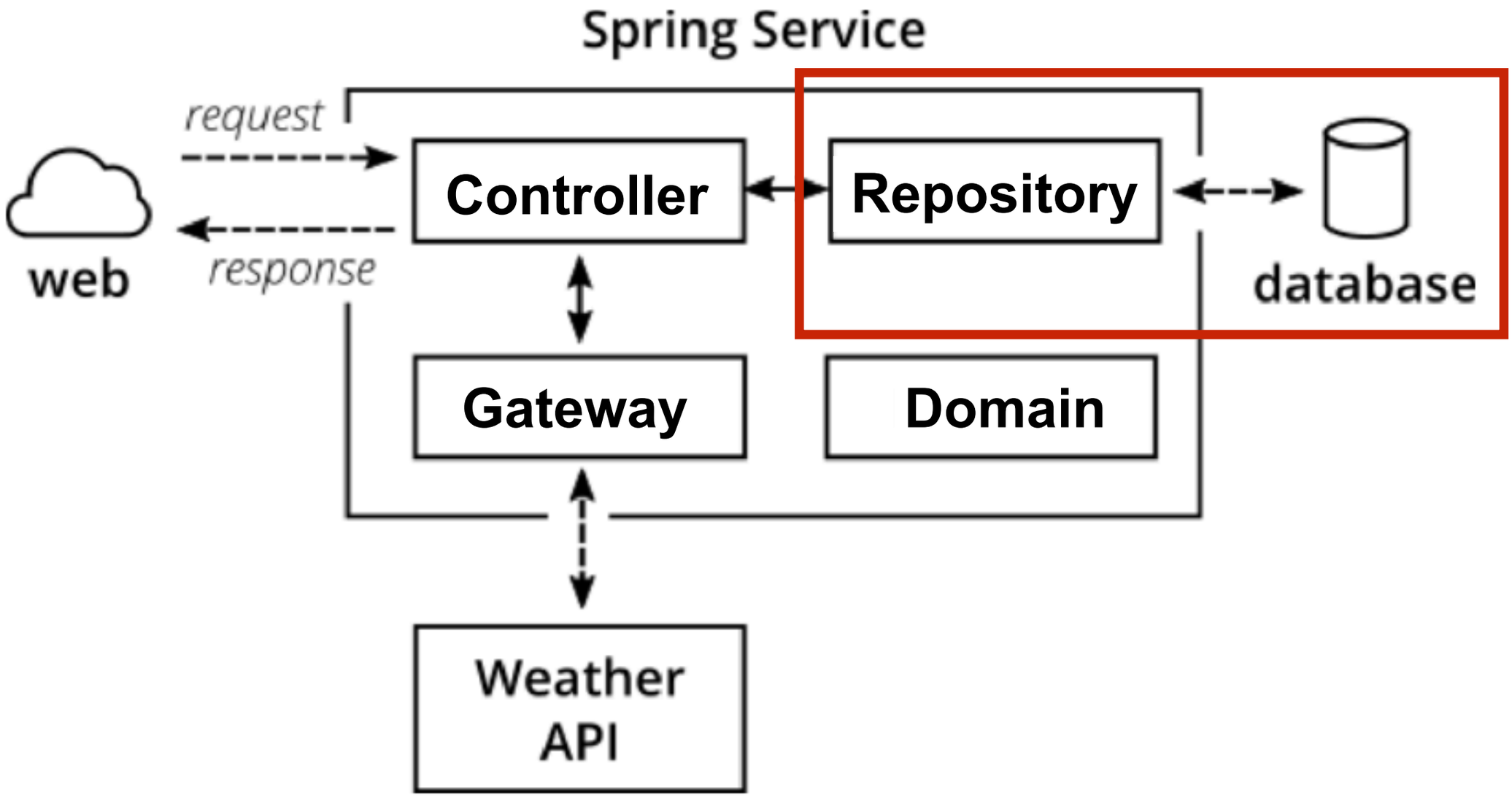
Component tests : limit the scope of the exercised software to a portion of the system under test, manipulating the system through internal code interfaces and using test doubles to isolate the code under test from other components.

Contract tests : verify interactions at the boundary of an external service asserting that it meets the contract expected by a consuming service.

End-to-end tests : verify that a system meets external requirements and achieves its goals, testing the entire system, from end to end.

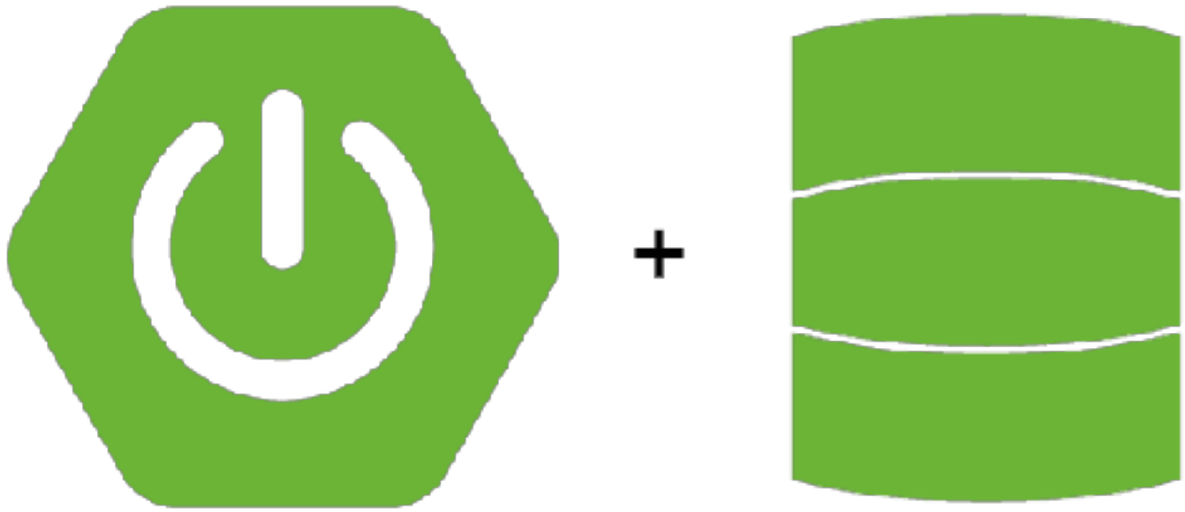
Let’s workshop

Working with repository



Working with repository

We're using Spring Data



Modify pom.xml

Add library of Spring Data

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

Modify pom.xml

Add library of Persistence/data store

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.1.1</version>
</dependency>
```

Add data store config

In src/main/resources

```
spring.datasource.url= jdbc:postgresql://127.0.0.1:15432/postgres
spring.datasource.username= user
spring.datasource.password= password
spring.datasource.platform= POSTGRESQL

spring.jpa.show-sql= true
spring.jpa.hibernate.ddl-auto= create-drop
spring.jpa.database-platform= org.hibernate.dialect.PostgreSQLDialect
```

Create repository interface

hello.repository.PersonRepository.java

```
public interface PersonRepository
        extends CrudRepository<Person, String> {

    Optional<Person> findByFirstName(String name);

}
```

Create Entity class

hello.repository.Person.java

```
@Entity
public class Person {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String firstName;
    private String lastName;

    public Person() {
    }

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
```

Create new controller

hello.repository.HelloControllerWithRepository.java

```java
public class HelloControllerWithRepository {

    private final PersonRepository personRepository;

    @Autowired
    public HelloControllerWithRepository(PersonRepository personRepository) {
        this.personRepository = personRepository;
    }

    @GetMapping("/hello/data/{name}")
    public Hello sayHi(@PathVariable String name) {
        Optional<Person> foundPerson = personRepository.findByName(name);
        String result = foundPerson
                .map(person -> String.format("Hello %s", person.getFirstName()))
                .orElse( other: "Data not found");
        return new Hello(result);
    }

}
```

Run test and package

$mvn clean package

```
Cobertura Report generation was successful.
Cobertura 2.1.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file
Cobertura: Loaded information on 3 classes.
time: 125ms

Cobertura Report generation was successful.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

Run your application

$java -jar target/hello.jar

```
org.postgresql.util.PSQLException: Connection to 127.0.0.1:15432 refused. Check that th
 the postmaster is accepting TCP/IP connections.
        at org.postgresql.core.v3.ConnectionFactoryImpl.openConnectionImpl(ConnectionFa
jar!/:42.1.1]
        at org.postgresql.core.ConnectionFactory.openConnection(ConnectionFactory.java:
        at org.postgresql.jdbc.PgConnection.<init>(PgConnection.java:194) ~[postgresql-
at org.postgresql.Driver.makeConnection(Driver.java:450) ~[postgresql-42.1.1.ja
at org.postgresql.Driver.connect(Driver.java:252) ~[postgresql-42.1.1.jar!/:42.
at com.zaxxer.hikari.util.DriverDataSource.getConnection(DriverDataSource.java:
at com.zaxxer.hikari.util.DriverDataSource.getConnection(DriverDataSource.java:
at com.zaxxer.hikari.pool.PoolBase.newConnection(PoolBase.java:365) [HikariCP-2
at com.zaxxer.hikari.pool.PoolBase.newPoolEntry(PoolBase.java:194) [HikariCP-2.
at com.zaxxer.hikari.pool.HikariPool.createPoolEntry(HikariPool.java:460) [Hika
at com.zaxxer.hikari.pool.HikariPool.checkFailFast(HikariPool.java:534) [Hikari
at com.zaxxer.hikari.pool.HikariPool.<init>(HikariPool.java:115) [HikariCP-2.7.
at com.zaxxer.hikari.HikariDataSource.getConnection(HikariDataSource.java:112)
```

Coverage report

Packages

All
hello
hello.controller
hello.domain
hello.repository

All Packages

Classes

Hello *(65%)*
HelloApplication *(33%)*
HelloController *(100%)*
HelloControllerWithRepository *(0%)*
Person *(0%)*
PersonRepository *(N/A)*

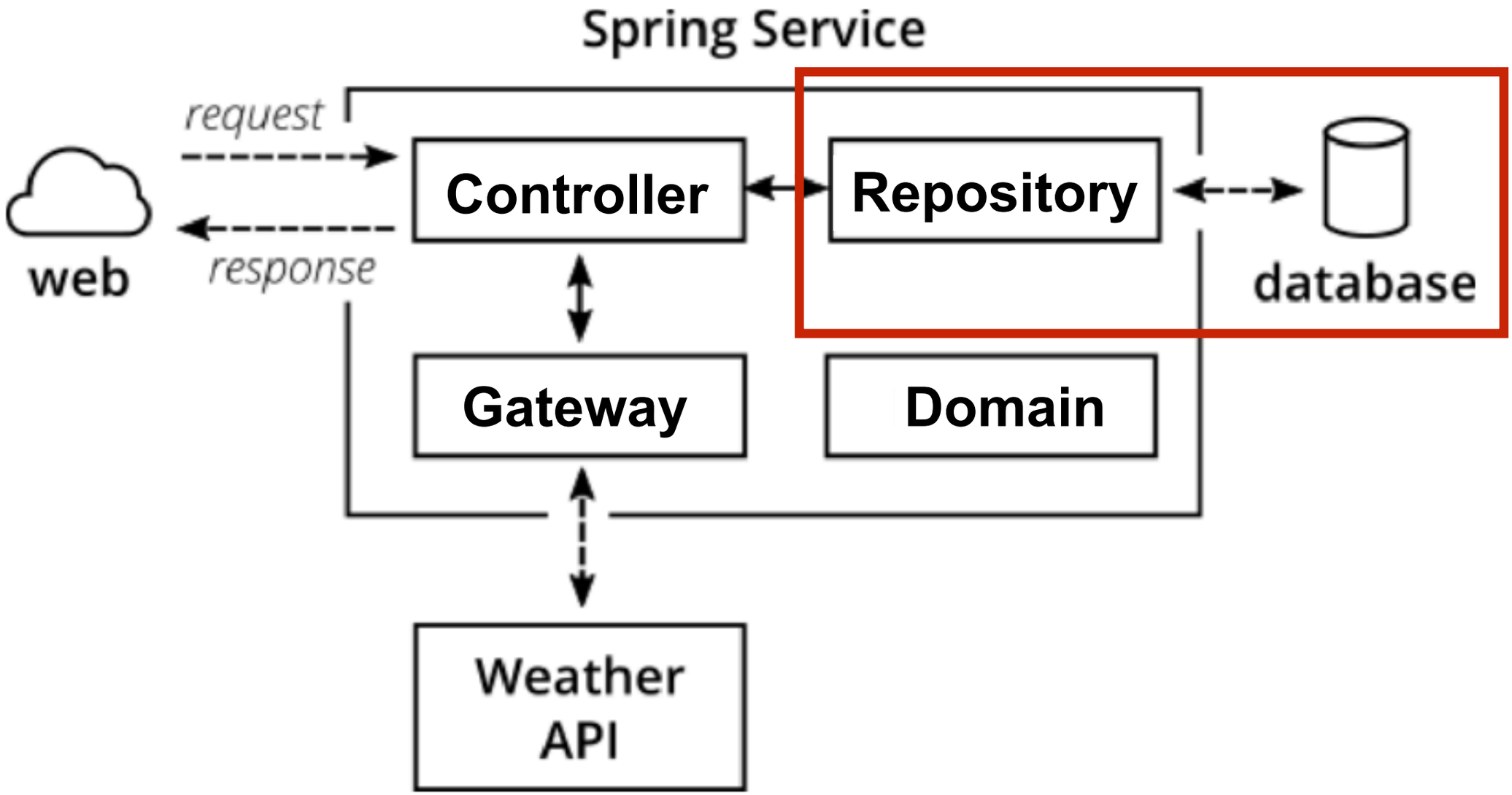
Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	6	25%	7/28	N/A	N/A	1
hello	1	33%	1/3	N/A	N/A	1
hello.controller	2	20%	2/10	N/A	N/A	1
hello.domain	1	66%	4/6	N/A	N/A	1
hello.repository	2	0%	0/9	N/A	N/A	1

Report generated by Cobertura 2.1.1 on 3/6/18 8:52 AM.

How to test ?

How to test with Repository ?



Spring boot provide DataJpaTest

should be add H2 library to pom.xml

```
<dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>test</scope>
</dependency>
```

Repository Testing (1)

```java
@RunWith(SpringRunner.class)
@DataJpaTest
public class PersonRepositoryTest {

    @Autowired
    private PersonRepository personRepository;

    @After
    public void clearData() {
        personRepository.deleteAll();
    }
```

Repository Testing (2)

Add a test case

```
@Test
public void shouldSaveAndGetData() throws Exception {
    //Arrange
    Person somkiat = new Person("somkiat", "pui");
    personRepository.save(somkiat);

    Optional<Person> shouldSomkiat
            = personRepository.findByFirstName("somkiat");

    assertEquals( expected: "somkiat",
            shouldSomkiat.get().getFirstName());

}
```

Run test and package

$mvn clean package

```
Cobertura Report generation was successful.
Cobertura 2.1.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file
Cobertura: Loaded information on 3 classes.
time: 125ms

Cobertura Report generation was successful.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

Coverage report

Packages

All
hello
hello.controller
hello.domain
hello.repository

All Packages

Classes

Hello *(66%)*
HelloApplication *(33%)*
HelloController *(100%)*
HelloControllerWithRepository *(0%*
Person *(88%)*
PersonRepository *(N/A)*

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	6	53%	15/28	N/A	N/A	1
hello	1	33%	1/3	N/A	N/A	1
hello.controller	2	20%	2/10	N/A	N/A	1
hello.domain	1	66%	4/6	N/A	N/A	1
hello.repository	2	88%	8/9	N/A	N/A	1

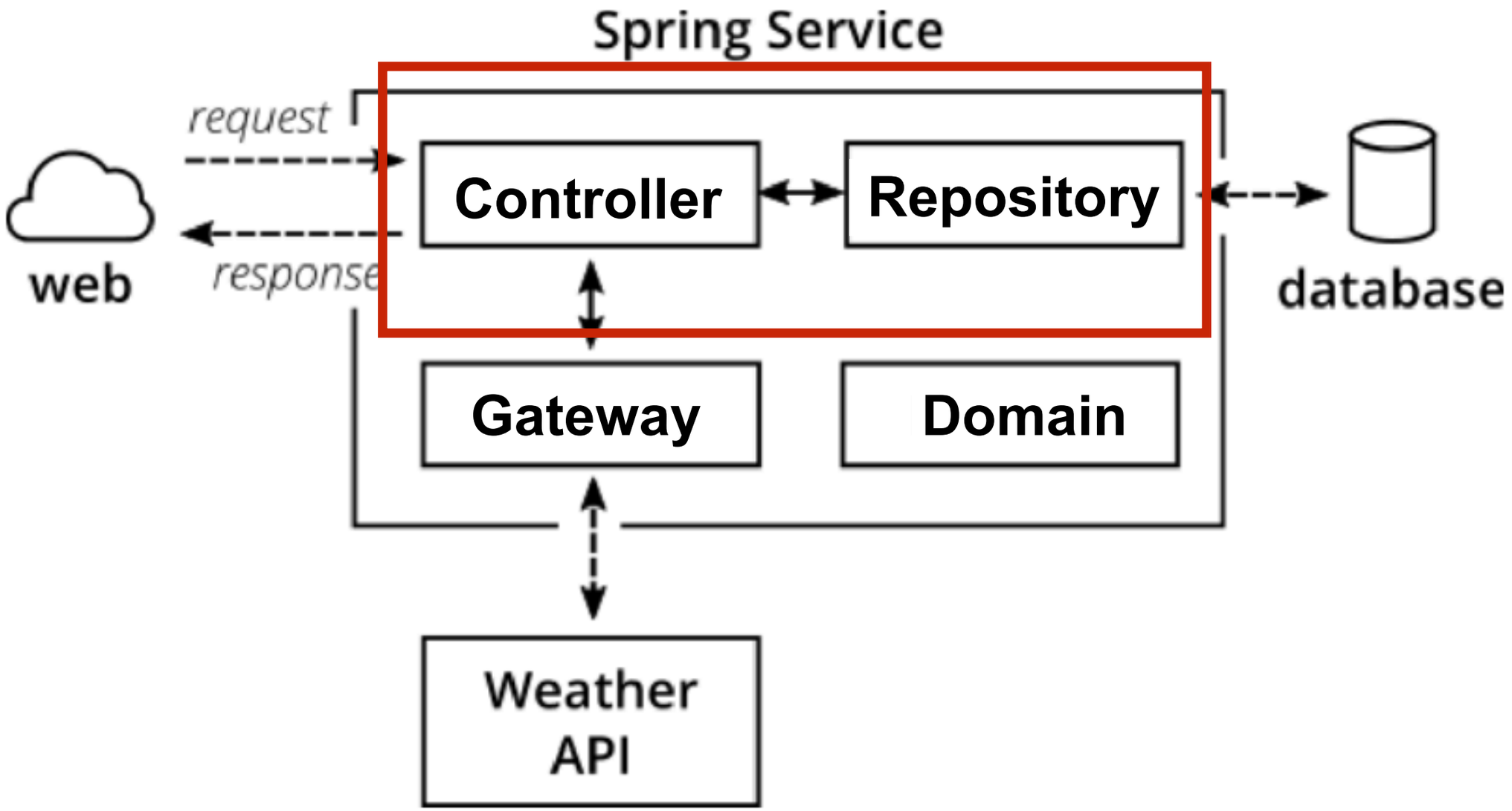
Report generated by Cobertura 2.1.1 on 3/6/18 9:32 AM.

Controller Testing

Unit testing ?

Spring Unit testing with MockMvc ?

Controller Testing with Unit test



Unit test

Use Test Double

In java, use Mockito library

http://site.mockito.org/

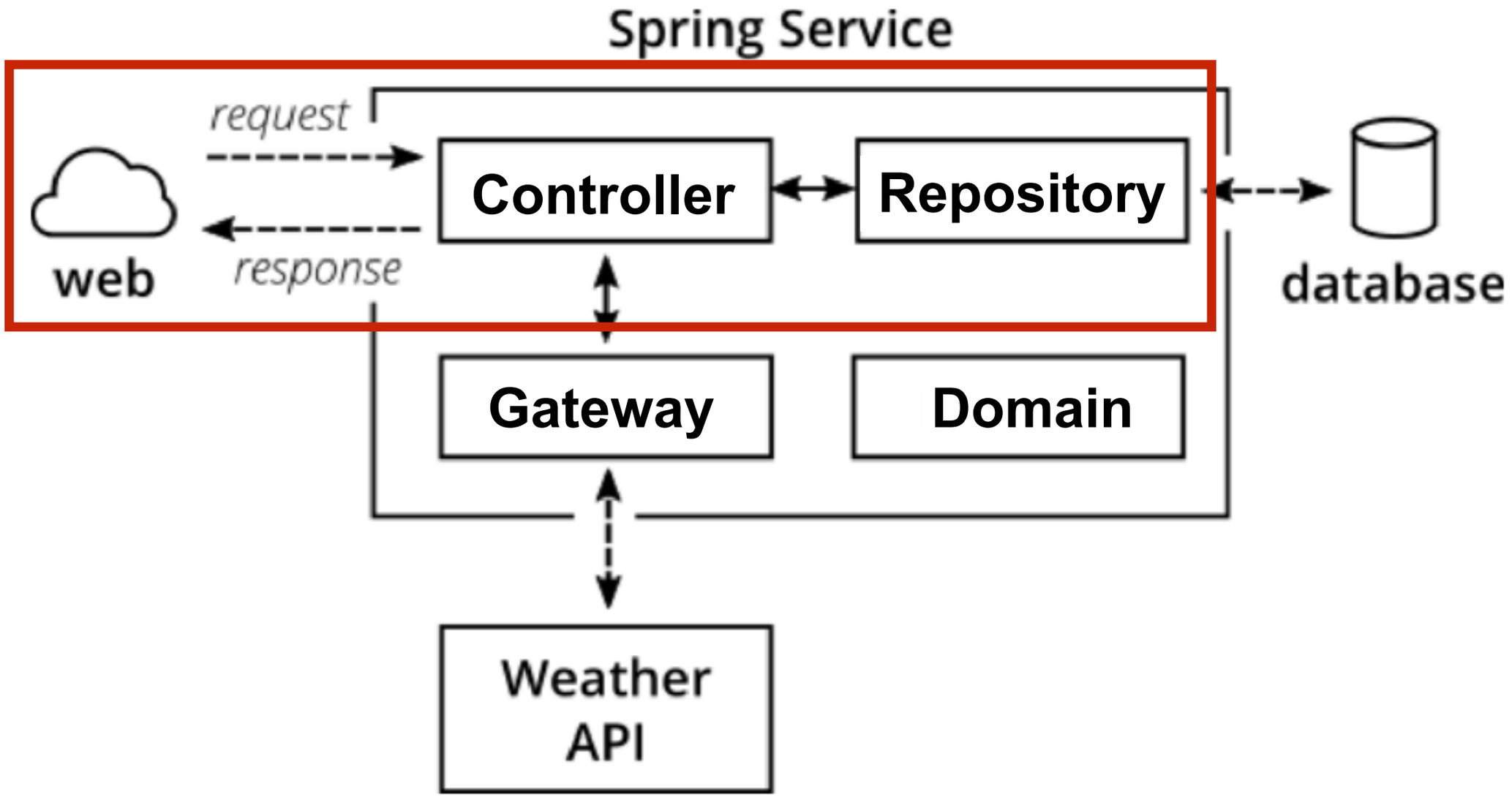
Unit test with Mockito (1)

```
public class HelloControllerWithRepositoryTest {

    private HelloControllerWithRepository controllerWithRepository;

    @Mock
    private PersonRepository personRepository;

    @Before
    public void init() {
        initMocks( testClass: this);
        controllerWithRepository
                = new HelloControllerWithRepository(personRepository);
    }
```

Unit test with Mockito (2)

```java
@Test
public void shouldReturnHelloSomkiat() {
    //Arrange
    Person somkiat = new Person("somkiat", "pui");
    given(personRepository.findByFirstName("somkiat"))
            .willReturn(Optional.of(somkiat));

    // Action
    Hello hello = controllerWithRepository.sayHi( name: "somkiat");

    // Assert
    assertEquals( expected: "Hello somkiat", hello.getMessage());
}
```

Controller Testing with MockMvc



Test with MockMvc (1)

```java
@RunWith(SpringRunner.class)
@WebMvcTest(controllers = HelloControllerWithRepository.class)
public class HelloControllerWithRepositoryMockMvcTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private PersonRepository personRepository;
```

Test with MockMvc (2)

```
@Test
public void shouldReturnHelloSomkiat() throws Exception {
    //Arrange
    Person somkiat = new Person("somkiat", "pui");
    given(personRepository.findByFirstName("somkiat"))
            .willReturn(Optional.of(somkiat));

    // Action and Assert
    mockMvc.perform(get( urlTemplate: "/hello/data/somkiat"))
            .andExpect(
                    jsonPath( expression: "$.message")
                            .value( expectedValue: "Hello somkiat"))
            .andExpect(status().is2xxSuccessful());
}
```

Coverage report

Packages

All
toystore
toystore.controller
toystore.domain
toystore.repository

All Packages

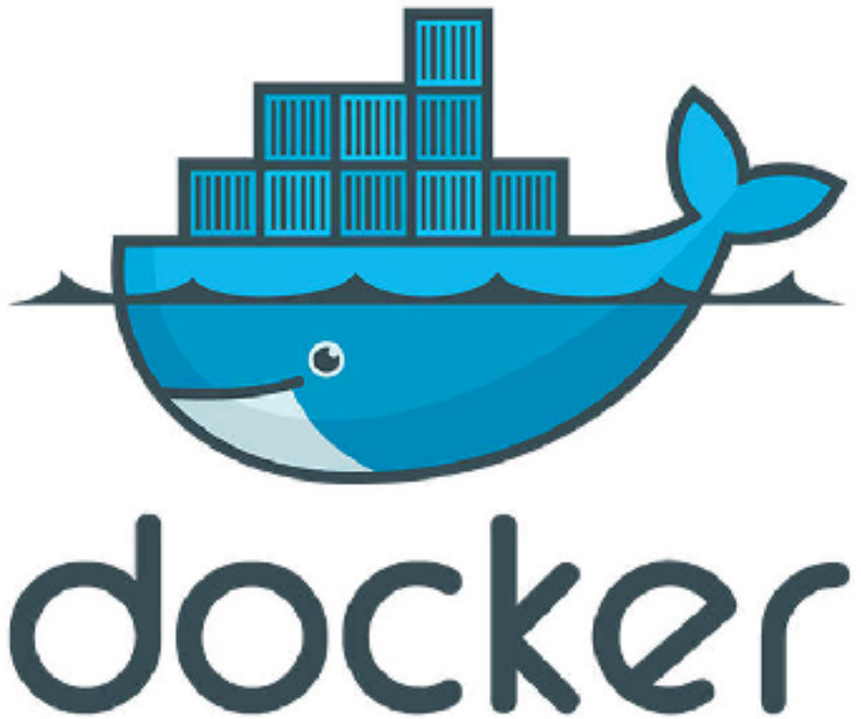
Classes

Hello *(100%)*
HelloController *(83%)*
HelloWithRepositoryController *(100*
Person *(100%)*
PersonRepository *(N/A)*
ToyStoreApplication *(33%)*

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	6	90%	29/32	50%	1/2	1.1
toystore	1	33%	1/3	N/A	N/A	1
toystore.controller	2	93%	15/16	50%	1/2	2
toystore.domain	1	100%	4/4	N/A	N/A	1
toystore.repository	2	100%	9/9	N/A	N/A	1

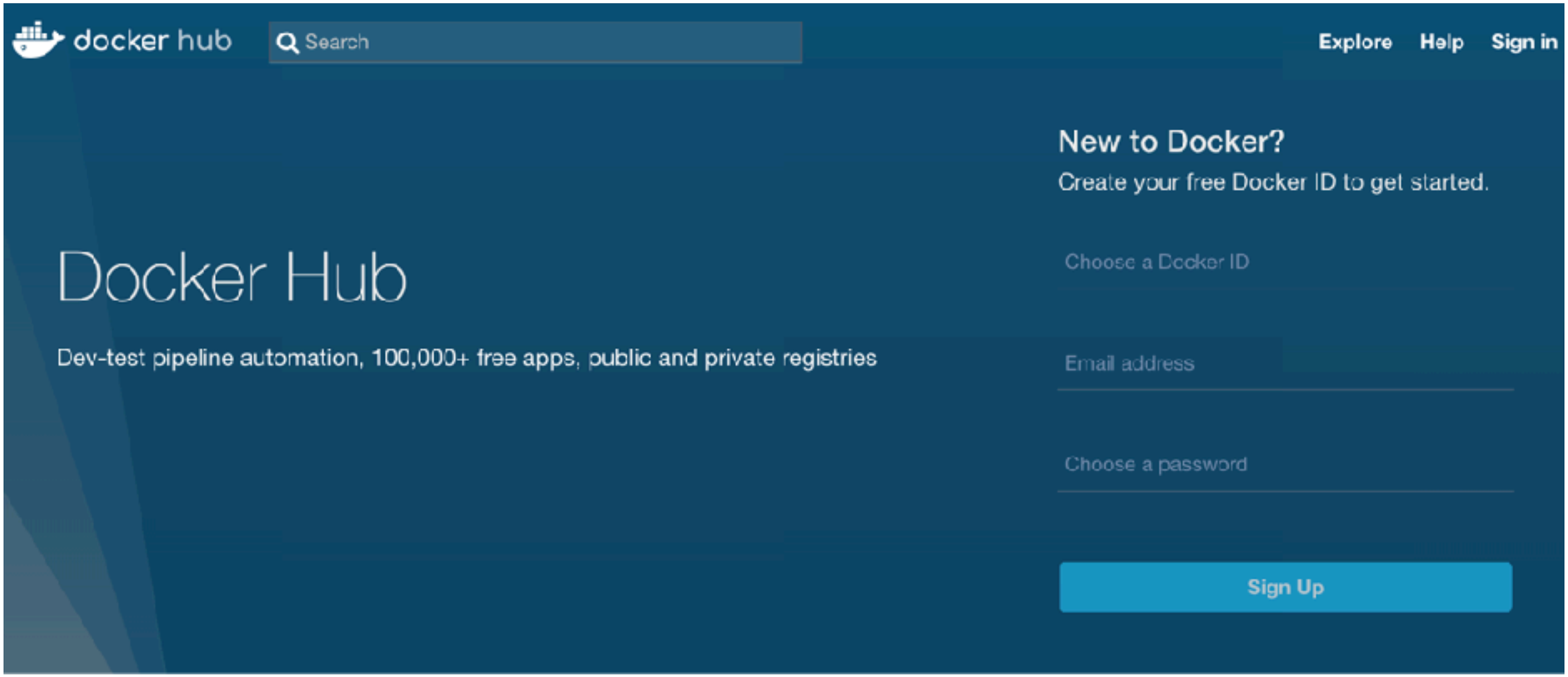
Report generated by Cobertura 2.1.1 on 3/6/18 5:20 PM.



Working with container

Choose your image

from hub.docker.com



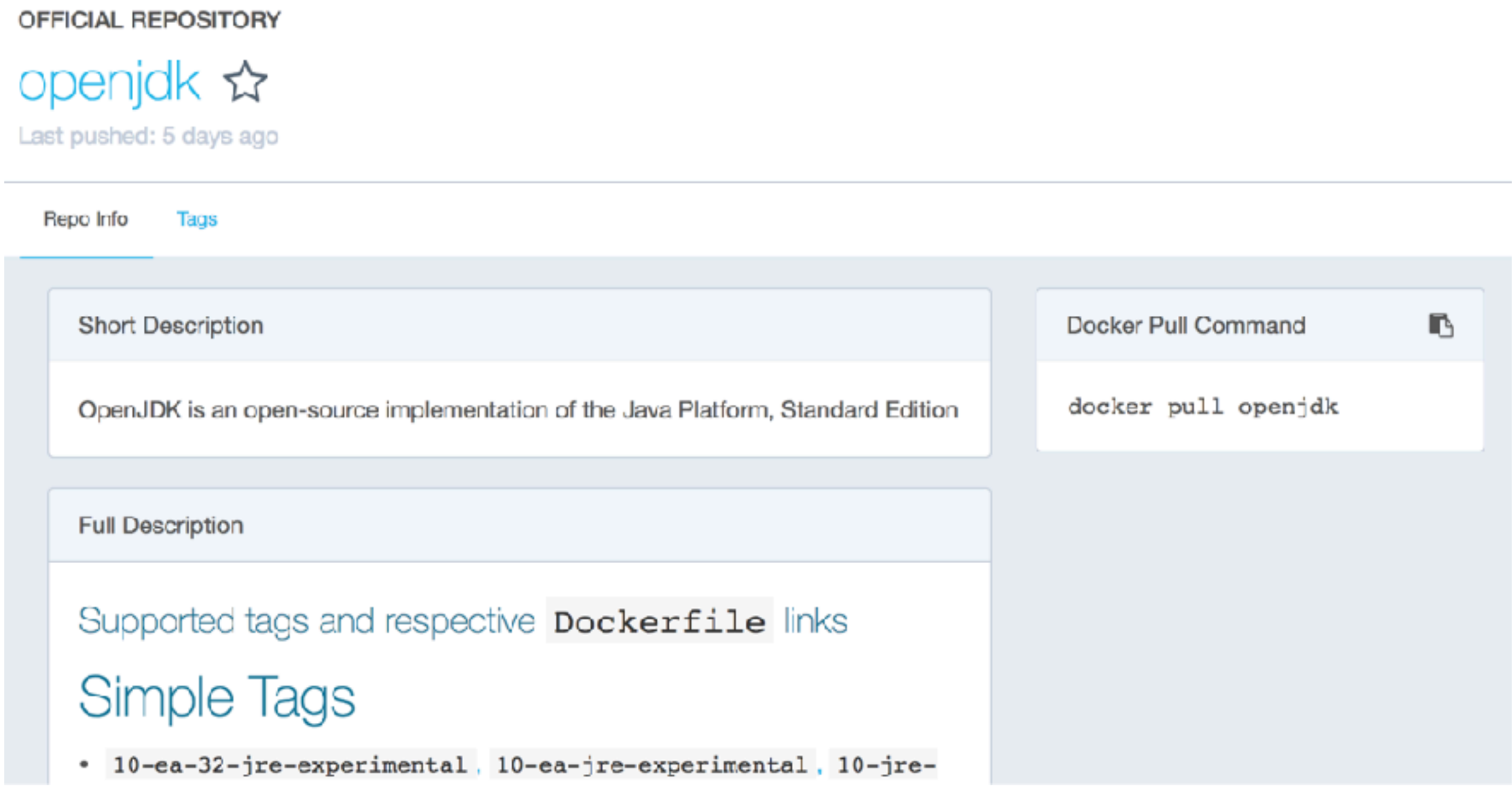
Images in workshop

Java => **OpenJDK**

Apache Maven => **maven**

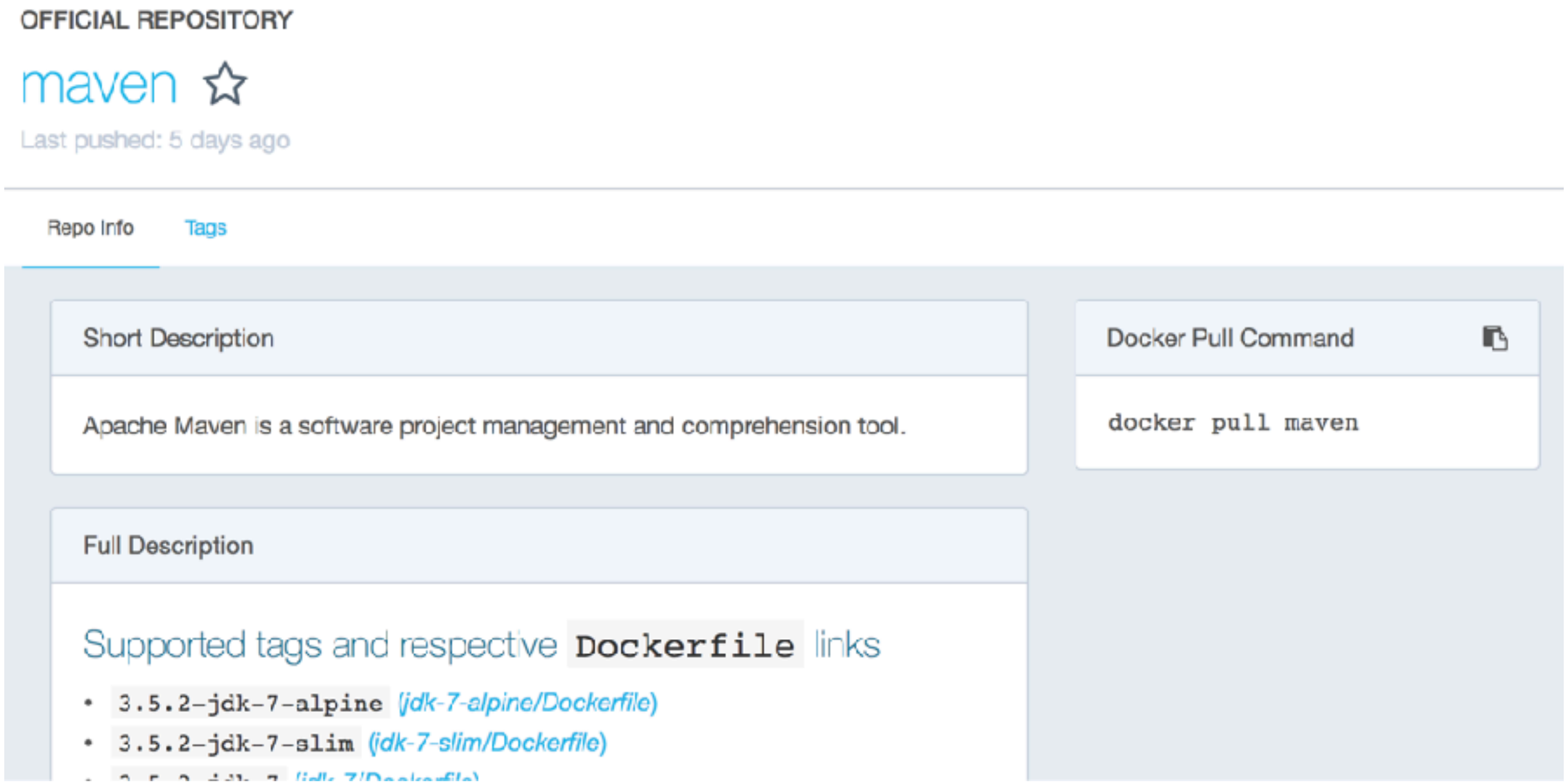
PostgreSQL => **progrès**

Java => OpenJDK



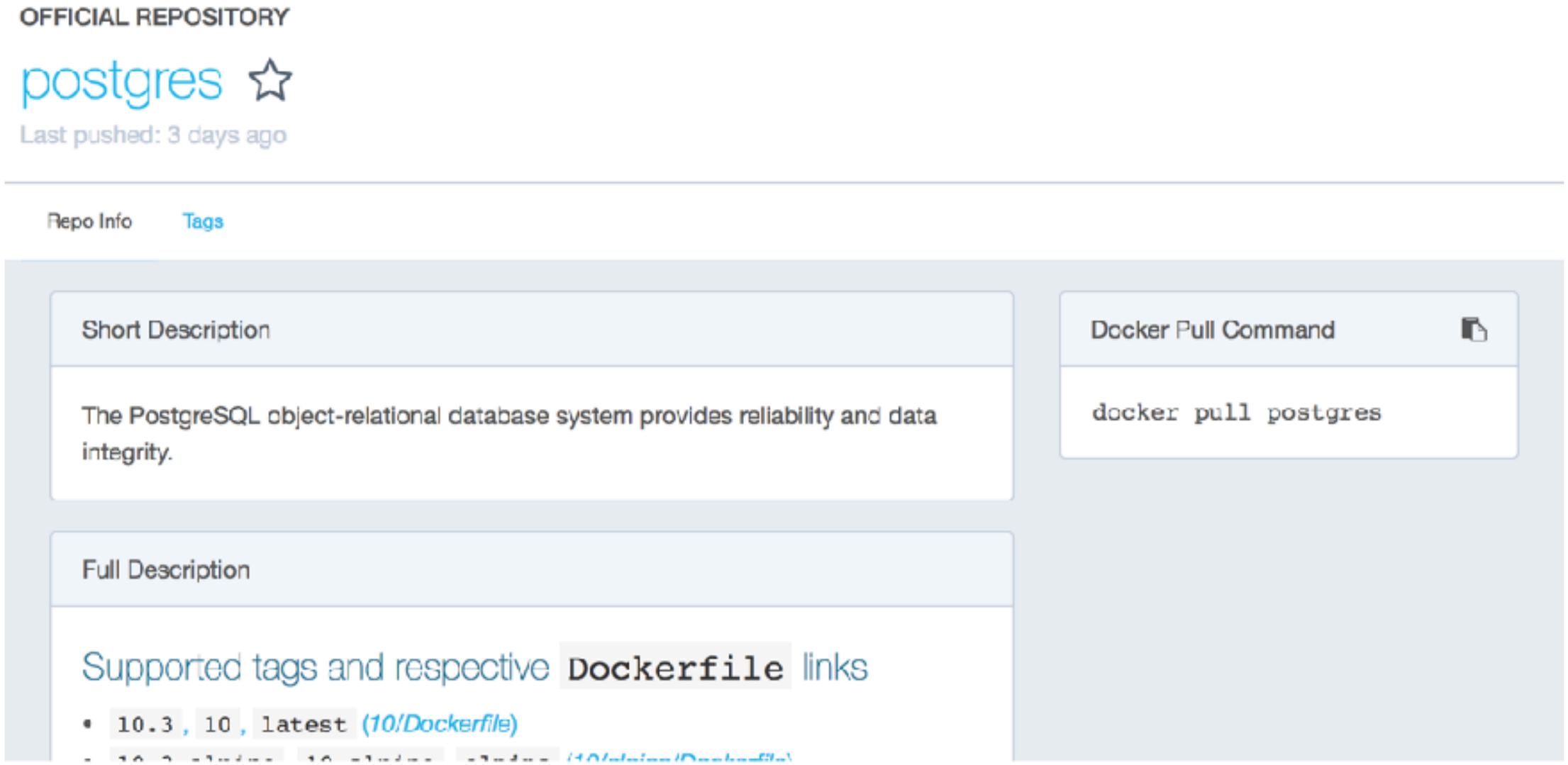
https://hub.docker.com/_/openjdk/

Apache Maven => maven



https://hub.docker.com/_/maven/

PostgreSQL => progrès



https://hub.docker.com/_/postgres/

Pull images from Docker Hub

$docker image pull openjdk:<tag>
$docker image pull maven:<tag>
$docker image pull postgres:<tag>

Create container to run Spring Boot

Create container to run Spring Boot

$docker container run **-d**
-v $(pwd)/target/toystore.jar:/xxx/toystore.jar
-p 8080:8080
--name web
openjdk:8-jre java -jar /xxx/toystore.jar

Create new Docker image (1)

from Dockerfile

```
FROM openjdk:8-jre
COPY ./target/toystore.jar /xxx/toystore.jar
CMD java -jar /xxx/toystore.jar
```

Create new Docker image (2)

Build image from Dockerfile

$docker image build -t toystore:0.1 .

```
Sending build context to Docker daemon  33.47MB
Step 1/3 : FROM openjdk:8-jre
 ---> e956268fd4ed
Step 2/3 : COPY ./target/toystore.jar /xxx/toystore.jar
 ---> 3dd837b158eb
Step 3/3 : CMD java -jar /xxx/toystore.jar
 ---> Running in 06994b290e74
Removing intermediate container 06994b290e74
 ---> 0bc2054f4ba8
Successfully built 0bc2054f4ba8
Successfully tagged toystore:0.1
```

Create container to run Spring Boot

$docker container run -d -p 8080:8080 **toystore:0.1**

$docker container run -d -p 8081:8080 **toystore:0.1**

$docker container run -d -p 8082:8080 **toystore:0.1**

Create container
to run Build Maven Project

Create container to build

```
$docker container run --rm
  -v $(pwd):/xxx
  -w /xxx
  maven:3.5.2-alpine mvn clean package
```

Run your application

$java -jar target/hello.jar

```
org.postgresql.util.PSQLException: Connection to 127.0.0.1:15432 refused. Check that th
 the postmaster is accepting TCP/IP connections.
        at org.postgresql.core.v3.ConnectionFactoryImpl.openConnectionImpl(ConnectionFa
jar!/:42.1.1]
        at org.postgresql.core.ConnectionFactory.openConnection(ConnectionFactory.java:
        at org.postgresql.jdbc.PgConnection.<init>(PgConnection.java:194) ~[postgresql-
at org.postgresql.Driver.makeConnection(Driver.java:450) ~[postgresql-42.1.1.ja
at org.postgresql.Driver.connect(Driver.java:252) ~[postgresql-42.1.1.jar!/:42.
at com.zaxxer.hikari.util.DriverDataSource.getConnection(DriverDataSource.java:
at com.zaxxer.hikari.util.DriverDataSource.getConnection(DriverDataSource.java:
at com.zaxxer.hikari.pool.PoolBase.newConnection(PoolBase.java:365) [HikariCP-2
at com.zaxxer.hikari.pool.PoolBase.newPoolEntry(PoolBase.java:194) [HikariCP-2.
at com.zaxxer.hikari.pool.HikariPool.createPoolEntry(HikariPool.java:460) [Hika
at com.zaxxer.hikari.pool.HikariPool.checkFailFast(HikariPool.java:534) [Hikari
at com.zaxxer.hikari.pool.HikariPool.<init>(HikariPool.java:115) [HikariCP-2.7.
at com.zaxxer.hikari.HikariDataSource.getConnection(HikariDataSource.java:112)
```

Create container of PostgreSQL

Create container of PostgreSQL

```
$docker container run
  -p 15432:5432
  -e POSTGRES_USER=user
  -e POSTGRES_PASSWORD=password
  postgres:10.3-alpine
```

Run your application again !!

$java -jar target/hello.jar

```
Hibernate: create sequence hibernate_sequence start 1 increment 1
Hibernate: create table person (id int4 not null, first_name varchar(255), last_name varchar(255), primary key (id))
2018-03-06 19:45:41.115  INFO 56495 --- [           main] o.h.t.schema.internal.SchemaCreatorImpl  : HHH000476: Executing import script 'org.hibernate.tool.schema.internal.exec.Scri
ptSourceInputNonExistentImpl@7e990ed7'
2018-03-06 19:45:41.121  INFO 56495 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2018-03-06 19:45:42.592  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.web.servlet.context.Anno
tationConfigServletWebServerApplicationContext@7cef4e59: startup date [Tue Mar 06 19:45:34 ICT 2018]; root of context hierarchy
2018-03-06 19:45:42.746  WARN 56495 --- [           main] aWebConfiguration$JpaWebMvcConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may b
e performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2018-03-06 19:45:42.857  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/hello/{name}],methods=[GET]}" onto public toystore.domain.Hello toyst
ore.controller.HelloController.sayHi(java.lang.String)
2018-03-06 19:45:42.881  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/hello/data/{name}],methods=[GET]}" onto public toystore.domain.Hello
toystore.controller.HelloWithRepositoryController.sayHi(java.lang.String)
2018-03-06 19:45:42.889  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.uti
l.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-03-06 19:45:42.893  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.ser
vlet.ModelAndView org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletRespons
e)
2018-03-06 19:45:43.046  INFO 56495 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.we
b.servlet.resource.ResourceHttpRequestHandler]
2018-03-06 19:45:43.046  INFO 56495 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class org.springframework.web.servle
t.resource.ResourceHttpRequestHandler]
2018-03-06 19:45:43.168  INFO 56495 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframewor
k.web.servlet.resource.ResourceHttpRequestHandler]
2018-03-06 19:45:43.783  INFO 56495 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2018-03-06 19:45:43.785  INFO 56495 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Bean with name 'dataSource' has been autodetected for JMX exposure
2018-03-06 19:45:43.793  INFO 56495 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Located MBean 'dataSource': registering with JMX server as MBean [com.zaxxer.hik
ari:name=dataSource,type=HikariDataSource]
2018-03-06 19:45:43.873  INFO 56495 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2018-03-06 19:45:43.877  INFO 56495 --- [           main] toystore.ToyStoreApplication             : Started ToyStoreApplication in 10.18 seconds (JVM running for 10.912)
2018-03-06 19:45:47.875  INFO 56495 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring FrameworkServlet 'dispatcherServlet'
2018-03-06 19:45:47.876  INFO 56495 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet': initialization started
2018-03-06 19:45:47.917  INFO 56495 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet': initialization completed in 41 ms
```

Monitoring and Metric

Metric in Spring Boot

Spring Boot **Actuator** for Spring Boot 1.x

MicroMeter for Spring Boot 2.0

Spring Boot Actuator (1)

Add library to pom.xml

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

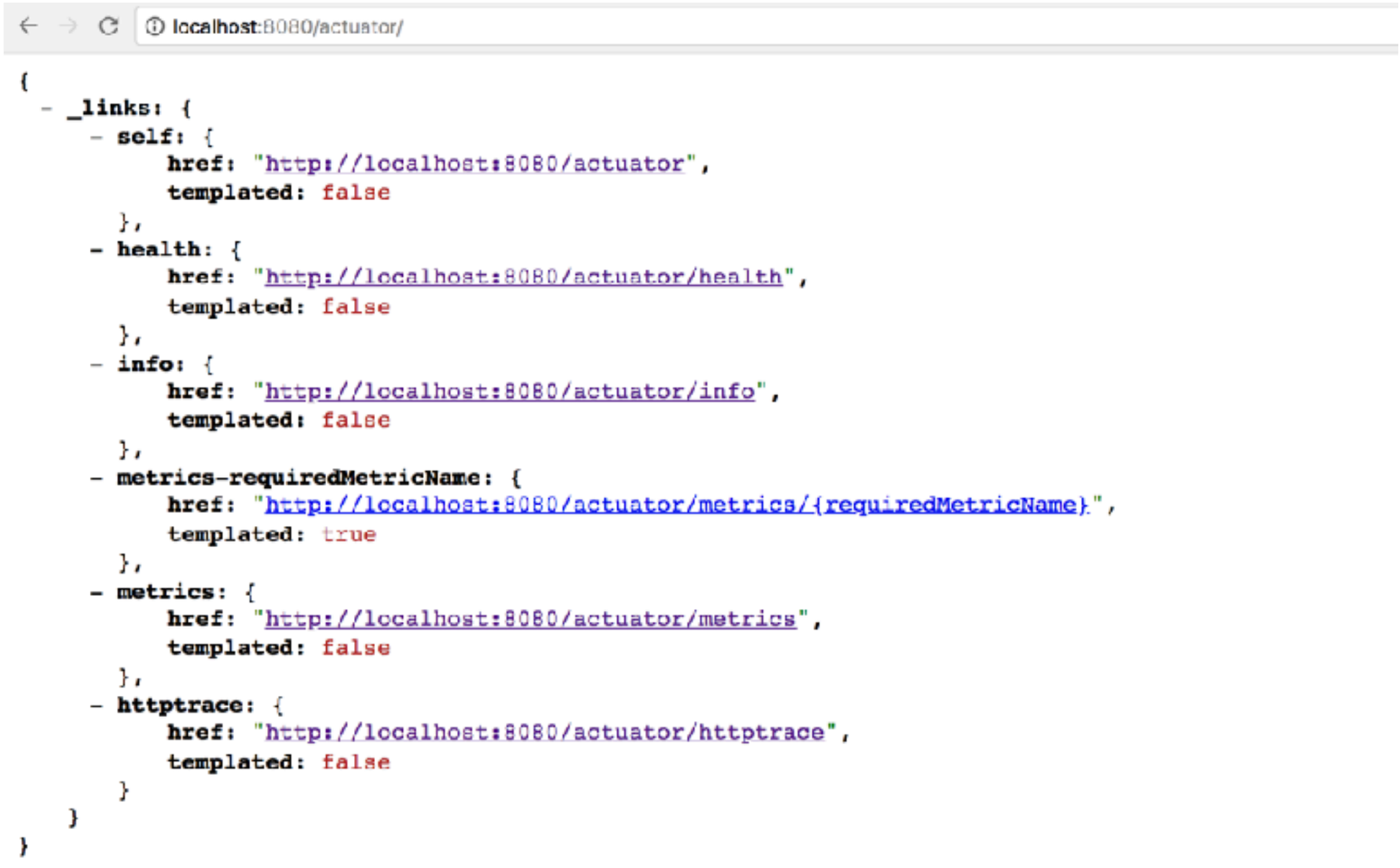
Spring Boot Actuator (2)

Enabled endpoint in application.properties

```
info.app.name=Toy Store
info.app.description=This is my first spring boot application
info.app.version=1.0.0

management.endpoints.web.exposure.include=health,info,metrics,httptrace
```

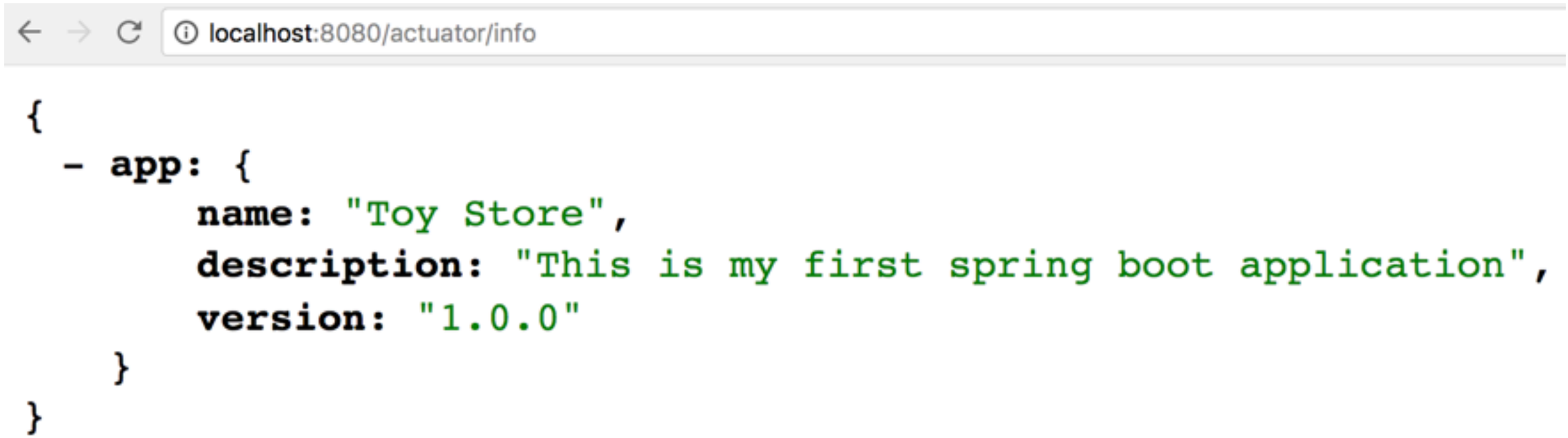
Spring Boot Actuator (3)

List of endpoints = /actuator/



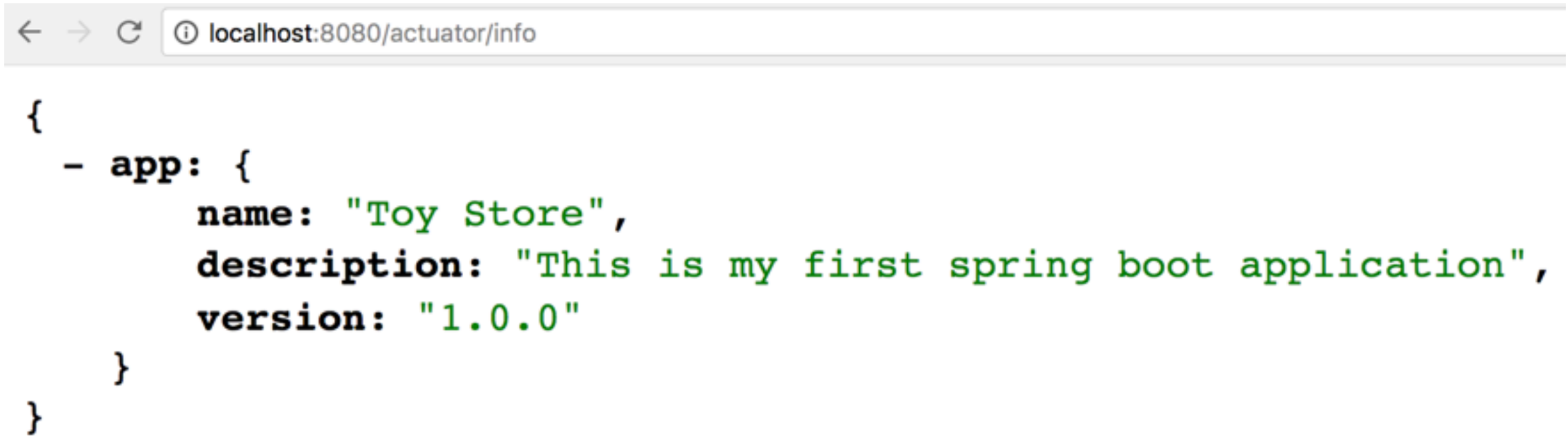
Spring Boot Actuator (4)

Info endpoint = /actuator/info



Spring Boot Actuator (5)

Info endpoint = /actuator/info



Spring Boot Actuator (6)

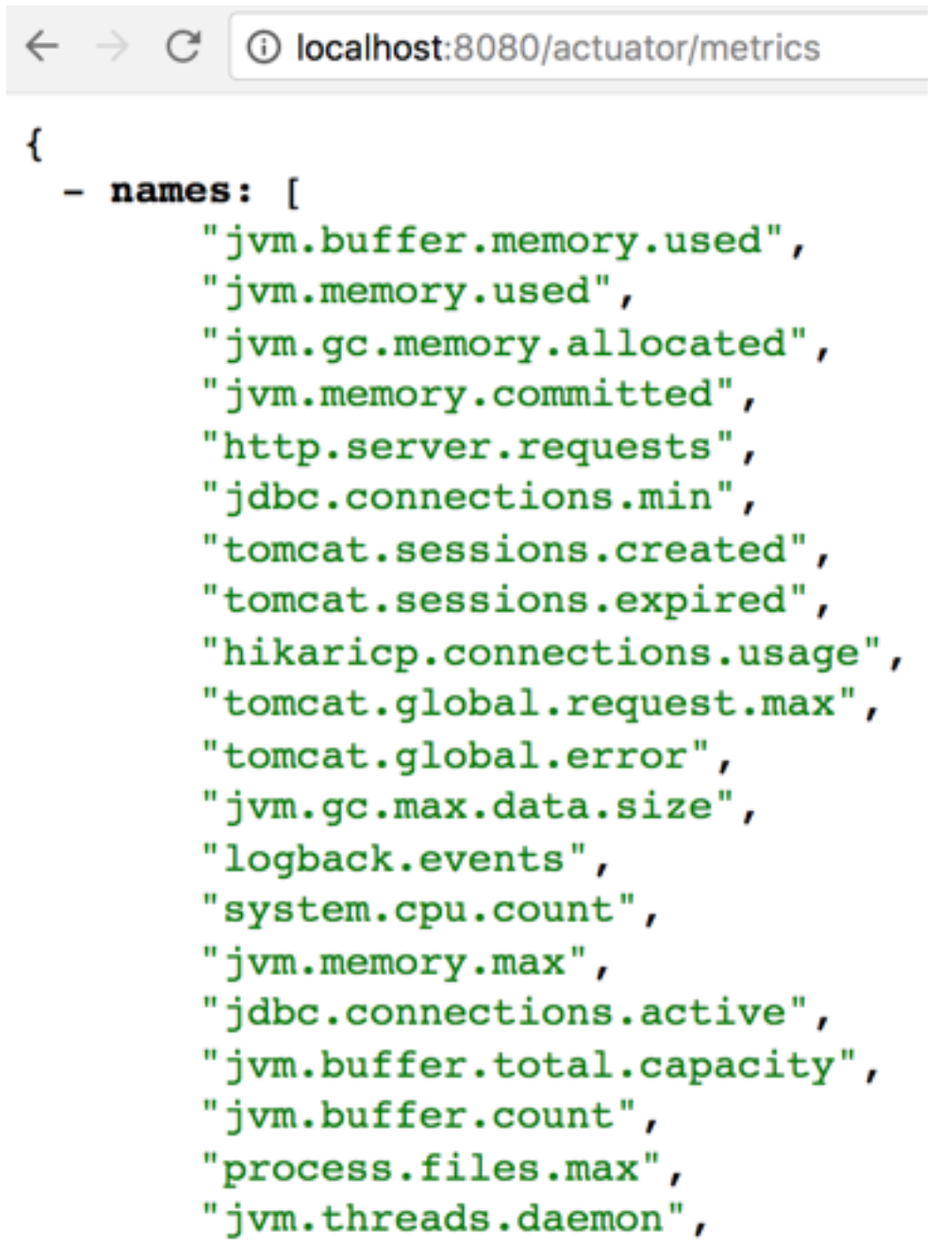
Info endpoint = /actuator/httptrace

localhost:8080/actuator/httptrace


```
{
  - traces: [
      - {
            timestamp: "2018-03-06T13:33:02.800Z",
            principal: null,
            session: null,
          - request: {
                method: "GET",
                uri: "http://localhost:8080/prometheus",
              - headers: {
                  - host: [
                        "localhost:8080"
                    ],
                  - user-agent: [
                        "Prometheus/2.0.0"
                    ],
                  - accept: [
                        "text/plain;version=0.0.4;q=1,*/*;q=0.1"
                    ],
                  - accept-encoding: [
                        "gzip"
                    ],
                  - x-prometheus-scrape-timeout-seconds: [
                        "5.000000"
                    ]
                },
                remoteAddress: null
            },
```


Spring Boot Actuator (7)

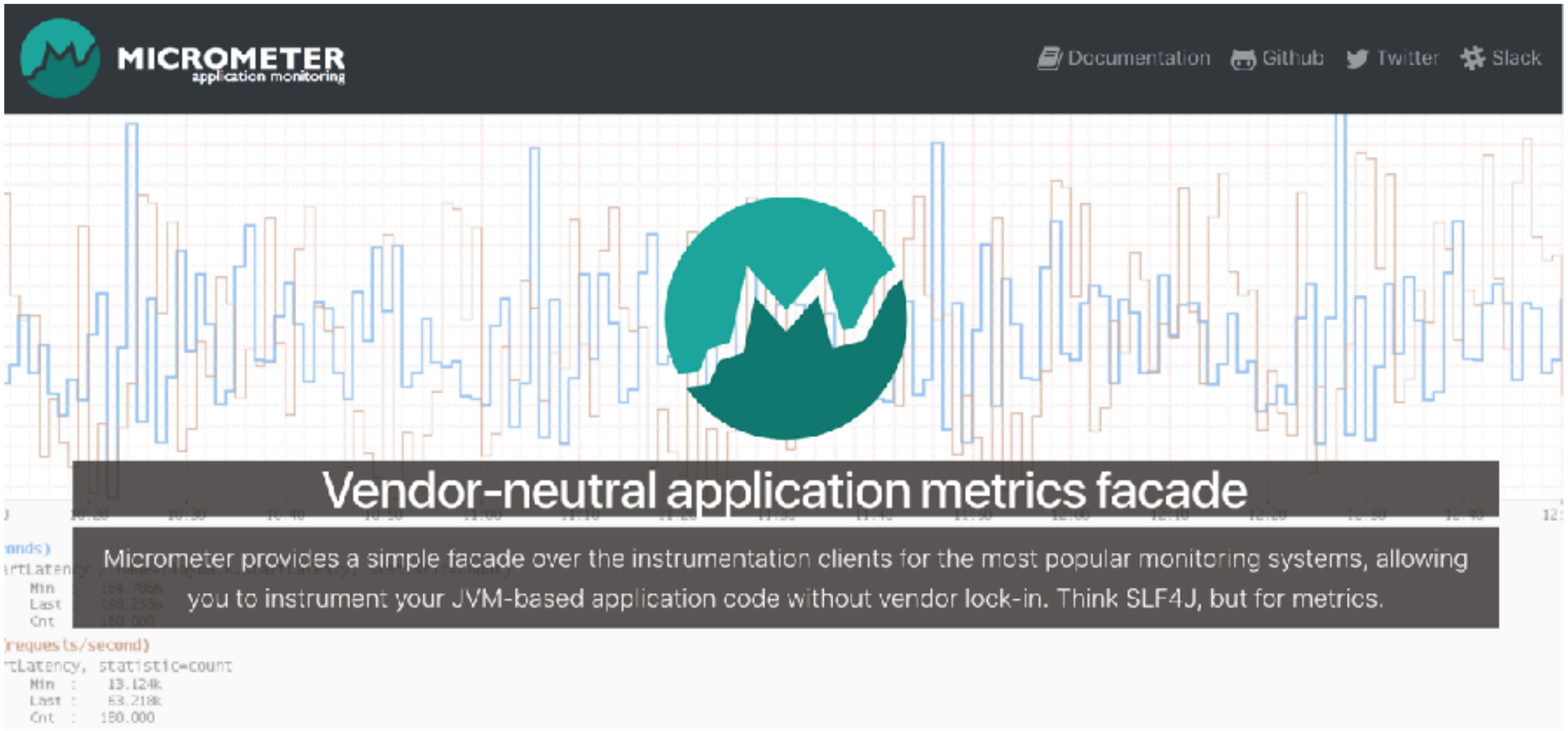
List of metrics endpoint = /actuator/metrics



Spring Boot Actuator (8)

/actuator/metrics/http.server.requests

Spring Boot 2.0 with MicroMeter

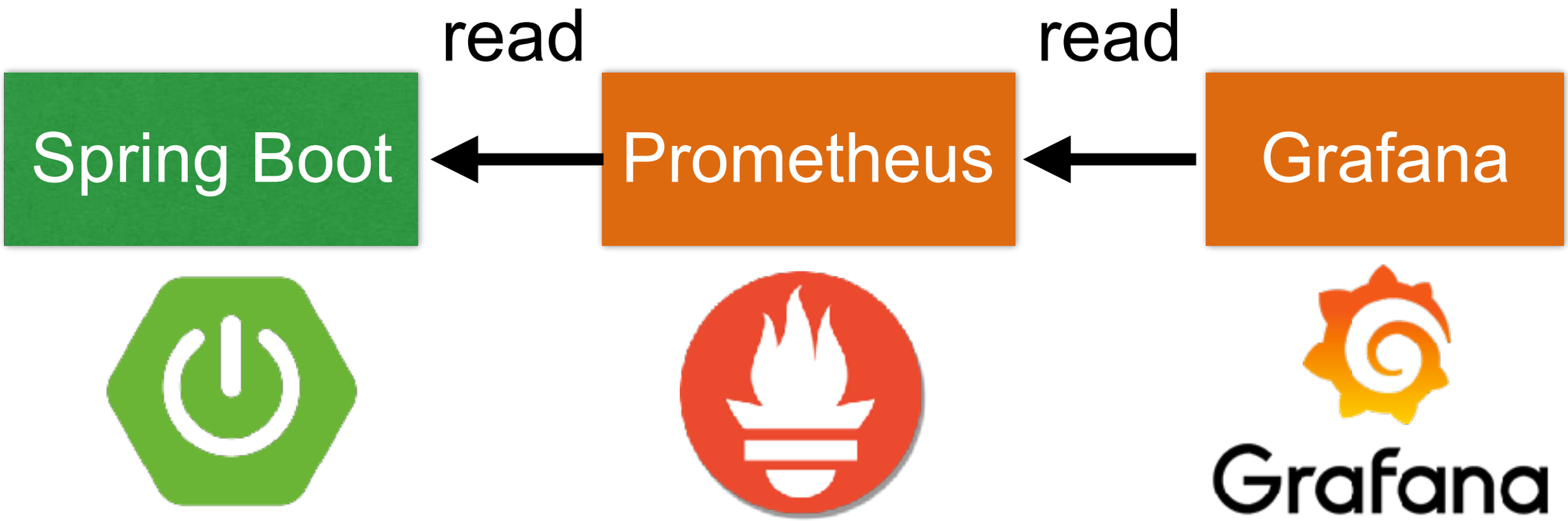


https://micrometer.io/

Keep and Visualize Metric of Spring Boot Services

Sample Architecture

Sample Architecture



Service metric for Prometheus

Enable Prometheus (1)

Add library to pom.xml

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
    <version>1.0.1</version>
</dependency>
```

Enable Prometheus (2)

Enabled endpoint in application.properties

management.endpoints.web.exposure.include =….,**prometheus**

Enable Prometheus (3)

New endpoint = actuator/prometheus

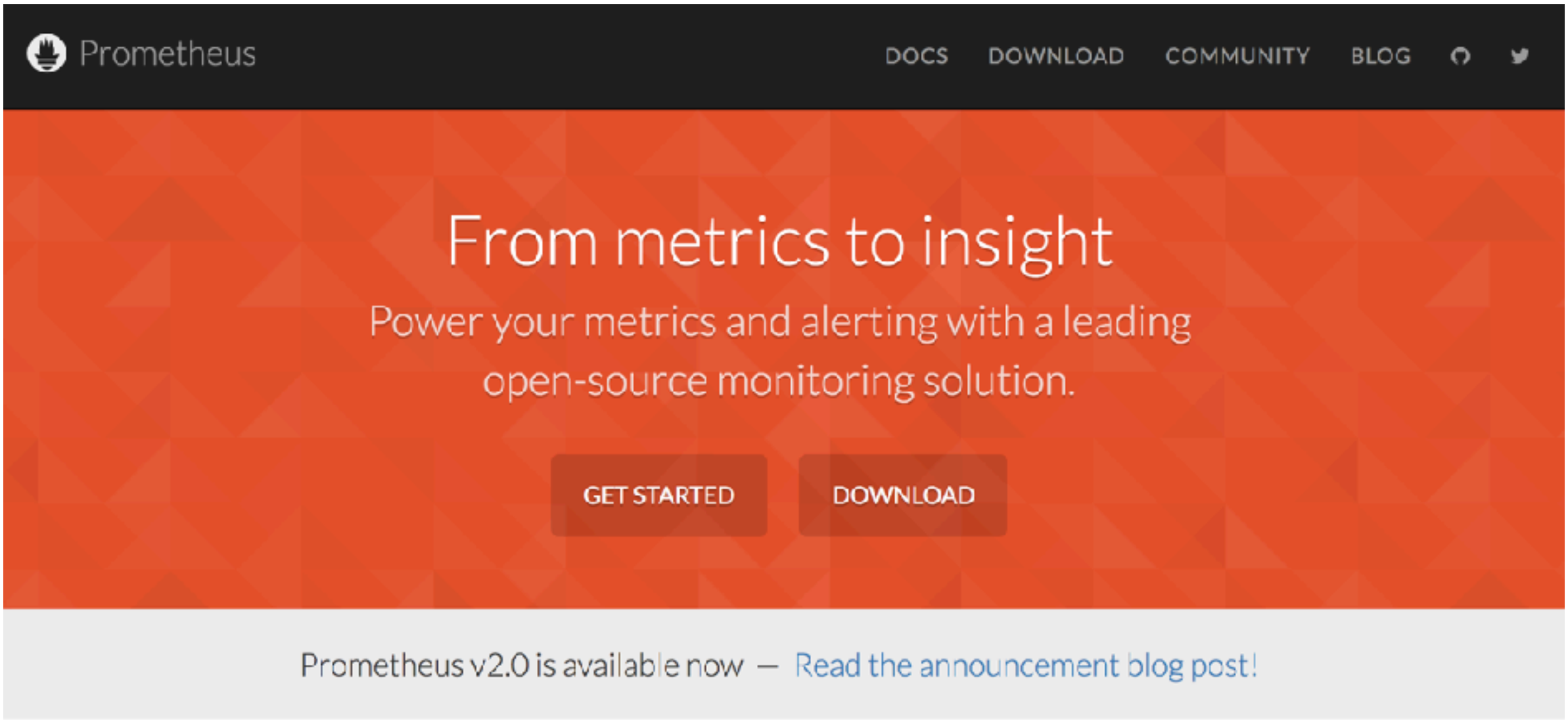
localhost:8080/actuator/prometheus

```
# HELP jvm_memory_used_bytes The amount of used memory
# TYPE jvm_memory_used_bytes gauge
jvm_memory_used_bytes{area="nonheap",id="Code Cache",} 1.49056E7
jvm_memory_used_bytes{area="nonheap",id="Metaspace",} 5.6766712E7
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",} 7617096.0
jvm_memory_used_bytes{area="heap",id="PS Eden Space",} 1.7135864E7
jvm_memory_used_bytes{area="heap",id="PS Survivor Space",} 1.6235192E7
jvm_memory_used_bytes{area="heap",id="PS Old Gen",} 2.1936456E7
# HELP hikaricp_connections_idle Idle connections
# TYPE hikaricp_connections_idle gauge
hikaricp_connections_idle{pool="HikariPool-1",} NaN
# HELP tomcat_threads_config_max
# TYPE tomcat_threads_config_max gauge
tomcat_threads_config_max{name="http-nio-8080",} 200.0
# HELP tomcat_servlet_error_total
# TYPE tomcat_servlet_error_total counter
tomcat_servlet_error_total{name="default",} 0.0
# HELP jvm_threads_peak The peak live thread count since the Java virtual machine start
# TYPE jvm_threads_peak gauge
jvm_threads_peak 28.0
# HELP hikaricp_connections_pending Pending threads
# TYPE hikaricp_connections_pending gauge
hikaricp_connections_pending{pool="HikariPool-1",} NaN
# HELP system_cpu_count The number of processors available to the Java virtual machine
```

Keep data in Prometheus

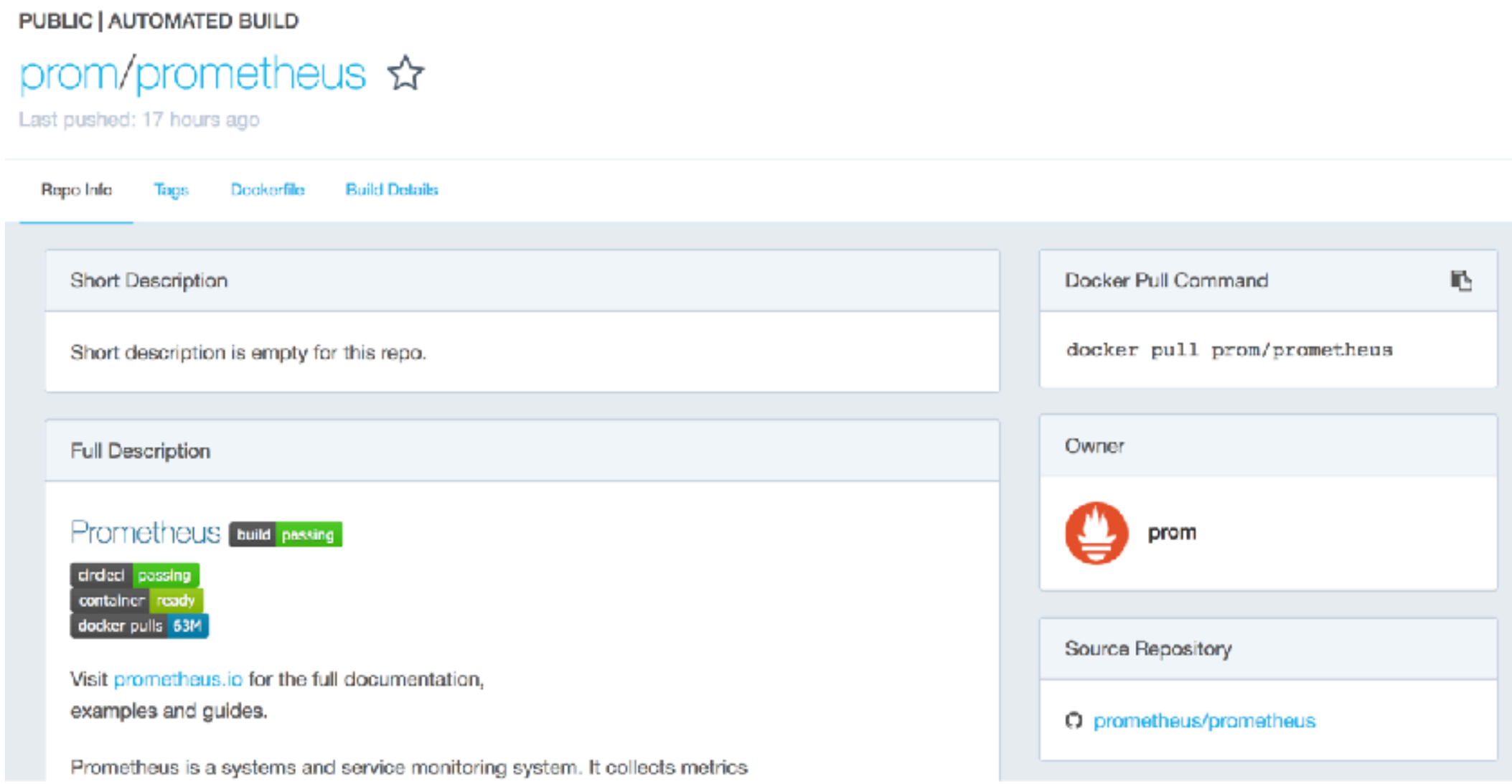
https://prometheus.io/

Prometheus



https://prometheus.io/

Prometheus



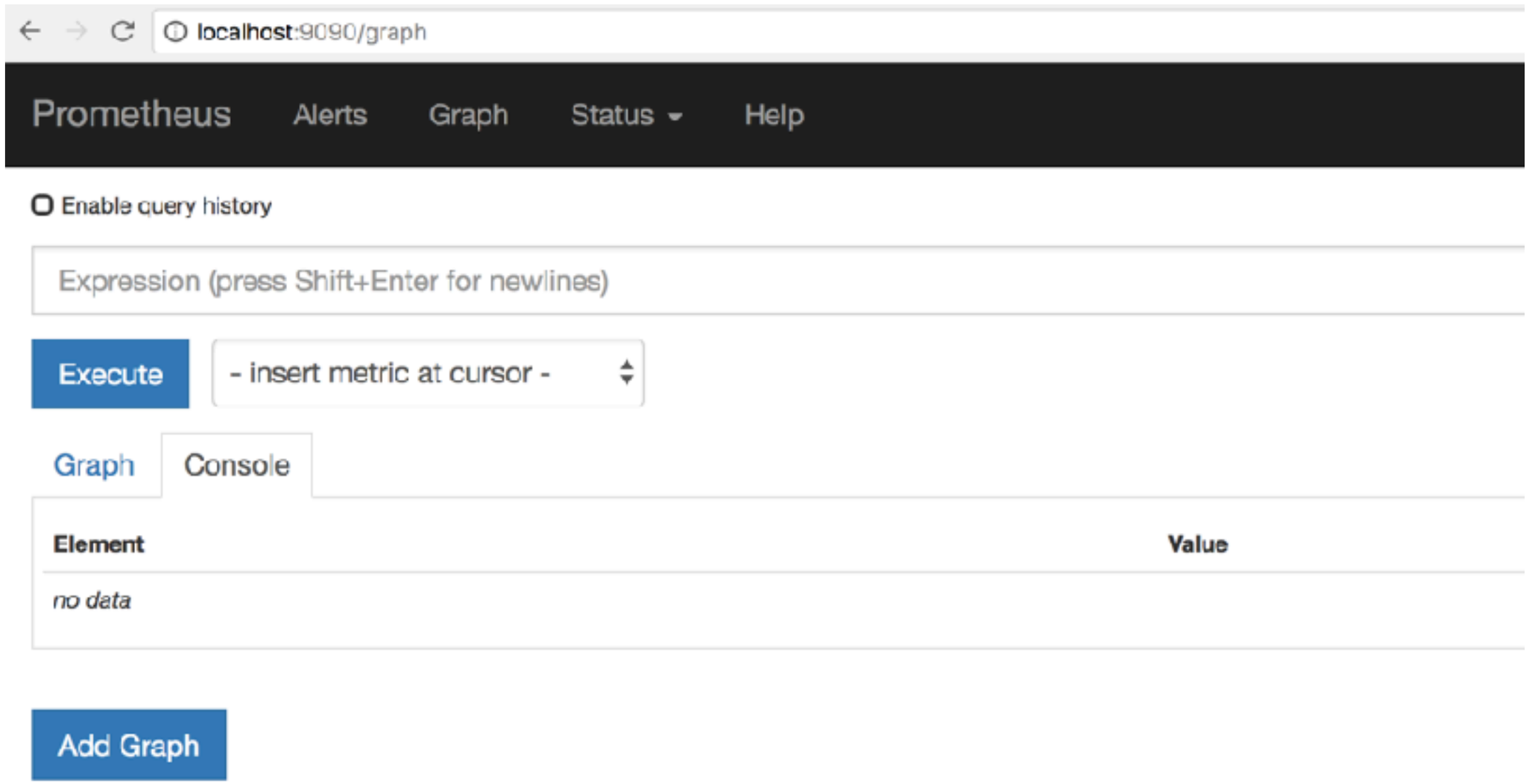
https://hub.docker.com/r/prom/prometheus/

Create container of Prometheus

$docker container run --rm
  -p 9090:9090
  -v $(pwd)/prometheus.yml:**/etc/prometheus/prometheus.yml**
  --name monitor prom/prometheus

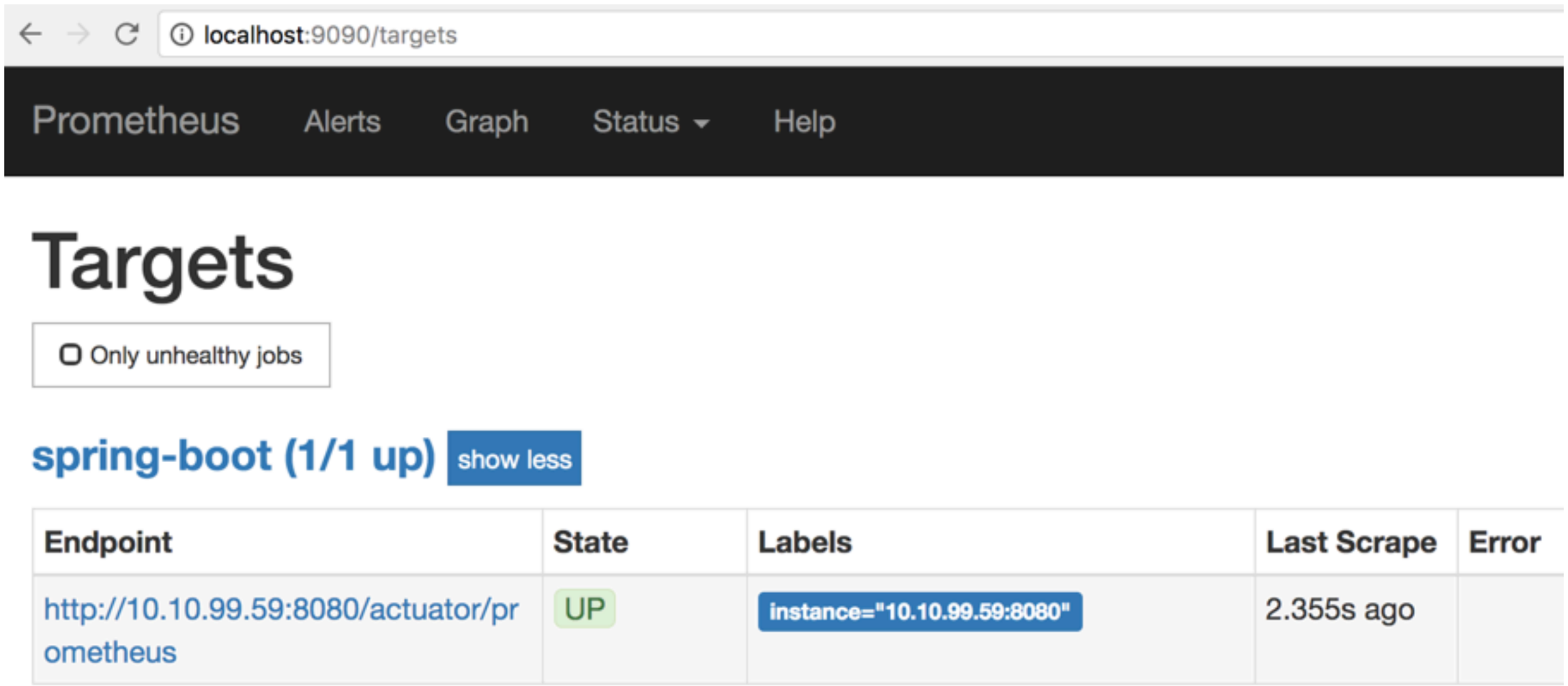
Check Data in Prometheus

http://localhost:9090/



Check Target in Prometheus

Status -> Targets



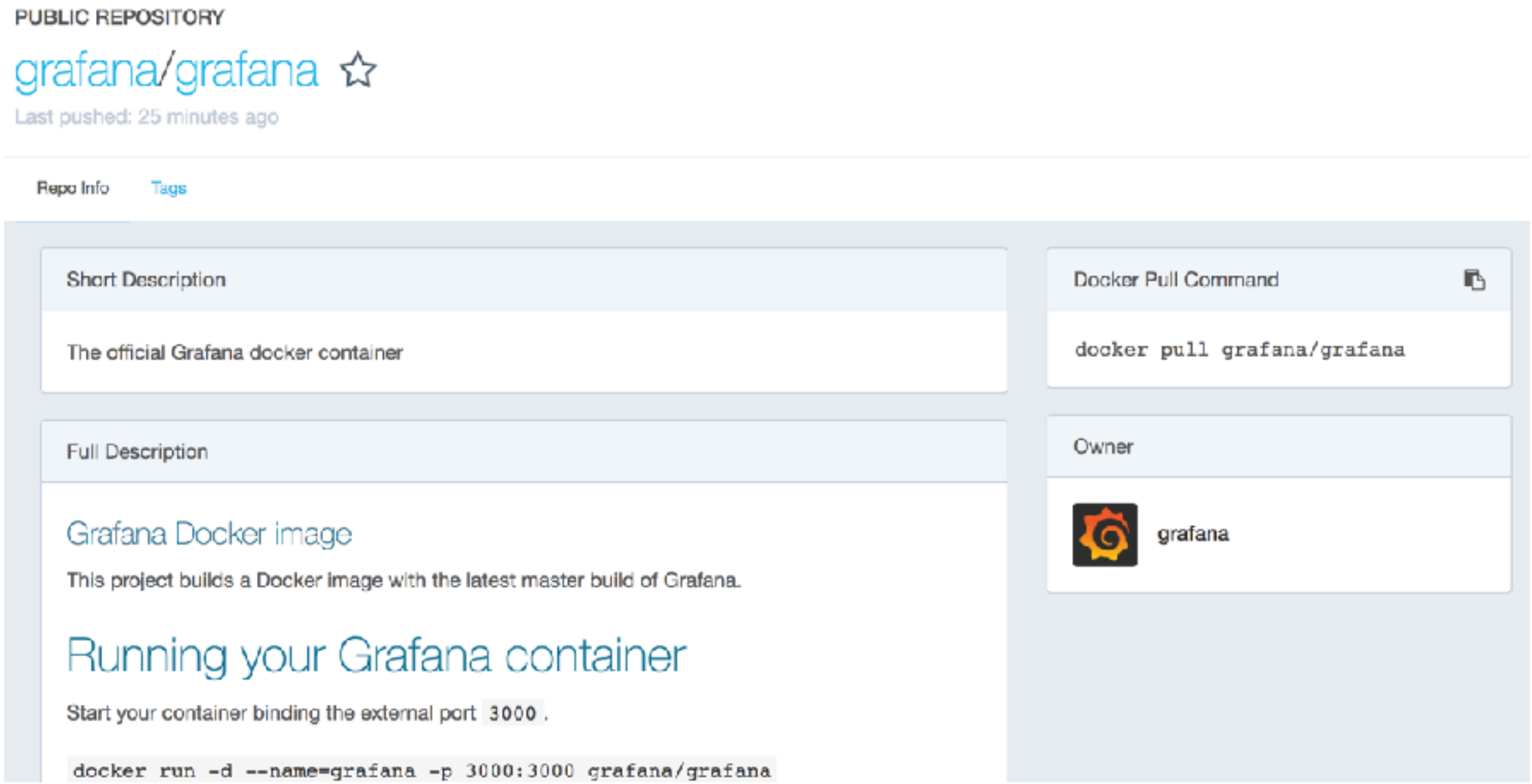
Show data in Grafana

https://grafana.com/

Grafana

https://grafana.com/

Grafana



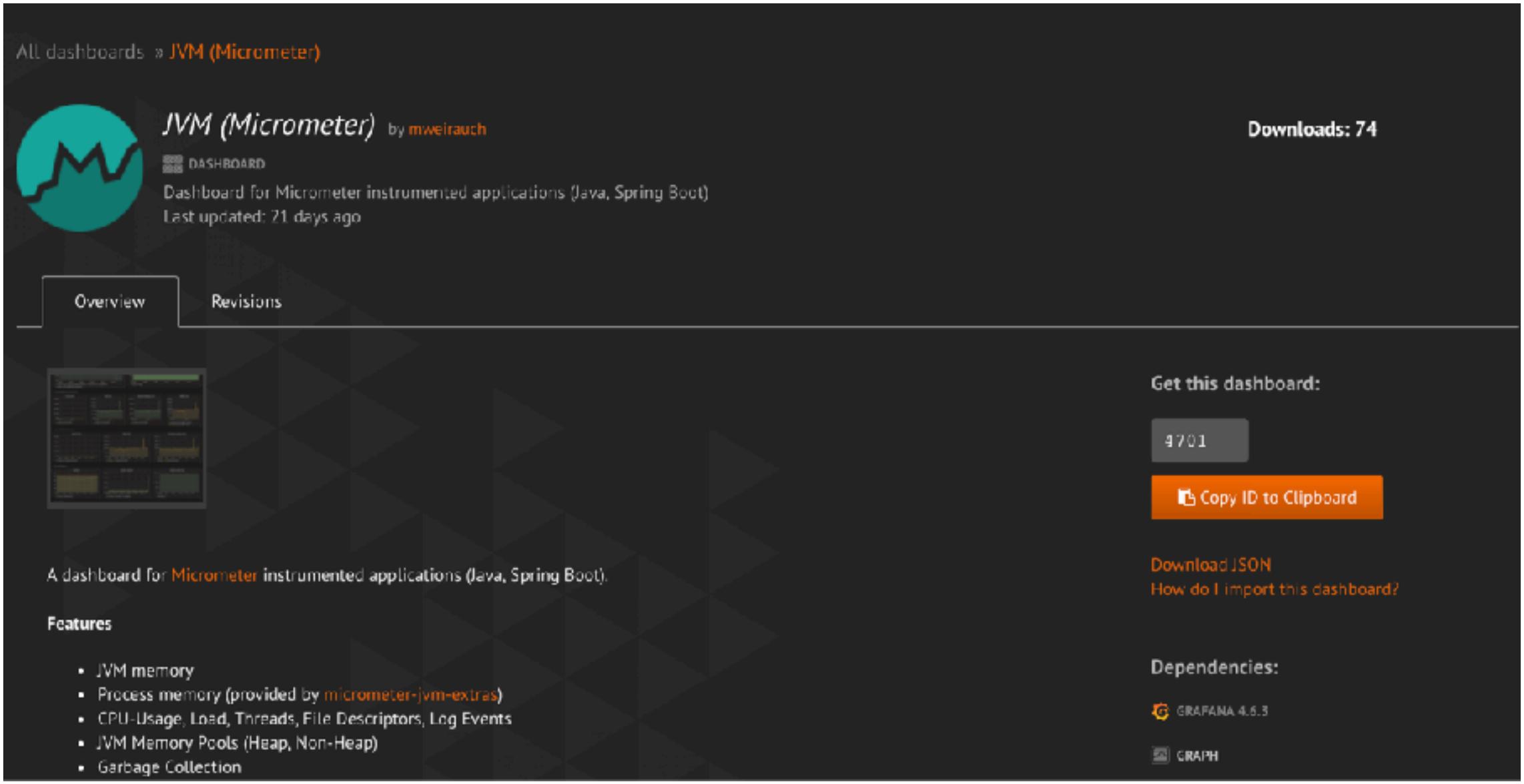
https://hub.docker.com/r/grafana/grafana/

Create container of Grafana

```
$docker container run
  --name=grafana
  -p 3000:3000 grafana/grafana
```

Grafana Dashboard

https://grafana.com/dashboards/4701



Take to your home

Always improve, always practice

Thank you
Q/A