



**Develop RESTful API  
Spring Boot  
NodeJS  
Go**



Somkiat Puisungnoen

Search

Somkiat | Home

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามชานาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata



Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

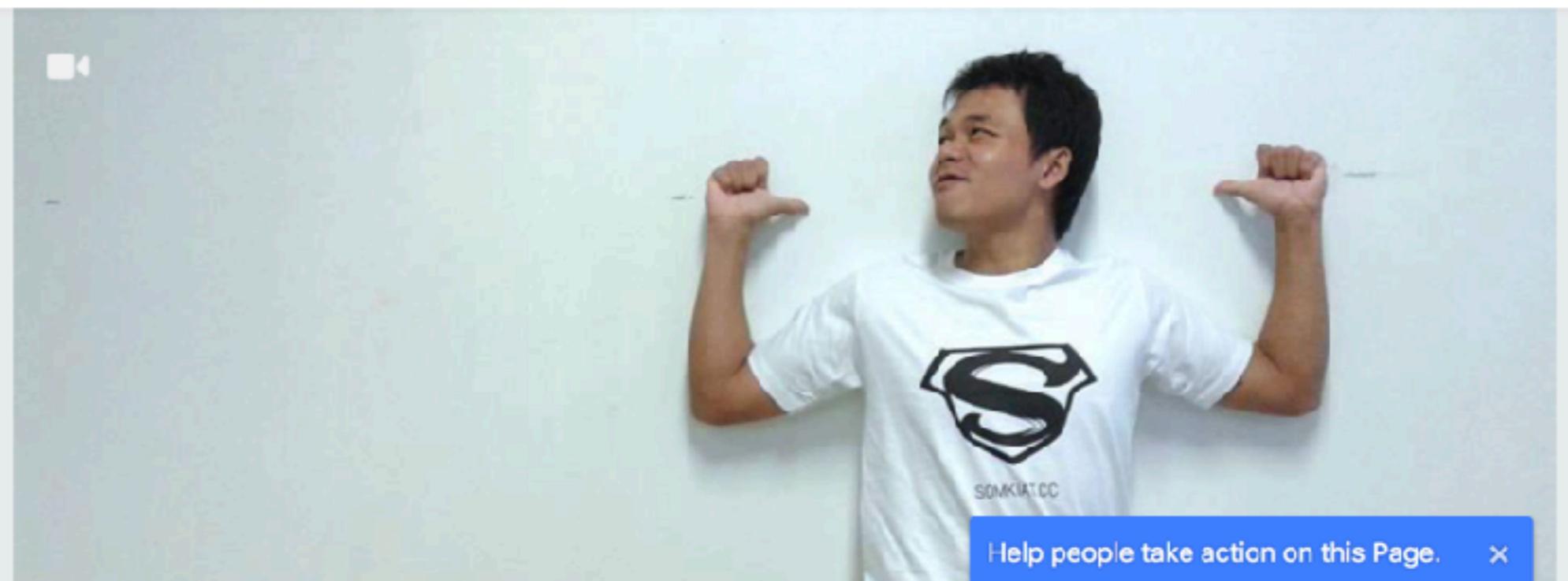
@somkiat.cc

Home

Posts

Videos

Photos



 Liked ▾

 Following ▾

 Share

...

+ Add a Button



Workshop

3

© 2020 - 2024 Siam Chamnankit Company Limited. All rights reserved.

**<https://github.com/up1/workshop-springboot-nodejs-go>**



# Develop RESTFul API



# Topics



# Basic of REST

REST (REpresentational State Transfer)

6 REST constraints

REST vs RESTful API

Design principles



# Spring Boot

Goals of Spring Boot

Create project

Better project structure of Spring Boot

Develop RESTful APIs

Layer in Spring Boot project

Error handling

Manage database with Spring Data

Testing

Observable service



# NodeJS

Basic of NodeJS

Create project with npm

Project structure of project

Develop RESTful APIs

Error handling

Manage database (postgresql and redis)

Testing

Observable service



# Go

Basic of go

Create project with go module

Project structure of project

Develop RESTful APIs

Error handling

Manage database (postgresql and redis)

Testing

Observable service

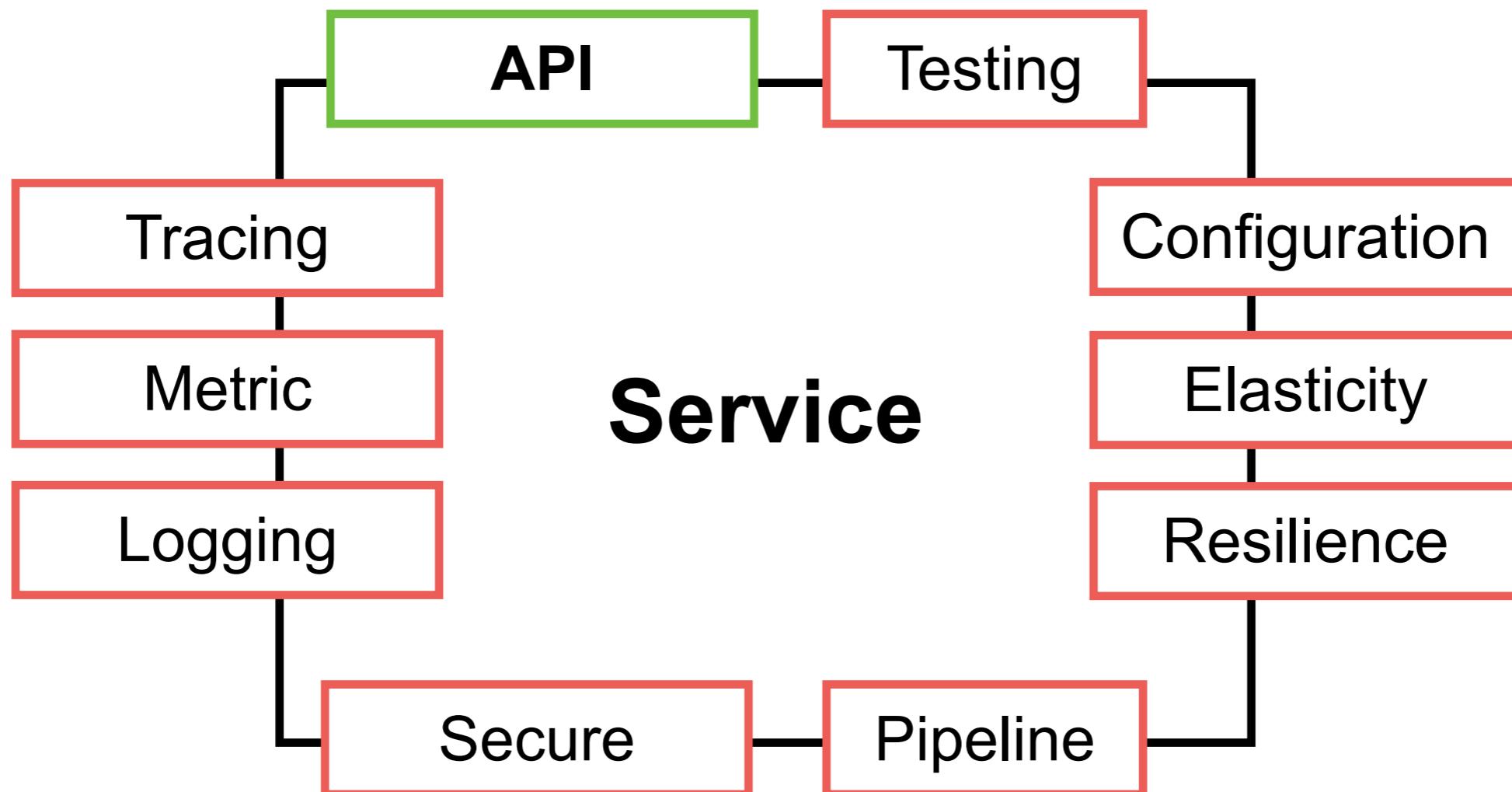


# My Rule

**Always write tests**



# Properties of Service



# APIs

## Application Programming Interface

**REST API**

gRPC

Messaging



# REST API



# **REST**

## **REpresentation State Transfer**

**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](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)



# Goals

Scalability  
Generality of interfaces  
Independent deployment of components



# REST properties

Performance

Scalability

Simplicity

Modification

Visibility

Portability

Reliability



# REST constraints

Client-server

Stateless

Cacheable

Layer system

Uniform  
interface

Code on  
demand

eg. JavaScript



# **RESTful service**

Implementation from REST



# REST Request Messages

RESTful request is typically in form of  
**Uniform Resource Identifiers (URI)**

Structure of URI depend on specific service

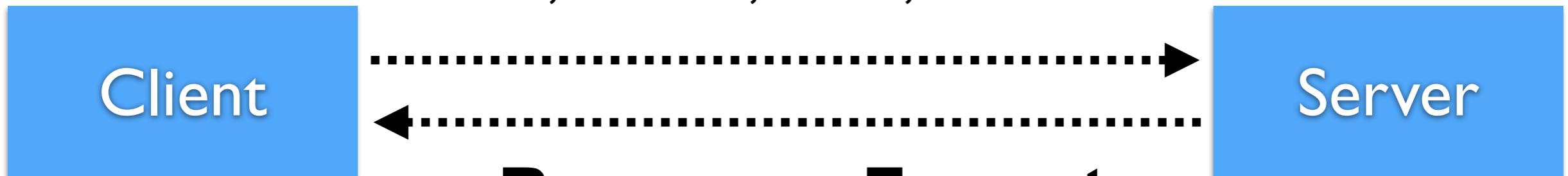
Request can include parameter and data in  
body of request as XML, JSON etc.



# REST Request & Response

**Request Method**

GET, POST, PUT, DELETE



**Response Format**

XML or JSON



# 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 ?



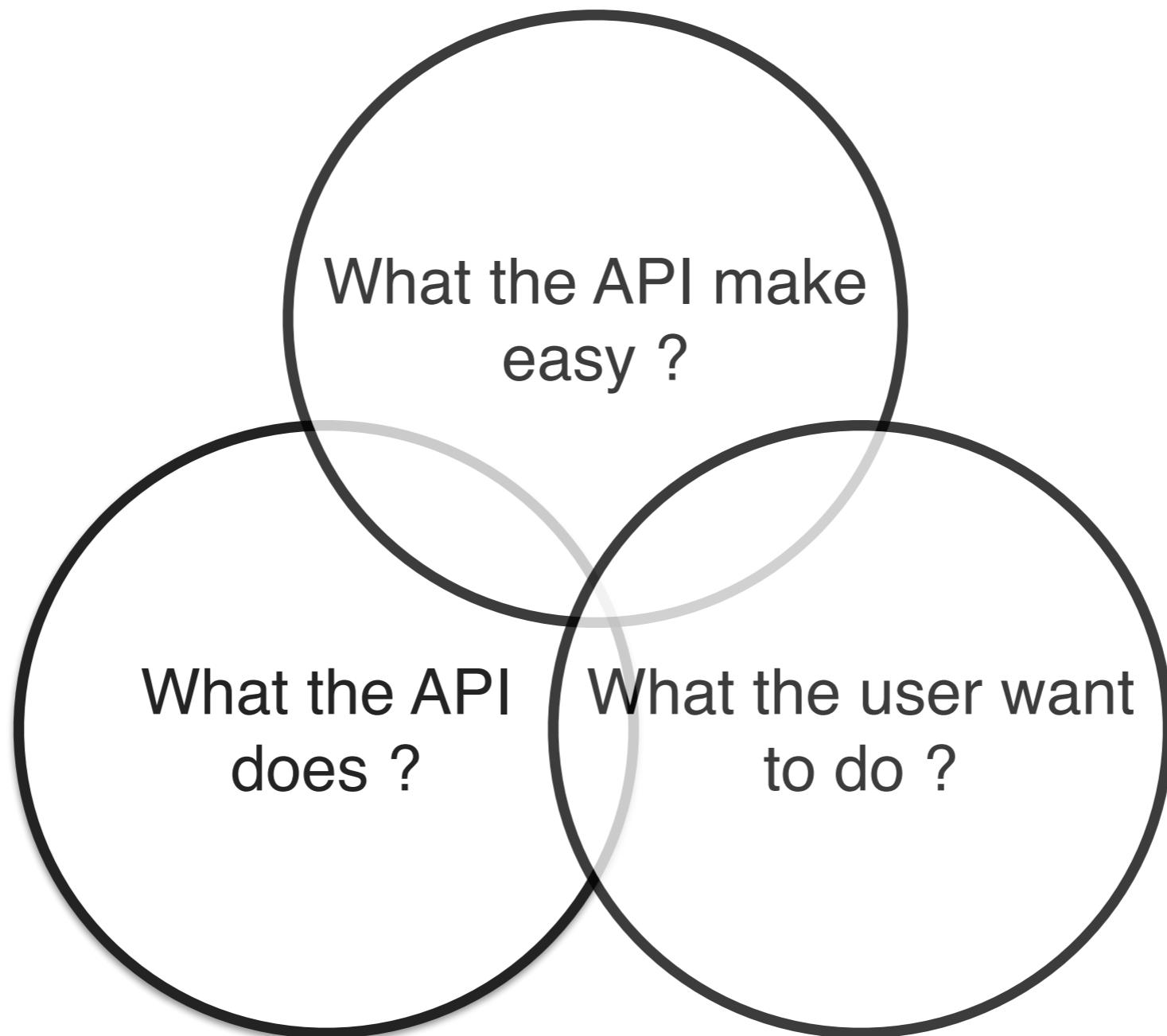
# Status Code

Code	Description
1xx	Information
2xx	Successful
3xx	Redirection
4xx	Client error
5xx	Server error

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>



# Good APIs ?



# Principles of API Design

Consistency

Statelessness

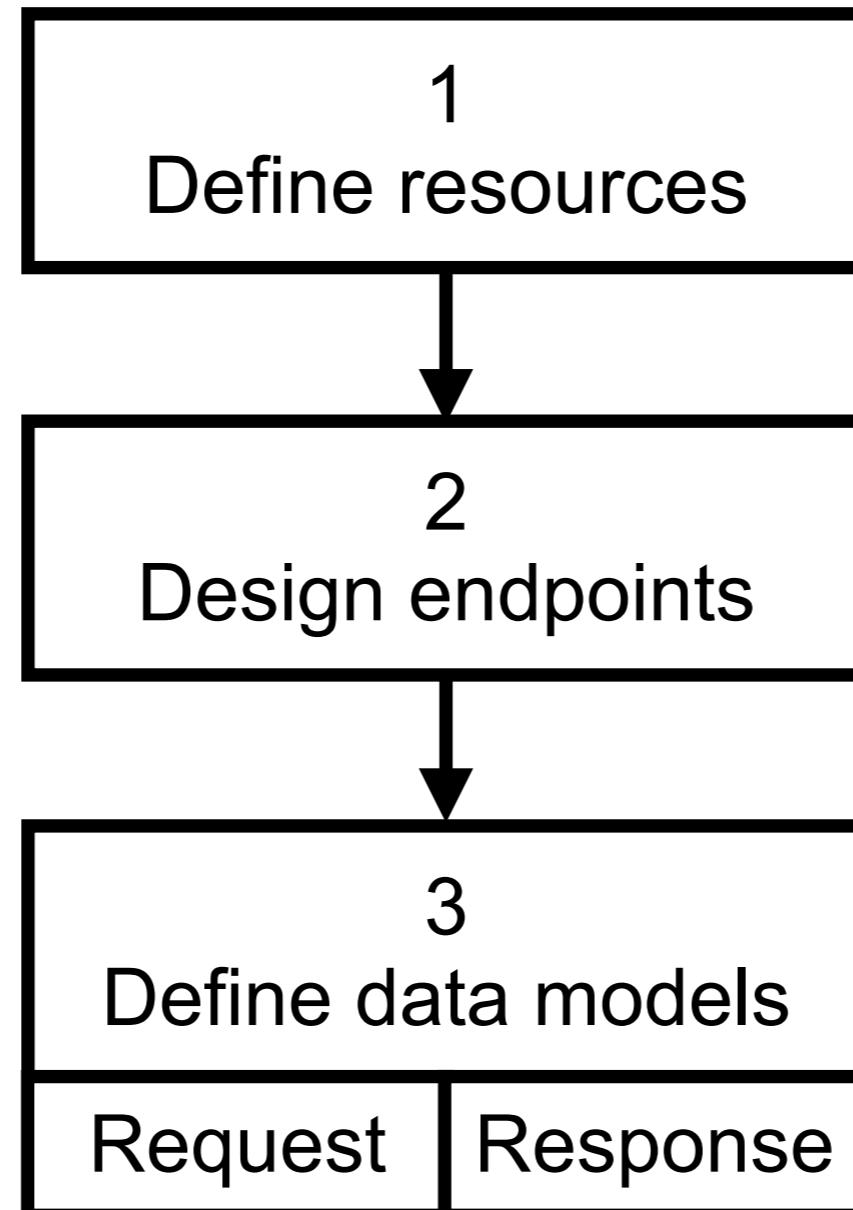
Resource-oriented design

Use standard  
HTTP methods

Versioning



# Steps to Design APIs



# More !!

Authentication  
Authorization

Pagination and  
filtering

Rate Limiting

Pagination and  
filtering

Error handling

Documentation

Testing

Monitoring and  
Observability



# Implementation

Coding

Testing

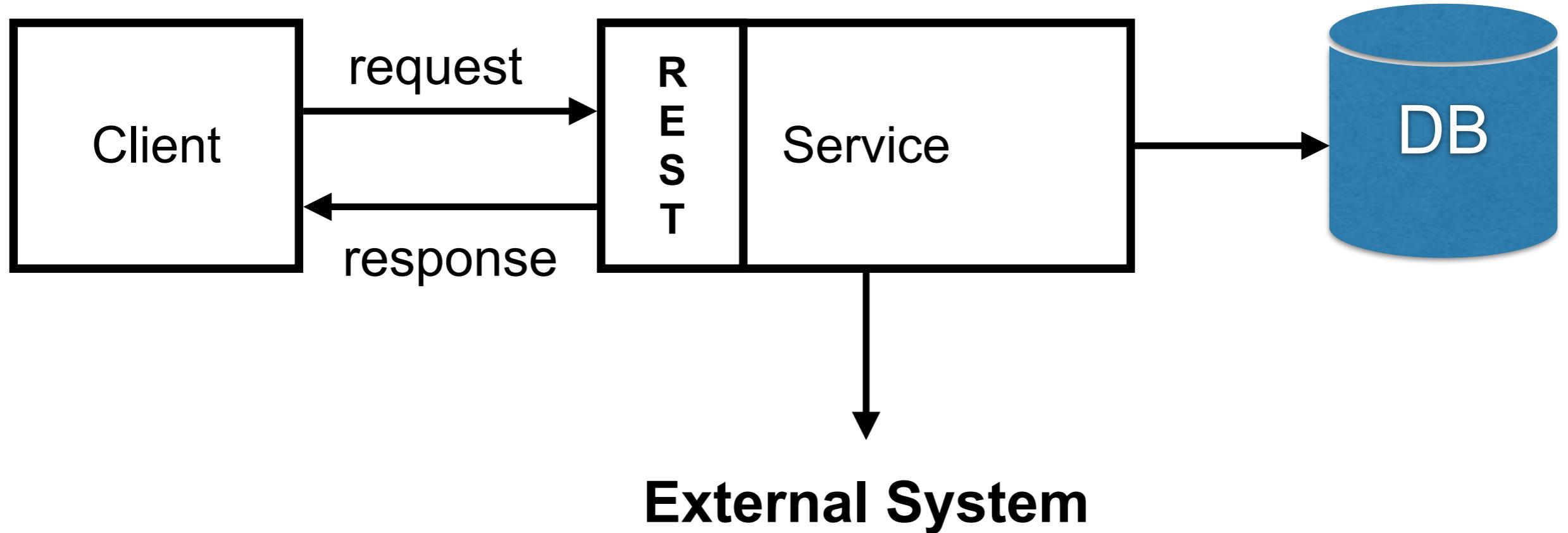
Observability

Performance

Security



# Structure of Service



# Design RESTful APIs



# Users

Method	Path	Description
GET	/users	Get all users
GET	/users/{id}	Get user by id
POST	/users	Create new user
PUT	/users/{id}	Update user
DELETE	/users/{id}	Delete user by id



# HTTP status codes

Code	Description	Example
1xx	Informational responses	100 = Continue 101 Switching protocols
2xx	Success	200 = OK 201 = Created
3xx	Redirection	300 = Multiple choices 301 = Moved permanently
4xx	Client errors	400 = Bad request 403 = Forbidden 404 = Not found
5xx	Server errors	500 = Internal server error 502 = Bad gateway 503 = Service unavailable



# API Documentation

## pet Everything about your Pets

**POST** /pet/{petId}/uploadImage uploads an image 

**POST** /pet Add a new pet to the store 

**PUT** /pet Update an existing pet 

**GET** /pet/findByStatus Finds Pets by status 

**GET** /pet/{petId} Find pet by ID 

**POST** /pet/{petId} Updates a pet in the store with form data 

**DELETE** /pet/{petId} Deletes a pet 

<https://swagger.io/>



Workshop

© 2020 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Develop RESTful API



# Spring Framework



# Spring Boot



# Create Spring Boot project

The screenshot shows the Spring Initializr web application interface. It has several sections:

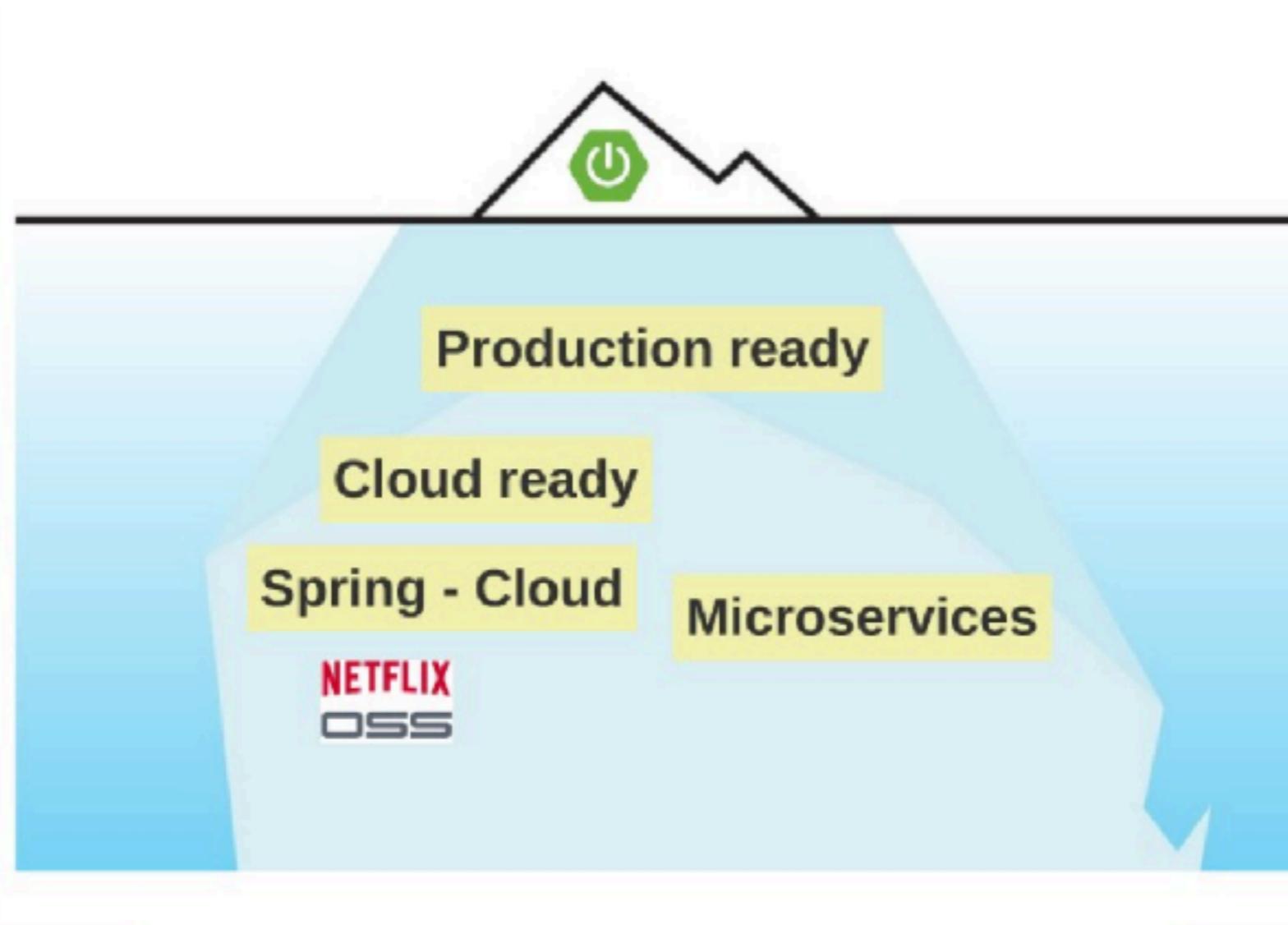
- Project**: Options for Gradle - Groovy, Gradle - Kotlin, Maven, and Groovy. Maven is selected.
- Language**: Options for Java (selected), Kotlin, and Groovy.
- Dependencies**: A button labeled "ADD DEPENDENCIES..." with a keyboard shortcut "⌘ + B". Below it, it says "No dependency selected".
- Spring Boot**: Options for 3.5.0 (SNAPSHOT), 3.5.0 (M3), 3.4.5 (SNAPSHOT), 3.4.4 (selected), 3.3.11 (SNAPSHOT), and 3.3.10.
- Project Metadata**: Fields for Group (com.example), Artifact (demo), Name (demo), Description (Demo project for Spring Boot), and Package name (com.example.demo).
- Buttons at the bottom**: "GENERATE ⌘ + ↵", "EXPLORE CTRL + SPACE", and "...".

<https://start.spring.io/>



# Why Spring Boot ?

Application skeleton generator  
Reduce effort to add new technologies



# What ?

Embedded application server

Integration with tools/technologies (starter)

Production tools (monitoring, health check)

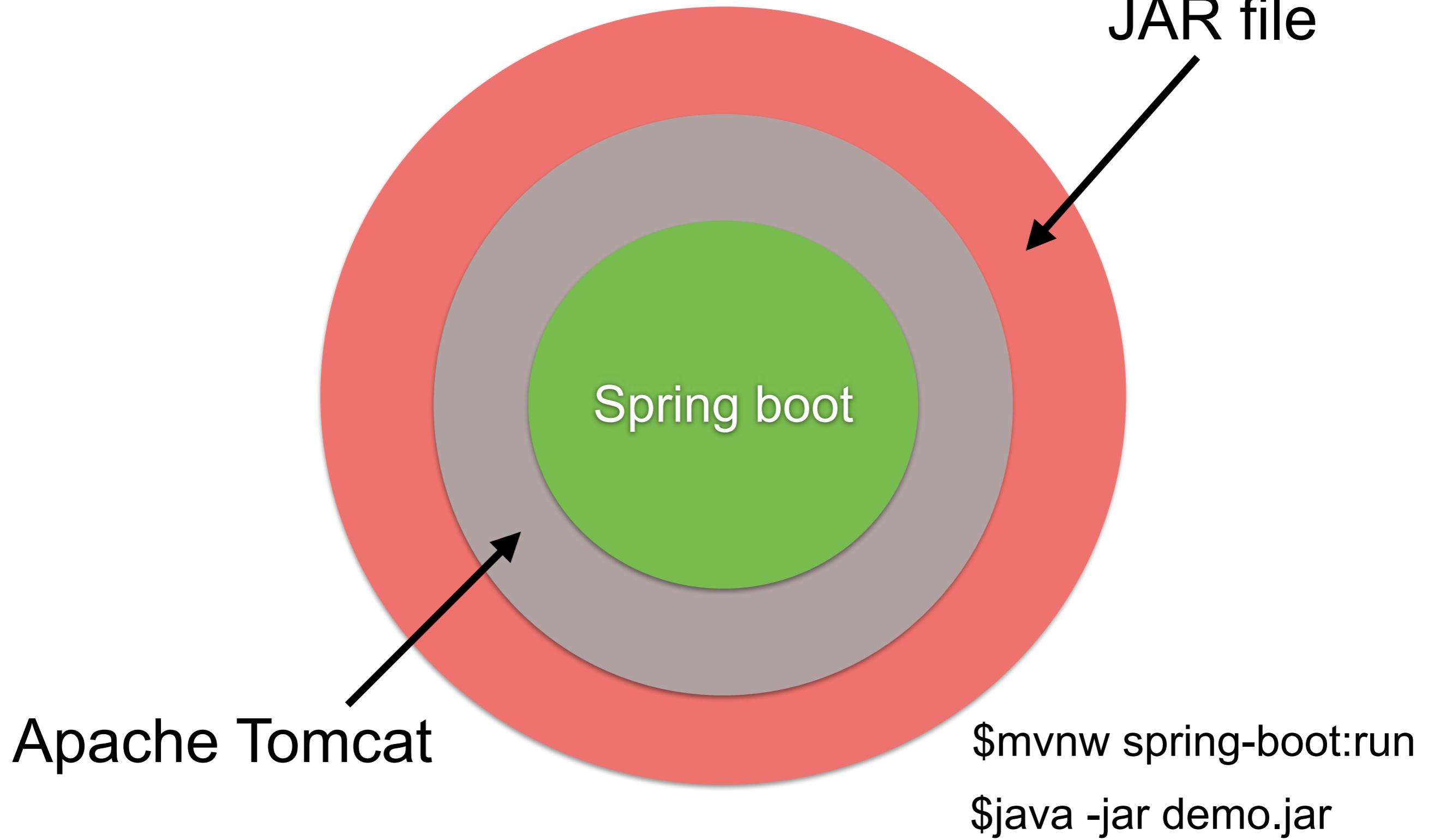
Configuration management

Dev tools

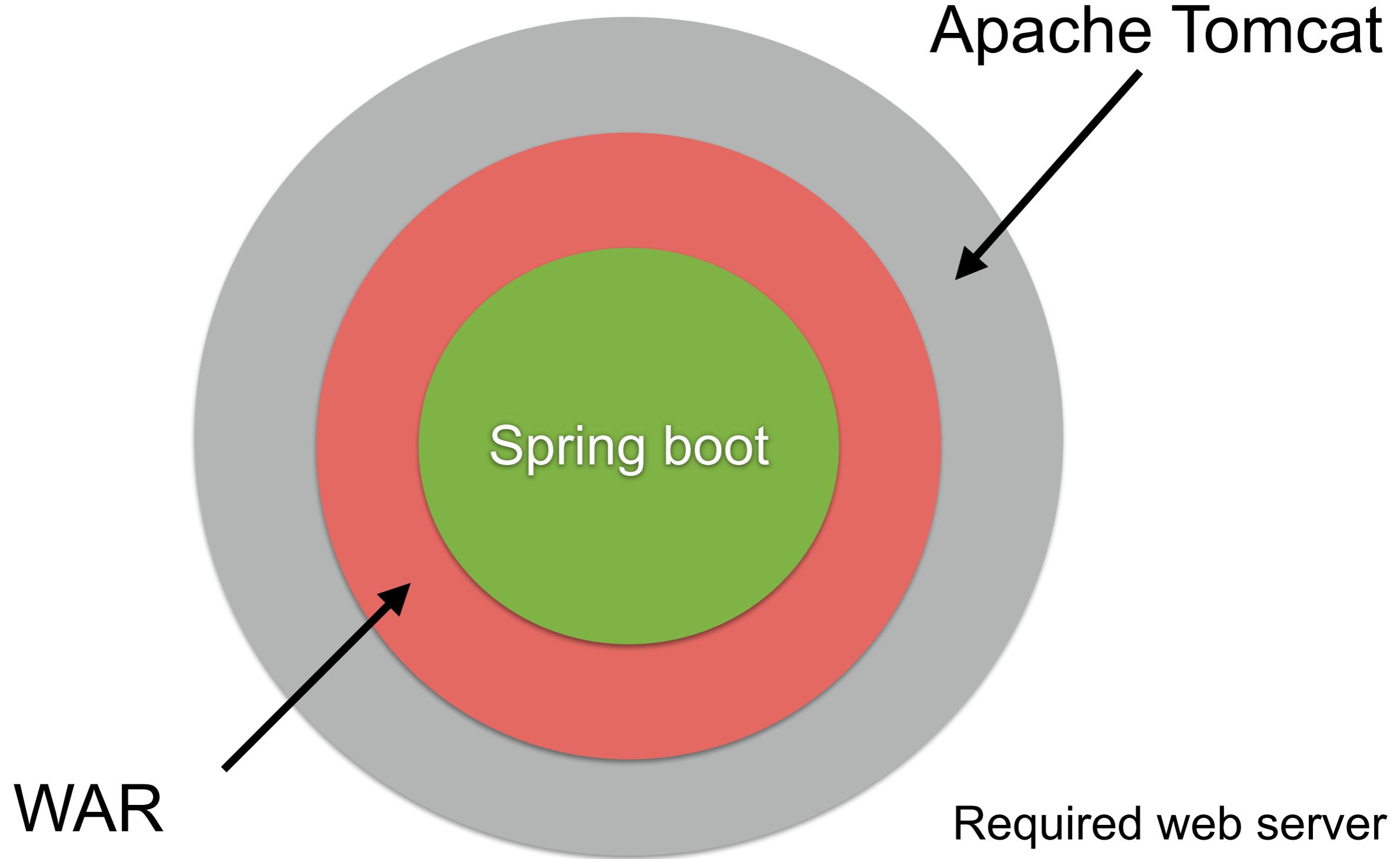
No source code generation, no XML



# How ?

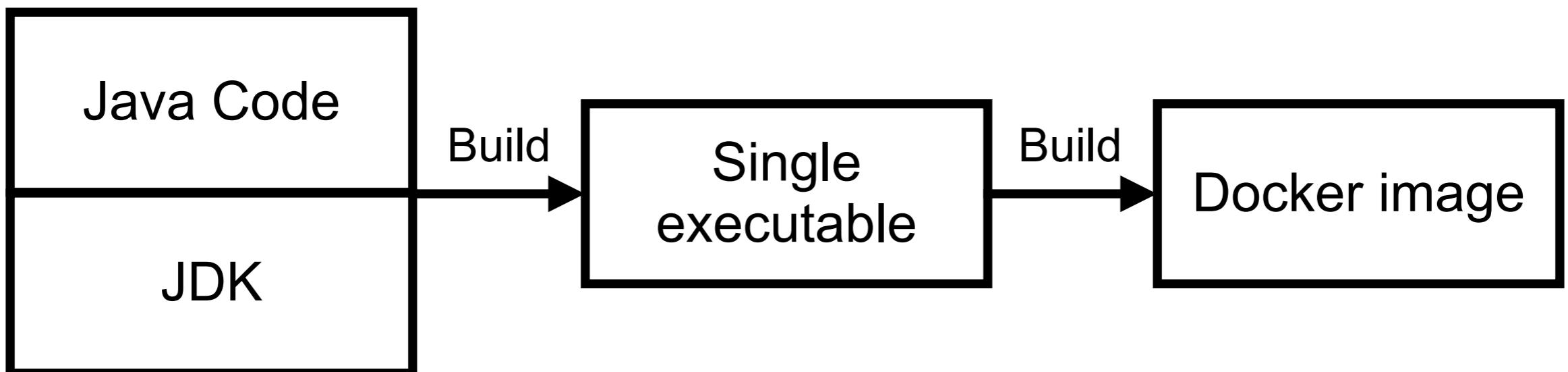


# Traditional Deployment



# GraalVM Native Support

Reduce size and memory usage



<https://www.graalvm.org/>



# Add to Spring Boot project



The screenshot shows the Spring Initializr web interface. On the left, there are sections for Project (Gradle - Groovy selected), Language (Java selected), and Spring Boot (3.1.4 selected). On the right, a red dashed box highlights the 'Dependencies' section, which includes 'GraalVM Native Support' (selected) under 'DEVELOPER TOOLS'. Below it, a note says 'Support for compiling Spring applications to native executables using the GraalVM native-image compiler.' At the bottom, the title '1. GraalVM support' is displayed in large red text.

Project

- Gradle - Groovy
- Gradle - Kotlin
- Maven

Language

- Java
- Kotlin
- Groovy

Spring Boot

- 3.2.0 (SNAPSHOT)
- 3.2.0 (M3)
- 3.1.5 (SNAPSHOT)
- 3.1.4
- 3.0.12 (SNAPSHOT)
- 3.0.11
- 2.7.17 (SNAPSHOT)
- 2.7.18

Project Metadata

Group: com.example

Artifact: demo

Name: demo

Description: Demo project for Spring Boot

Package name: com.example.demo

Packaging:  Jar  War

Java:  21  17  11  8

Dependencies

**GraalVM Native Support** DEVELOPER TOOLS

Support for compiling Spring applications to native executables using the GraalVM native-image compiler.

## 1. GraalVM support

<https://docs.spring.io/spring-boot/docs/current/reference/html/native-image.html>



# **Building RESTful API with Spring Boot**



# Create new project



Project  
 Gradle - Groovy  Gradle - Kotlin  
 Maven

Language  
 Java  Kotlin  Groovy

Spring Boot  
 3.2.0 (SNAPSHOT)  3.2.0 (M2)  3.1.4 (SNAPSHOT)  3.1.3  
 3.0.11 (SNAPSHOT)  3.0.10  2.7.16 (SNAPSHOT)  2.7.15

Project Metadata  
**Group**: com.example  
**Artifact**: demo  
**Name**: demo  
**Description**: Demo project for Spring Boot  
**Package name**: com.example.demo  
**Packaging**:  Jar  War  
**Java**:  20  17  11  8

## Dependencies

[ADD DEPENDENCIES...](#) 36 + 3

### Spring Web WEB

Build web, including RESTful applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

### H2 Database SQL

Provides a fast in-memory database that supports JDBC API and H2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

### Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

1. Spring Web
2. Spring Data JPA
3. H2 Database



# Run project (Dev mode)

```
./mvnw spring-boot:run
```

```
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
```



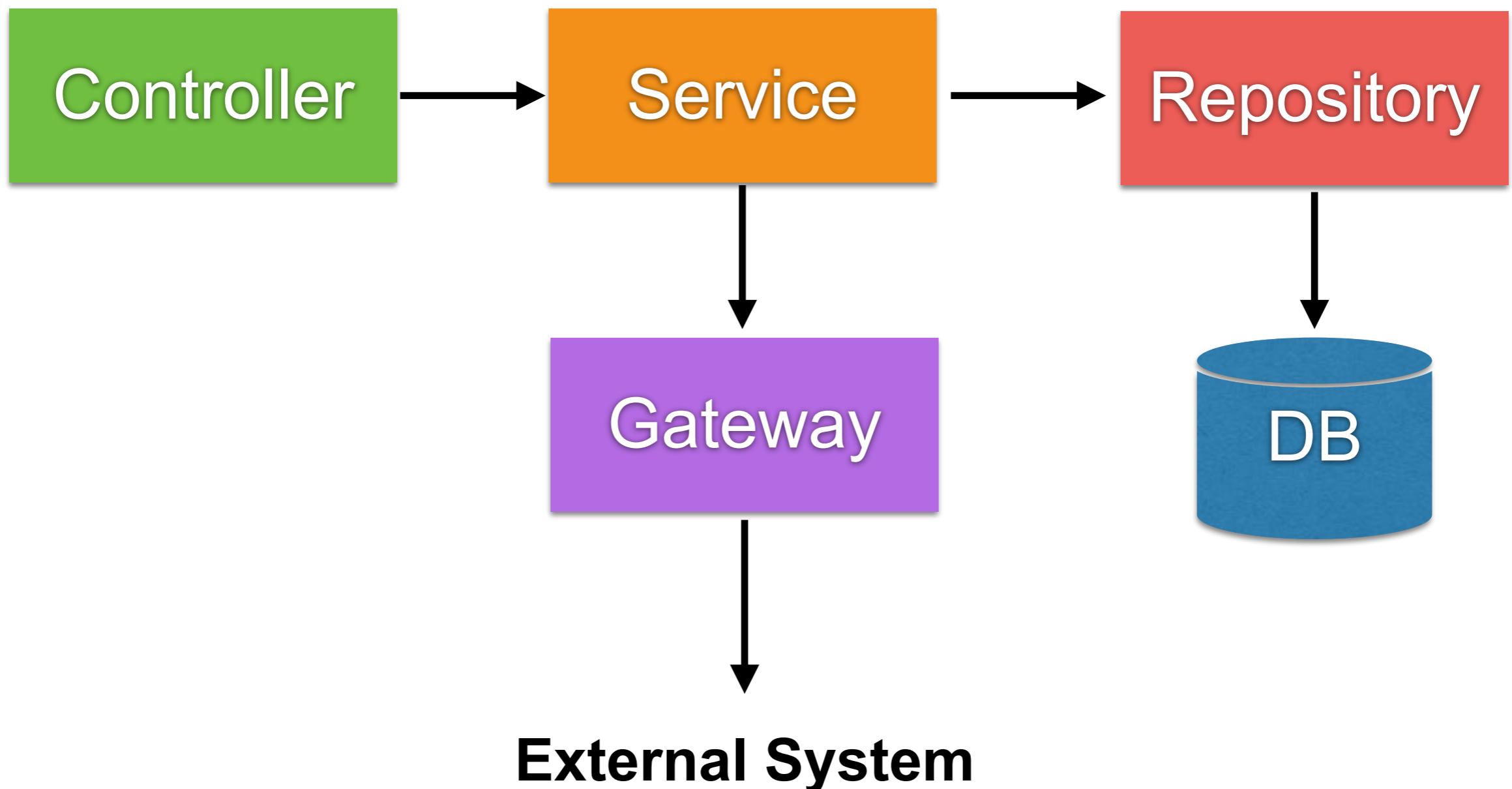
# Run project (production mode)

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

```
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
```



# Structure of Spring Boot project



# Controller

Request and Response  
Validation input

Delegate to others classes such as service and repository



# Service

Control the flow of main business logic  
Manage database transaction  
Don't reuse service



# Repository

Manage data in data store such as RDBMS and  
NoSQL



# Gateway

Call external service via network such as  
WebServices and RESTful APIs



# Spring Boot Structure (1)

Separate by function/domain/feature

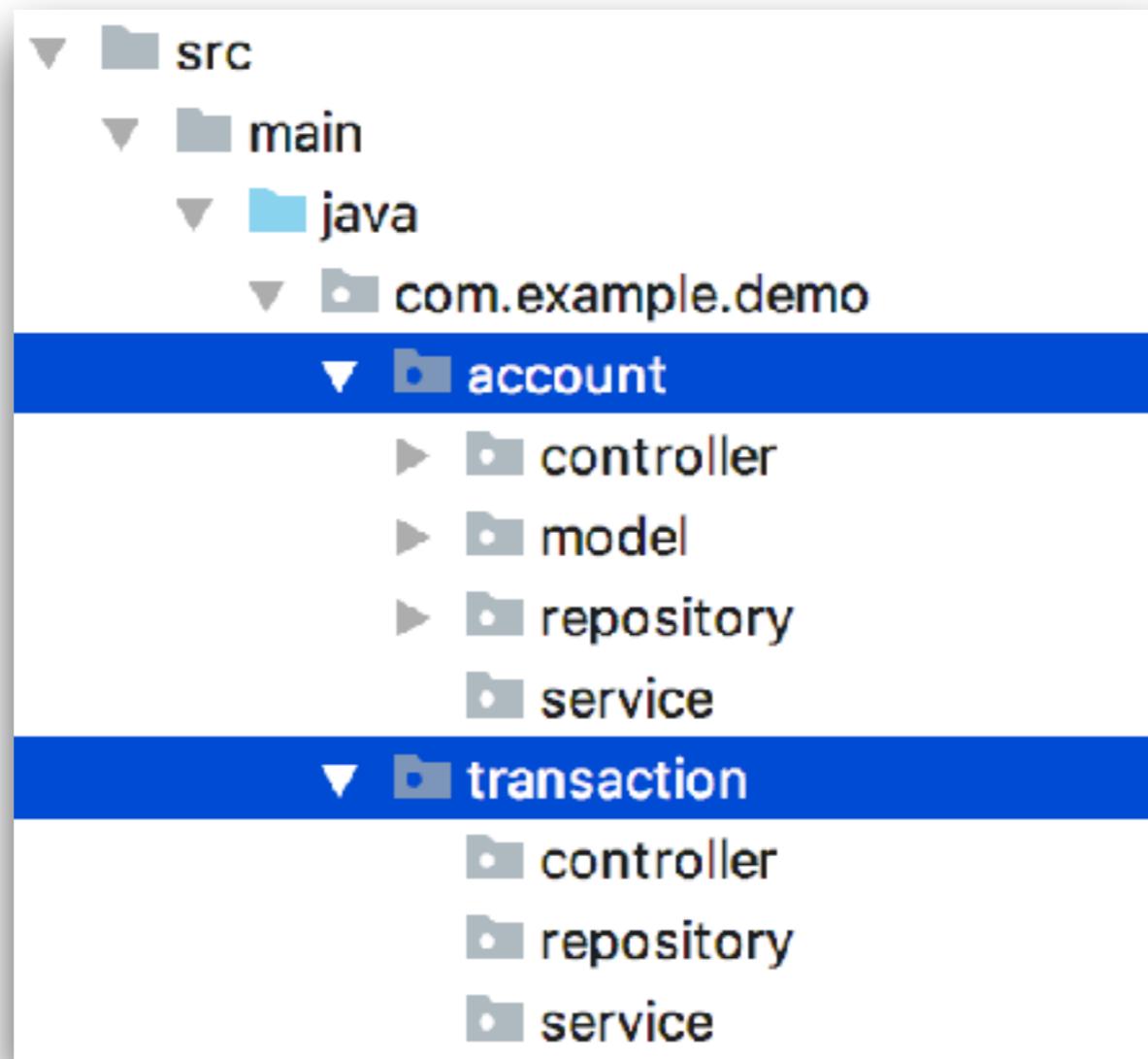
```
feature1
  - controller
  - service
  - repository
feature2
  - controller
  - service
  - repository
```

<https://docs.spring.io/spring-boot/reference/using/structuring-your-code.html>



# Spring Boot Structure (2)

Separate by function/domain/feature



# Configurations

A twelve-factor app should externalize all such configurations that vary between deployments

## File application.properties

```
spring.datasource.url=jdbc:mysql://${MYSQL_HOST}:${MYSQL_PORT}/movies  
spring.datasource.username=${MYSQL_USER}  
spring.datasource.password=${MYSQL_PASSWORD}
```

## Usage

```
set MYSQL_HOST=localhost  
set MYSQL_PORT=3306  
set MYSQL_USER=movies  
set MYSQL_PASSWORD=password
```

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



# Convert from properties to YAML

## Yaml to properties / Properties to Yaml converter

[\*\*<< to yaml\*\*](#) [\*\*to properties >>\*\*](#)

**Yaml**

```
spring:  
  h2:  
    console:  
      enabled: 'true'  
  profiles: test  
  kafka:  
    streams:  
      bootstrap-servers: localhost:9092,  
      192.168.0.1:9092
```

**Properties**

```
spring.profiles=test  
spring.h2.console.enabled=true  
spring.kafka.streams.bootstrap-  
servers=localhost:9092, 192.168.0.1:9092
```

[\*\*convert >>\*\*](#)

[\*\*<< convert\*\*](#)

[\*\*<< convert\*\*](#) [\*\*convert >>\*\*](#)

**YAML**

<https://mageddo.com/tools/yaml-converter>



# Convert in IntelliJ IDEA

The screenshot shows the IntelliJ IDEA plugin marketplace interface. A red dashed box highlights the search results for 'convert YAML'. The results list includes:

- Convert YAML and Properties File** by chencn, 111.9K downloads, 3.49 rating, Install button.
- Properties to YAML Converter** by Jakub Kubrynski, 103.4K downloads, 3.50 rating, Install button.
- convert yaml to properties** by zhang min, 11.3K downloads, 1.90 rating, Install button.
- JSON-YAML-XML Converter** by Syncro Soft, 2.6K downloads, 3.57 rating, Paid status, Install button.
- Format-converter** by Dmitrii\_Priporov, 596 downloads, 4.47 rating, Install button.

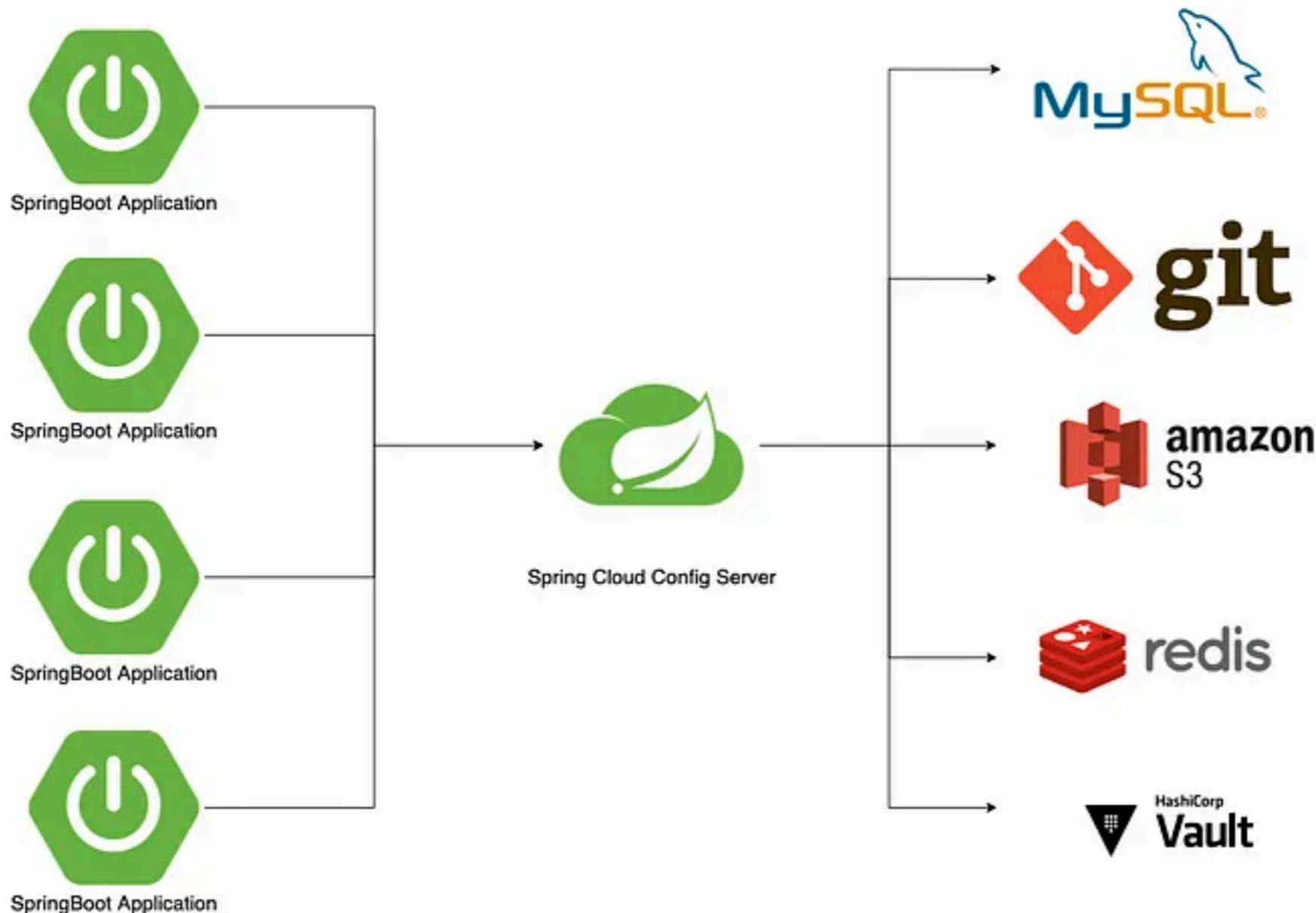
A second red dashed box highlights the details page for the 'Convert YAML and Properties File' plugin. The page shows:

- Code Tools** category.
- Convert YAML and Properties File** by chencn, Plugin homepage link.
- Install** button and version 1.0.5.
- Reviews** and **Additional Info** sections.
- Overview** tab selected.
- A screenshot of the IntelliJ IDEA interface showing the plugin in action.

<https://plugins.jetbrains.com/plugin/13804-convert-yaml-and-properties-file>



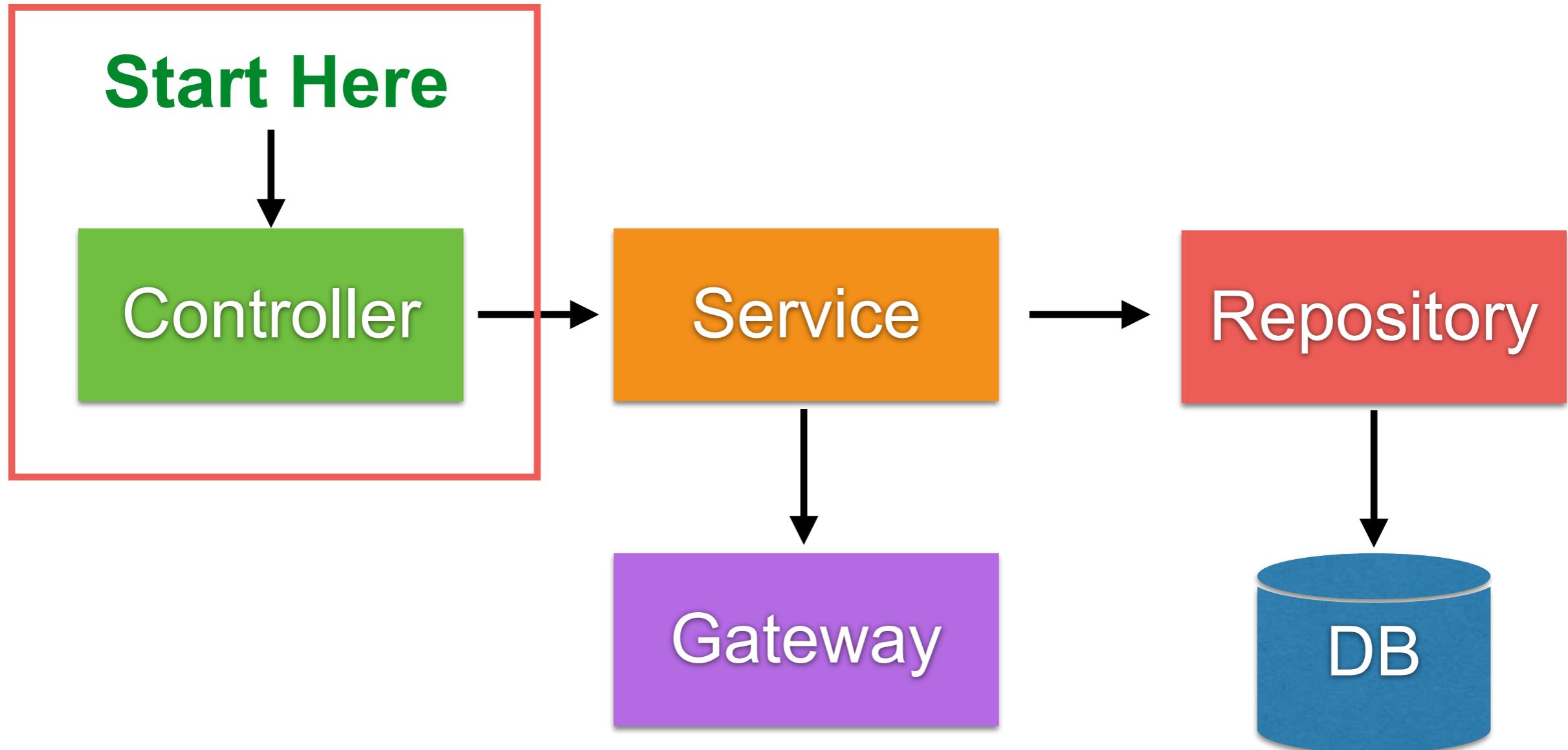
# Centralized Configuration



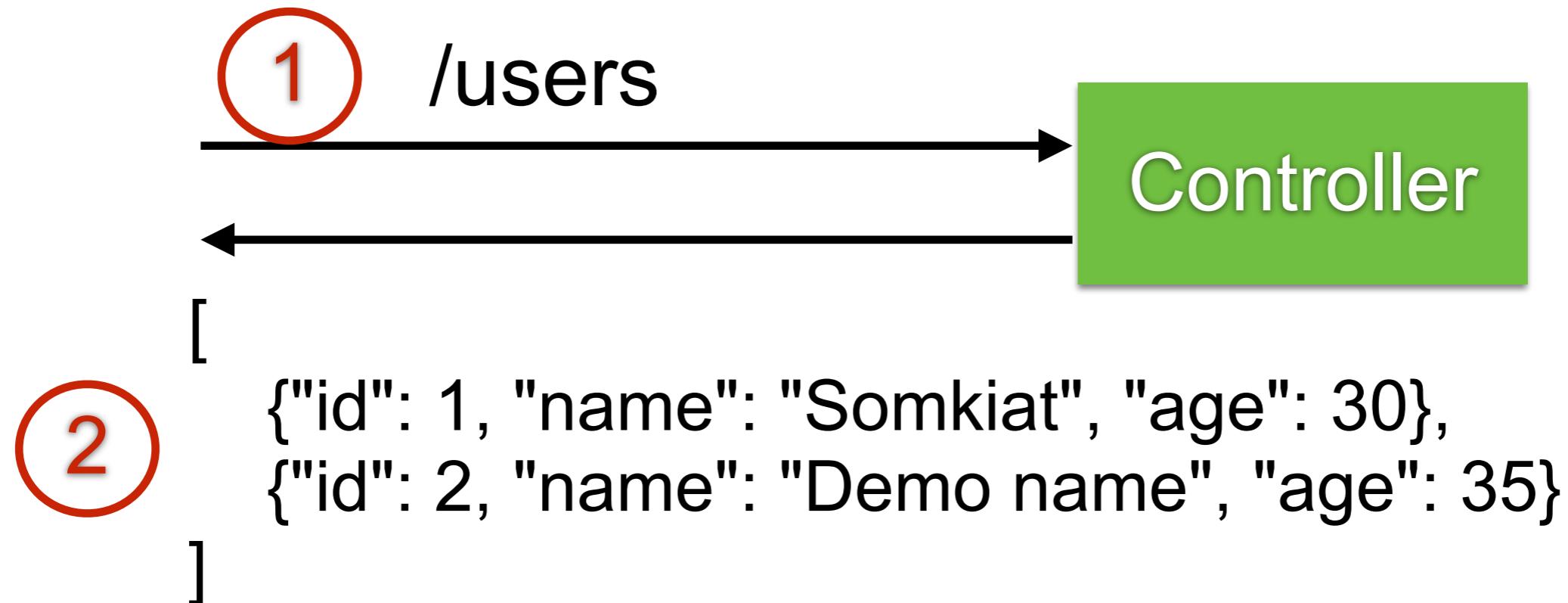
<https://spring.io/projects/spring-cloud-config>



# Basic structure of Spring Boot



# Get all users



# 1. Create REST Controller

## UserController.java

```
@RestController
public class UserController {

    @GetMapping("/users")
    public List<UserResponse> getAllUsers() {
        List<UserResponse> userResponseList = new ArrayList<>();
        userResponseList.add(new UserResponse(1, "demo 1", 30));
        userResponseList.add(new UserResponse(2, "demo 2", 35));
        return userResponseList;
    }
}
```



# 2. Create model class

## UserResponse.java

```
public class UserResponse {  
    private int id;  
    private String name;  
    private int age;  
  
    public UserResponse() {  
    }  
  
    public UserResponse(int id, String name, int age) {  
        this.id = id;  
        this.name = name;  
        this.age = age;  
    }  
}
```



# 3. Compile and Packaging

\$mvnw clean package



# 4. Run

```
$java -jar target/hello.jar
```



# 5. Open in browser

<http://localhost:8080/users>

```
[  
  {  
    "id": 1,  
    "name": "demo 1",  
    "age": 30  
  },  
  {  
    "id": 2,  
    "name": "demo 2",  
    "age": 35  
  }]  
]
```



# Create more APIs

Get user by id

Create a new user

Update user by id

Delete user by id



# Get user by id

GET /users/{id}

```
@GetMapping("/users/{id}")
public UserResponse getUserById(
    @PathVariable int id) {

    UserResponse userResponse = new UserResponse(id, "Demo", 40);
    return userResponse;
}
```



# Create UserResponse

**POJO**  
Setter/Getter

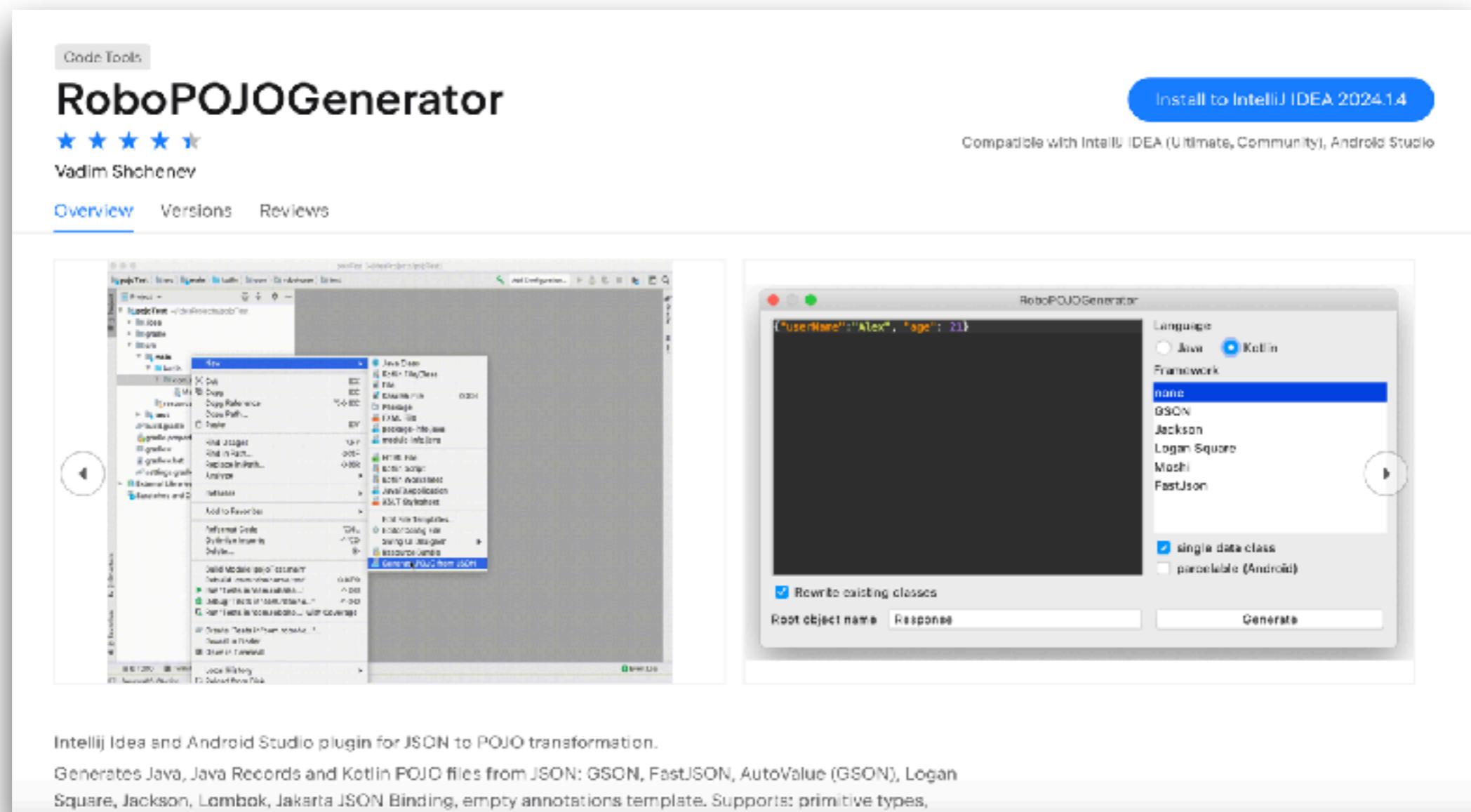
**Lombok**

**Record**  
Java 14+



# Convert JSON to POJO

## IntelliJ IDEA



<https://plugins.jetbrains.com/plugin/8634-robopojogenerator>

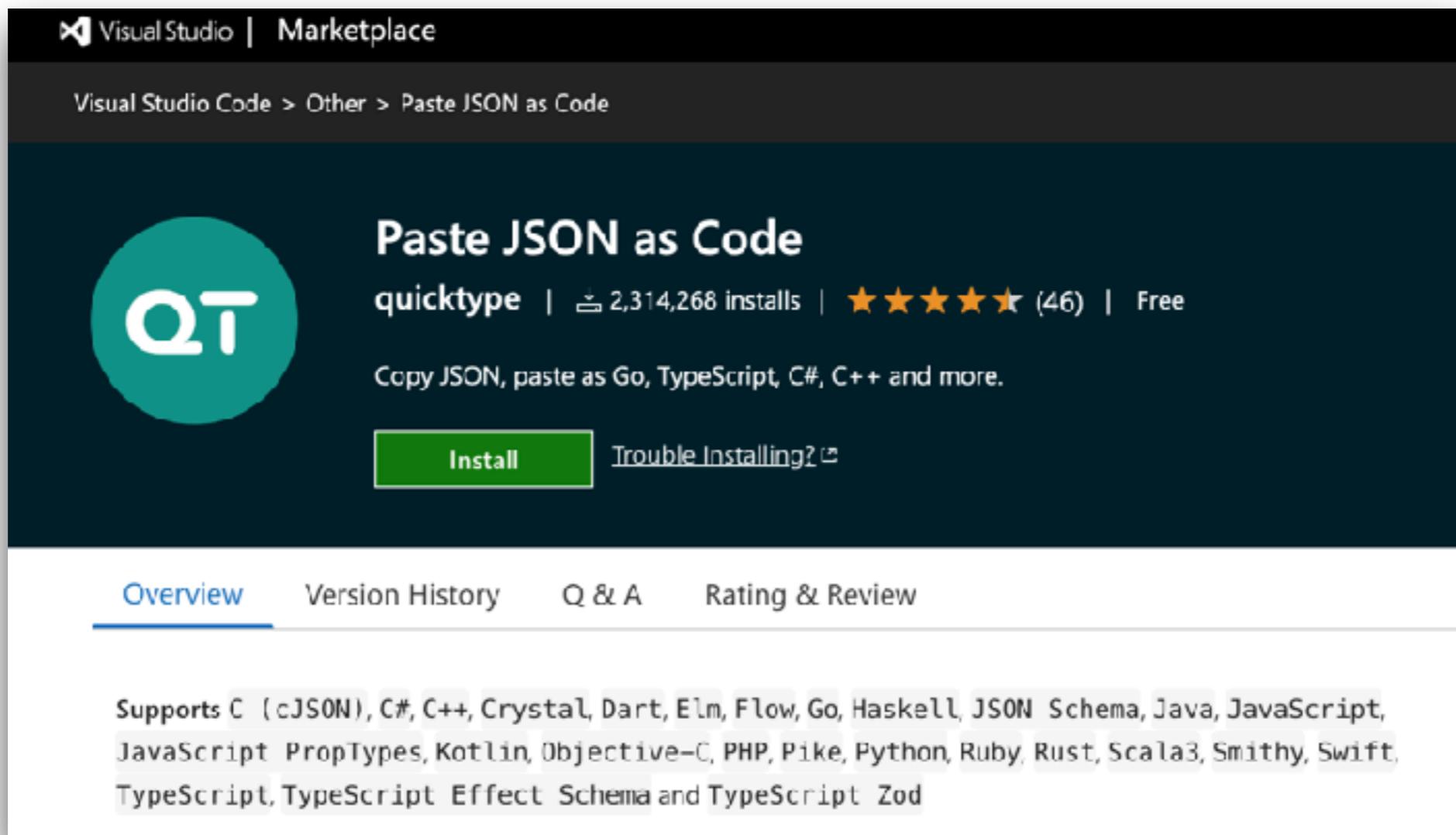


Workshop

© 2020 - 2024 Siam Chamnkit Company Limited. All rights reserved.

# Convert JSON to POJO

## VS Code



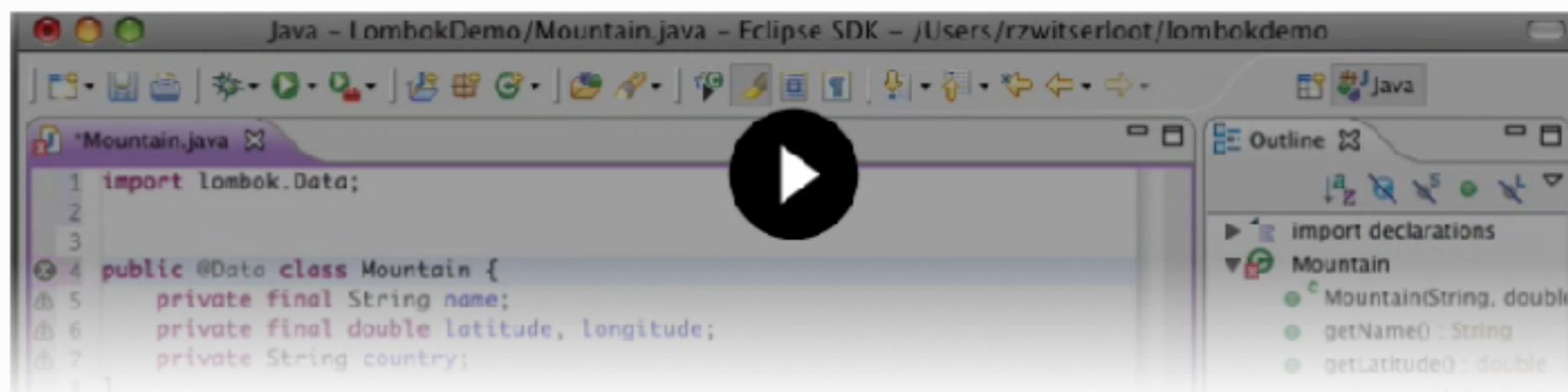
<https://marketplace.visualstudio.com/items?itemName=quicktype.quicktype>



# Using Lombok

## Project Lombok

Project Lombok is a Java library that automatically plugs into your editor and build tools, spicing up your Java. Never write another getter or equals method again, with one annotation your class has a fully featured builder, Automate your logging variables, and much more.



Project Lombok is **powered by:**

<https://projectlombok.org/>



Workshop

© 2020 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# IntelliJ IDEA plug-in

The screenshot shows the IntelliJ IDEA plugin marketplace interface. A red dashed box highlights the search results for 'lombok'. The first result, 'Lombok' by JetBrains s.r.o., is selected and has a checkmark next to its name. The right side of the screen displays the details for the 'Lombok' plugin, including its homepage, a 'Disable' button, and version information (232.9921.47). Below this, tabs for 'Overview', 'What's New', 'Reviews', and 'Additional Info' are visible. The 'Overview' tab is active, showing the title 'IntelliJ Lombok plugin' and a description: 'A plugin that adds first-class support for Project Lombok Features'. A bulleted list of features is provided:

- `@Getter` and `@Setter`
- `@FieldNameConstants`
- `@ToString`
- `@EqualsAndHashCode`
- `@AllArgsConstructor`, `@RequiredArgsConstructor` and `@NoArgsConstructor`

<https://plugins.jetbrains.com/plugin/6317-lombok>



# Use Lombok

```
@Data  
@Builder  
@NoArgsConstructor  
@AllArgsConstructor  
public class UserRequest {  
    private String email;  
    private String password;  
}
```



# Create more controller ..



# Create a new user

POST /users

```
@PostMapping("/users")
public UserResponse createNewUser(
    @RequestBody UserRequest newUser) {
    UserResponse newUserResponse = new UserResponse(
        1,
        newUser.getName(),
        newUser.getAge());
    return newUserResponse;
}
```



# Update user by id

PUT /users/{id}

```
@PutMapping("/users/{id}")
public UserResponse updateUser(@RequestBody UserRequest newUser,
                               @PathVariable int id) {
    // TODO
    // 1. find by id
    // 2. found => update user
    // 3. not found => ?? (create ? or throw error)
    UserResponse updatedUserResponse = new UserResponse(
        id,
        newUser.getName(),
        newUser.getAge());
    return updatedUserResponse;
}
```



# Delete user by id

DELETE /users/{id}

```
@DeleteMapping("/users/{id}")
public void deleteUser(@PathVariable int id) {
    // TODO
}
```



# API Documentation



# Swagger/ Open API

The screenshot shows the official Swagger website. At the top, there's a navigation bar with links for Learn, Tools, Resources, a search bar, Sign In, and Try Free. Below the header, a large green banner reads "API Development for Everyone". Underneath, a sub-section says "Simplify your API development with our open-source and professional tools, built to help you and your team efficiently design and document APIs at scale." To the right, a sidebar lists various API resources and methods: GET, POST, PUT, PATCH, DELETE, OPTIONS, and HEAD, each with a corresponding color-coded button.

API Development for Everyone

Simplify your API development with our open-source and professional tools, built to help you and your team efficiently design and document APIs at scale.

Find your tool    Read the docs →

TRUSTED BY

Microsoft    NATIONAL GEOGRAPHIC    ZUORA

API Resources

GET /example

POST /example

PUT /example

PATCH /example

DELETE /example

OPTIONS /example

HEAD /example

<https://swagger.io/>



# Swagger + Spring Boot 3

## Spring Doc



<https://springdoc.org/>



# **How to test UserController ?**



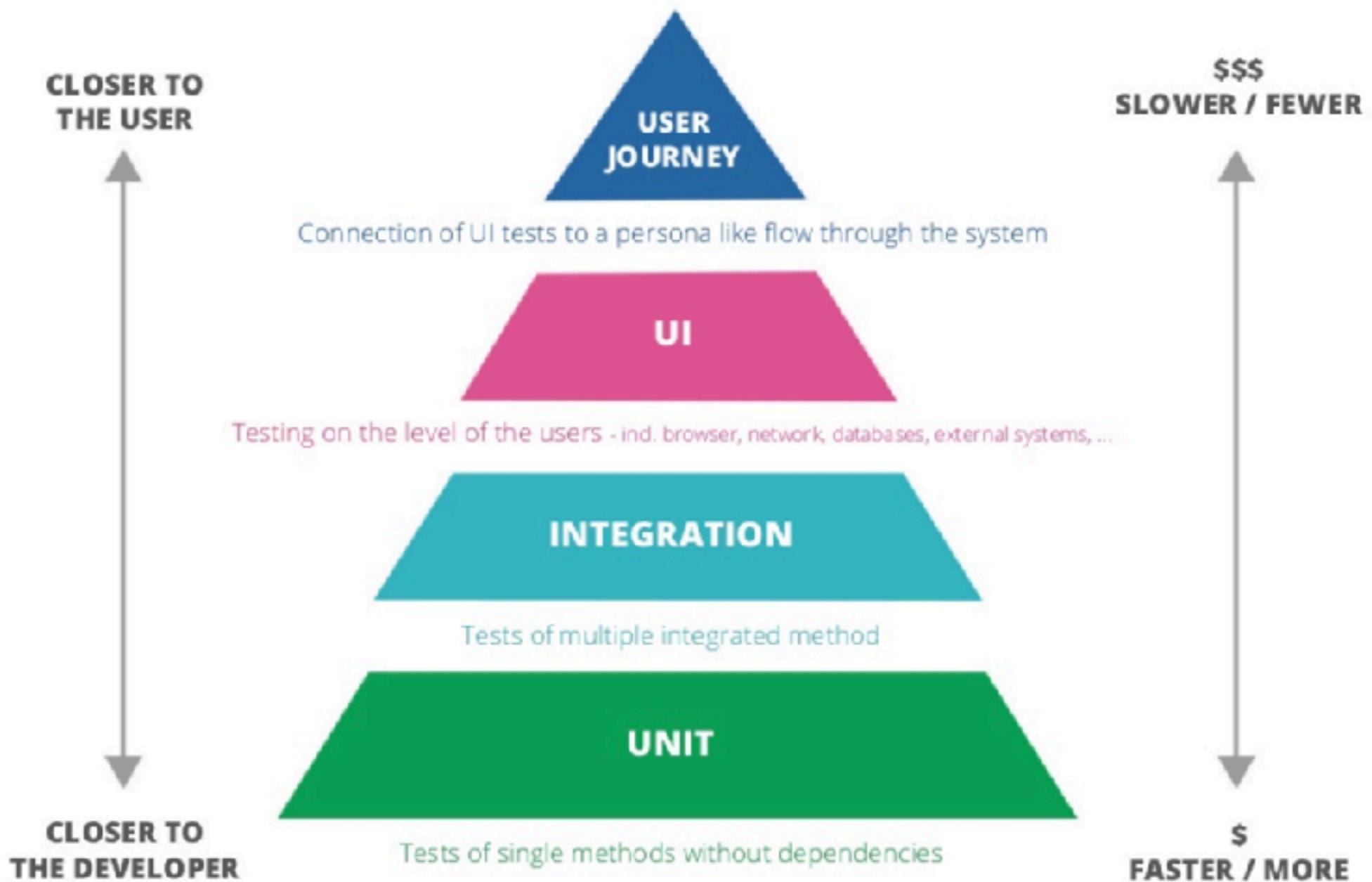
# Postman



POSTMAN

<https://www.postman.com/downloads/>





# Controller testing

How to testing with Spring Boot ?

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

<https://docs.spring.io/spring-boot/docs/current/reference/html/features.html#features.testing>



# Spring Boot Testing

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



<https://docs.spring.io/spring-boot/docs/current/reference/html/features.html#features.testing>



# Testing in Spring Boot

**@SpringBootTest**  
**@WebMVCTest**  
**@JsonTest**  
**@DataJpaTest**  
**@RestClientTest**



# Testing in Spring Boot

@SpringBootTest

@WebMVC Test

@JsonTest

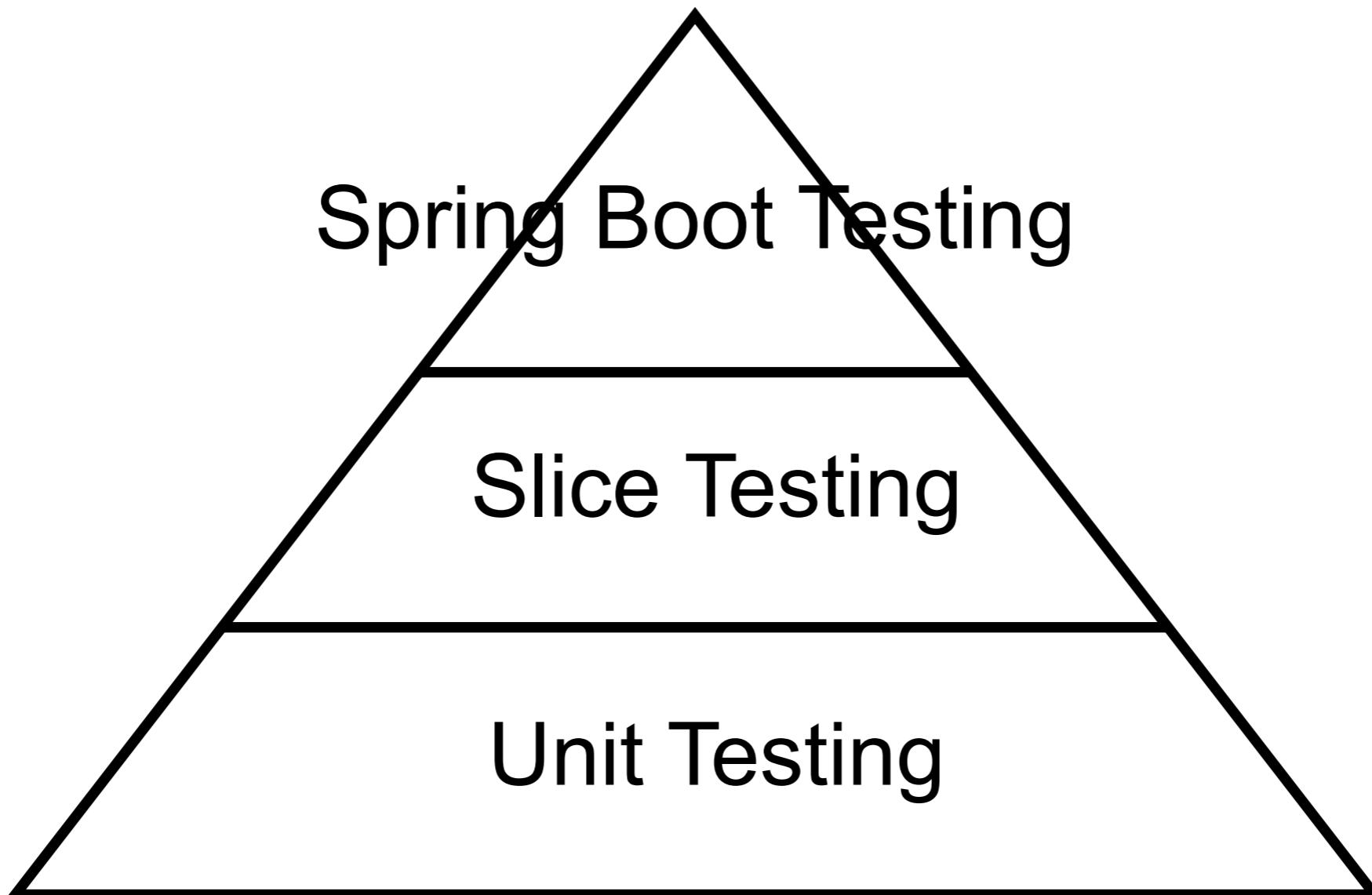
@DataJpaTest

@RestClientTest

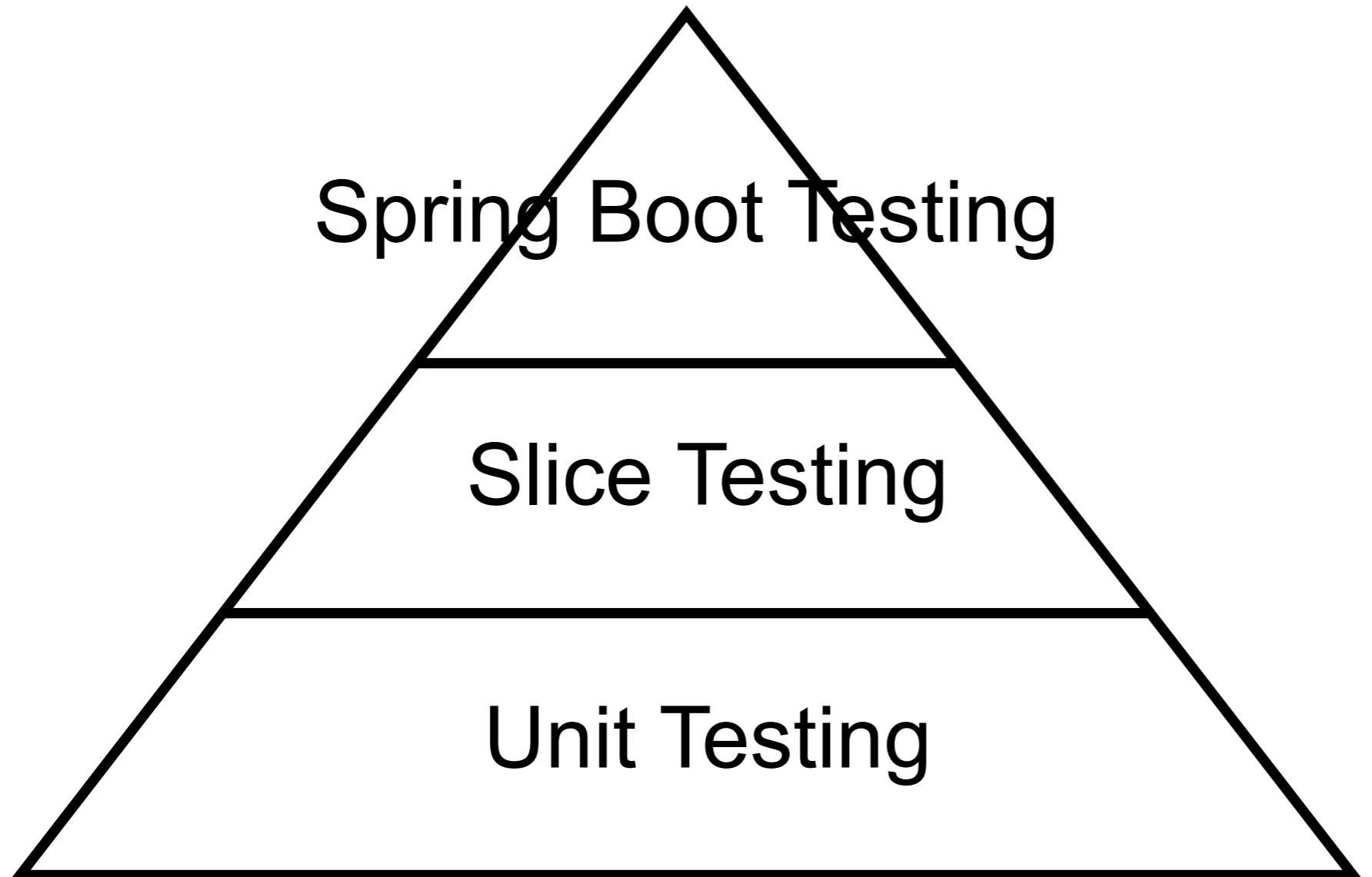
**Slice testing**



# Testing in Spring Boot



# Testing in Spring Boot



# Controller testing

1. Spring Boot Testing
2. Slice Testing with MockMvc
3. Unit Testing



# 1. SpringBootTest

## Application context

Controller Advice

Controllers

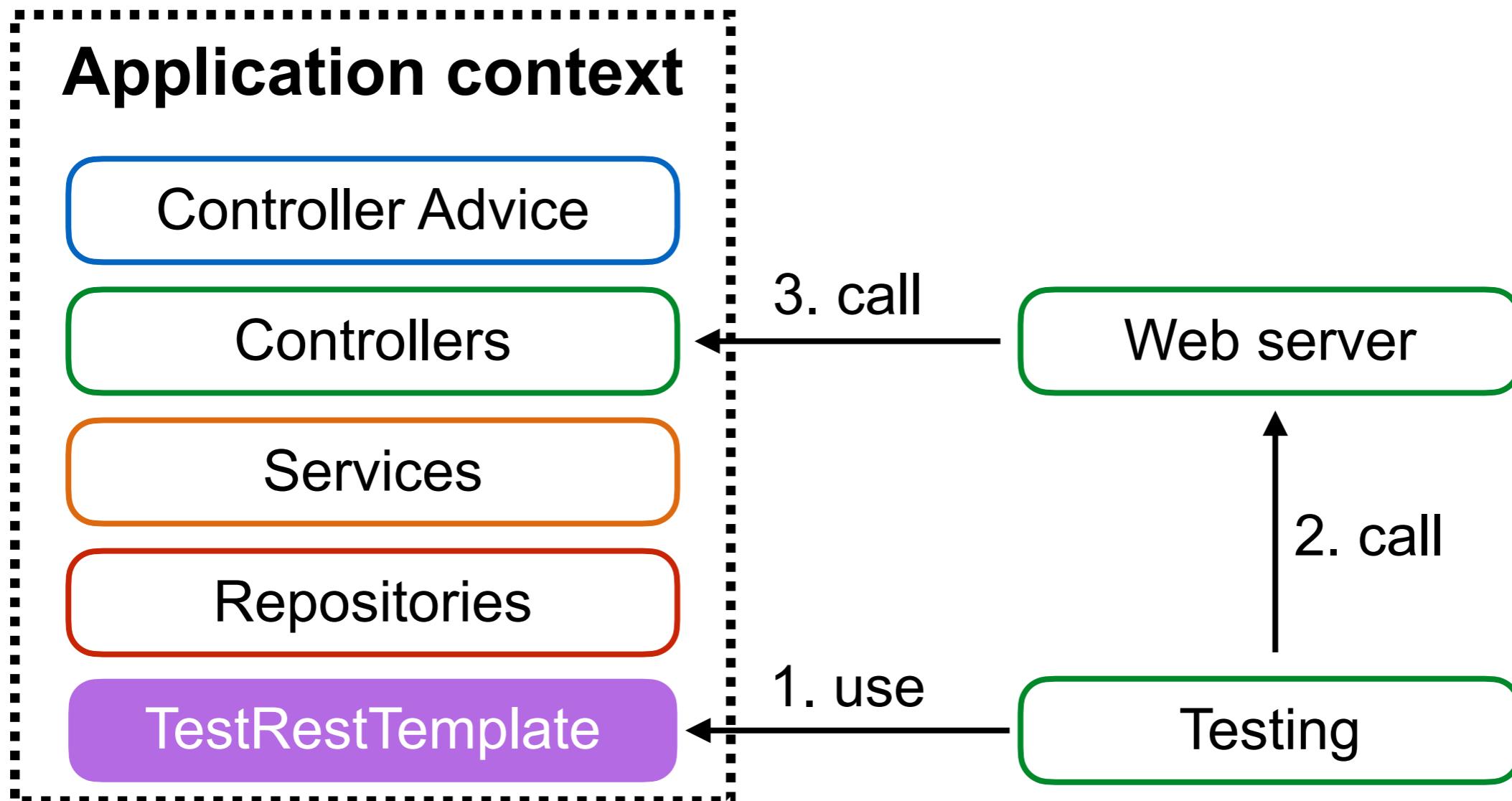
Services

Repositories

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#features.testing>



# 1. SpringBootTest



# SpringBootTest #1

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment
        = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class HelloControllerTest {

    @Autowired
    private TestRestTemplate testRestTemplate;

    @Test
    public void sayHi() {
        // Action :: Call controller
        Hello actualResult
            = testRestTemplate.getForObject("/hello/somkiat",
                                            Hello.class);

        // Assertion :: Check result with expected result
        assertEquals("Hello, somkiat", actualResult.getMessage());
    }
}
```



# SpringBootTest #2

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment
        = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class HelloControllerTest {

    @Autowired
    private TestRestTemplate testRestTemplate;

    @Test
    public void sayHi() {
        // Action :: Call controller
        Hello actualResult
            = testRestTemplate.getForObject("/hello/somkiat",
                Hello.class);

        // Assertion :: Check result with expected result
        assertEquals("Hello, somkiat", actualResult.getMessage());
    }
}
```



# SpringBootTest #3

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment
        = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class HelloControllerTest {

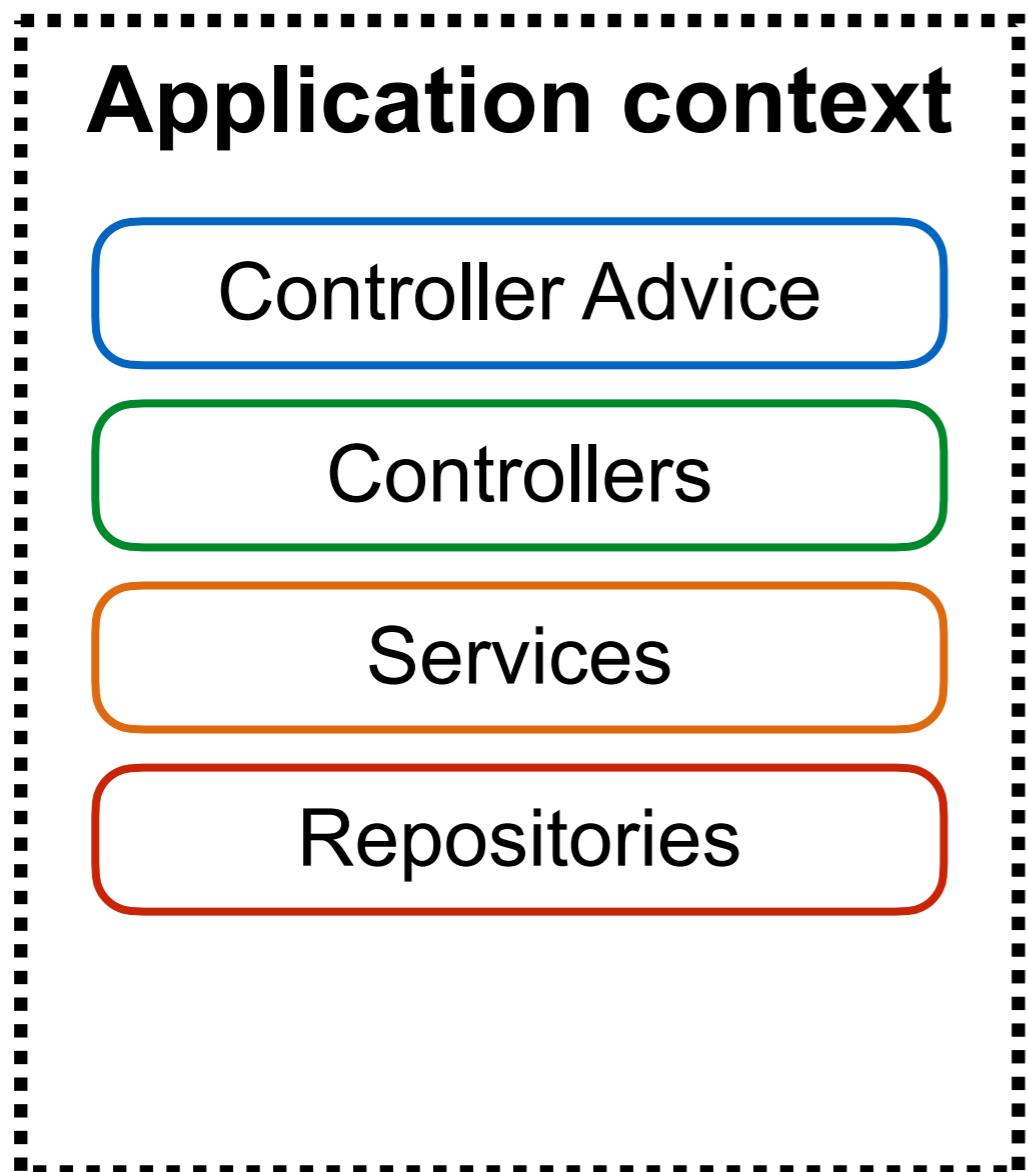
    @Autowired
    private TestRestTemplate testRestTemplate;

    @Test
    public void sayHi() {
        // Action :: Call controller
        Hello actualResult
                = testRestTemplate.getForObject("/hello/somkiat",
                                                Hello.class);

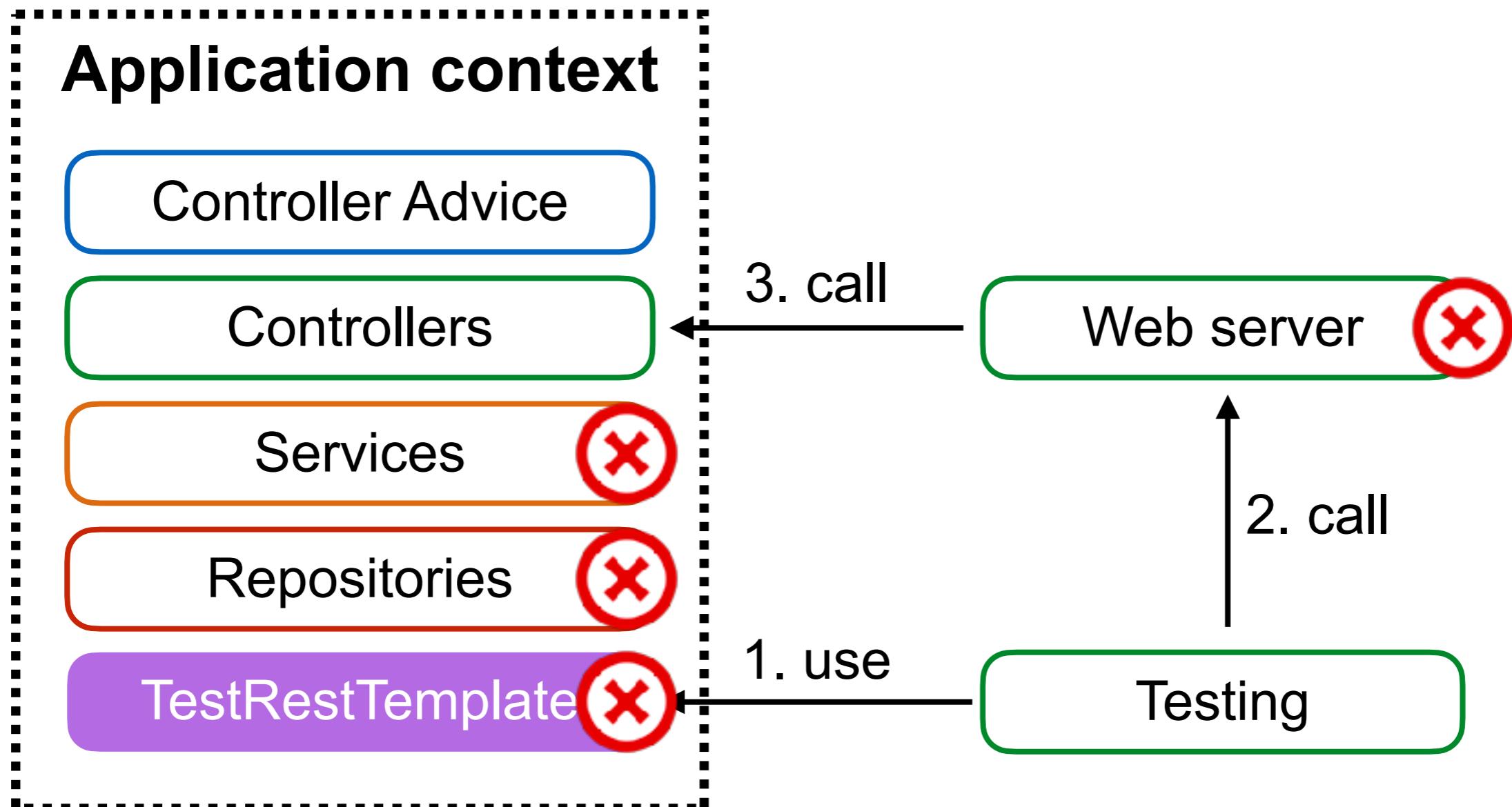
        // Assertion :: Check result with expected result
        assertEquals("Hello, somkiat", actualResult.getMessage());
    }
}
```



# 2. Slice Testing with MockMVC



# 2. Slice Testing with MockMVC



# MockMvcTest #1

```
@RunWith(SpringRunner.class)
@WebMvcTest(NumberController.class)
public class NumberControllerMockMvcTest {

    @MockBean
    private MyRandom stubRandom;

    @Autowired
    private MockMvc mvc;

    @Test
    public void success() throws Exception {
        NumberControllerResponse expected
            = new NumberControllerResponse("5555");

        // Stub
        given(stubRandom.nextInt(10)).willReturn(5555);
    }
}
```



# MockMvcTest #2

```
@RunWith(SpringRunner.class)
@WebMvcTest(NumberController.class)
public class NumberControllerMockMvcTest {

    @MockBean
    private MyRandom stubRandom;

    @Autowired
    private MockMvc mvc;

    @Test
    public void success() throws Exception {
        NumberControllerResponse expected
            = new NumberControllerResponse("5555");

        // Stub
        given(stubRandom.nextInt(10)).willReturn(5555);

        // ...
    }
}
```



# MockMvcTest #3

## Use ObjectMapper to convert JSON to object

```
// Call API HTTP response code = 200
String response =
    this.mvc.perform(get("/number"))
    .andExpect(status().isOk())
    .andReturn()
    .getResponse().getContentAsString();

// Convert JSON message to Object
ObjectMapper mapper = new ObjectMapper();
NumberControllerResponse actual =
    mapper.readValue(response,
        NumberControllerResponse.class);
```



# 3. Unit testing with Controller



# Unit test

Use Test Double

In java, use Mockito library



<http://site.mockito.org/>



Workshop

© 2020 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Unit testing with Mockito #1

```
@ExtendWith(MockitoExtension.class)
public class NumberControllerUnitTest {

    @Mock
    private MyRandom stubRandom;

    @Test
    public void success() throws Exception {
        NumberControllerResponse expected
            = new NumberControllerResponse("5555");

        // Stub
        given(stubRandom.nextInt(10)).willReturn(5555);
    }
}
```



# Unit testing with Mockito #2

```
@ExtendWith(MockitoExtension.class)
public class NumberControllerUnitTest {
    @Mock
    private MyRandom stubRandom;

    @Test
    public void success() throws Exception {
        NumberControllerResponse expected
            = new NumberControllerResponse("5555");

        // Stub
        given(stubRandom.nextInt(10)).willReturn(5555);
    }
}
```



# Unit testing with Mockito #3

```
@Test
public void success() throws Exception {
    NumberControllerResponse expected
        = new NumberControllerResponse("5555");

    // Stub
    given(stubRandom.nextInt(10)).willReturn(5555);

    // Call
    NumberController controller = new NumberController(stubRandom);
    NumberControllerResponse actual = controller.randomNumber();

    // Assert
    assertEquals("5555", actual.getValue());
    assertEquals(expected, actual);
}
```



# Error handling



# Working with Error Responses

**ErrorResponse**

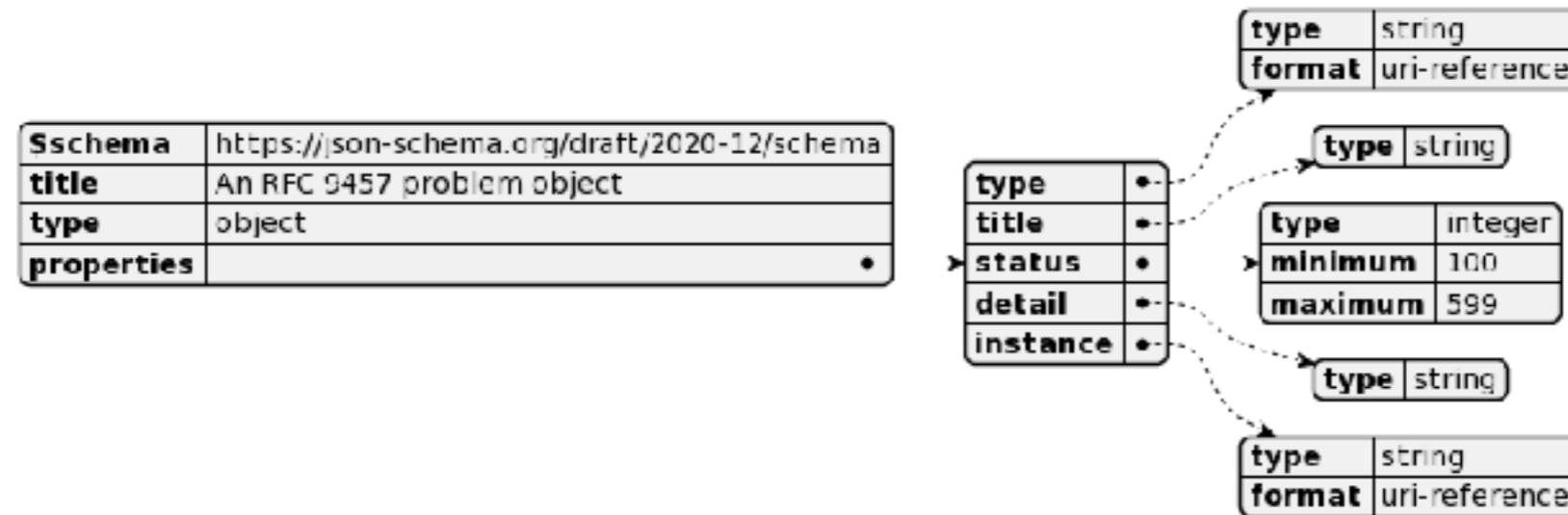
**ProblemDetail**

**Customization**

<https://docs.spring.io/spring-framework/reference/web/webmvc/mvc-ann-rest-exceptions.html>



# ProblemDetail



```
{  
  "type": "https://problems-registry.smartbear.com/missing-body-property",  
  "status": 400,  
  "title": "Missing body property",  
  "detail": "The request is missing an expected body property.",  
  "code": "400-09",  
  "instance": "/logs/registrations/d24b2953-ce05-488e-bf31-67de50d3d085",  
  "errors": [  
    {  
      "detail": "The body property {name} is required",  
      "pointer": "/name"  
    }  
  ]  
}
```

<https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/http/ProblemDetail.html>



# ProblemDetail

Enable ProblemDetail for error by default

## application.properties

```
spring.mvc.problemdetails.enabled=true
```



# Example in ControllerAdvice

```
@RestControllerAdvice
class HelloControllerAdvice {

    @ExceptionHandler(InvalidInputException.class)
    public ProblemDetail handleInvalidInputException(
        InvalidInputException e, WebRequest request) {

        ProblemDetail problemDetail
            = ProblemDetail
                .forStatusAndDetail(HttpStatus.BAD_REQUEST, e.getMessage());
        problemDetail.setInstance(URI.create("demo-instance"));
        problemDetail.setTitle("demo");

        return problemDetail;
    }
}
```



# Error handling

```
@Service
public class UserService {

    private AccountRepository accountRepository;

    @Autowired
    public UserService(AccountRepository accountRepository) {
        this.accountRepository = accountRepository;
    }

    public Account getAccount(int id) {
        Optional<Account> account = accountRepository.findById(id);
        if(account.isPresent()) {
            return account.get();
        }
        throw new MyAccountNotFoundException(
            String.format("Account id=[%d] not found", id));
    }
}
```



# MyAccountNotFoundException

```
public class MyAccountNotFoundException  
    extends RuntimeException {
```

```
    public MyAccountNotFoundException(String message) {  
        super(message);  
    }
```

```
}
```



# Response Status

404 = Not Found

Status	Description
400	Request body doesn't meet API spec
401	Authentication/Authorization fail
403	User can't perform the operation
404	Resource does not exist
405	Unsupported operation
500	Error on server



# Handling error in Spring Boot

```
@RestControllerAdvice  
public class AccountControllerHandler {  
  
    @ExceptionHandler(MyAccountNotFoundException.class)  
    public ResponseEntity<ExceptionResponse> accountNotFound(  
        MyAccountNotFoundException exception) {  
  
        ExceptionResponse response =  
            new ExceptionResponse(exception.getMessage(),  
                "More detail");  
  
        return new ResponseEntity<ExceptionResponse>(response,  
            HttpStatus.NOT_FOUND);  
    }  
}
```



# Handling error in Spring Boot

```
@RestControllerAdvice  
public class AccountControllerHandler {  
  
    @ExceptionHandler(MyAccountNotFoundException.class)  
    public ResponseEntity<ExceptionResponse> accountNotFound(  
        MyAccountNotFoundException exception) {  
  
        ExceptionResponse response =  
            new ExceptionResponse(exception.getMessage(),  
                "More detail");  
  
        return new ResponseEntity<ExceptionResponse>(response,  
            HttpStatus.NOT_FOUND);  
    }  
}
```

2



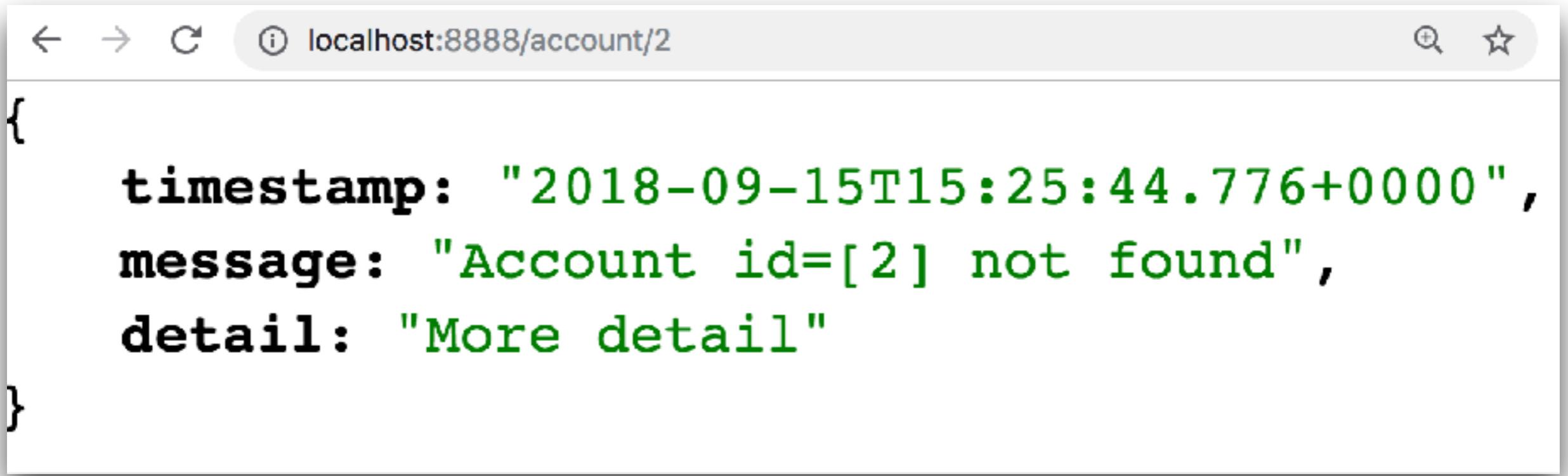
# ExceptionResponse

Response format of error

```
public class ExceptionResponse{  
  
    private Date timestamp = new Date();  
    private String message;  
    private String detail;  
  
    public ExceptionResponse(String message, String detail) {  
        this.message = message;  
        this.detail = detail;  
    }  
}
```



# Result of API



A screenshot of a web browser window. The address bar shows the URL `localhost:8888/account/2`. The main content area displays a JSON object:

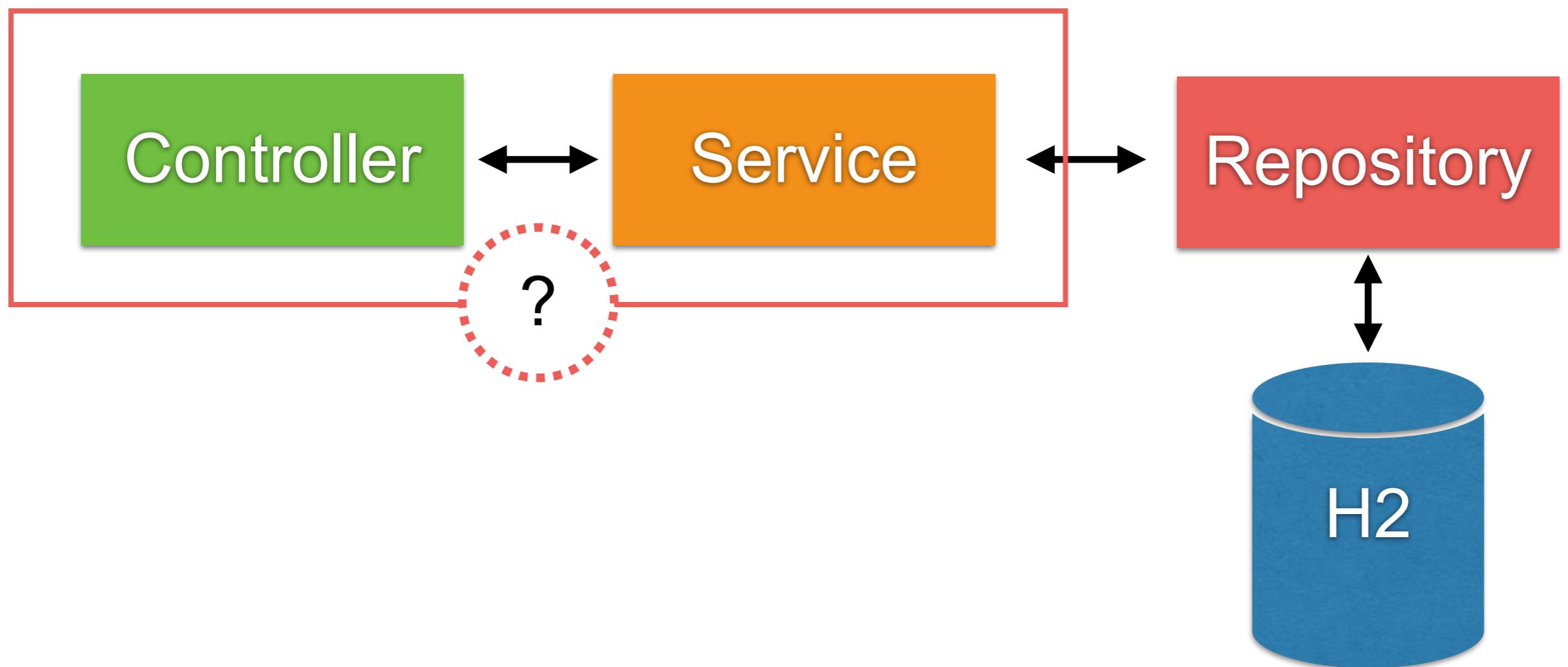
```
{  
  timestamp: "2018-09-15T15:25:44.776+0000",  
  message: "Account id=[ 2 ] not found",  
  detail: "More detail"  
}
```



# How to test ?



# How to test with Error/Exception ?



# Testing with WebMvcTest and MockMvc

Try to check data in response

```
@Test
public void getByIdWithNotFoundAccount() throws Exception {
    // Stub
    given(userService.getAccount(2))
        .willThrow(new MyAccountNotFoundException("Not found"));

    mockMvc.perform(
        get("/account/2")
            .accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isNotFound())
        .andExpect(content().contentType(MediaType.APPLICATION_JSON_UTF8))
        .andExpect(jsonPath("$.message", is("Not found")));
}
```

1



# Testing with WebMvcTest and MockMvc

Try to check data in response

```
@Test  
public void getByIdWithNotFoundAccount() throws Exception {  
    // Stub  
    given(userService.getAccount(2))  
        .willThrow(new MyAccountNotFoundException("Not found"));  
  
    mockMvc.perform(  
        get("/account/2")  
            .accept(MediaType.APPLICATION_JSON))  
        .andExpect(status().isNotFound())  
        .andExpect(content().contentType(MediaType.APPLICATION_JSON_UTF8))  
        .andExpect(jsonPath("$.message", is("Not found"))));  
}
```

2



# Testing with Service

Try to check when exception is thrown

```
@ExtendWith(MockitoExtension.class)
public class UserServiceTest {

    @Mock
    private UserRepository userRepository;

    @Test
    public void user_not_found_with_exception() {
        given(userRepository.findById(1))
            .willReturn(Optional.empty());
        UserService userService = new UserService();
        userService.setRepository(userRepository);

        Assertions.assertThrows(RuntimeException.class, () -> {
            userService.getData(1);
        });
    }
}
```



# Compile with testing

\$mvnw clean test

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



# Code coverage



**"Code coverage can show the high risk areas in a program, but never the risk-free."**

Paul Reilly, 2018, Kotlin TDD with Code Coverage



# Code coverage

A tool to measure how much of your code is covered by tests that break down into classes, methods and lines.



# Code coverage

But 100% of code coverage **does not mean** that your code is 100% correct



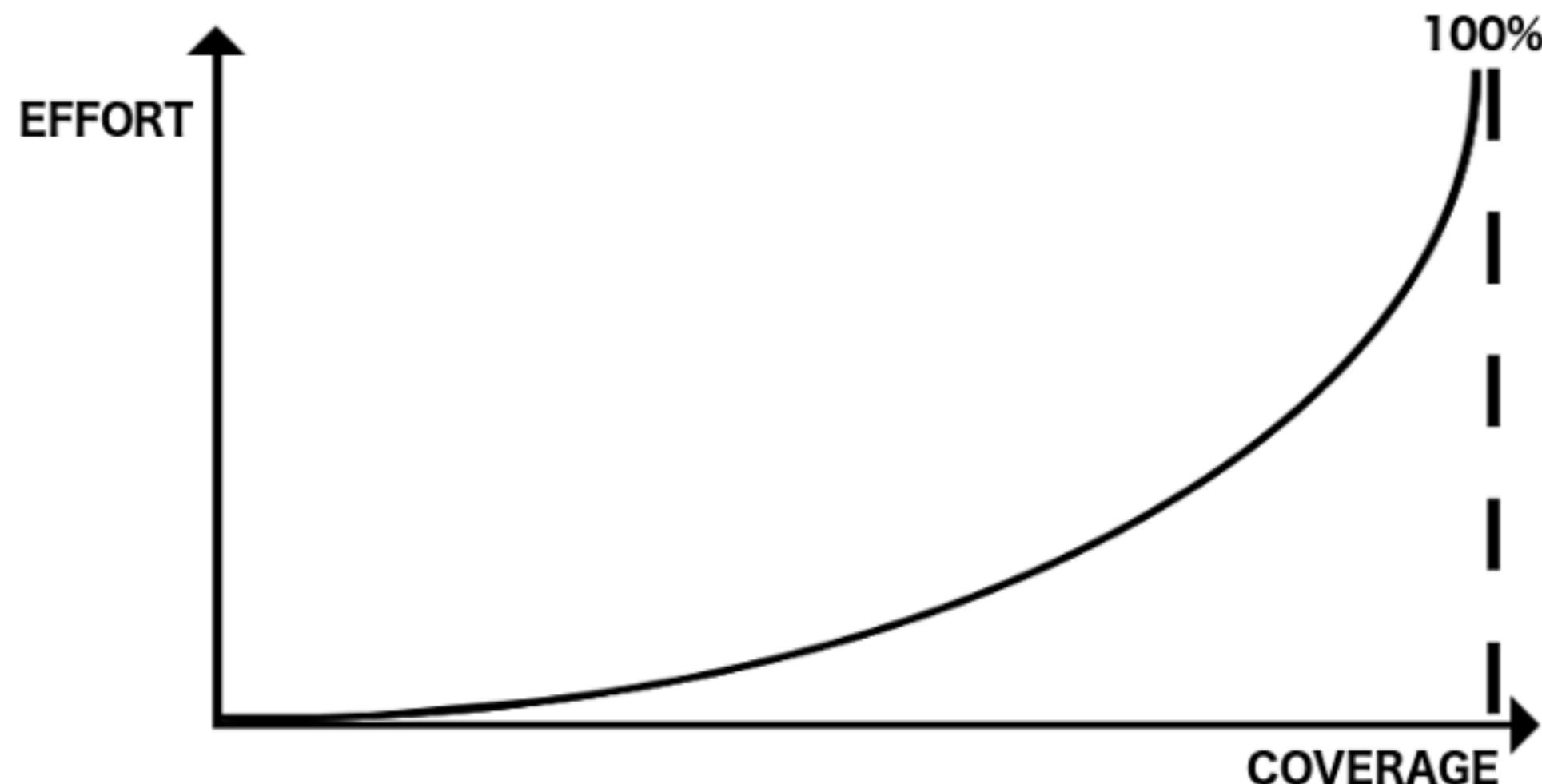
# Code coverage

Powerful tool to improve the quality of your code

**Code coverage != quality of tests**



# Code coverage 100% ?



# Code coverage with Java

Cobertura  
Jacoco

<http://bit.ly/2DIlsDeX>



# Run test again

\$mvnw 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.

---

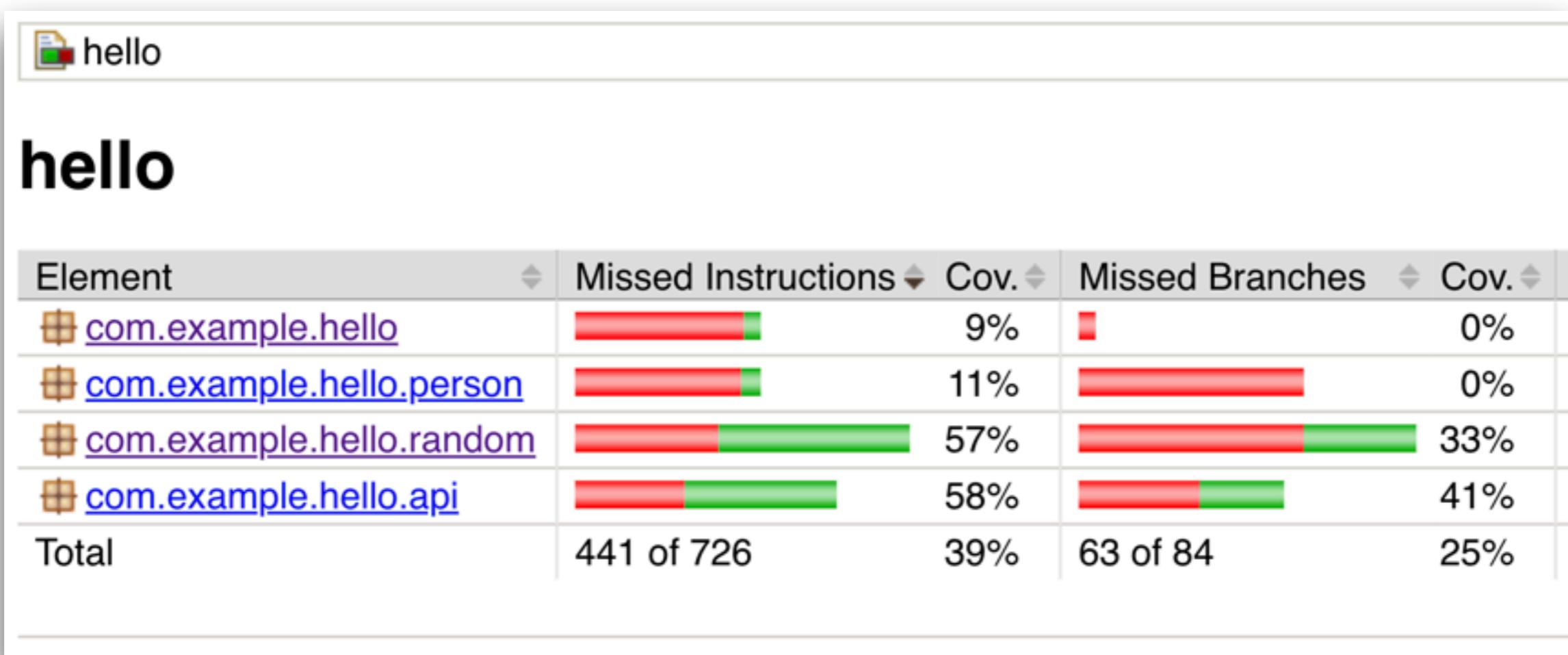
BUILD SUCCESS

---



# Coverage report

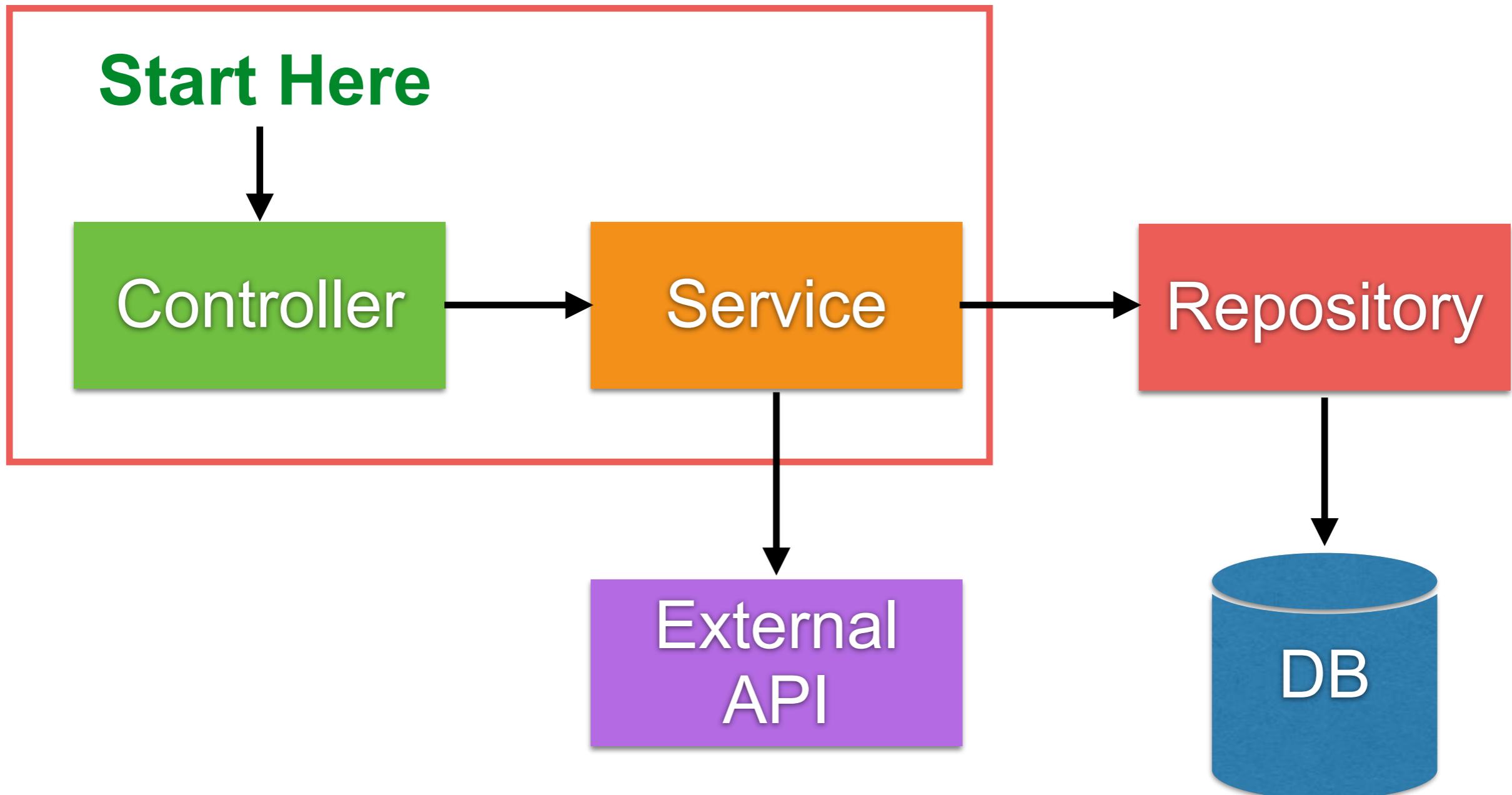
open target/site/jacoco/index.html



# **Move business logic to Service layer**

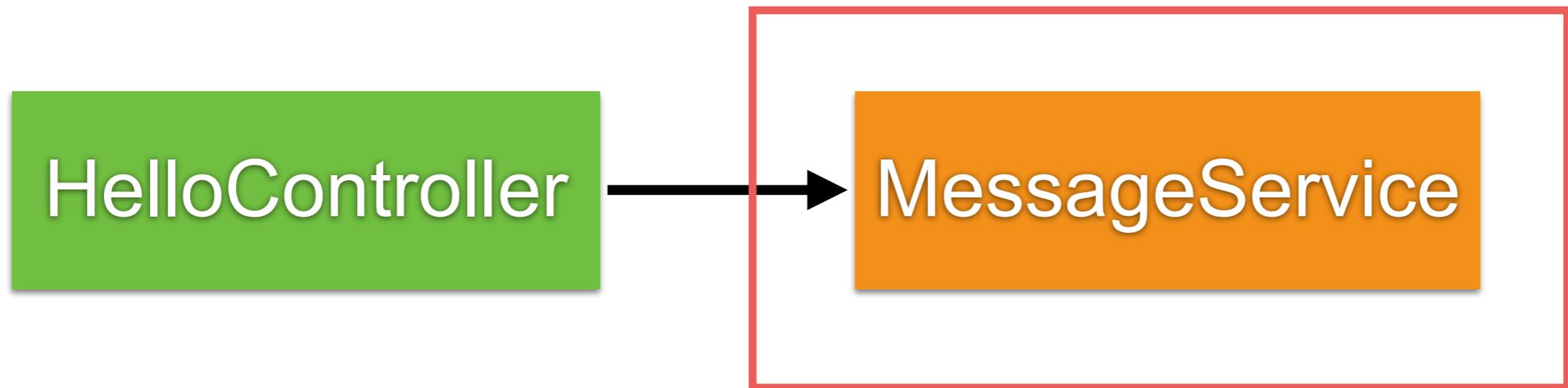


# Working with service



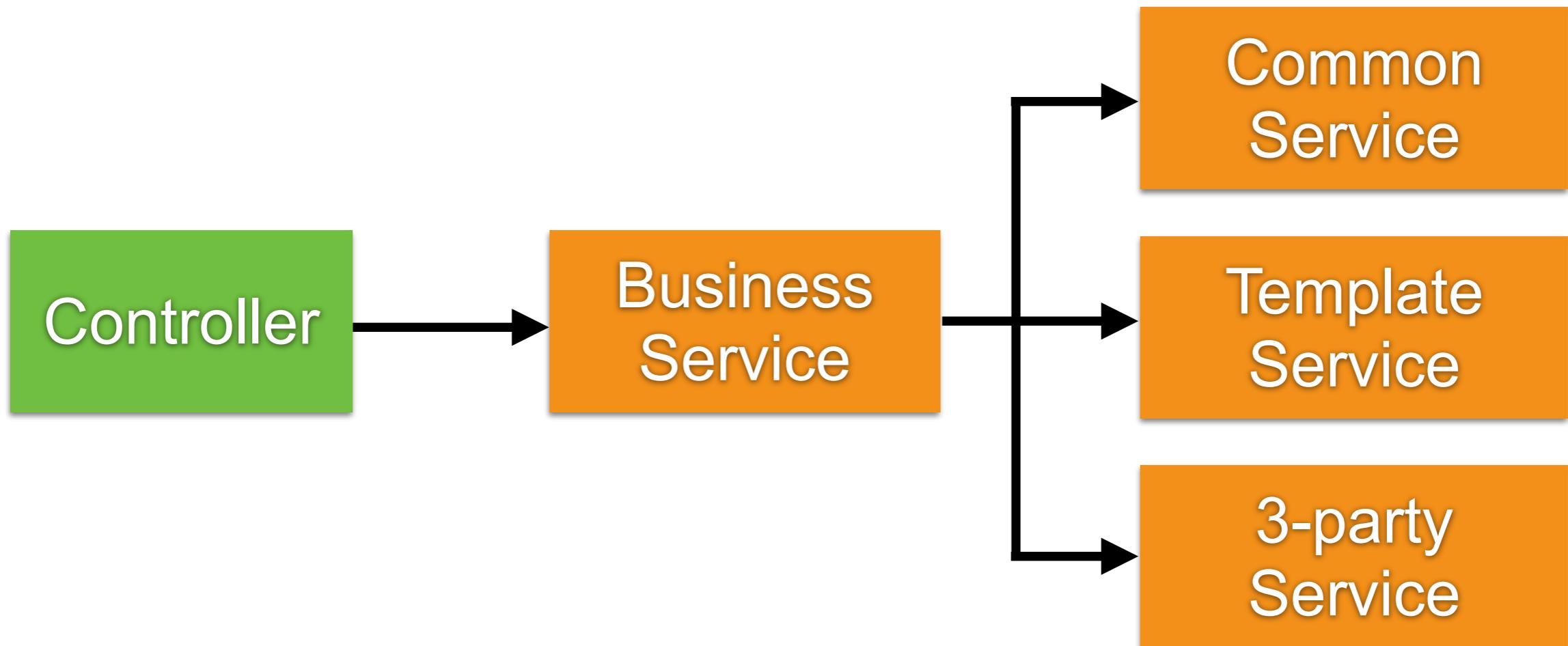
# Move business logic to service

Service class or interface ?

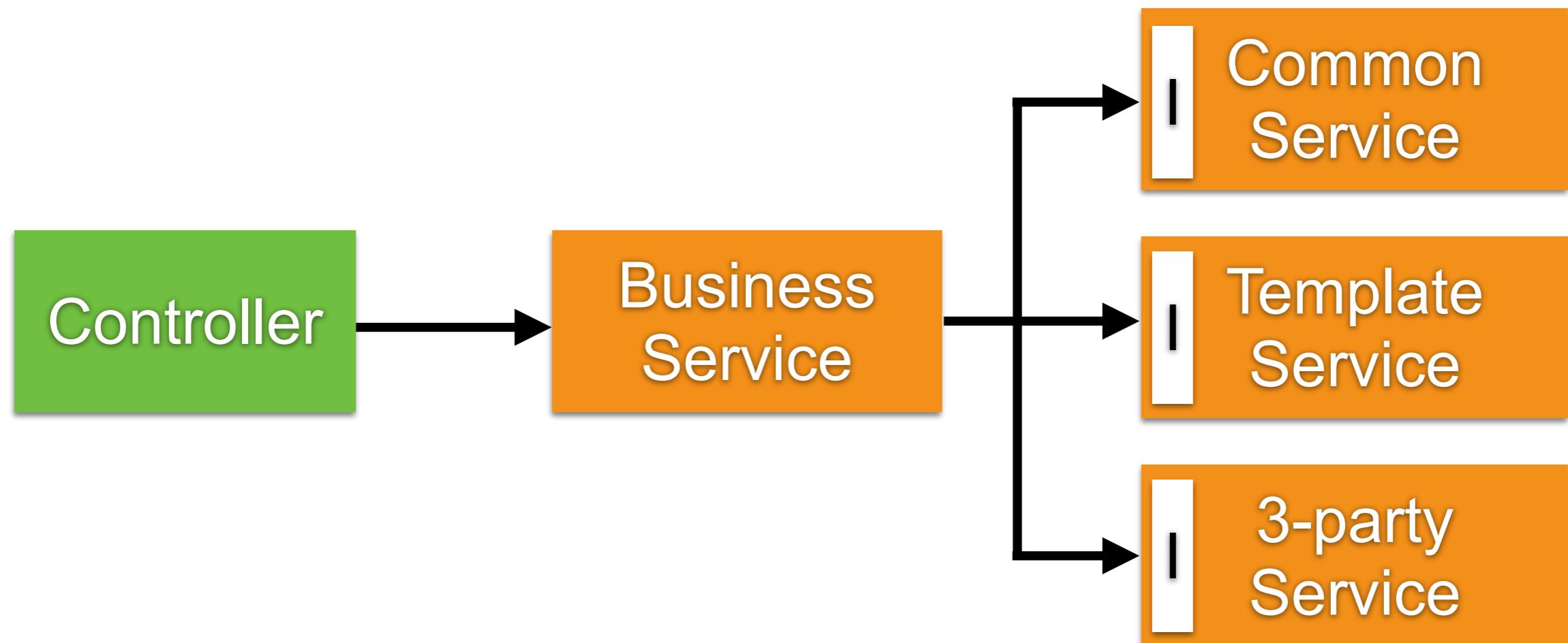


# Types of service

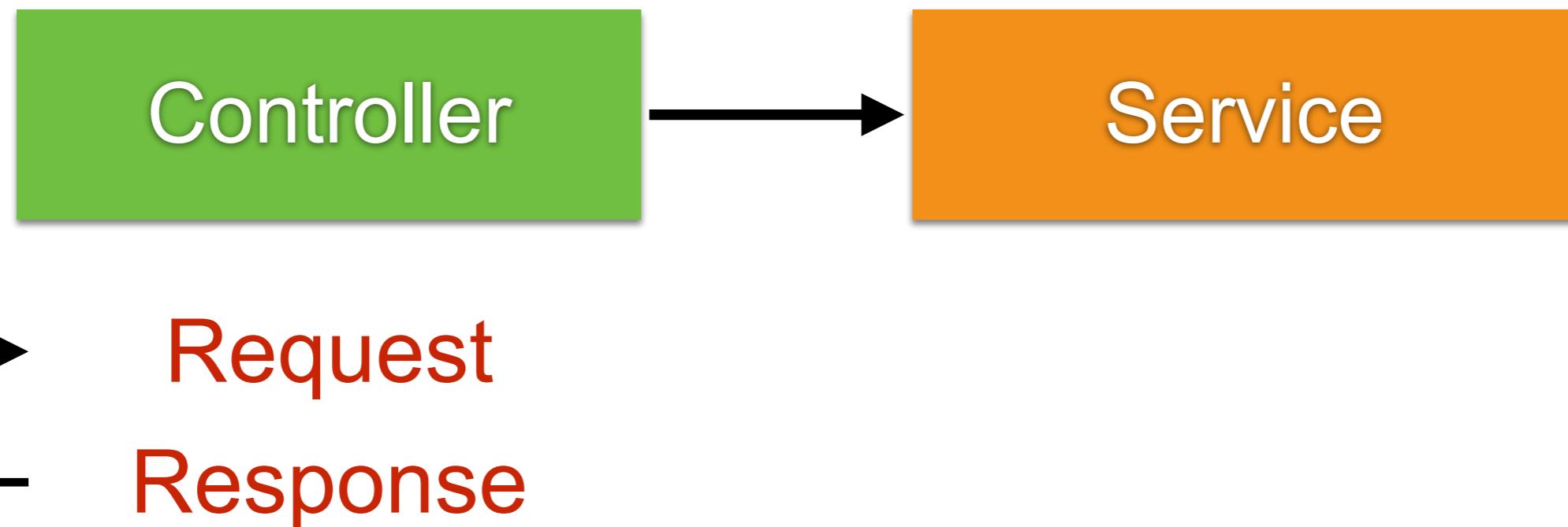
Business services (not reuse)  
Common/Template/3-party services



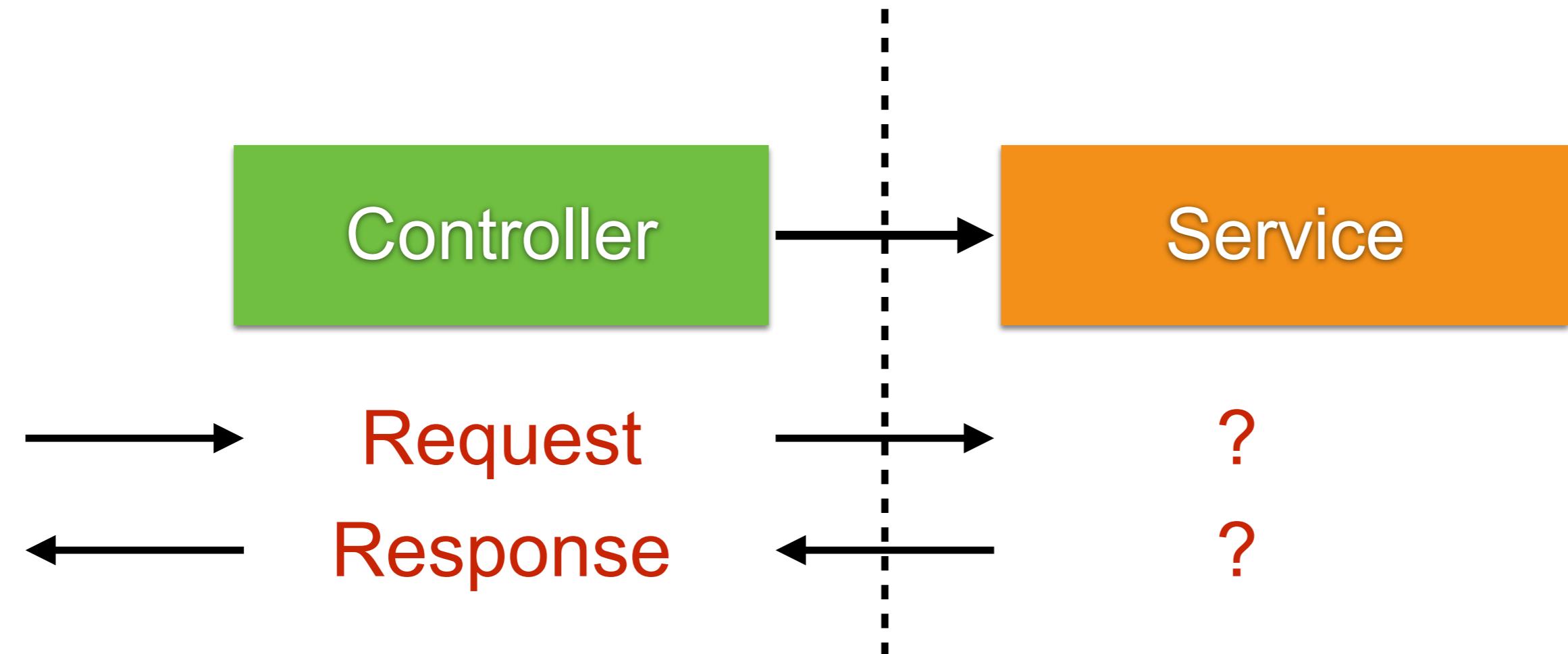
# Interface for common service



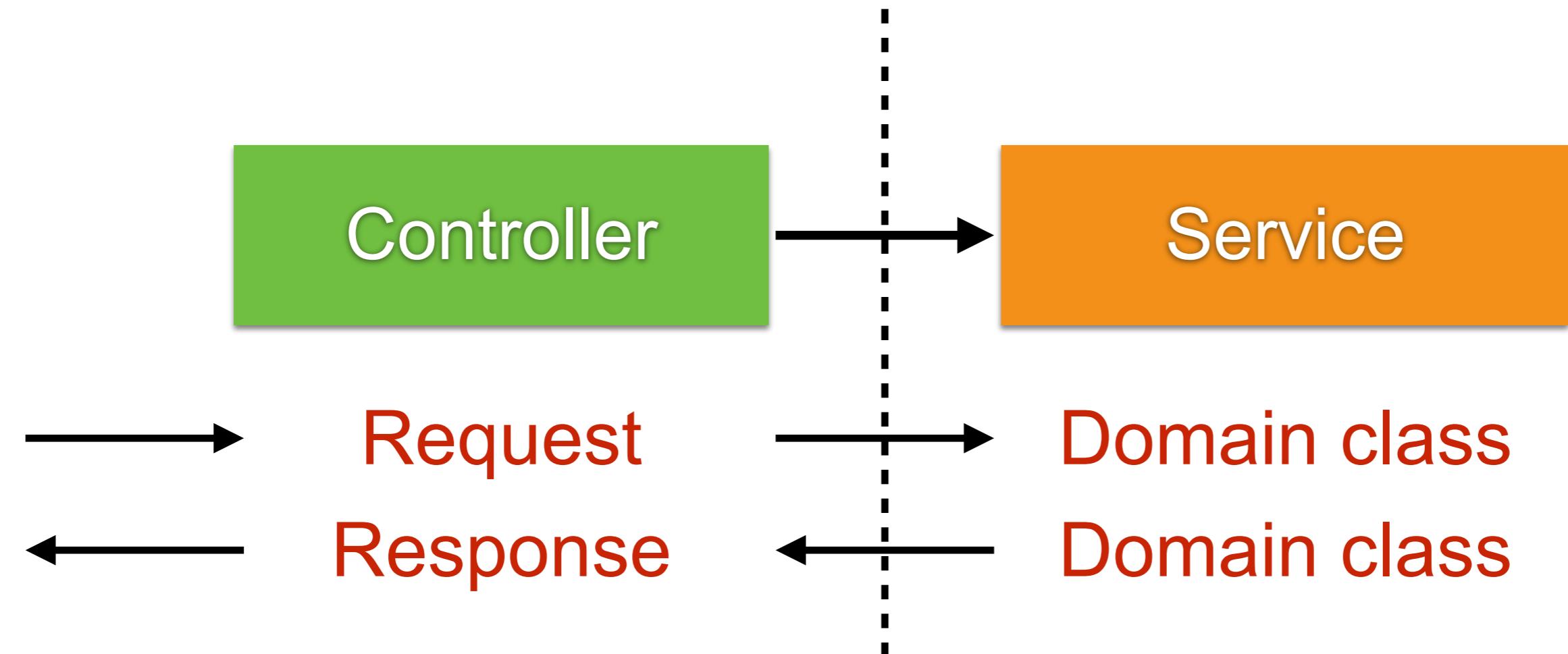
# Data Model for service ?



# Data Model ?

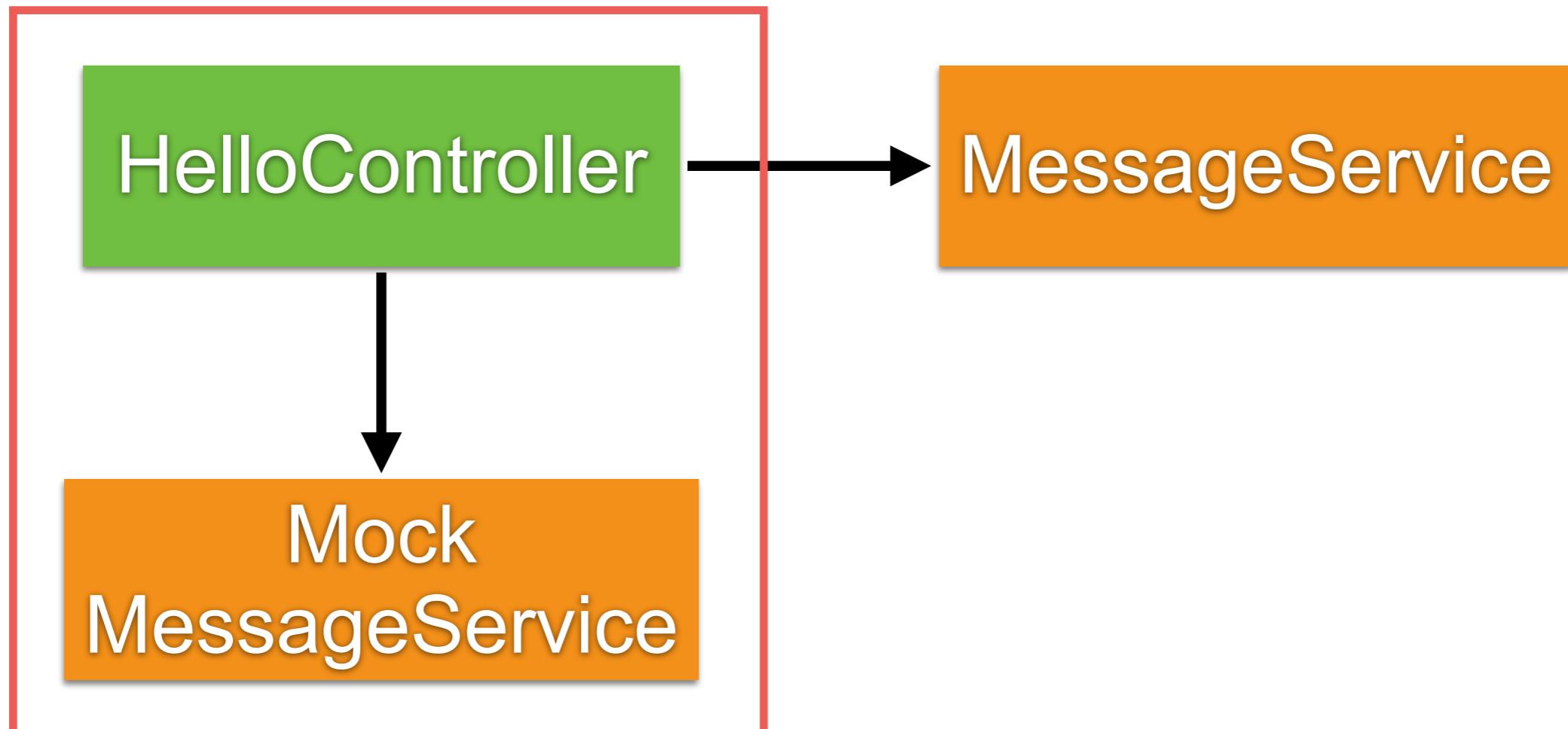


# Data Model ?



# Testing controller with service

Try to mocking service with Mockito



# Run all tests !!

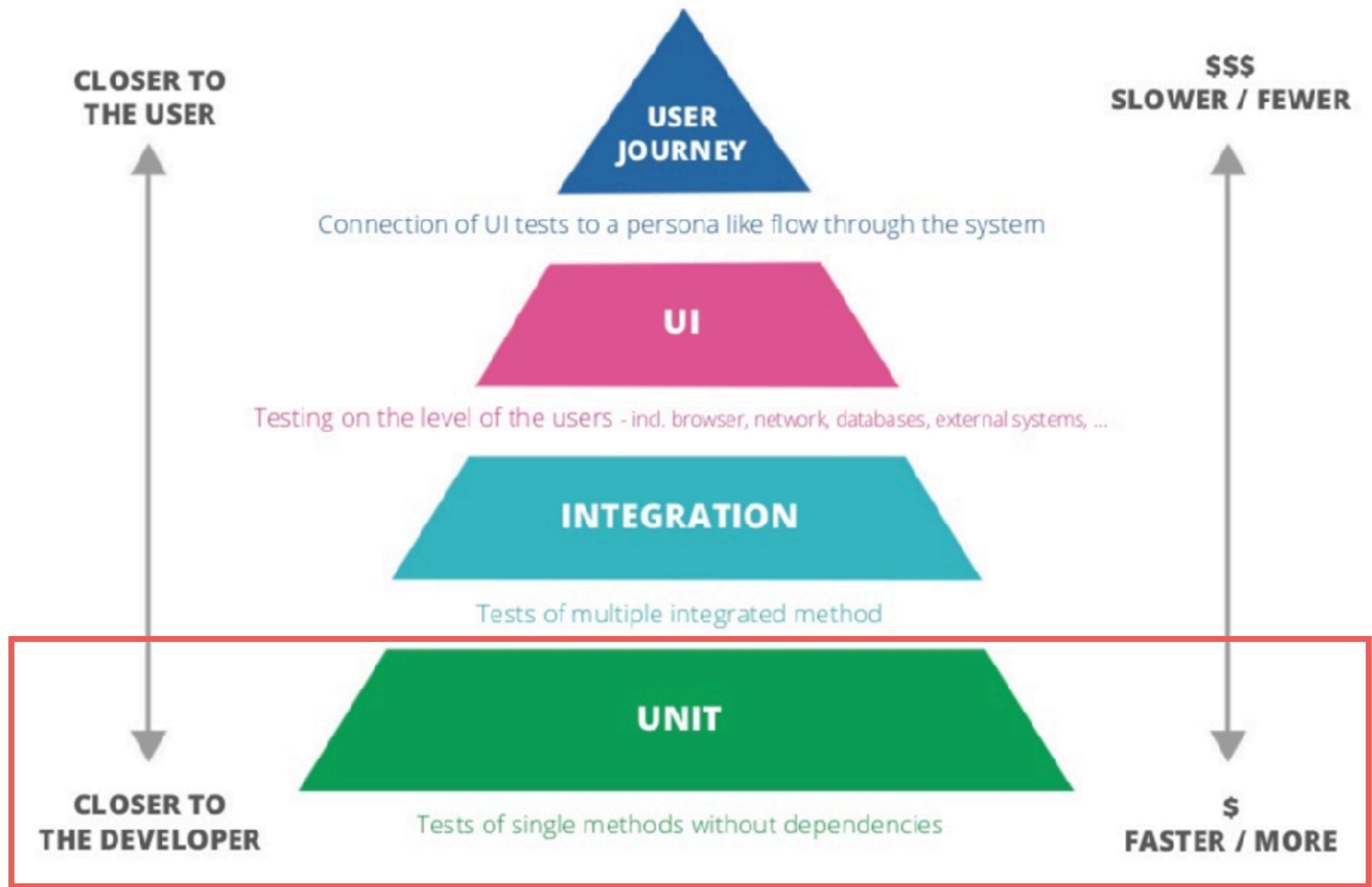
\$mvnw clean test

```
[INFO]
[INFO] Results:
[INFO]
[WARNING] Tests run: 10, Failures: 0, Errors: 0,
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 18.299 s
[INFO] Finished at: 2018-08-20T23:36:31+07:00
[INFO] -----
```

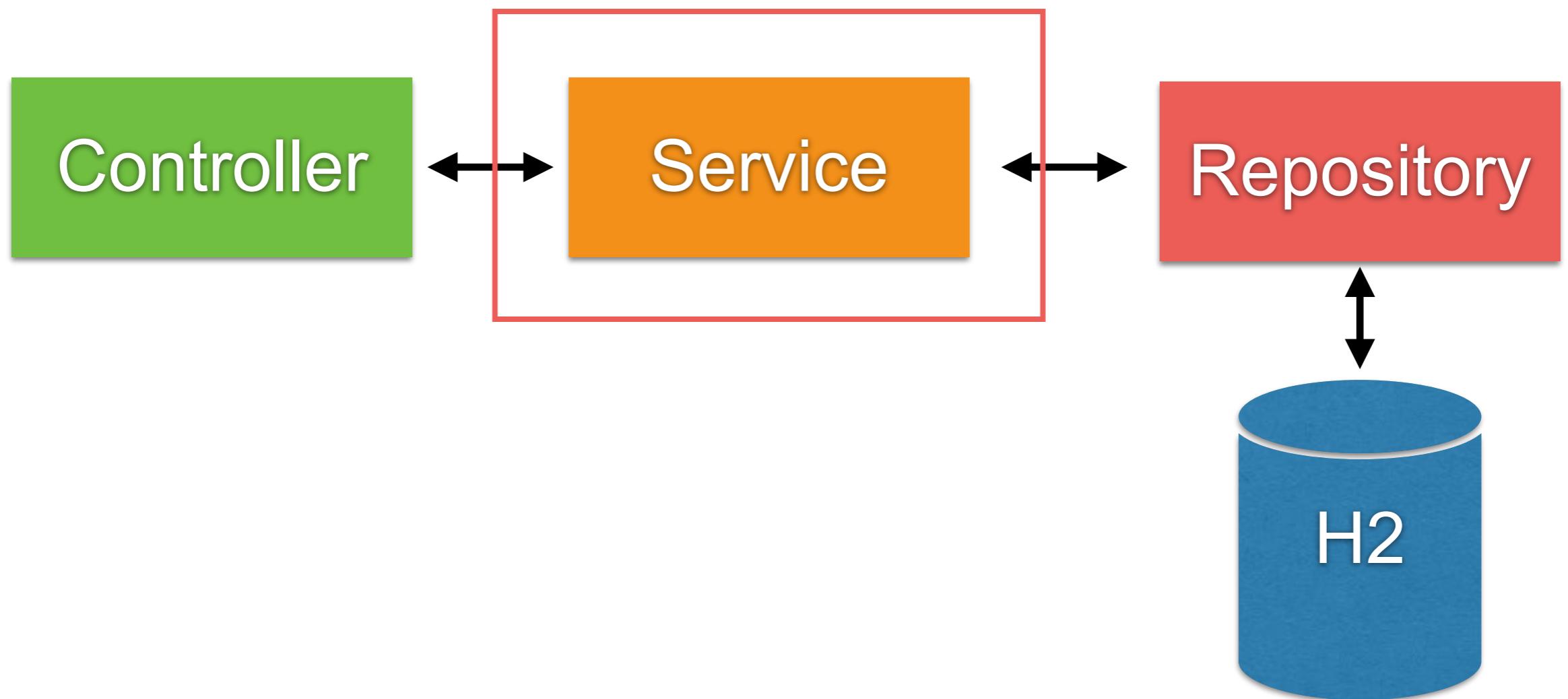


# **How to improve the speed of testing ?**





# Service Testing ?



# Service Testing

```
@ExtendWith(MockitoExtension.class)
public class UserServiceTest {

    @Mock
    private AccountRepository accountRepository;

    @Test
    public void getAccount() {
        // Stub
        Account account = new Account();
        account.setUserName("user");
        account.setPassword("pass");
        account.setSalary(1000);
        given(accountRepository.findById(1))
            .willReturn(Optional.of(account));

        UserService userService = new UserService(accountRepository);
        Account actualAccount = userService.getAccount(1);
        assertNotNull(actualAccount);
    }
}
```

1



# Service Testing

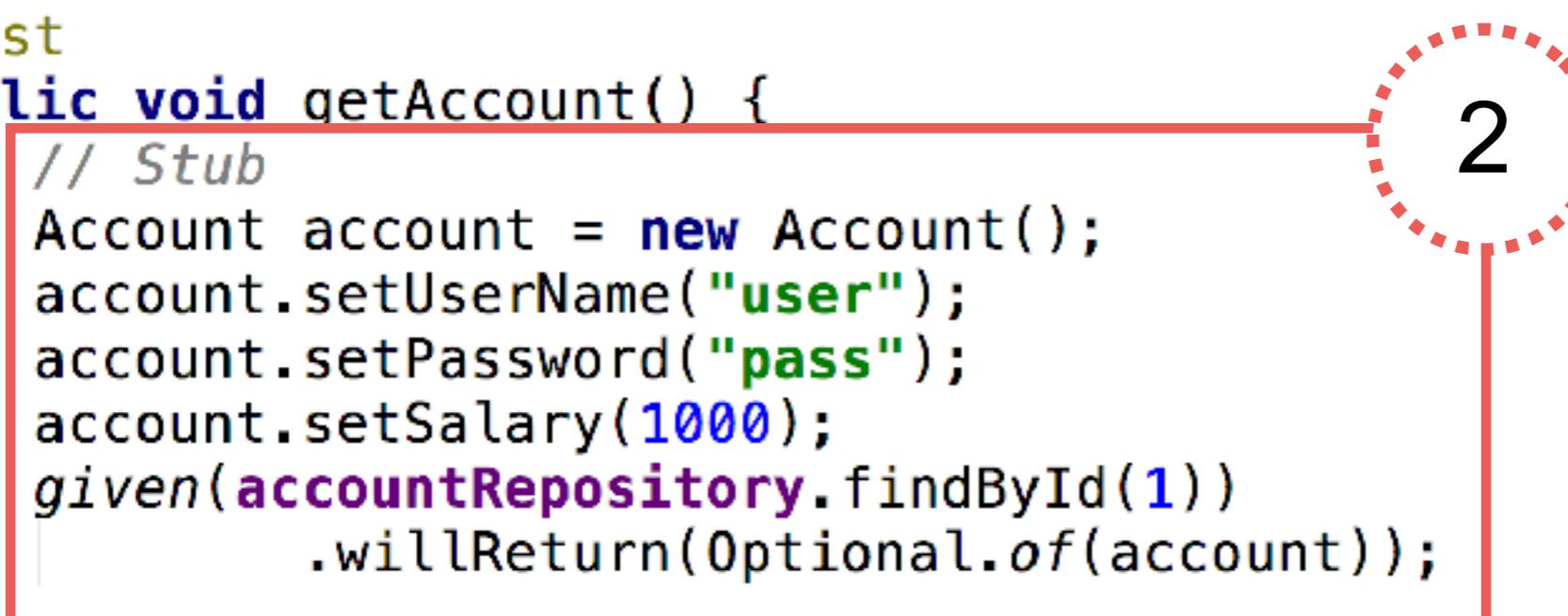
```
@ExtendWith(MockitoExtension.class)
public class UserServiceTest {

    @Mock
    private AccountRepository accountRepository;

    @Test
    public void getAccount() {
        // Stub
        Account account = new Account();
        account.setUserName("user");
        account.setPassword("pass");
        account.setSalary(1000);
        given(accountRepository.findById(1))
            .willReturn(Optional.of(account));
    }

    UserService userService = new UserService(accountRepository);
    Account actualAccount = userService.getAccount(1);
    assertNotNull(actualAccount);
}

}
```



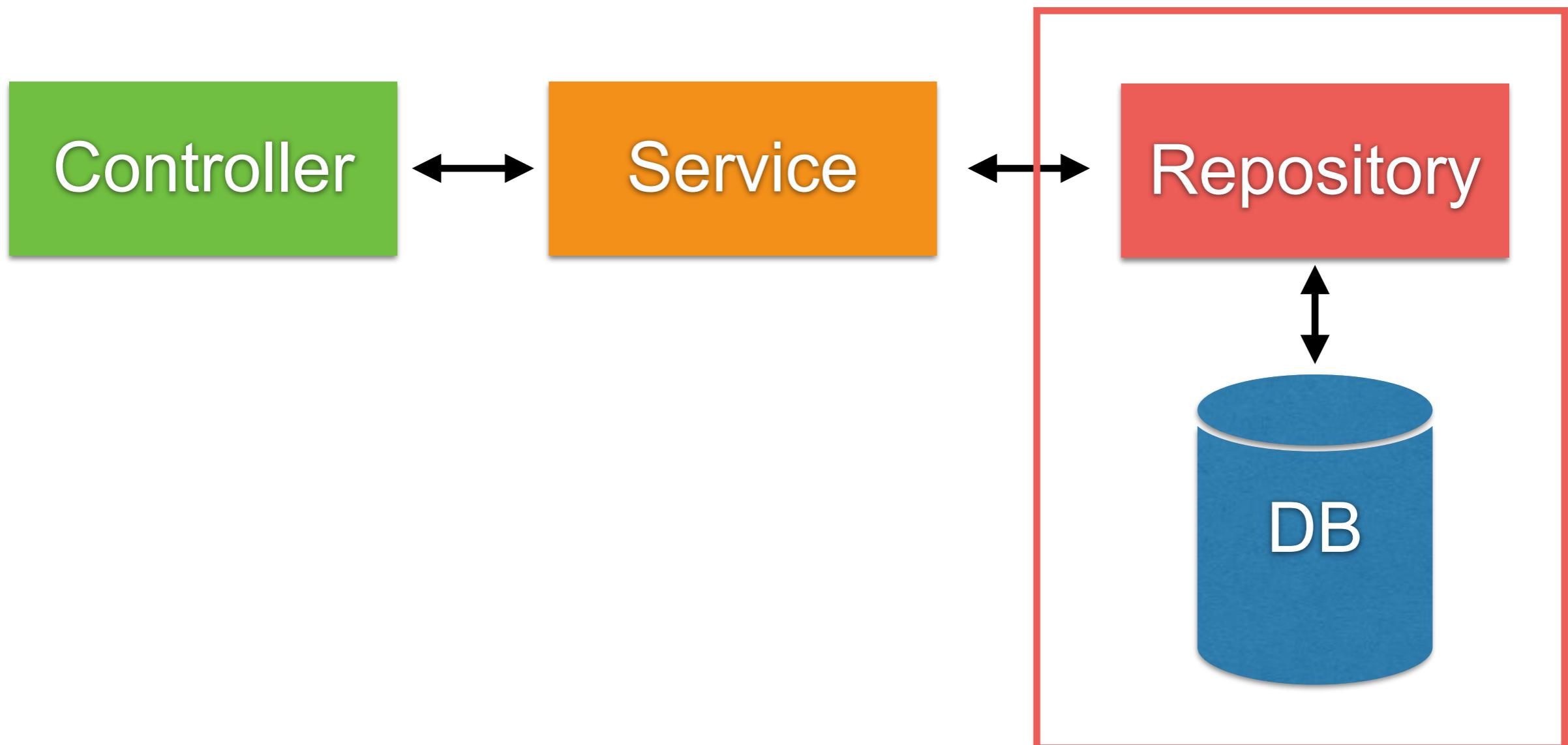
The diagram shows a red rectangular box highlighting the stub code within the `getAccount()` method. A red dashed circle containing the number "2" is positioned to the right of the box, likely indicating a step or iteration number.



# Working with Repository



# Working with repository



# Basic of JDBC

```
// 1. Load jdbc driver
Class.forName("postgresql");

// 2. Create connection
Connection connection = DriverManager.getConnection("", "", "");

// 3. Prepared Statement
String sql = "SELECT * FROM TABLE WHERE name=?";
PreparedStatement pStmt = connection.prepareStatement(sql);

// 4. Query
ResultSet resultSet = pStmt.executeQuery();
while(resultSet.next()) {

}

// 5. Release resource
if(resultSet != null) {
    resultSet.close();
    resultSet = null;
}
```



# Framework !!

```
// 1. Load jdbc driver  
Class.forName("postgresql");  
// 2. Create connection  
Connection connection = DriverManager.getConnection("", "", "");  
Manage by Framework
```

```
// 3. Prepared Statement  
String sql = "SELECT * FROM TABLE WHERE name=?";  
PreparedStatement pStmt = connection.prepareStatement(sql);  
Manage by Framework
```

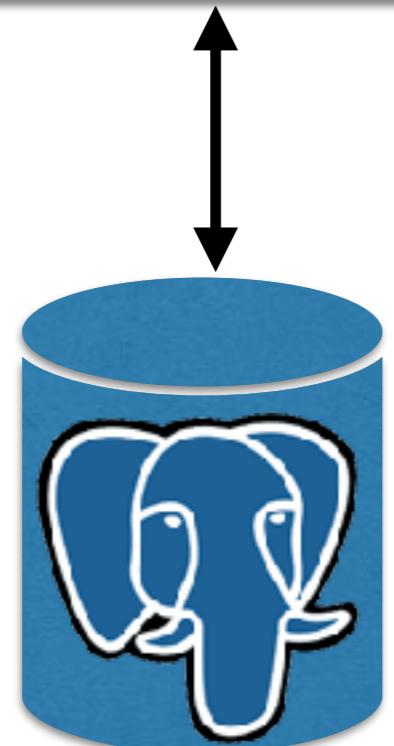
```
// 4. Query  
ResultSet resultSet = pStmt.executeQuery();  
while(resultSet.next()) {  
}  
Manage by Framework
```

```
// 5. Release resource  
if(resultSet != null) {  
    resultSet.close();  
    resultSet = null;  
}  
Manage by Framework
```



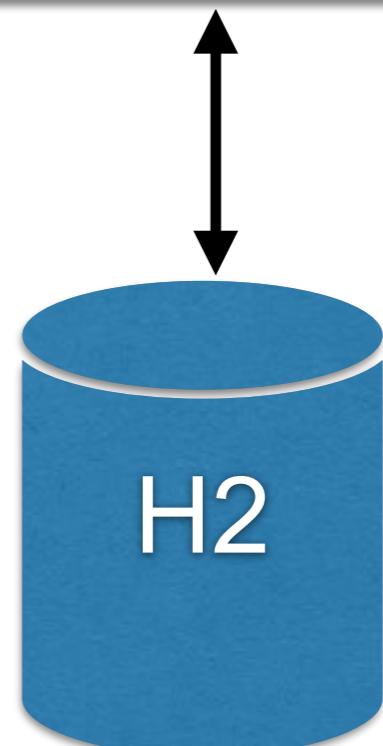
# Working with Database ?

Production



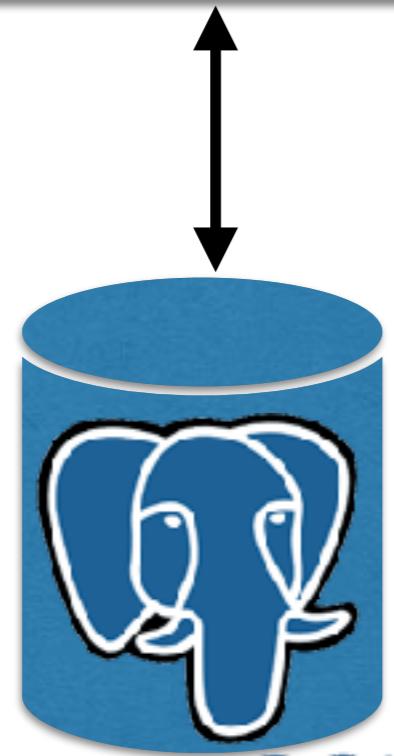
PostgreSQL

Testing



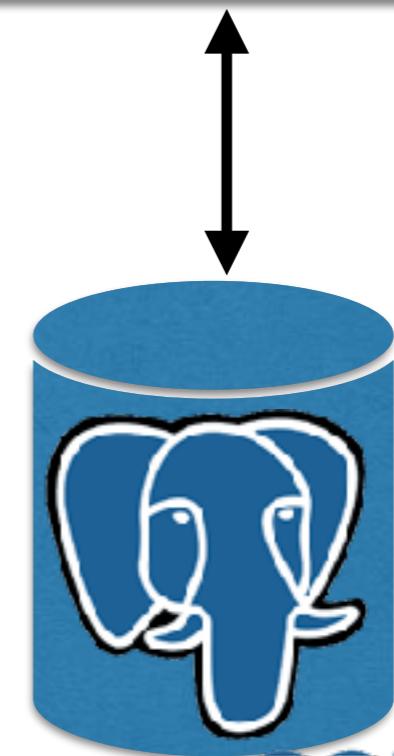
# Working with Database ?

Production



PostgreSQL

Testing



PostgreSQL



# Working with repository

We're using Spring Data



<https://spring.io/projects/spring-data>

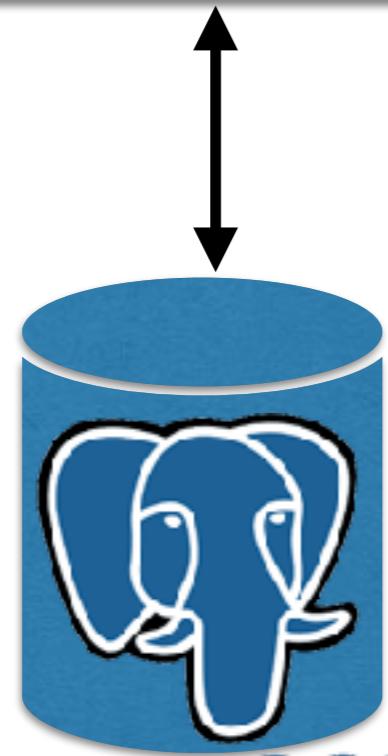


# Working with Spring Data JPA



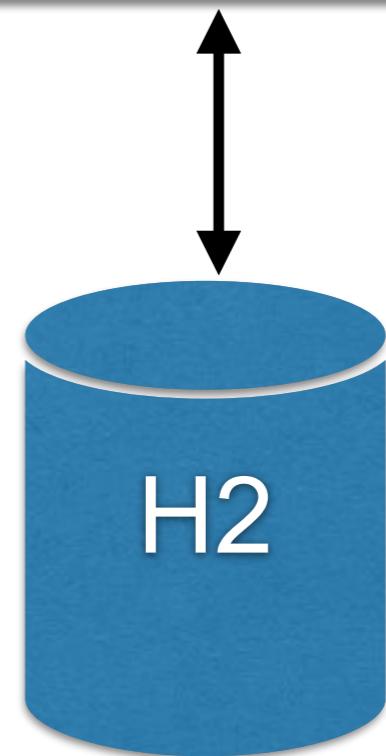
# Working with Database

Production



PostgreSQL

Testing



# Modify pom.xml

Add library of Spring Data JPA, PostgreSQL, H2

**Dependencies**

**ADD DEPENDENCIES... ⌂ + B**

---

**PostgreSQL Driver** SQL

A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

---

**H2 Database** SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

---

**Spring Data JPA** SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

<https://start.spring.io/>



# Modify pom.xml

## H2 for testing

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

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

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```



# Modify pom.xml

## PostgreSQL for production

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

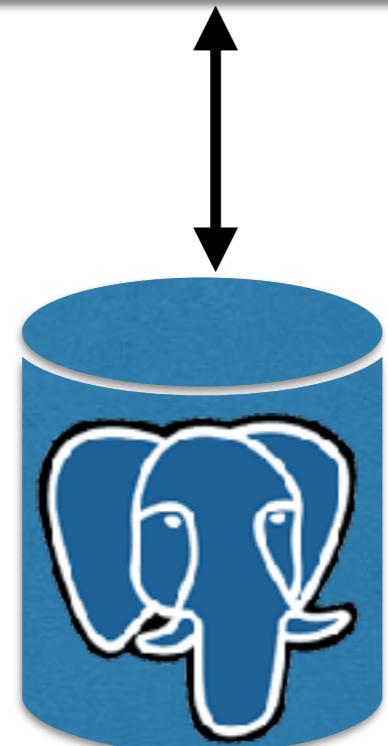
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```



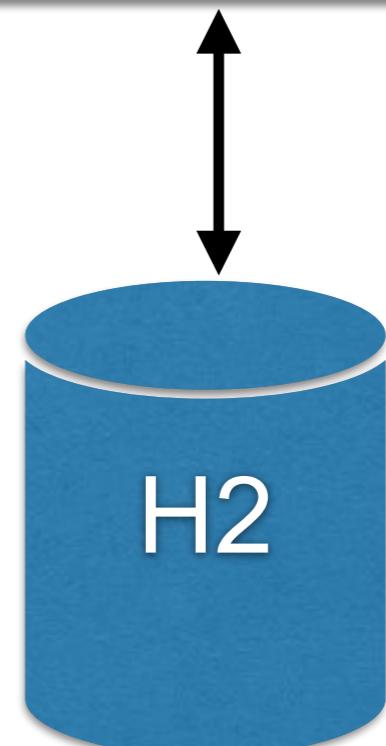
# Start in testing scope

Production

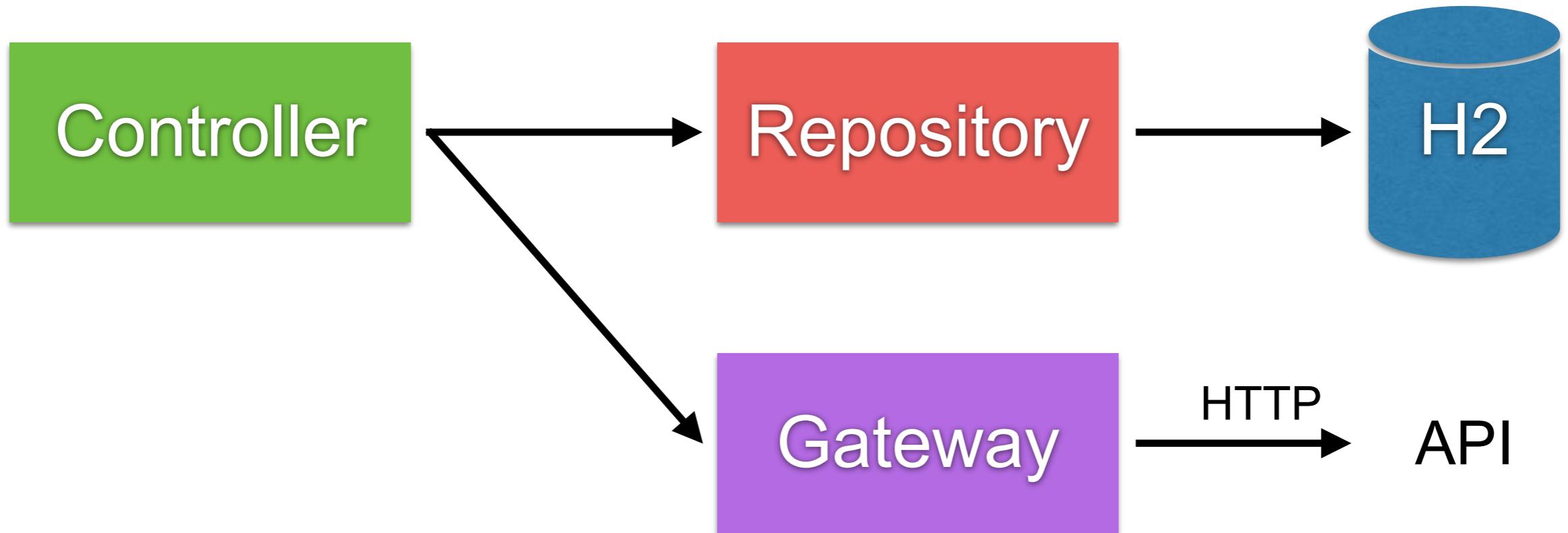


PostgreSQL

Testing

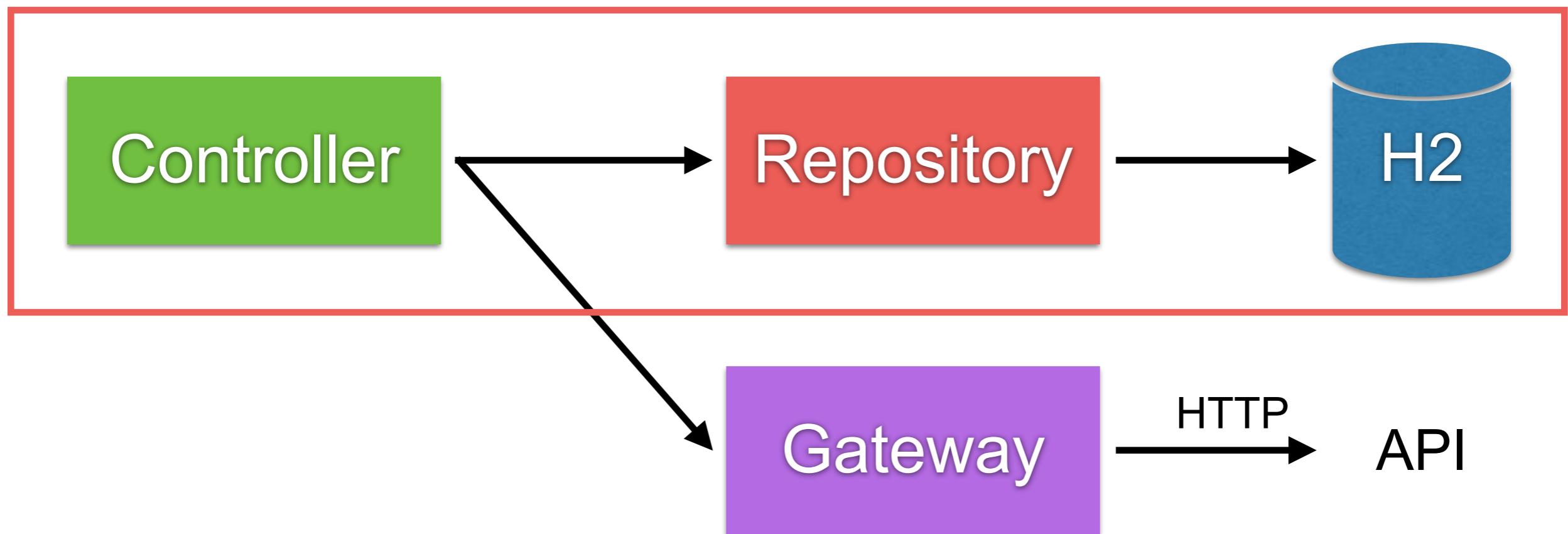


# Use cases



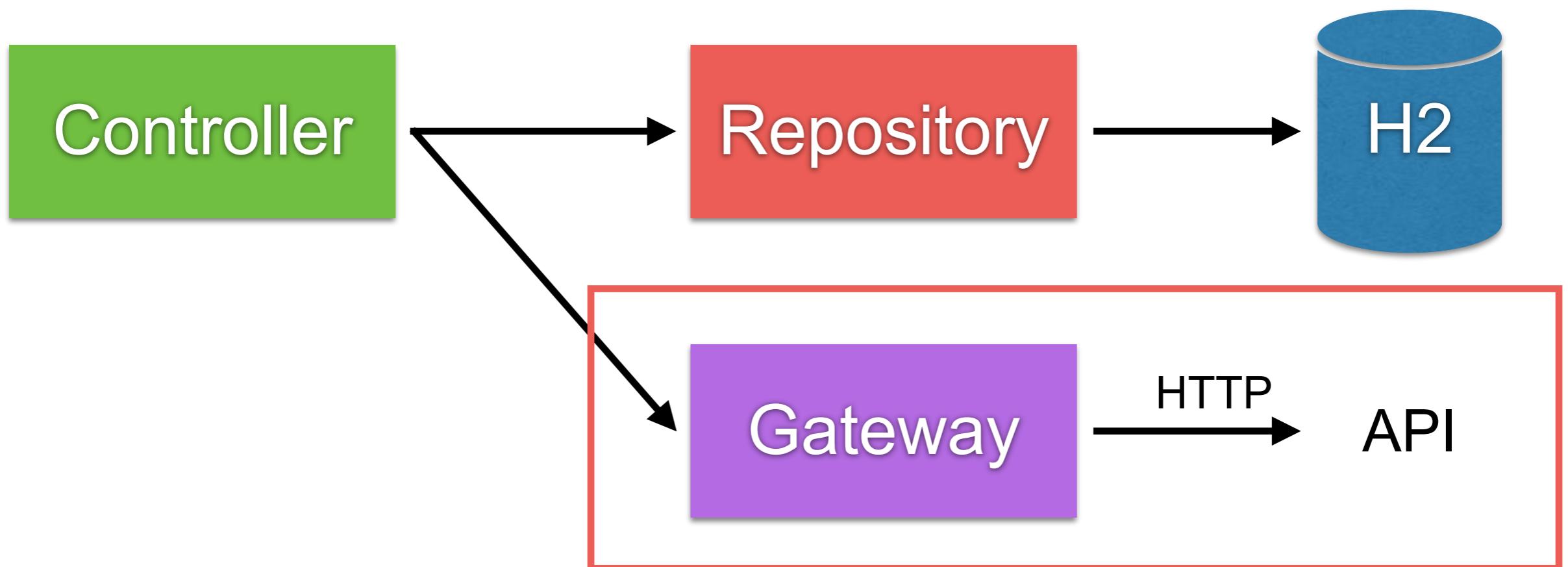
# Use case 1

## Working with repository



# Use case 2

## Working with API



# **Use case 1**

## **Working with Database**



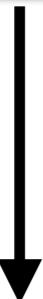
# Use case 1

Working with repository

3. HelloController



2. PersonRepository



1. Person

4. PersonResponse



# 1. Create Entity class

In package person

```
@Entity  
public class Person {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private long id;  
    private String firstName;  
    private String lastName;  
  
    public Person() {  
    }  
}
```



# 2. Create repository with JPA

## PersonRepository.java

```
import java.util.Optional;  
  
import org.springframework.data.repository.CrudRepository;  
  
public interface PersonRepository  
    extends CrudRepository<Person, Long> {  
  
    Optional<Person> findByLastName(String lastName);  
  
}
```



# 2. Create repository with JPA

## PersonRepository.java

```
import java.util.Optional;  
  
import org.springframework.data.repository.CrudRepository;  
  
public interface PersonRepository  
    extends CrudRepository<Person, Long> {  
  
    Optional<Person> findByLastName(String lastName);  
}
```

*SELECT \* FROM Person WHERE LastName=?*

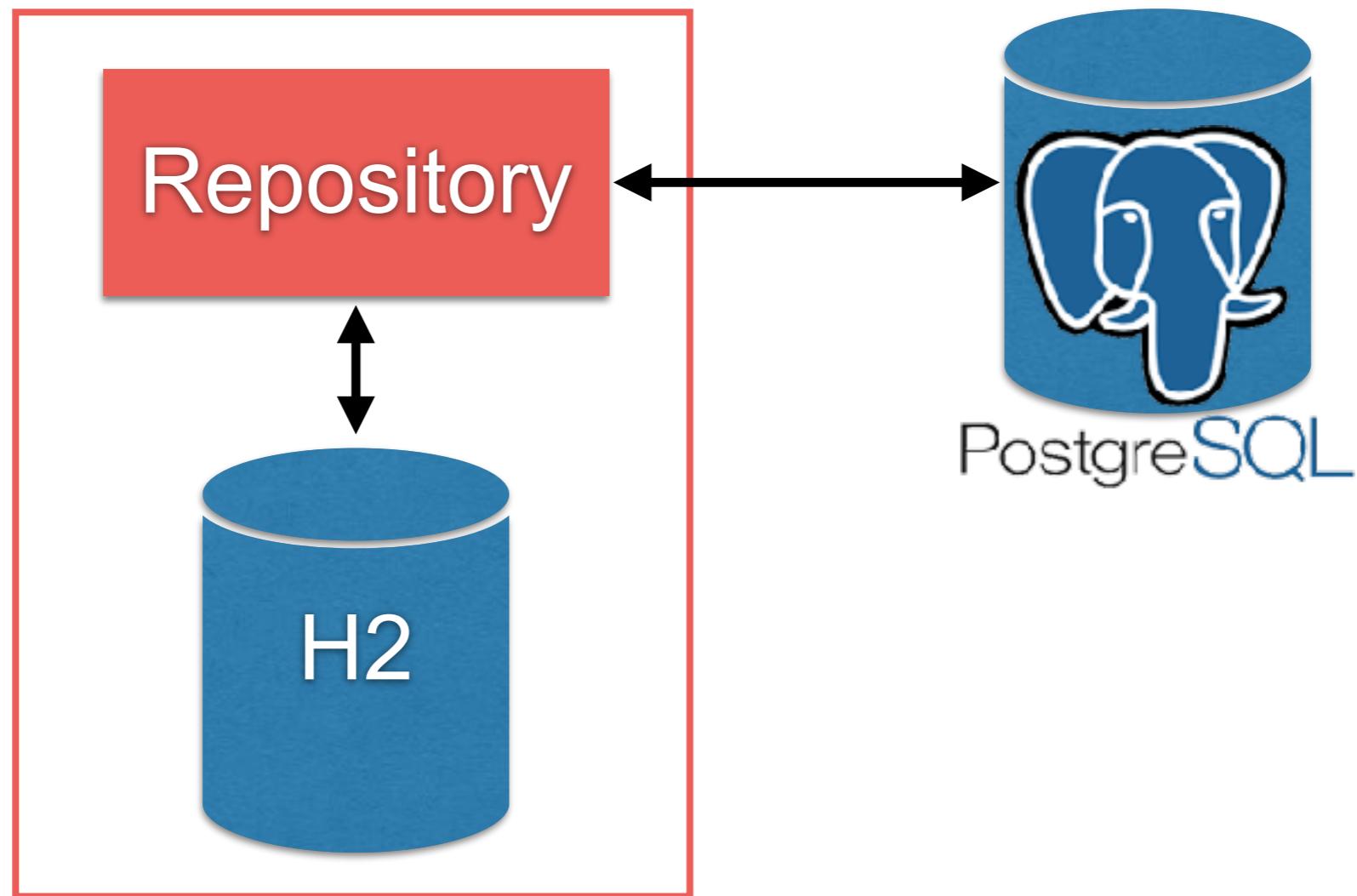


# How to testing repository ?



# Repository Testing

Using `@DataJpaTest` (slice testing)



Working with In-memory database



# Repository Testing #1

## Setup test with @DataJpaTest

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

    @Autowired
    private PersonRepository repository;

    @After
    public void tearDown() throws Exception {
        repository.deleteAll();
    }

}
```



# Repository Testing #2

## Auto wired repository for testing

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

    @Autowired
    private PersonRepository repository;

    @After
    public void tearDown() throws Exception {
        repository.deleteAll();
    }

}
```



# Repository Testing #3

Clear data in table after executed each test case

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

    @Autowired
    private PersonRepository repository;

    @After
    public void tearDown() throws Exception {
        repository.deleteAll();
    }

}
```



# Repository Testing #3

Write your first test case

```
@Test  
public void should_save_fetch_a_person() {  
  
    Person somkiat = new Person("Somkiat", "Pui");  
    repository.save(somkiat);  
  
    Optional<Person> maybeSomkiat  
        = repository.findByLastName("Pui");  
  
    assertEquals(maybeSomkiat, Optional.of(somkiat));  
}
```



# Run test

\$mvnw clean test

Hibernate: drop table person if exists

Hibernate: drop sequence if exists hibernate\_sequence

Hibernate: create sequence hibernate\_sequence start with 1 increment by 1

Hibernate: create table person (id varchar(255) not null, first\_name varchar(255), last\_name varchar(255), primary key (id))

Insert data

Hibernate: call next value for hibernate\_sequence

Hibernate: insert into person (first\_name, last\_name, id) values (?, ?, ?)

Hibernate: select person0\_.id as id1\_0\_, person0\_.first\_name as first\_na2\_0\_, person0\_.last\_name as last\_nam3\_0\_ from person person0\_ where person0\_.last\_name=?

Hibernate: select person0\_.id as id1\_0\_, person0\_.first\_name as first\_na2\_0\_, person0\_.last\_name as last\_nam3\_0\_ from person person0\_

2

Query data

Hibernate: drop table person if exists

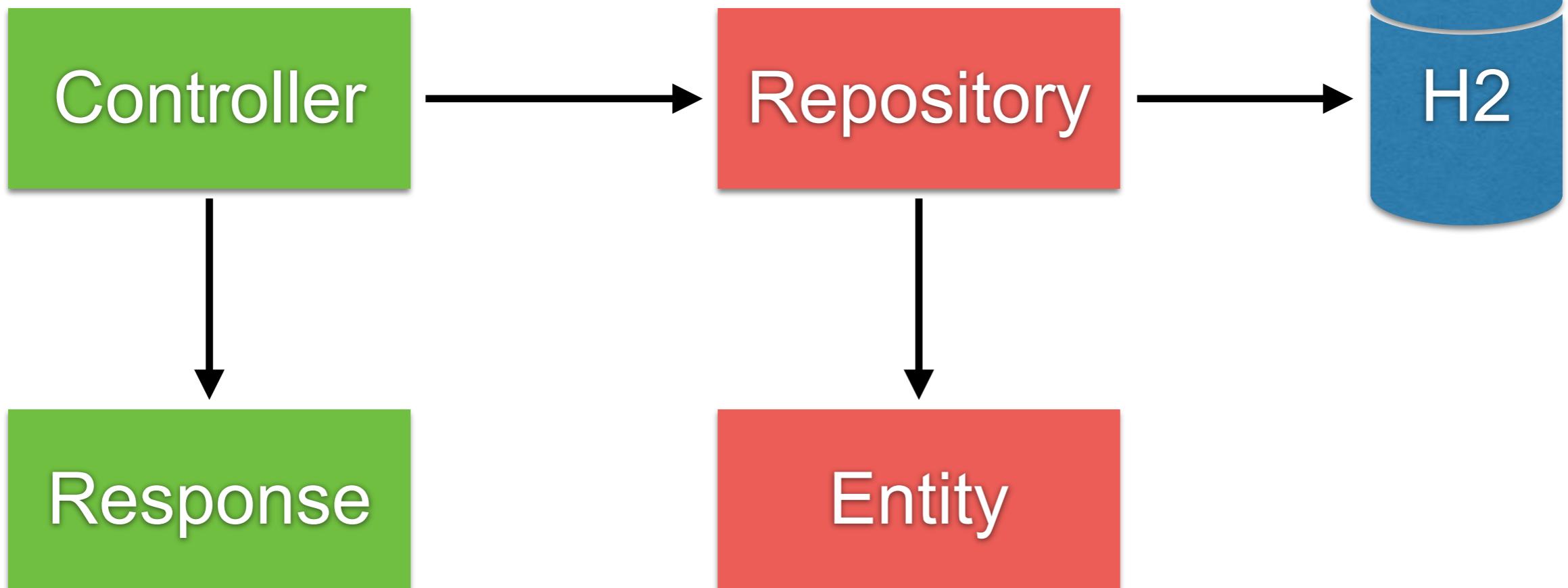
Hibernate: drop sequence if exists hibernate\_sequence



# Use case 1

Integrate repository with controller

3. HelloController      2. PersonRepository



1. Person

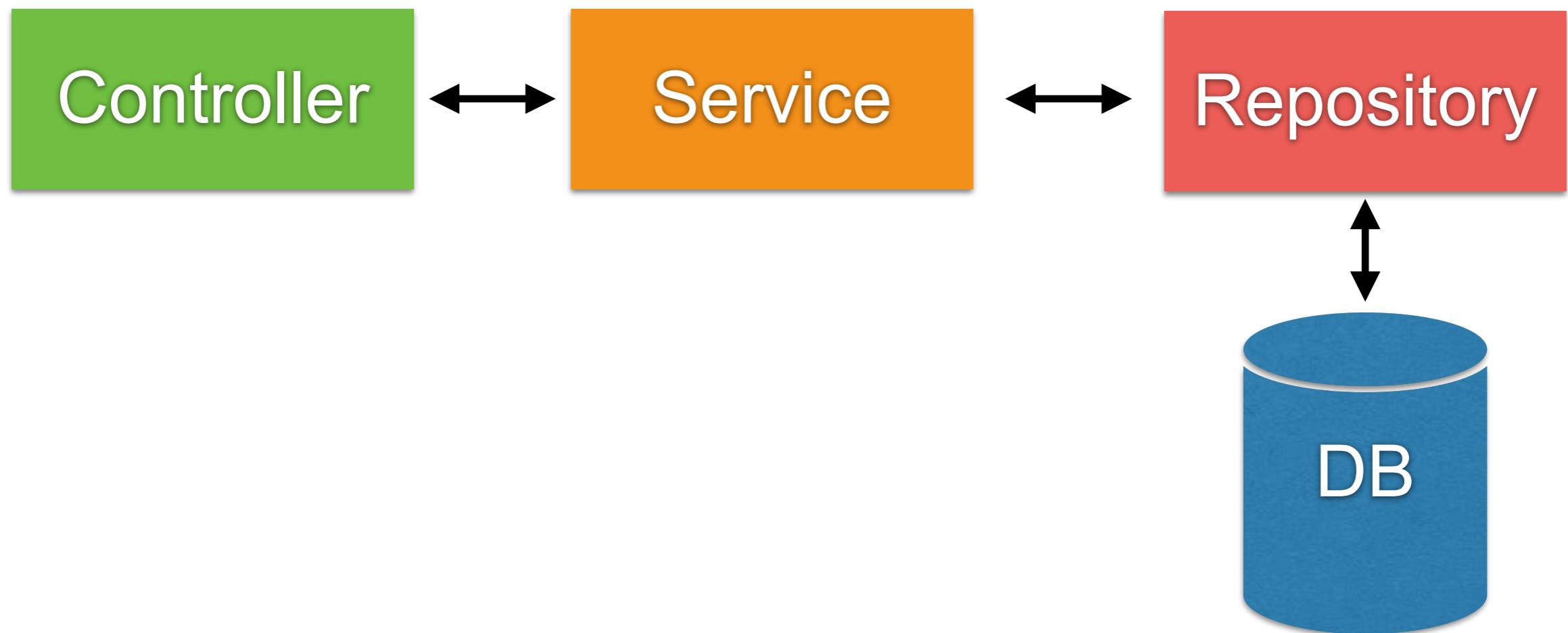
4. PersonResponse



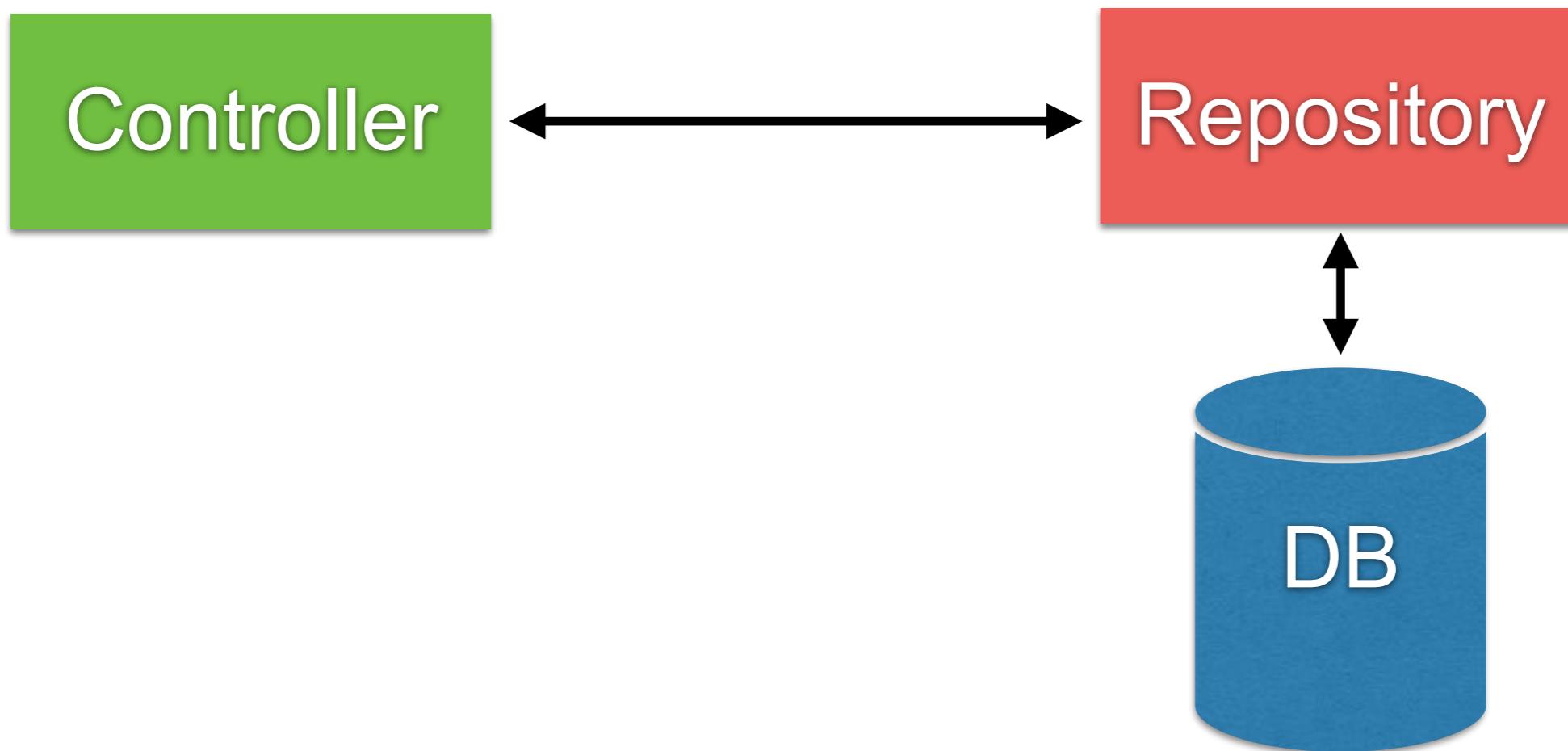
# **Integrate repository with service/controller**



# Service use repository ?



# Controller use repository ?



# Controller call repository

## Create HelloController.java

```
@RestController
public class HelloController {

    private final PersonRepository personRepository;

    @Autowired
    public HelloController(final PersonRepository personRepository) {
        this.personRepository = personRepository;
    }
}
```



# Controller call repository

## Create HelloController.java

```
@GetMapping("/hello/{lastName}")
public HelloResponse hello(@PathVariable final String lastName) {

    Optional<Person> foundPerson
        = personRepository.findByLastName(lastName);

    return foundPerson
        .map(person ->
            new HelloResponse(person.getFirstName(),
                person.getLastName()))
        .orElseThrow(() -> new RuntimeException());
}
```



# Run spring boot

\$mvnw spring-boot:run



# Fix !!!

## Modify src/main/resources/application.properties

server.port=8088

spring.datasource.url=jdbc:postgresql://127.0.0.1:15432/postgres

spring.datasource.username=testuser

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

## Start database server !!



# Fix !!!

Modify pom.xml  
Delete or comment postgresql dependency

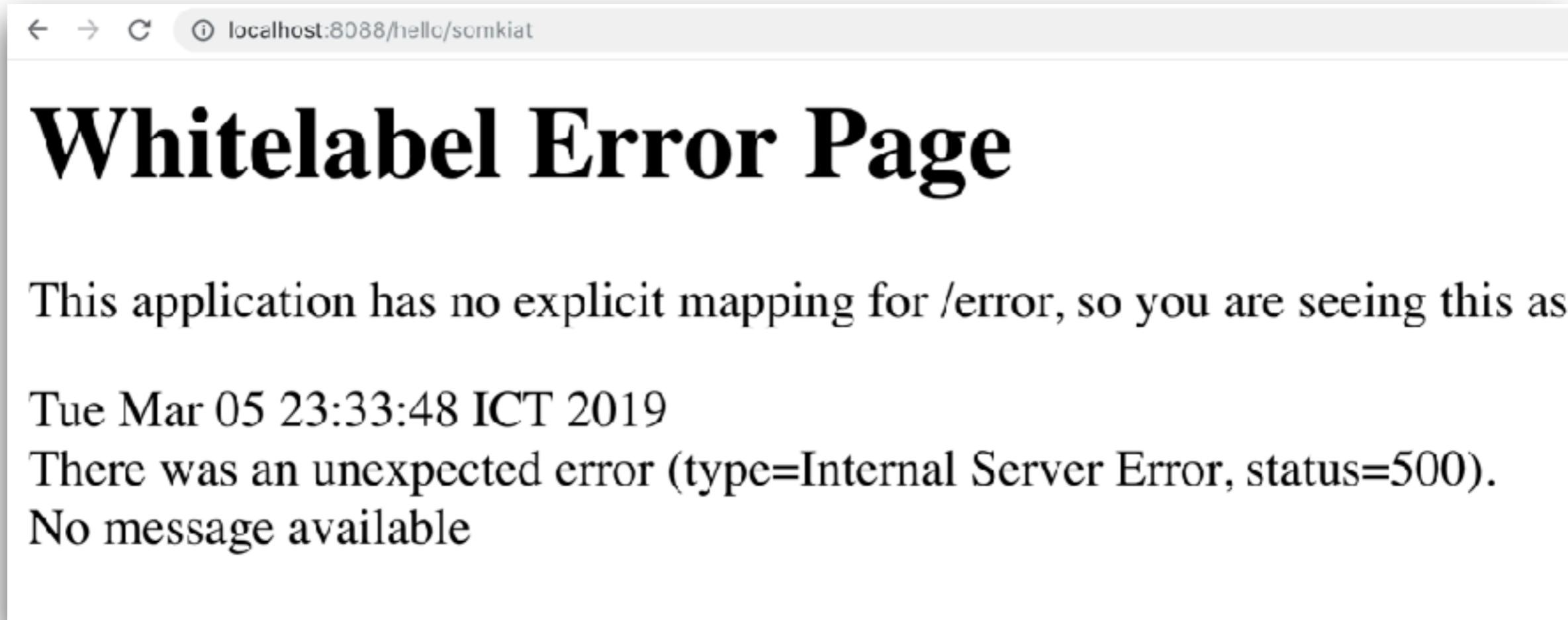
```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>

<!-- <dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency> -->
```



# Run spring boot

\$mvnw spring-boot:run



# Initial data in database



# Initial database #1

## Using @Bean and CommandLineRunner

```
@Bean  
public CommandLineRunner initData(MessageRepository repository) {  
    return new CommandLineRunner() {  
  
        @Override  
        public void run(String... args) throws Exception {  
            repository.save(new Message("somkiat1"));  
            repository.save(new Message("somkiat2"));  
        }  
    };  
}
```



# Initial database #2

## Using @PostConstruct

```
@PostConstruct  
public void initData() {  
    Account account1 = new Account();  
    account1.setAccountId("01");  
    accountRepository.save(account1);  
    Account account2 = new Account();  
    account2.setAccountId("02");  
    accountRepository.save(account2);  
}
```



# Initial database #3

**Schema** (resources/schema.sql)

**Data** (resources/data.sql)

## Schema.sql

```
CREATE TABLE account(
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    account_Id VARCHAR(16) NOT NULL UNIQUE,
    mobile_No VARCHAR(10),
    name VARCHAR(50),
    account_Type CHAR(2)
);
```

## Data.sql

```
INSERT INTO account (account_Id) VALUES ('01');
INSERT INTO account (account_Id) VALUES ('02');
```



# Initial database 2

Disable auto generate DDL from JPA in file application.yml

```
spring:  
  jpa:  
    show-sql: true  
    hibernate:  
      ddl-auto: none
```



# Initial database 2

## Problem with naming strategy !!

```
spring:  
  jpa:  
    show-sql: true  
    hibernate:  
      ddl-auto: none  
      naming:  
        physical-strategy:  
          org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy  
        implicit-strategy:  
          org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
```

<https://github.com/spring-projects/spring-boot/tree/master/spring-boot-project/spring-boot/src/main/java/org/springframework/boot/orm/jpa/hibernate>



# Run and see from logging

Execute file schema.sql and data.sql

```
.datasource.init.ScriptUtils      : Executing SQL script from URL [file:/Users/somki...  
.datasource.init.ScriptUtils      : Executed SQL script from URL [file:/Users/somki...  
.datasource.init.ScriptUtils      : Executing SQL script from URL [file:/Users/somki...  
.datasource.init.ScriptUtils      : Executed SQL script from URL [file:/Users/somki...
```



# Run spring boot

\$mvnw spring-boot:run



# **Write controller testing ?**

**\$mvnw clean test**



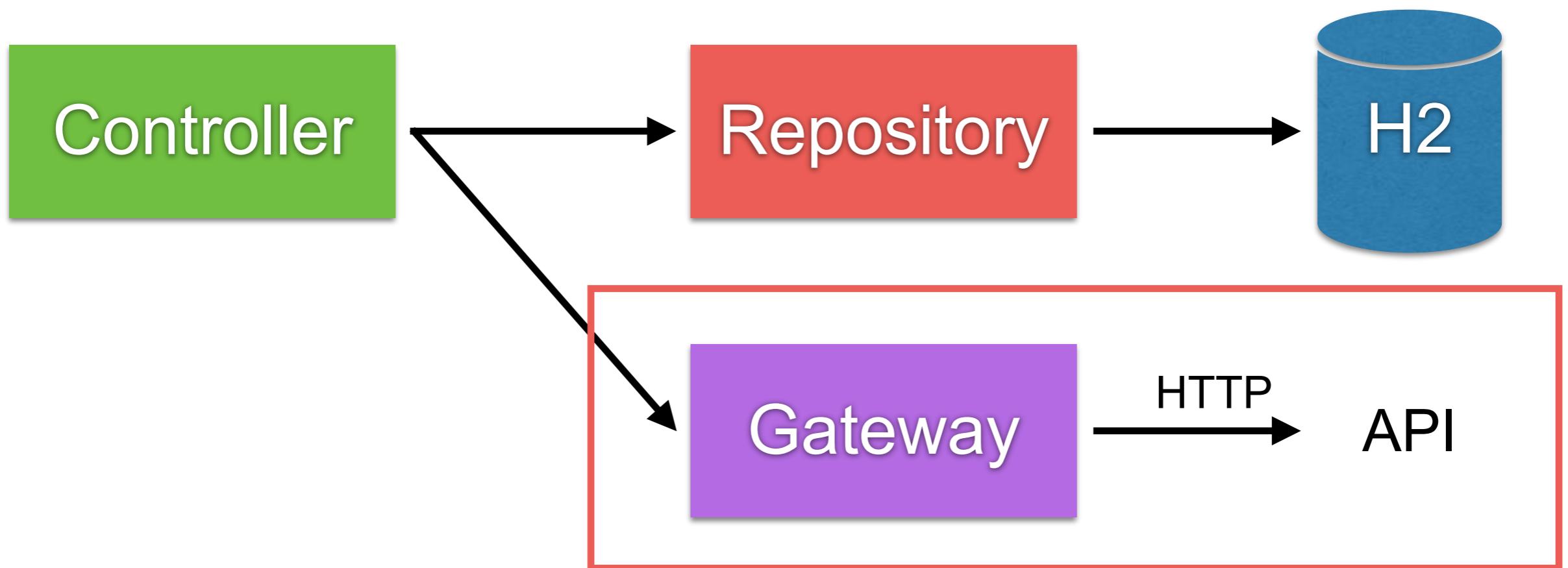
# Use case 2

## Working with API



# Use case 2

## Working with API



# JSON Place Holder

<https://jsonplaceholder.cypress.io/posts/1>

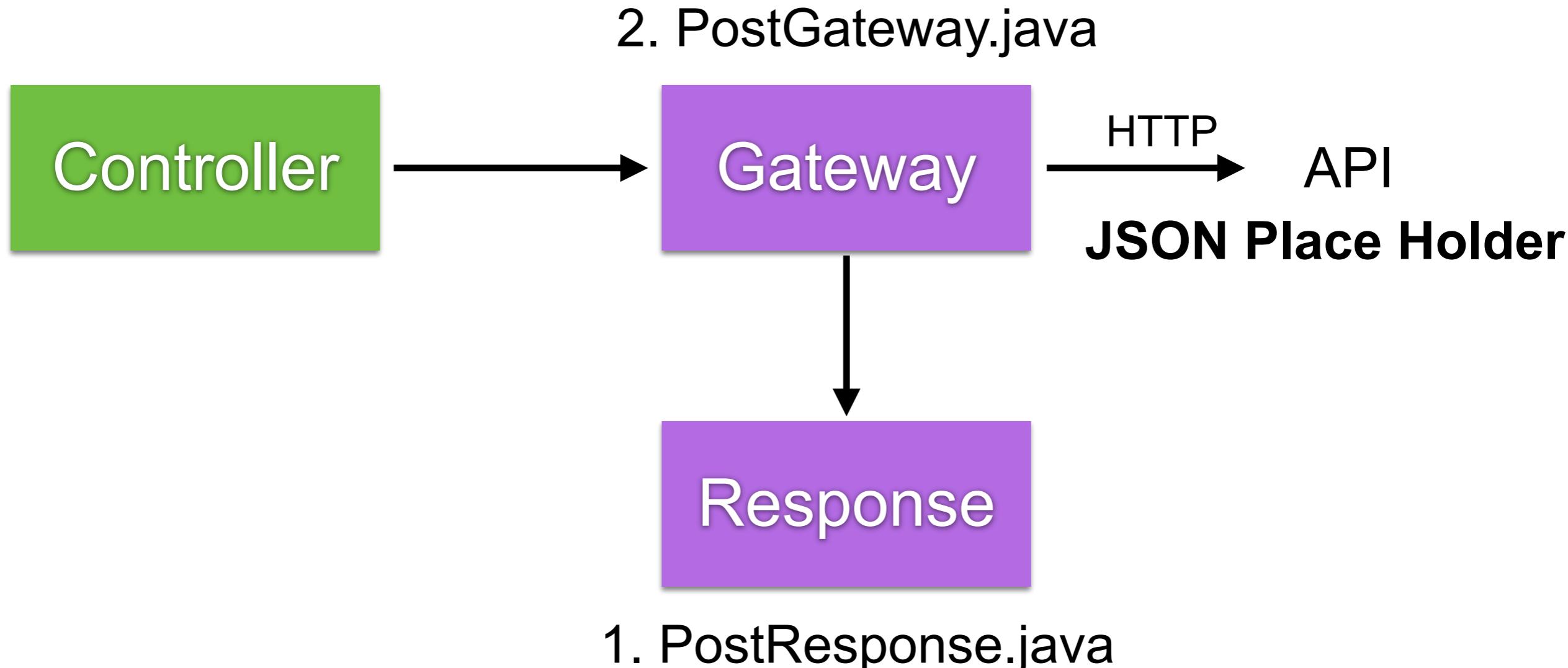
The screenshot shows a web-based API testing interface for JSONPlaceholder. At the top, the title "JSONPlaceholder" is displayed in large, bold letters. Below it, a subtitle reads "Fake Online REST API for Testing and Prototyping" and "Powered by [JSON Server + LowDB](#)". A code snippet in a light gray box illustrates an asynchronous JavaScript fetch call:

```
fetch('https://jsonplaceholder.cypress.io/todos/1')
  .then(response => response.json())
  .then(json => console.log(json))
```

At the bottom of the interface is a blue "Try it" button.



# Working with API



# Library to call external APIs

Rest Template

OpenFeign

HTTP Interface

Spring framework 6



# 1. Create Response class

In package post

```
public class PostResponse {  
    private int id;  
    private int userId;  
    private String title;  
    private String body;
```



# 2. Create PostGateway class #1

In package post

```
@Component
public class PostGateway {

    private final RestTemplate restTemplate;
    private final String postApiUrl;

    @Autowired
    public PostGateway(final RestTemplate restTemplate,
                       @Value("${post.api.url}") final String postApiUrl) {
        this.restTemplate = restTemplate;
        this.postApiUrl = postApiUrl;
    }
}
```



# 2. Create PostGateway class #1

In package post

```
@Component
public class PostGateway {

    private final RestTemplate restTemplate;
    private final String postApiUrl;

    @Autowired
    public PostGateway(final RestTemplate restTemplate,
        @Value("${post.api.url}") final String postApiUrl) {
        this.restTemplate = restTemplate;
        this.postApiUrl = postApiUrl;
    }
}
```

Configuration ?



# Configuration

Configuration in file application.properties

```
post.api.url=https://jsonplaceholder.cypress.io
```



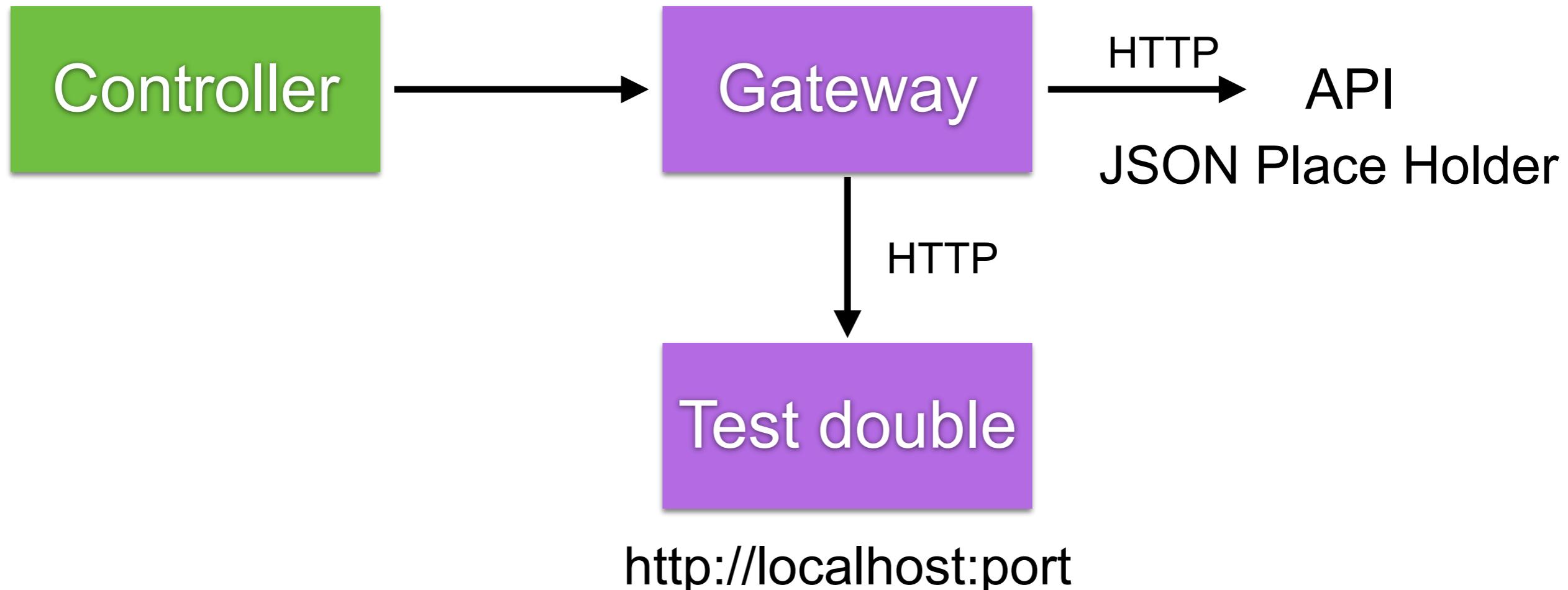
# 2. Create PostGateway class #2

Get data from API

```
public Optional<PostResponse> getPostById(int id) {  
    String url = String.format("%s/posts/%d", postApiUrl, id);  
  
    try {  
        return Optional.ofNullable(  
            restTemplate.getForObject(url, PostResponse.class));  
    } catch (RestClientException e) {  
        return Optional.empty();  
    }  
}
```



# Testing with API



# Testing with API

Unit testing with Mockito

**Component testing with WireMock**

Consumer testing with Pact



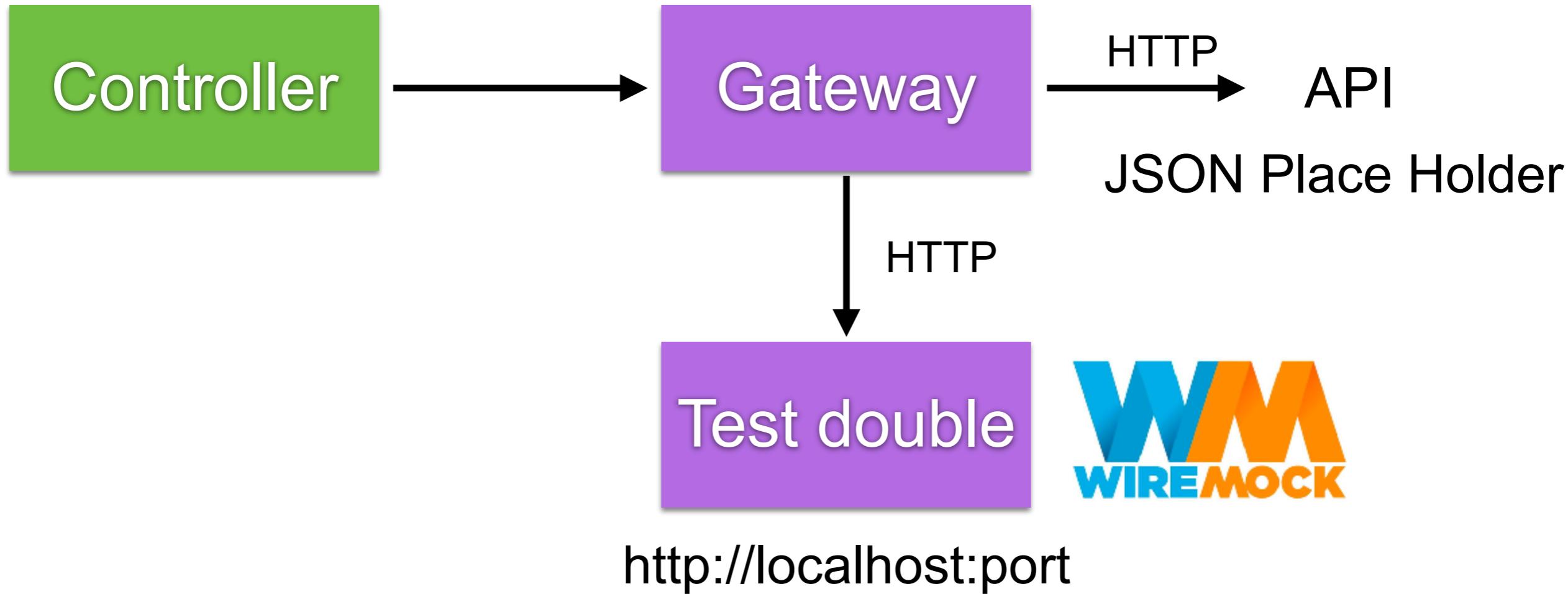
<https://docs.spring.io/spring-cloud-contract/reference/project-features-wiremock.html>



Workshop

© 2020 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Component testing with WireMock



**/src/test/resources/application.properties**

post.api.url=http://localhost:9999



# Component testing #1

## Working with WireMock

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@AutoConfigureWireMock(port = 9999)
public class PostGatewayComponentTest {

    @Autowired
    private PostGateway postGateway;
```

*“Default port = 9999”*



# Component testing #2

## Success case

```
@Test  
public void getPostById() throws IOException {  
    stubFor(get(urlPathEqualTo("/posts/1"))  
        .willReturn(aResponse()  
            .withBody(read("classpath:postApiResponse.json"))  
            .withHeader(CONTENT_TYPE, MediaType.APPLICATION_JSON_VALUE)  
            .withStatus(200));  
  
    Optional<PostResponse> postResponse = postGateway.getPostById(1);  
    assertEquals(11, postResponse.get().getId());  
    assertEquals(11, postResponse.get().getUserId());  
    assertEquals("Test Title", postResponse.get().getTitle());  
    assertEquals("Test Body", postResponse.get().getBody());  
}
```



# Component testing #3

## Read data from resources folder

```
public static String read(String filePath) throws IOException {  
    File file = ResourceUtils.getFile(filePath);  
    return new String(Files.readAllBytes(file.toPath()));  
}
```



# Component testing #4

## File postApiResponse.json

```
{  
  "userId": 11,  
  "id": 11,  
  "title": "Test Title",  
  "body": "Test Body"  
}
```



# **Write controller testing ?**

**\$mvnw clean test**



# Important Quality Service



# Important Quality Service

Security  
Observability  
Configurability



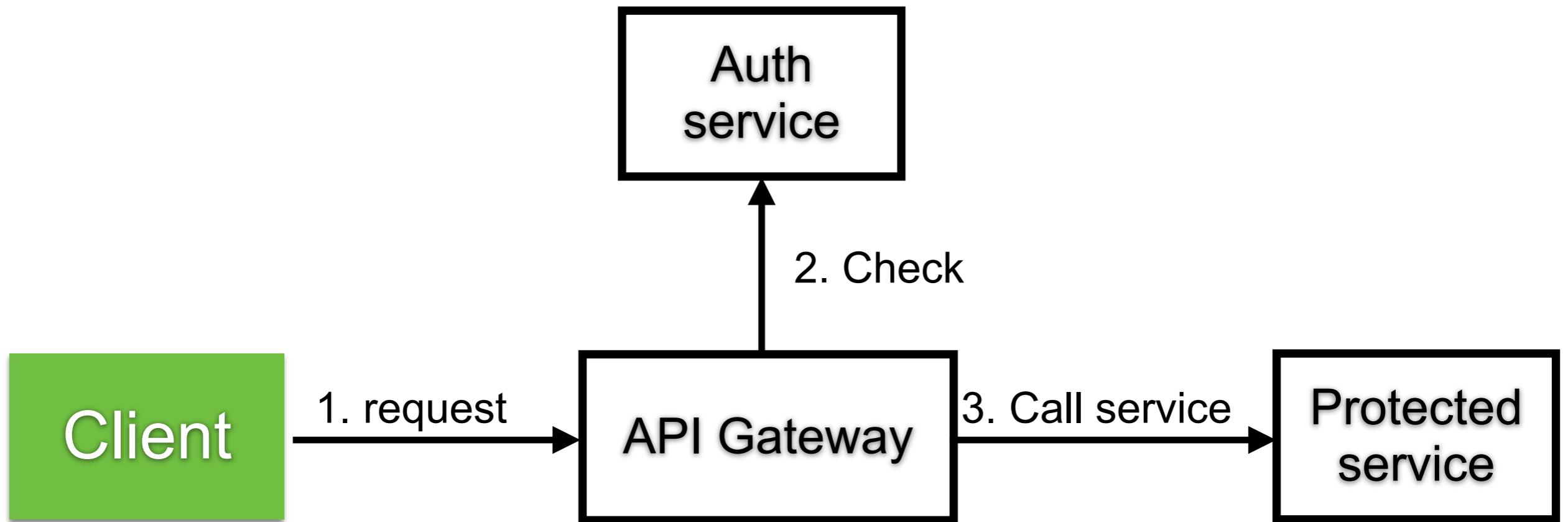
# **Secure Service ?**

## **API Gateway**

**Integrate in your system/service**



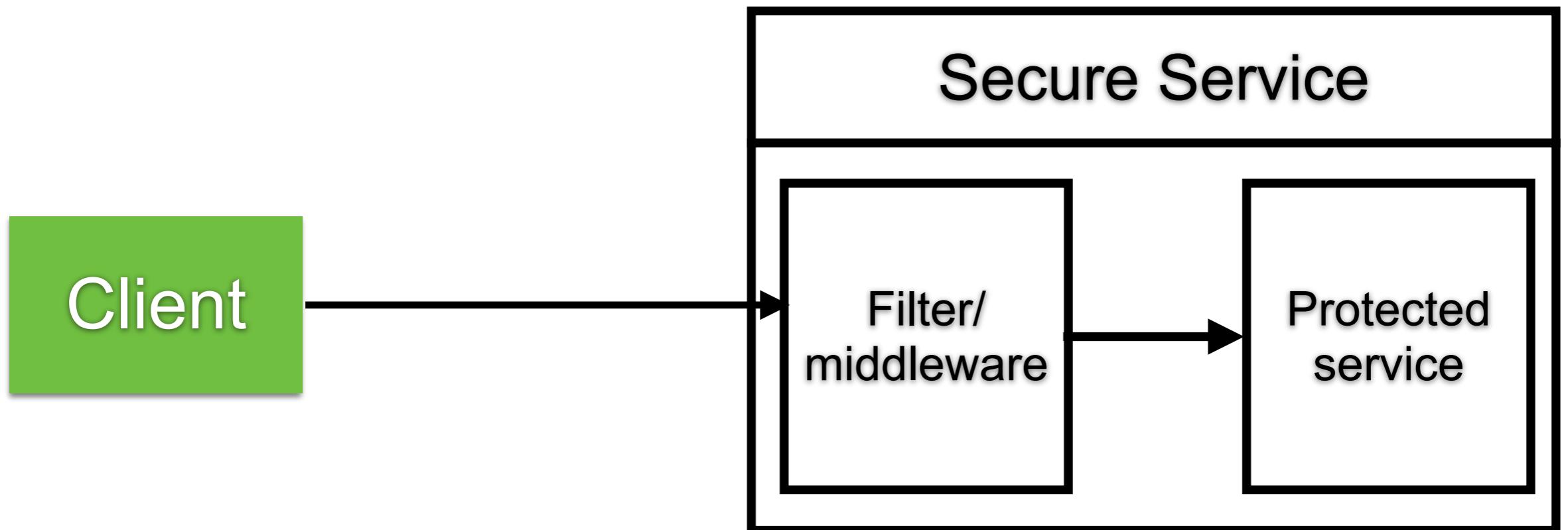
# API Gateway



Kong



# Within service



<https://github.com/up1/demo-spring-security>



# Secure Service with Spring

Working with Spring Security  
Support authentication and authorization

Spring Boot 3

Spring Security 6

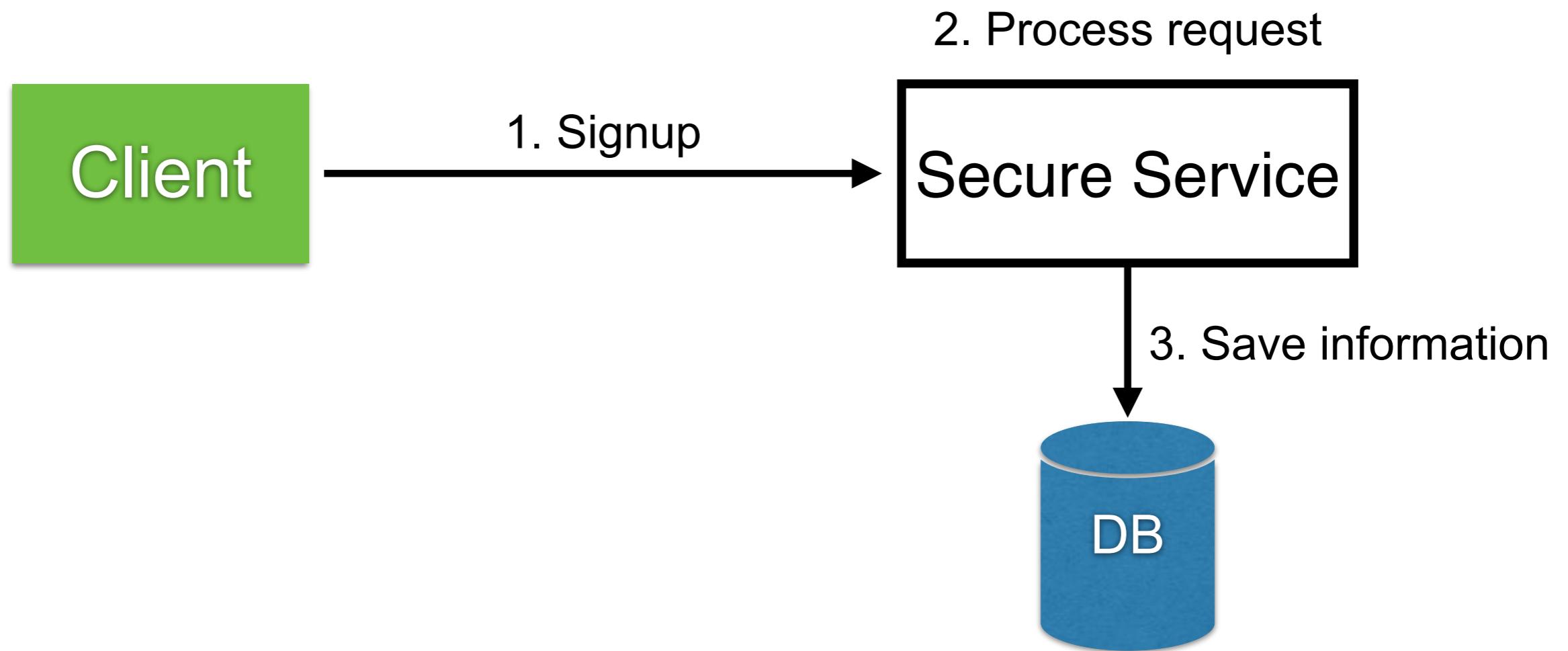
JDK 17

<https://spring.io/projects/spring-security>

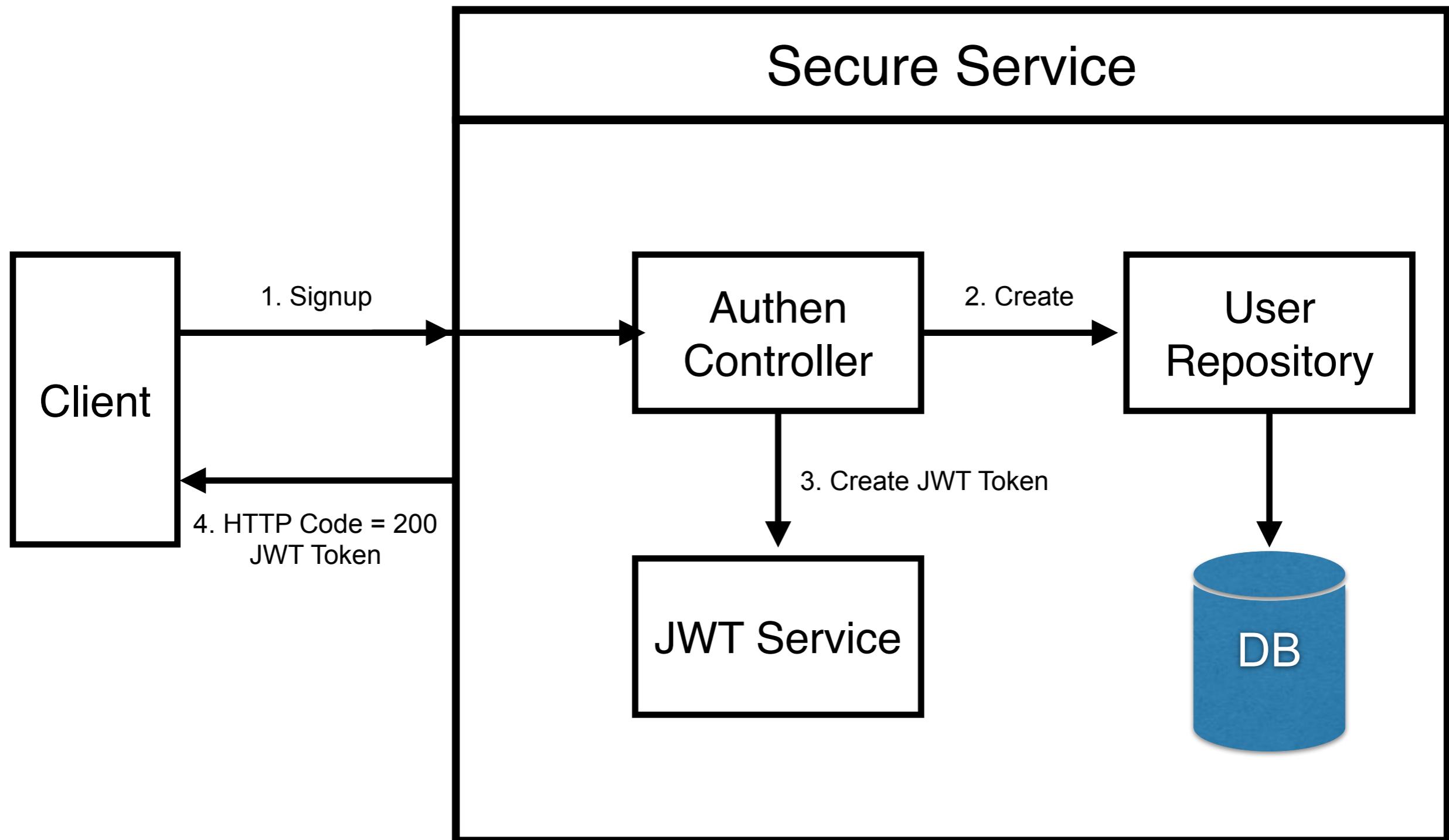


# Simple Flow of Secure Service

## Step 1 :: Signup process

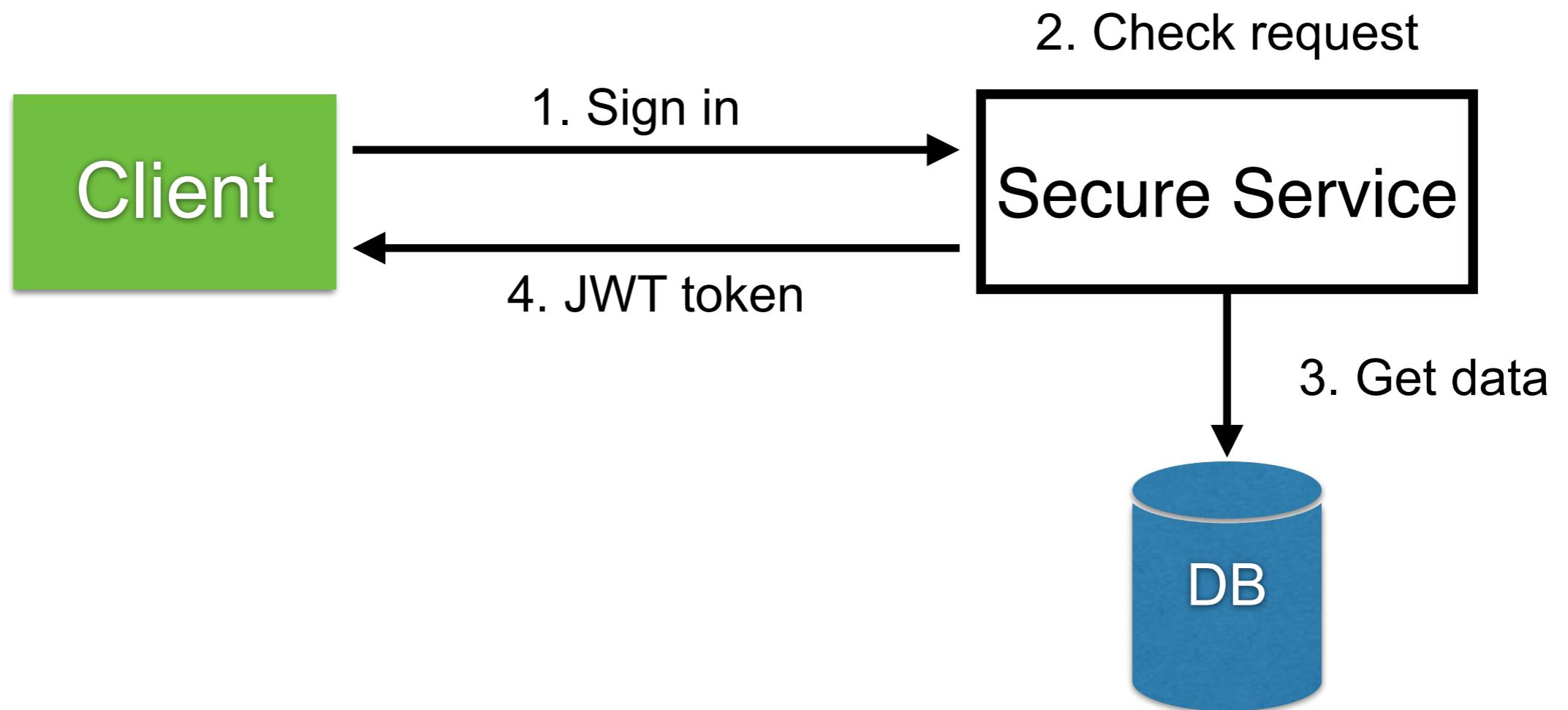


# Flow diagram of Signup

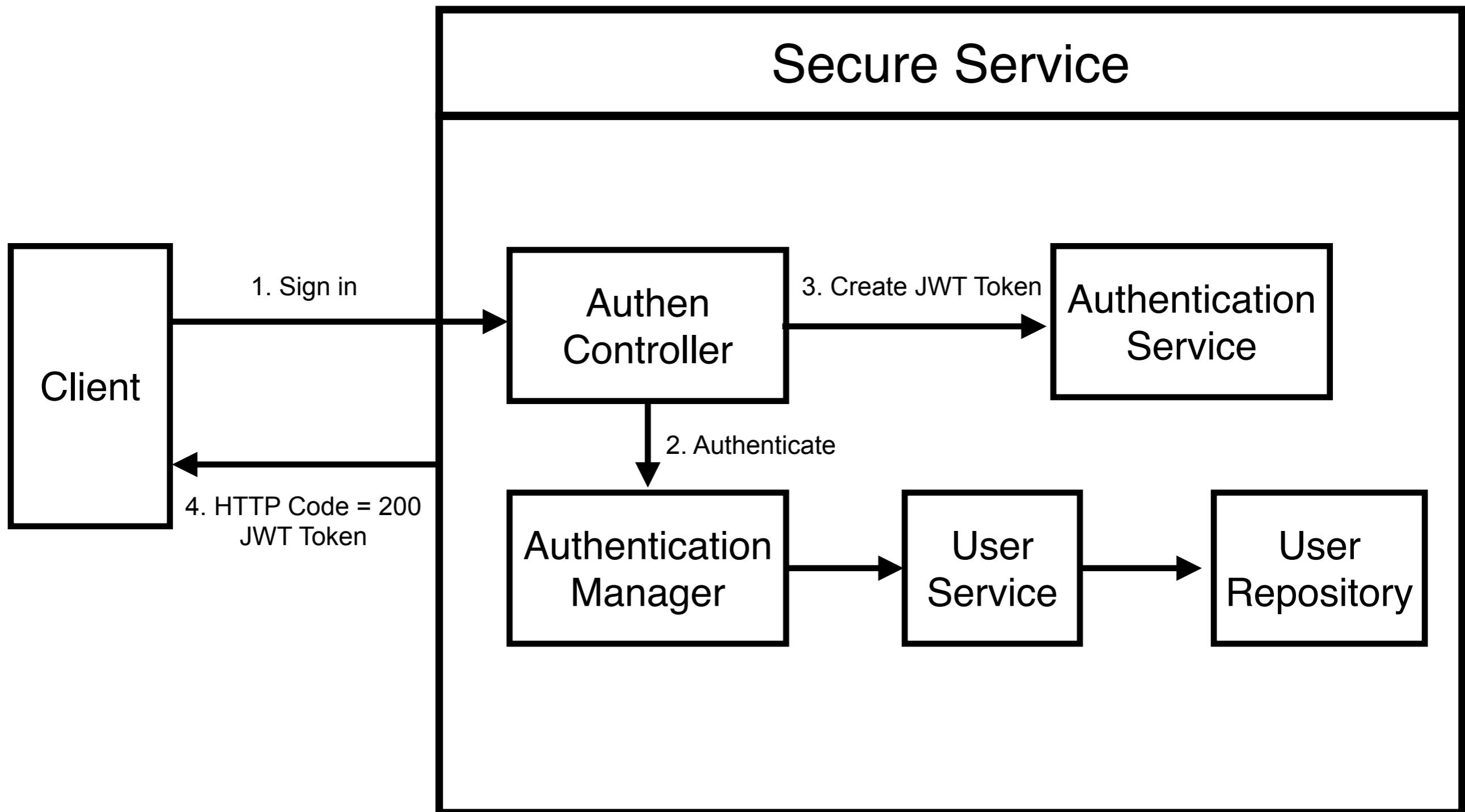


# Simple Flow of Secure Service

## Step 2 :: Sign in process

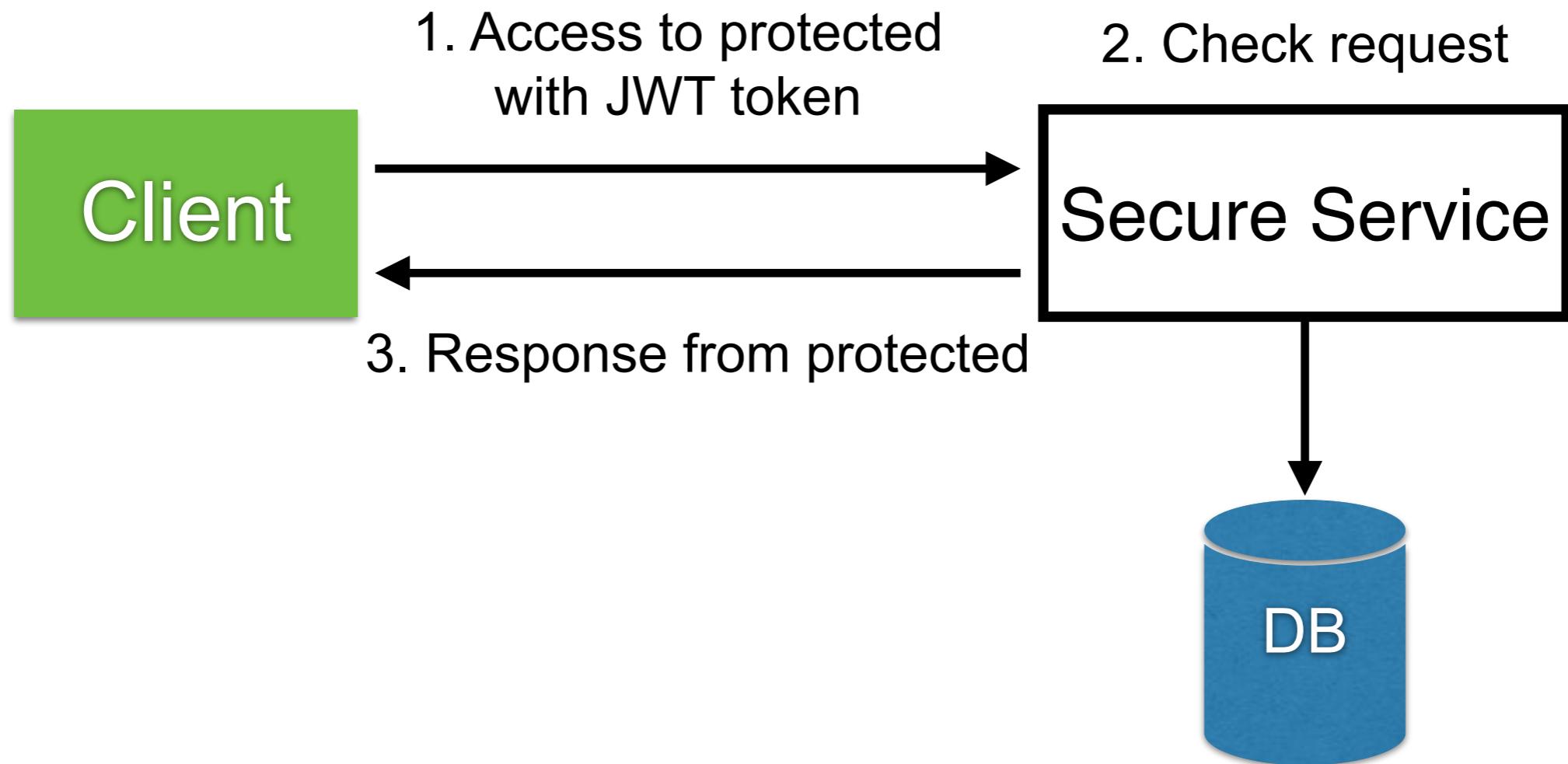


# Flow diagram of Sign in

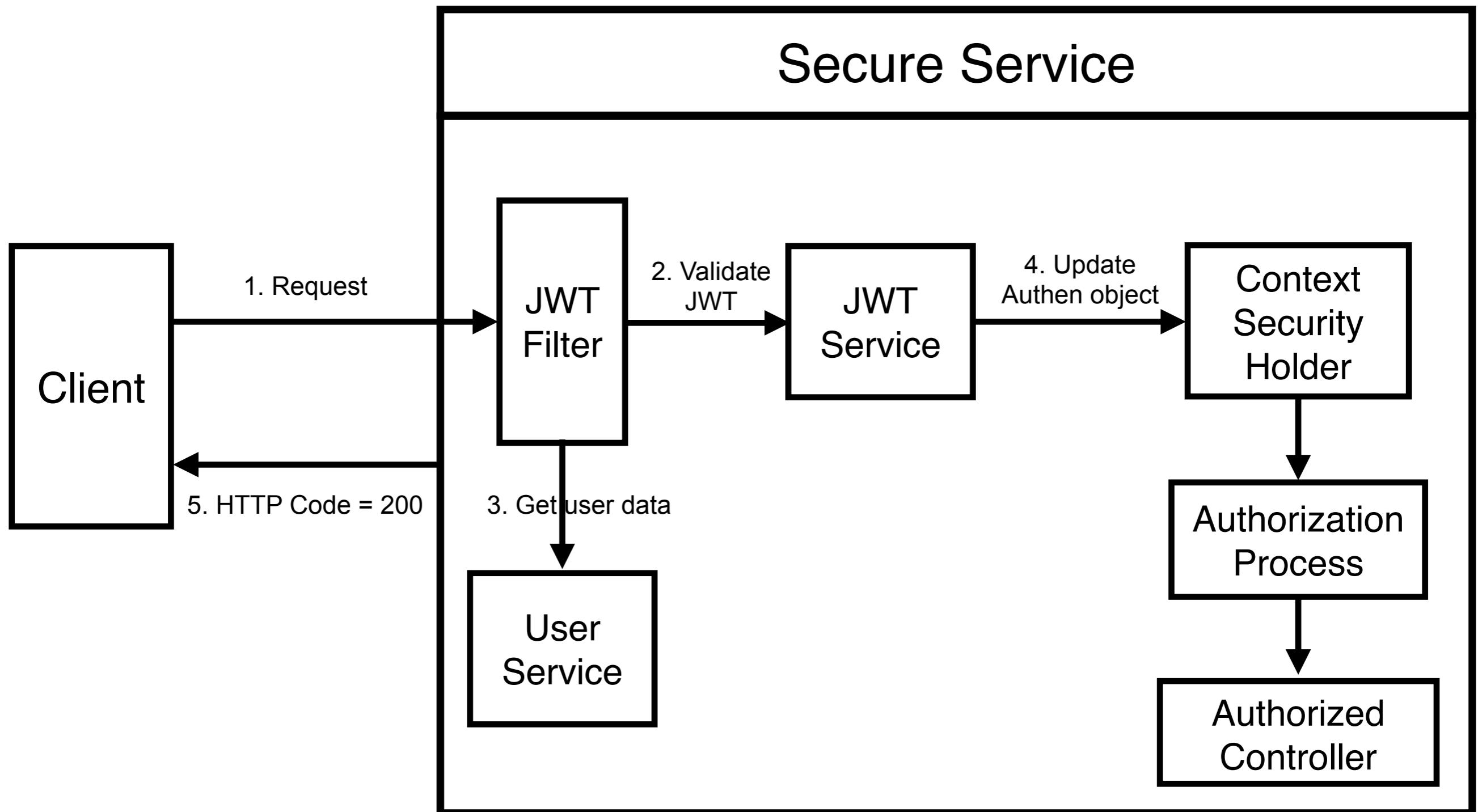


# Simple Flow of Secure Service

## Step 3 :: Access to protected services



# Flow diagram of call service



# JSON Web Tokens (JWT)

**Encoded** PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf1KxwRJSMeKKF2QT4fwpNeJf36P0k6yJV_adQssw5c
```

**Decoded** EDIT THE PAYLOAD AND SECRET

**HEADER: ALGORITHM & TOKEN TYPE**

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

**PAYLOAD: DATA**

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

**VERIFY SIGNATURE**

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  secret: base64 encoded
```

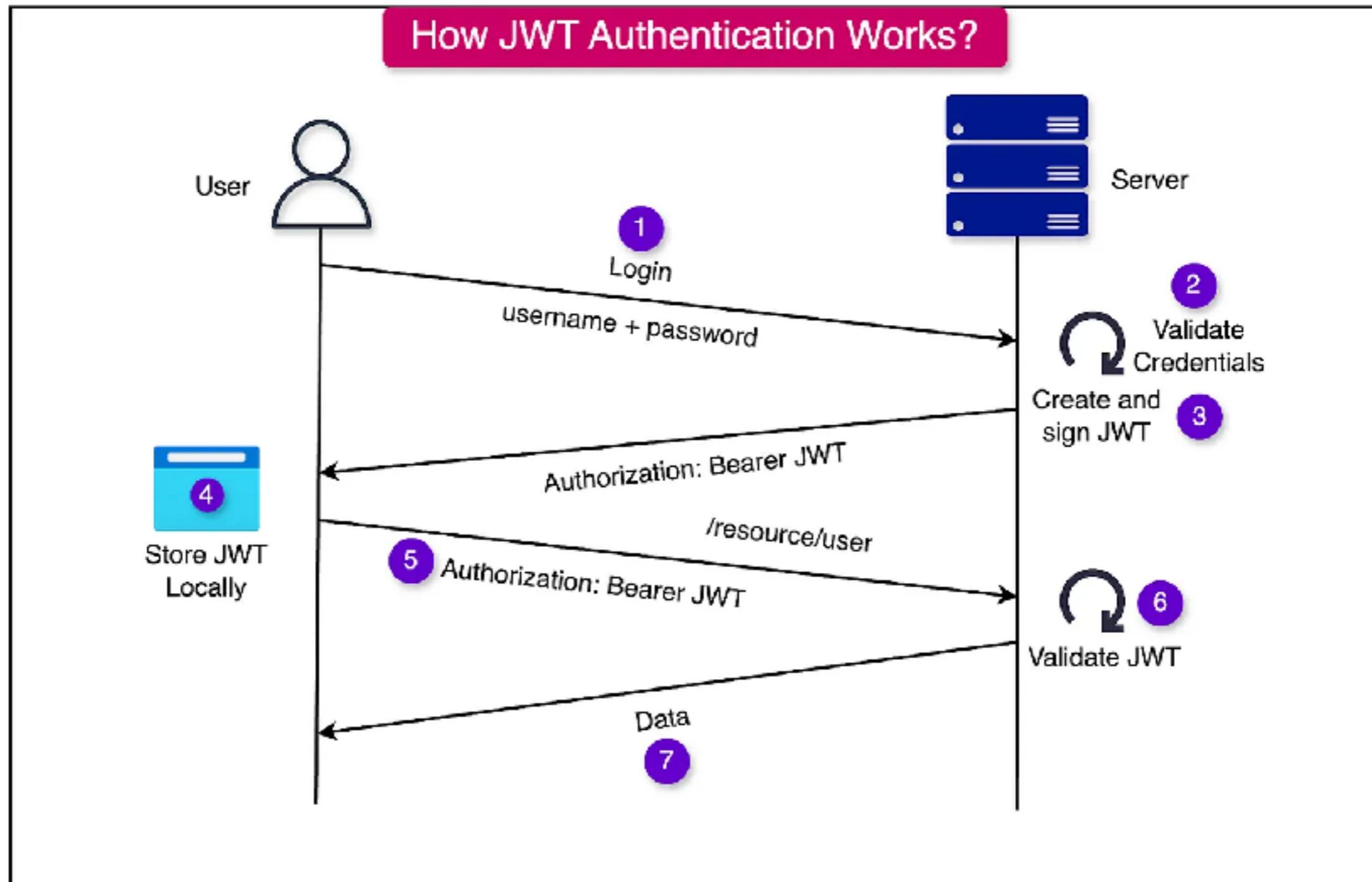
 **Signature Verified**

**SHARE JWT**

<https://jwt.io/>



# Working with JWT



<https://blog.bytebytogo.com/p/mastering-modern-authentication-cookies>



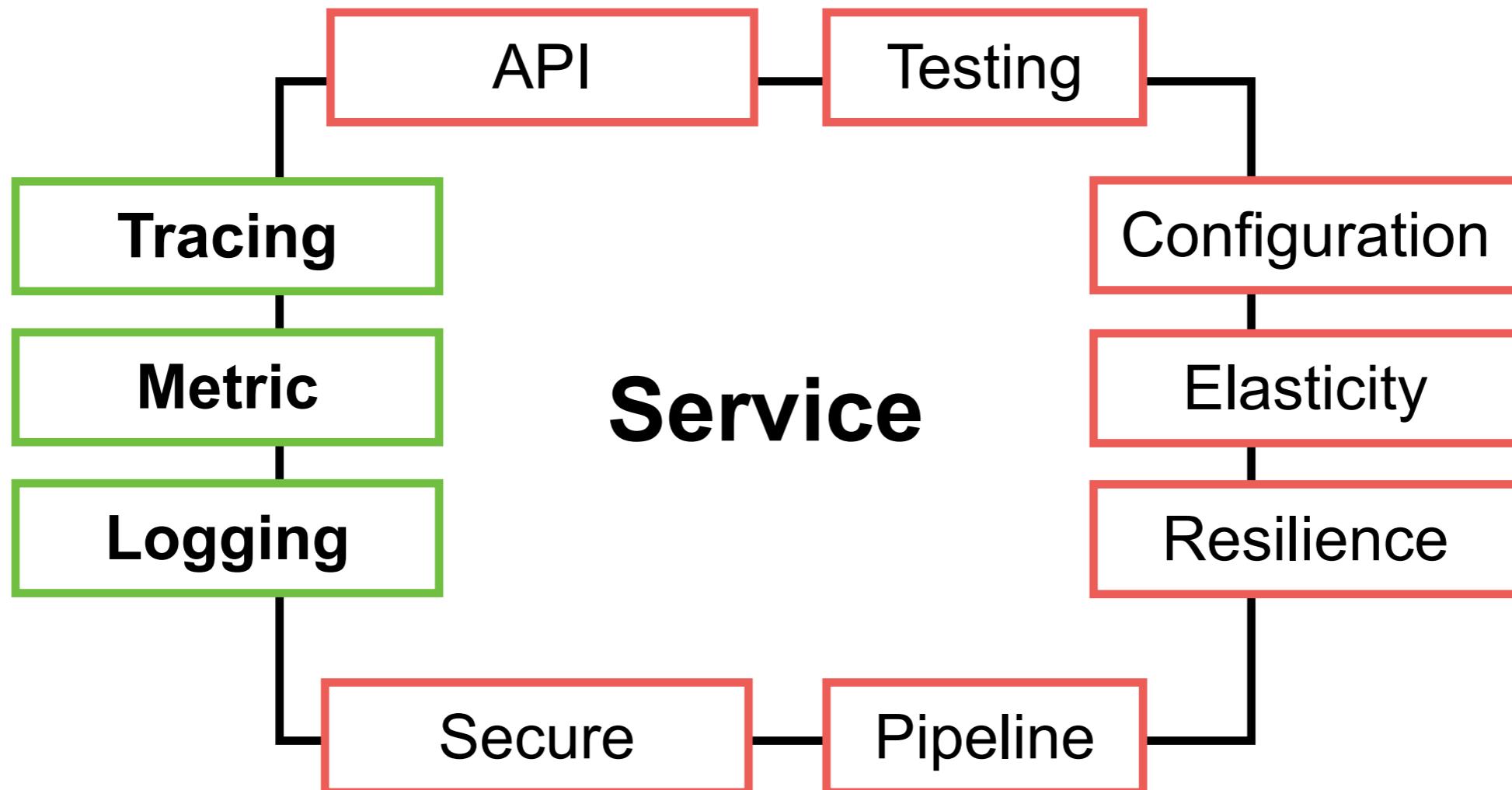
Go workshop

© 2022 - 2024 Siam Chamnkit Company Limited. All rights reserved.

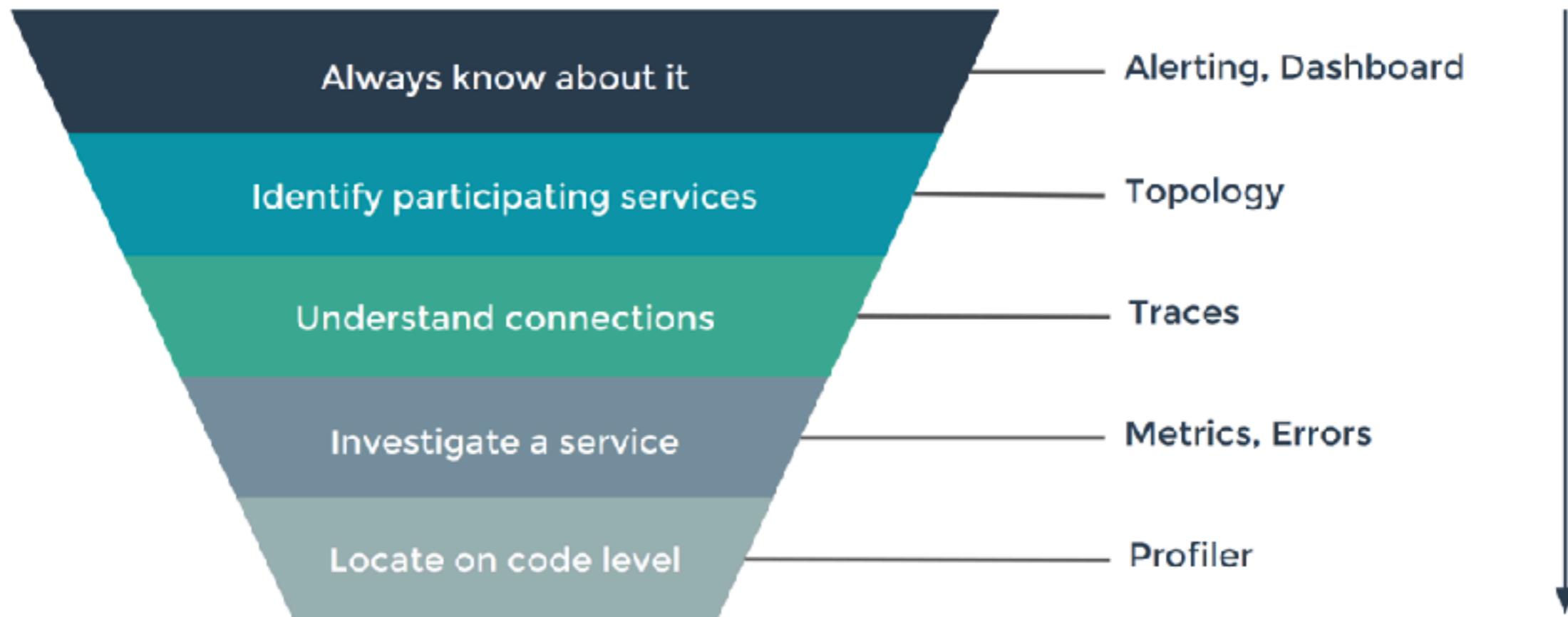
# Observable Service



# Properties of Service



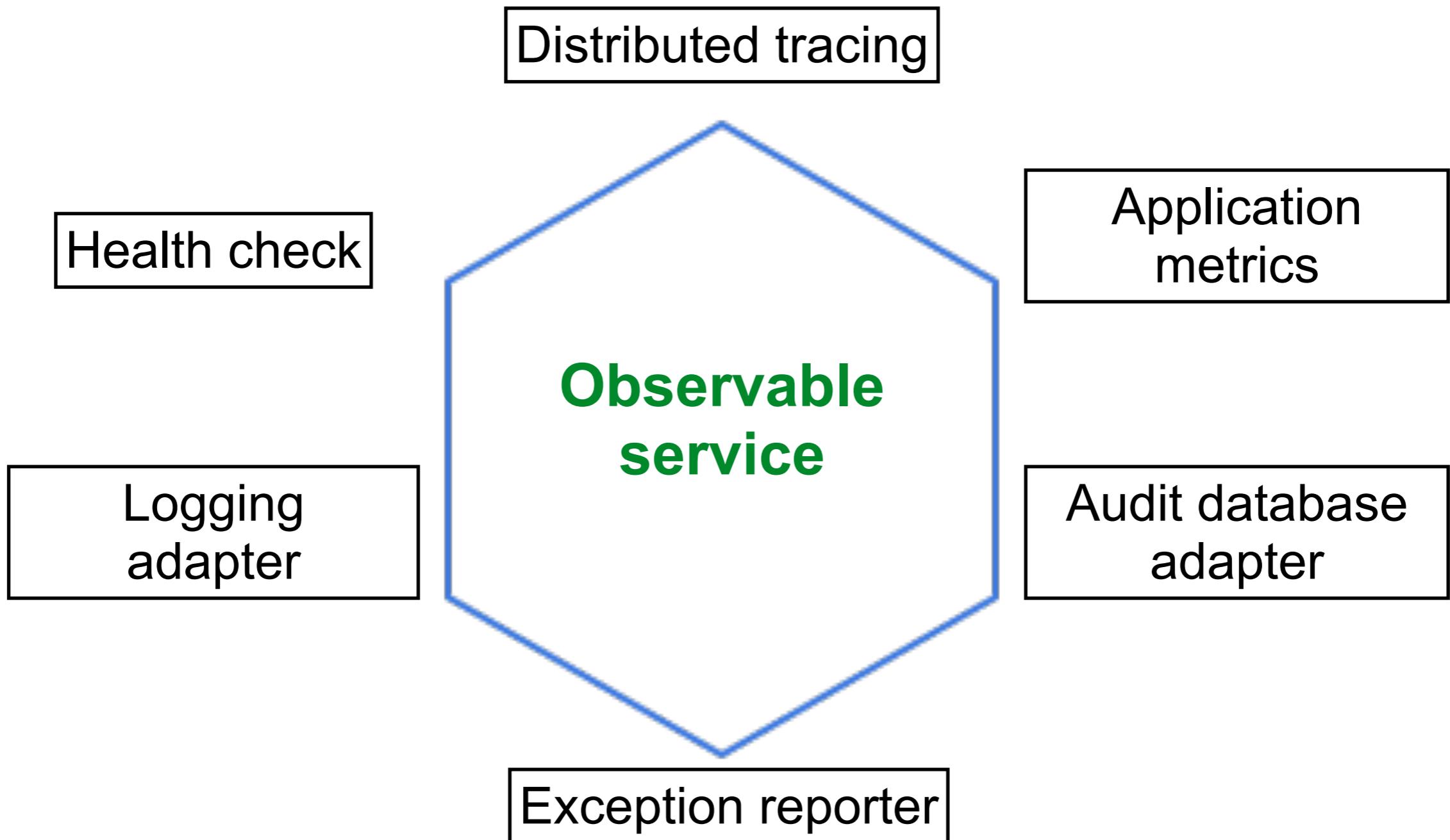
# How to find an issue ?



# Observability vs Monitoring

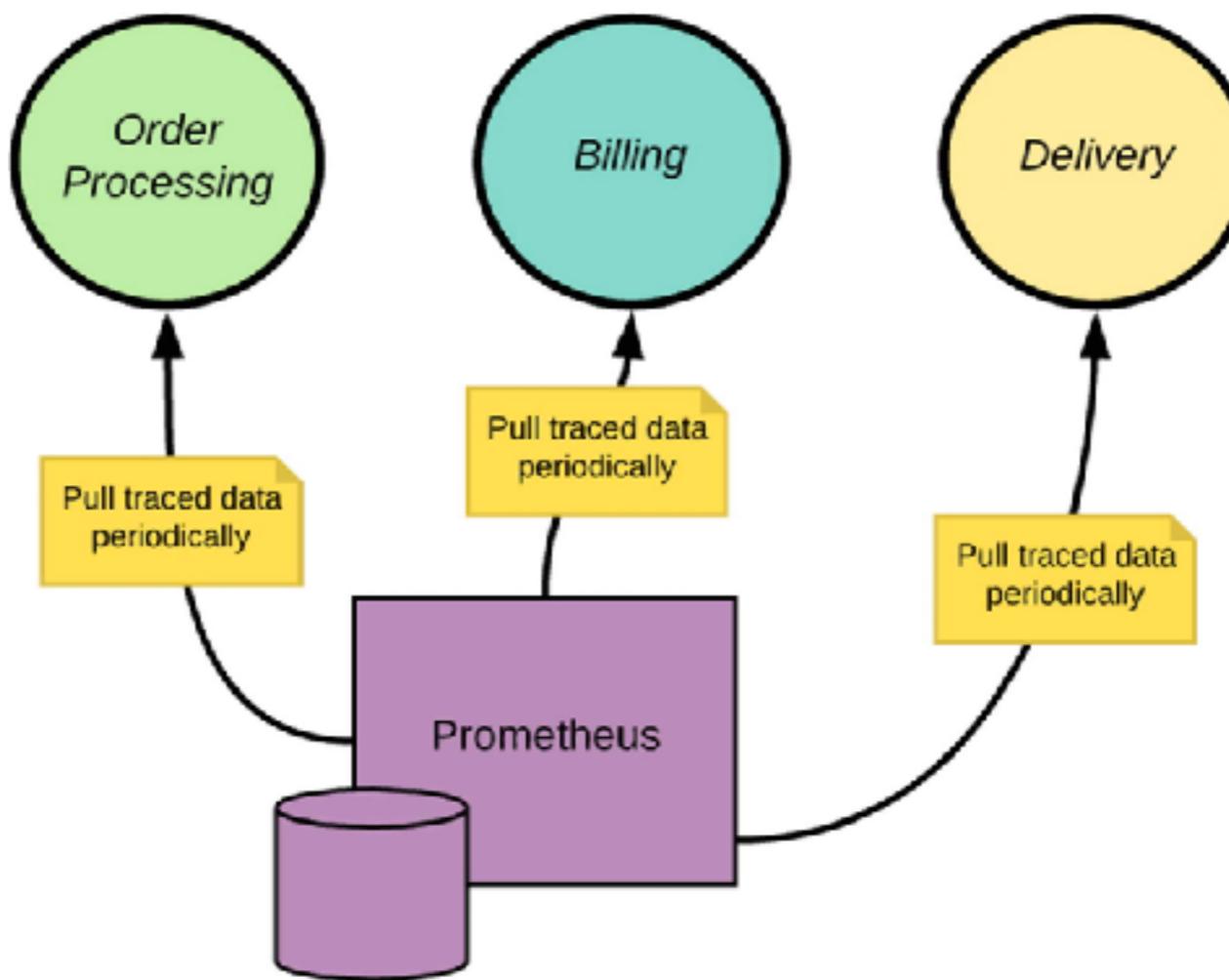


# Observable services



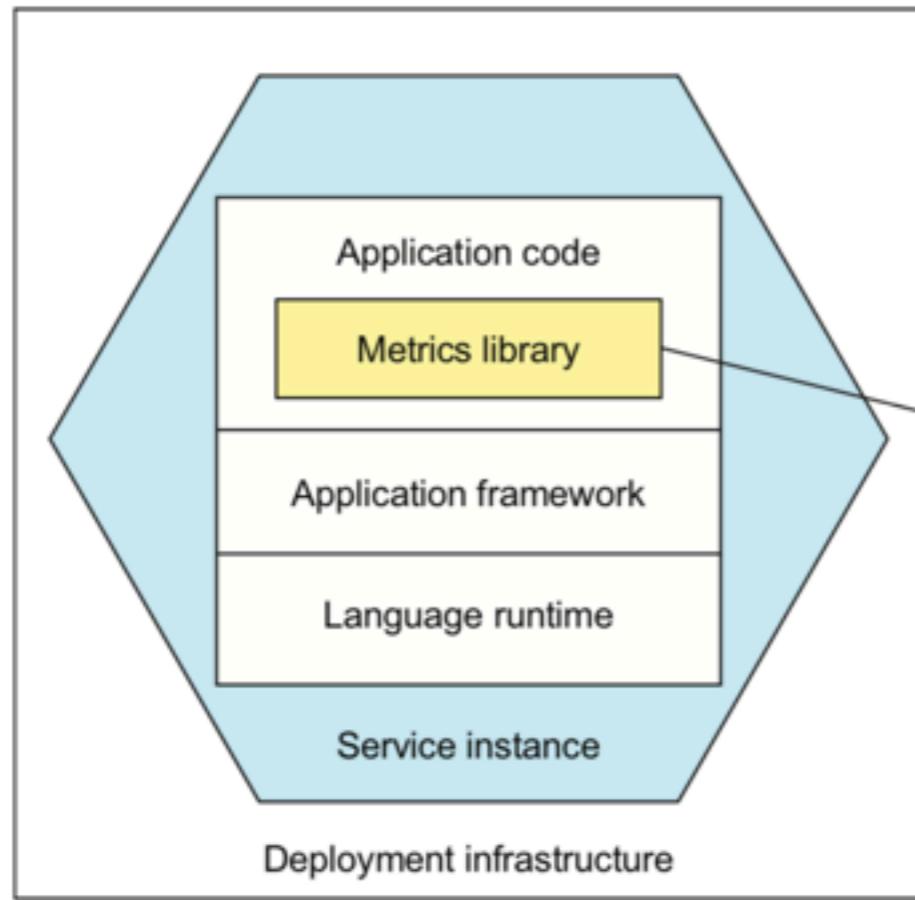
# Application metrics

Services maintain metrics and expose to metric server (counters, gauges)

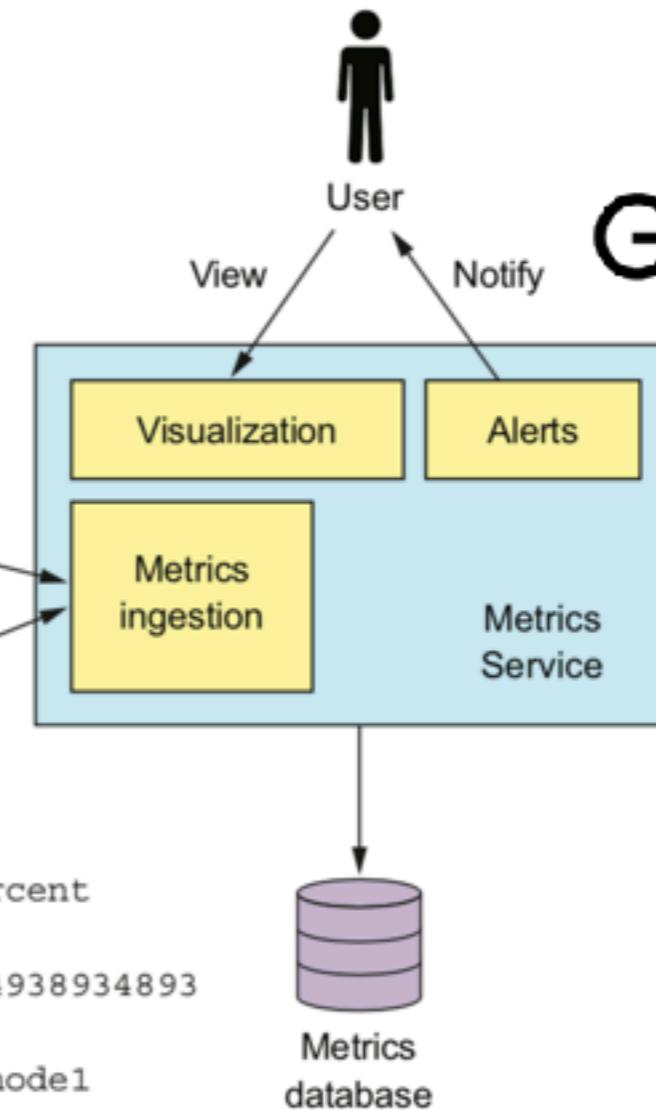


# Application metrics tools

## Spring Boot Actuator



Metrics sample:  
name=cpu\_percent  
value=68  
timestamp=34938934893  
dimensions:  
machine=node1  
...



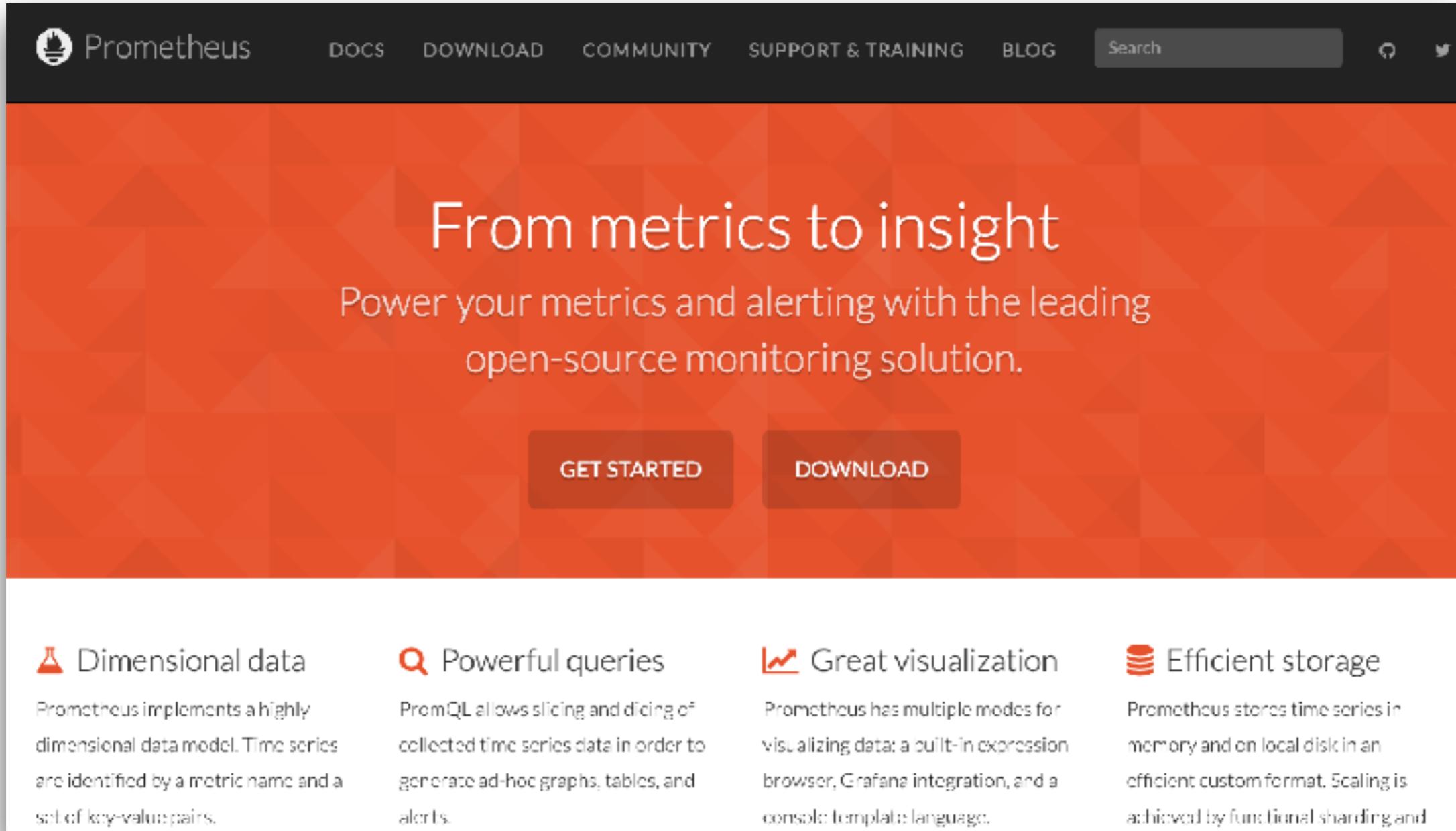
Grafana



Prometheus



# Prometheus



The screenshot shows the official Prometheus website. At the top, there's a dark navigation bar with the Prometheus logo, a search bar, and social media links for GitHub and Twitter. The main heading "From metrics to insight" is prominently displayed in white on an orange background. Below it, a subtitle reads "Power your metrics and alerting with the leading open-source monitoring solution." Two large buttons, "GET STARTED" and "DOWNLOAD", are visible. The bottom section features four cards with icons and descriptions: "Dimensional data" (Prometheus implements a highly dimensional data model), "Powerful queries" (PromQL allows slicing and dicing of collected time series data), "Great visualization" (Prometheus has multiple modes for visualizing data), and "Efficient storage" (Prometheus stores time series in memory and on local disk in an efficient custom format). Each card also includes a short explanatory paragraph.

Dimensional data

Prometheus implements a highly dimensional data model. Time series are identified by a metric name and a set of key-value pairs.

Powerful queries

PromQL allows slicing and dicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts.

Great visualization

Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.

Efficient storage

Prometheus stores time series in memory and on local disk in an efficient custom format. Scaling is achieved by functional sharding and

<https://prometheus.io/>



# Grafana

Grafana Labs Products Open source Solutions Learn Company Downloads Contact us Sign in

Compose and scale observability with one or all pieces of the stack

Your observability wherever you need it

Cloud

Self-managed

API

Grafana

Plugins

Dashboards

Alerts

Usage insights

Reports

Governance

Metrics

Logs

Traces

Applications

Infrastructure

Cloud

Self-managed

<https://grafana.com/>

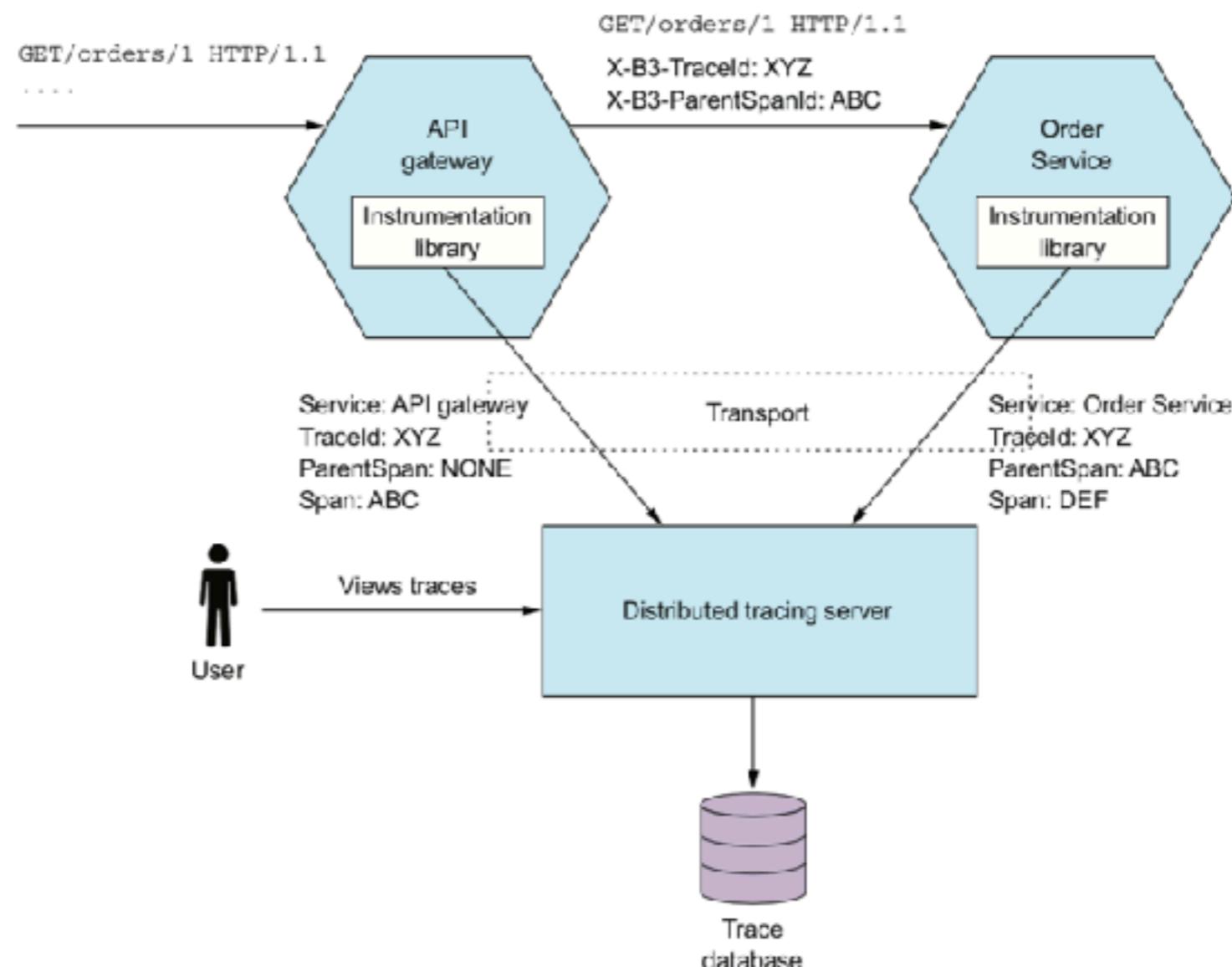


Workshop

© 2020 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Distributed tracing

Assign each external request a unique ID and trace requests as flow between services



# Distributed tracing tools

Format standard with OpenTelemetry  
Zipkin, Jaeger, Tempo, AWS X-Ray, Elastic APM



JAEGER



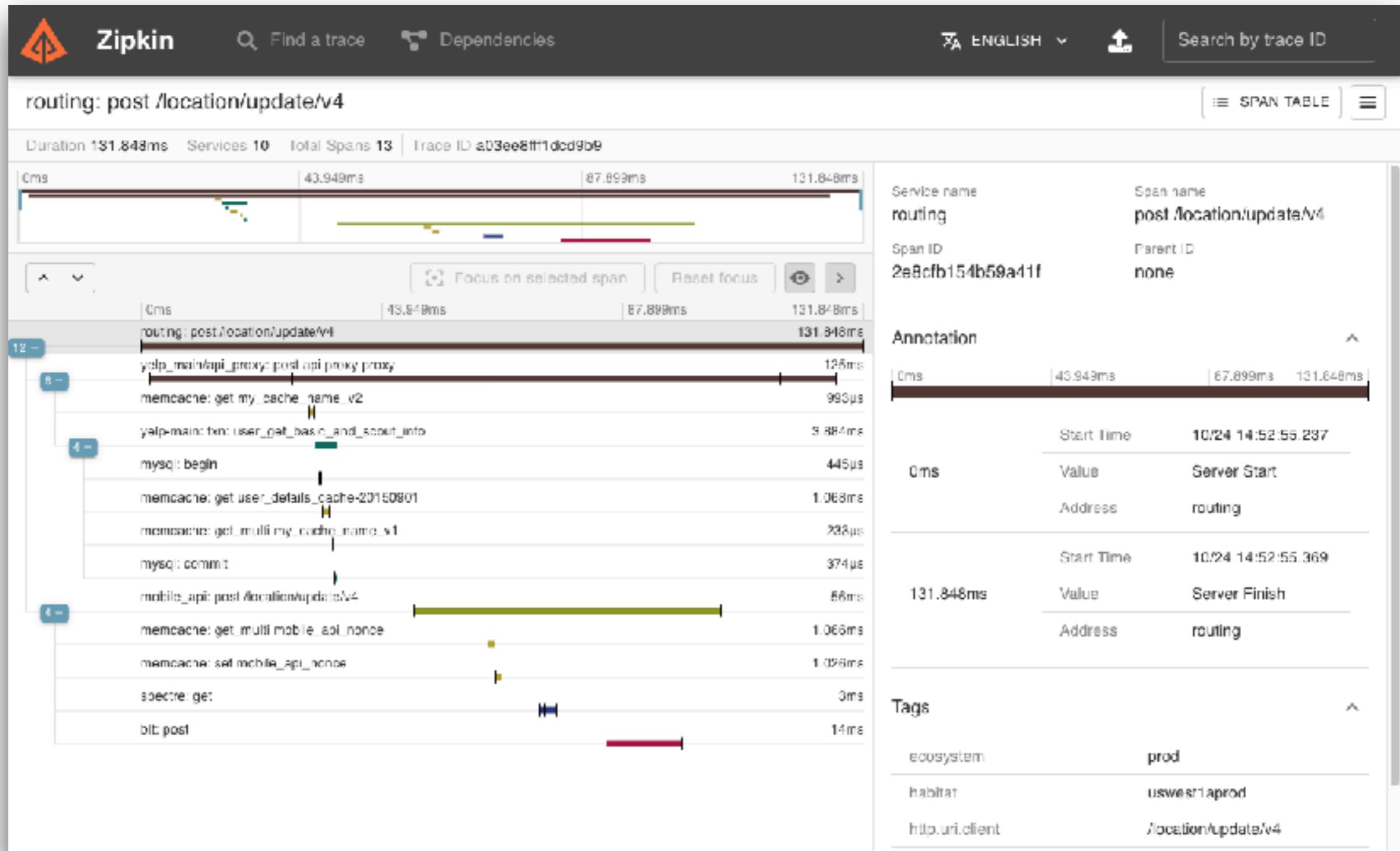
Grafana Tempo



OpenTelemetry



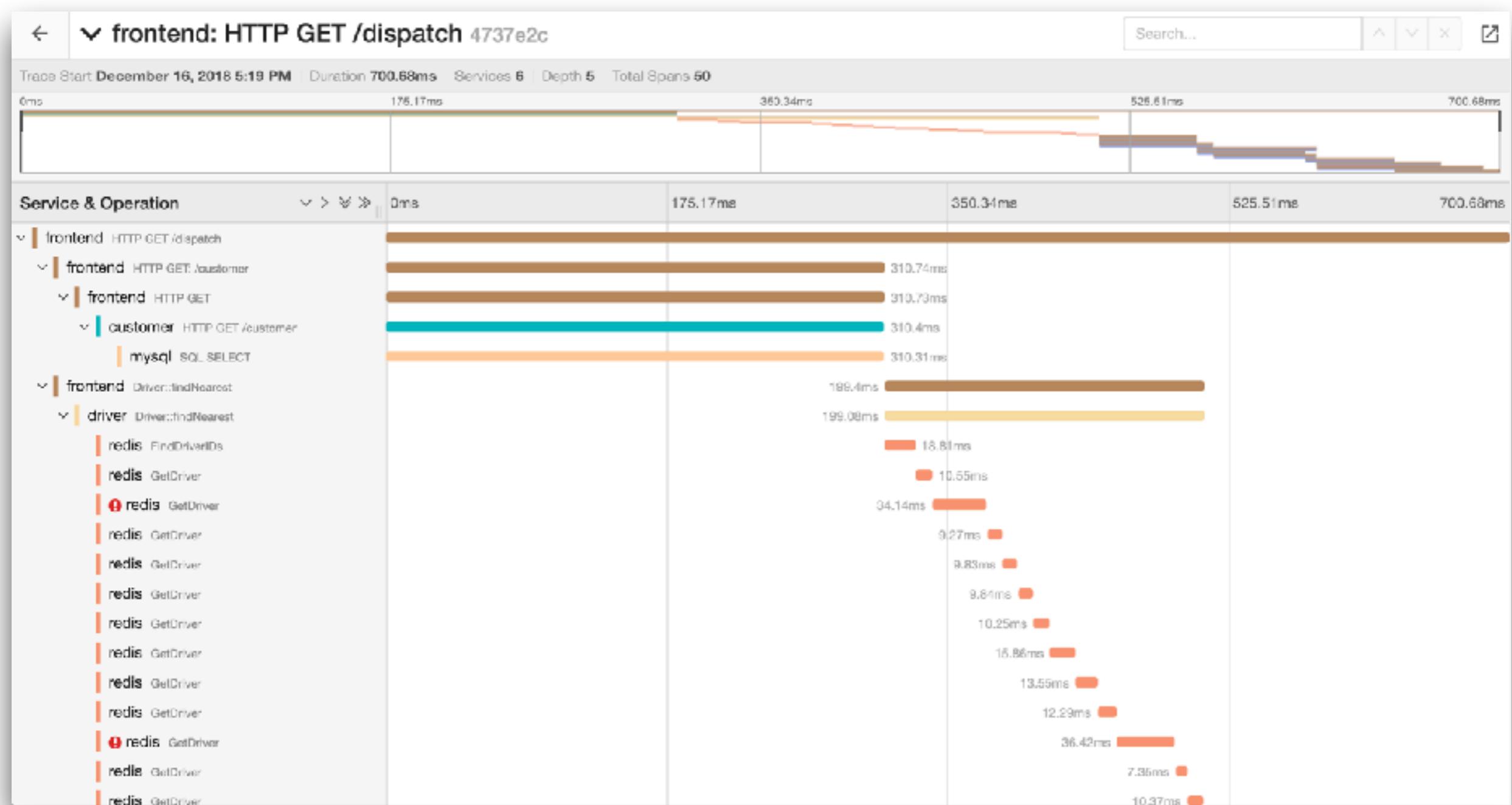
# Zipkin



<https://zipkin.io/>



# Jaeger



<https://www.jaegertracing.io>

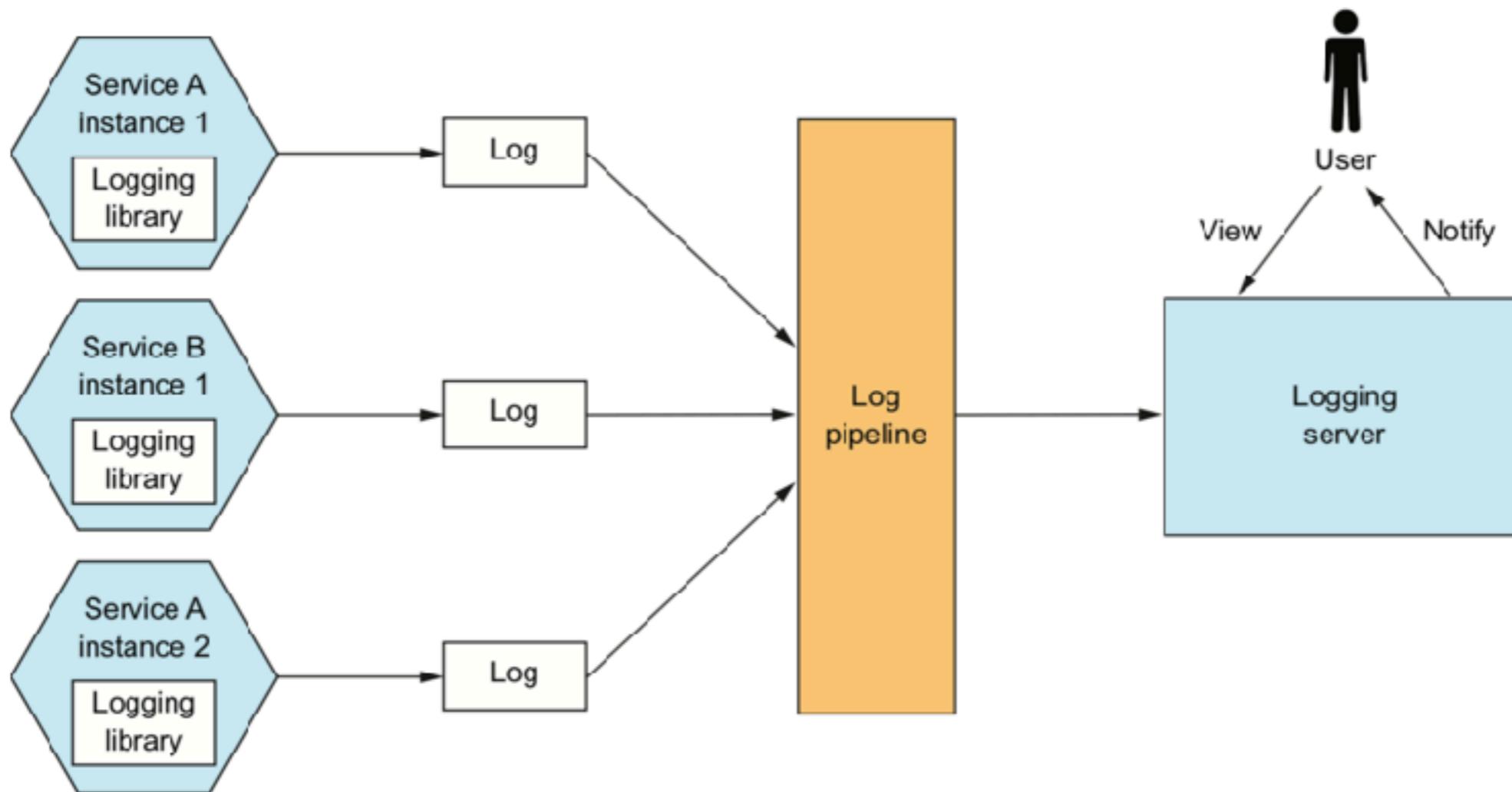


Workshop

© 2020 - 2024 Siam Chamnkit Company Limited. All rights reserved.

# Centralized logging

Log service activity and write logs into a centralized logging server. (searching, alerting)



# Effective Logging

Define event to log

Use structured logging

Exclude sensitive information

Log at the correct level

Be specific in your message

Don't log large message

Make sure you keep trace Id in the log

[https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Logging\\_Cheat\\_Sheet.md](https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Logging_Cheat_Sheet.md)



# Consistent Structure across all logs

Property	Description	Example
<b>Timestamp</b>	Date and time of the log	2023-07-01
<b>Log level</b>	DEBUG, INFO, ERROR	
<b>Trace Id or Correlation Id</b>	Unique identifier that refer to other logs from all services	
<b>Event/Action Name</b>	Identify to event or action of log	Authentication fail
<b>Service ID/ Name</b>	Identify to service	
<b>Request path</b>	Path for the request	/api/products

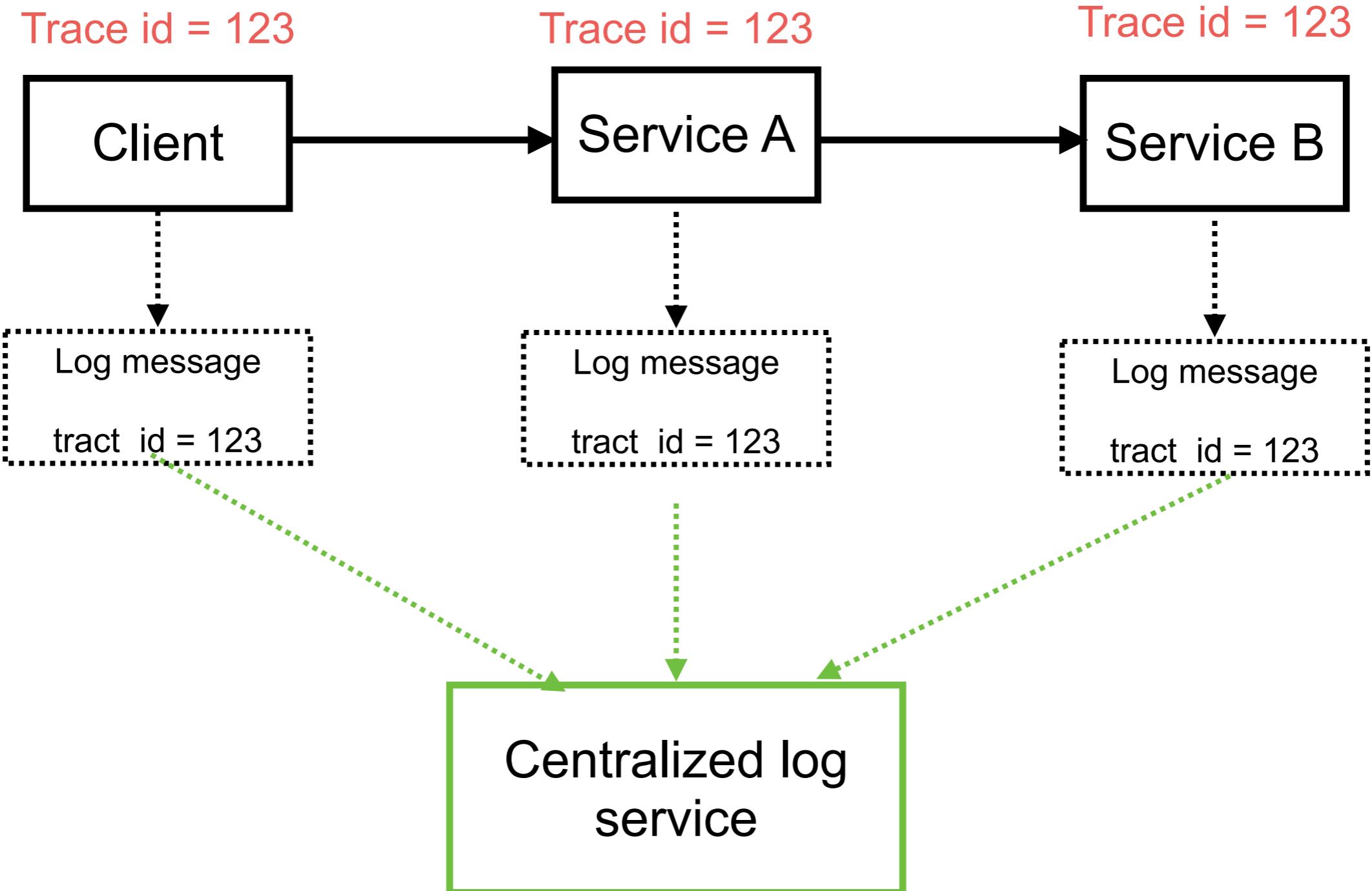


# Don't keep !!

```
com.framework.FrameworkException: Error in web request
  at com.framework.ApplicationStarter.lambda$start$0(ApplicationStarter.java:15)
  at spark.RouteImpl$1.handle(RouteImpl.java:72)
  at spark.http.matching.Routes.execute(Routes.java:61)
  at spark.http.matching.MatcherFilter.doFilter(MatcherFilter.java:134)
  at spark.embeddedserver.jetty.JettyHandler.doHandle(JettyHandler.java:50)
  at org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.java:1568)
  at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:144)
  at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:132)
  at org.eclipse.jetty.server.Server.handle(Server.java:503)
  at org.eclipse.jetty.server.HttpChannel.handle(HttpChannel.java:364)
  at org.eclipse.jetty.server.HttpConnection.onFillable(HttpConnection.java:260)
  at org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:305)
  at org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:103)
  at org.eclipse.jetty.io.ChannelEndPoint$2.run(ChannelEndPoint.java:118)
  at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:765)
  at org.eclipse.jetty.util.thread.QueuedThreadPool$2.run(QueuedThreadPool.java:683)
  at java.base/java.lang.Thread.run(Thread.java:834)
Caused by: com.project.module.MyProjectFooBarException: The number of FooBars cannot be zero
  at com.project.module.MyProject.anotherMethod(MyProject.java:20)
  at com.project.module.MyProject.someMethod(MyProject.java:12)
  at com.framework.ApplicationStarter.lambda$start$0(ApplicationStarter.java:13)
  ... 16 more
Caused by: java.lang.ArithmetricException: The denominator must not be zero
  at org.apache.commons.lang3.math.Fraction.getFraction(Fraction.java:143)
  at com.project.module.MyProject.anotherMethod(MyProject.java:18)
  ... 18 more
```



# Centralized logging



# Observable Service Workshop

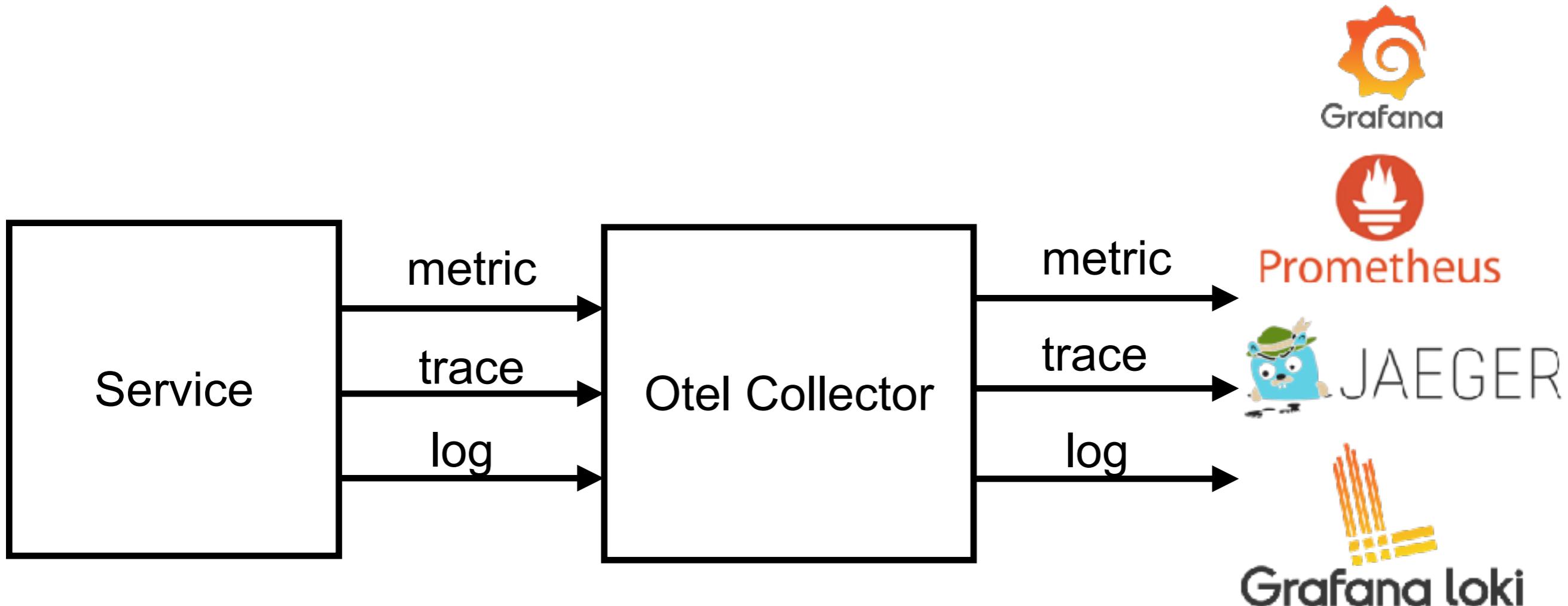
Metric

Tracing

Logging



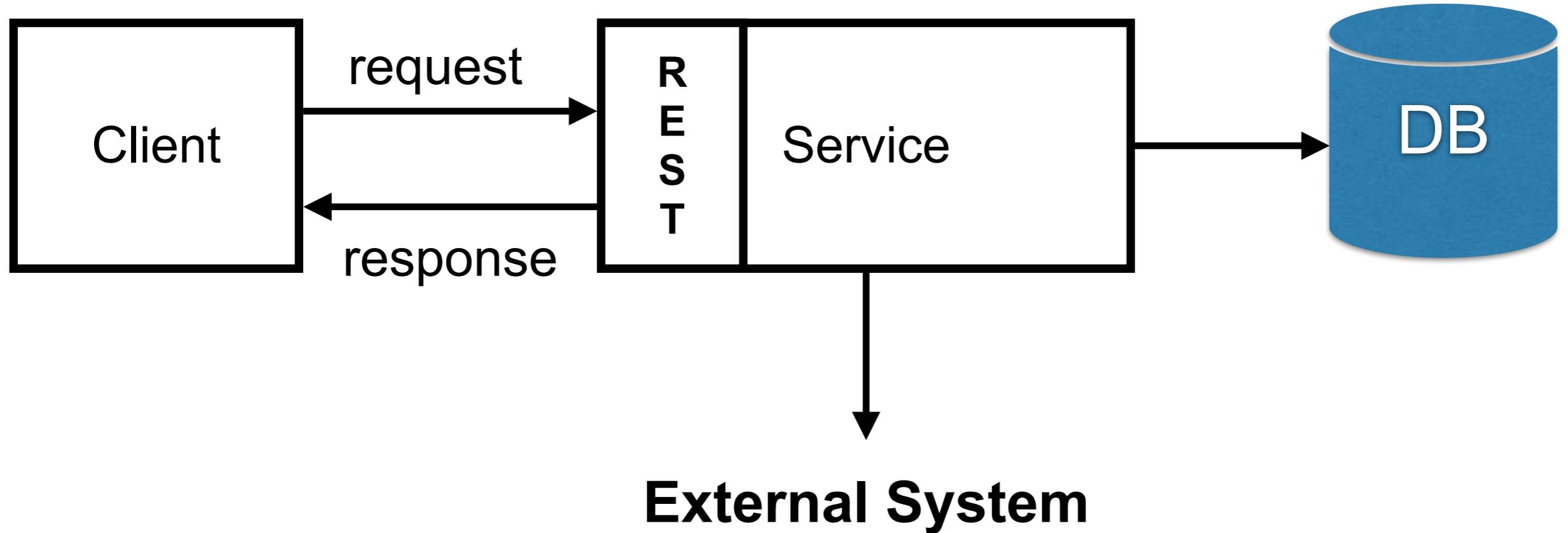
# Working with OpenTelemetry



# Develop RESTful API



# Structure of Service



# TypeScript



Download Docs Handbook Community Tools

Search Docs

## TypeScript is JavaScript with syntax for types.

TypeScript is a strongly typed programming language that builds on JavaScript, giving you better tooling at any scale.

Try TypeScript Now

Online or via npm

...

[Editor Checks](#) [Auto-complete](#) [Interfaces](#) [JSX](#)

```
const user = {  
    firstName: "Angela",  
    lastName: "Davis",  
    role: "Professor",  
}
```

```
console.log(user.name)
```

Property 'name' does not exist on type '{  
 firstName: string; lastName: string; role:  
 string; }'.



<https://www.typescriptlang.org/>



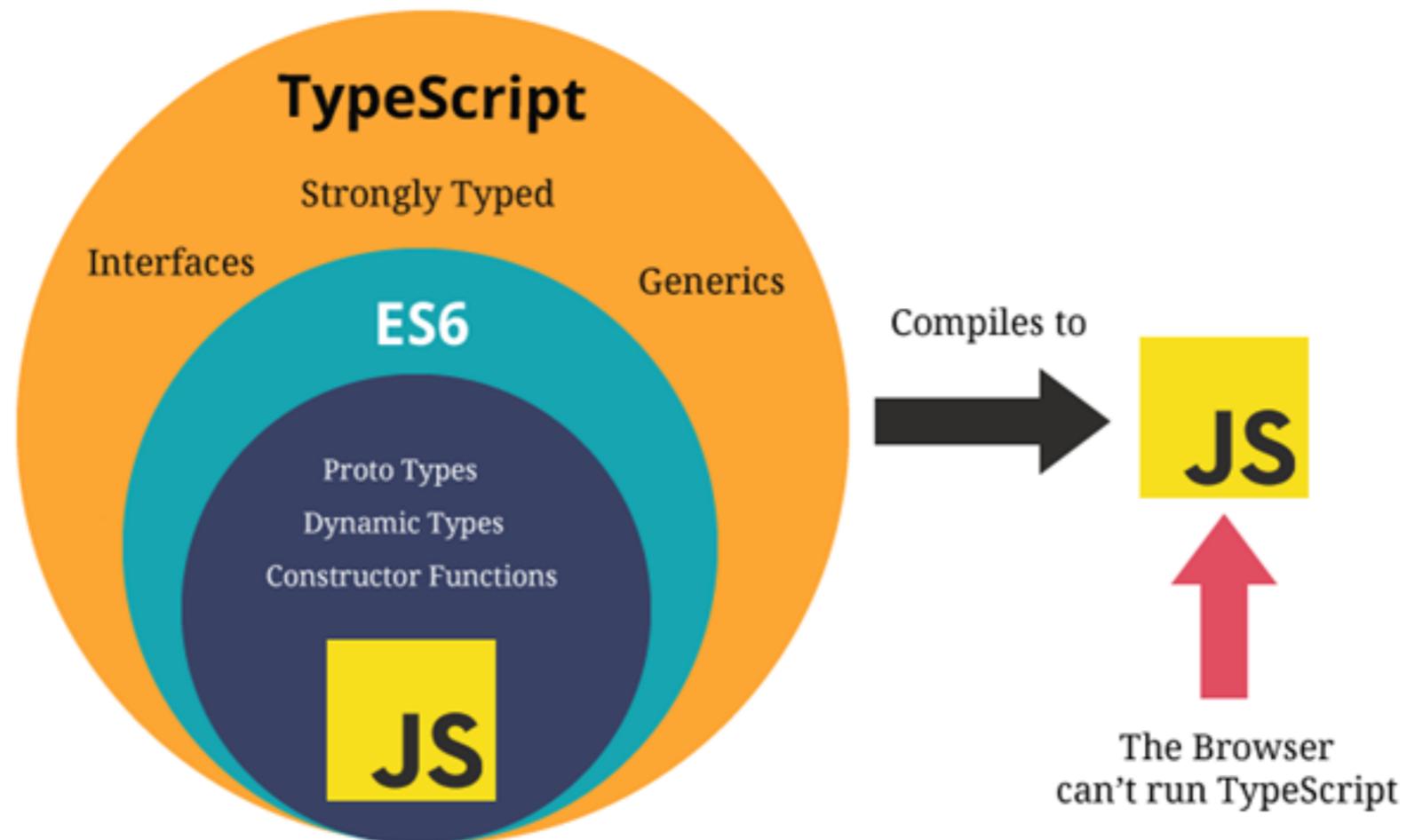
NodeJS

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

250

# TypeScript

A superset of JavaScript



<https://www.typescriptlang.org/>



# TypeScript Essentials

Basic

Type in TypeScript

Advance

Design patterns



# Key idea of TypeScript

It adds **static-typing** to JavaScript

More maintainable

Easy to learn and use



# Software Requirements

NodeJS 18+

The screenshot shows the official Node.js website at <https://nodejs.org/en>. The main heading is "Run JavaScript Everywhere". Below it, a paragraph describes Node.js as a free, open-source, cross-platform JavaScript runtime environment. A green button labeled "Download Node.js (LTS)" is visible. To the right, a code editor window displays a script named "server.mjs" for creating an HTTP server. The code uses ES6 syntax like import statements and arrow functions. Below the code, there are tabs for "JavaScript" and a "Copy to clipboard" button. A search bar and navigation menu are also present at the top.

```
1 // server.mjs
2 import { createServer } from 'node:http';
3
4 const server = createServer((req, res) =>
5   res.writeHead(200, { 'Content-Type': 'text/plain' });
6   res.end('Hello World!\n');
7 );
8
9 // starts a simple http server locally on
10 server.listen(3000, '127.0.0.1', () => {
11   console.log('Listening on 127.0.0.1:3000');
12 });
13
14 // run with 'node server.mjs'
```

<https://nodejs.org/en>



# Software Requirements

## Node Version Manager



<https://github.com/nvm-sh/nvm>

<https://github.com/coreybutler/nvm-windows>



# Create project (1)

```
$npm install -g typescript
```

```
$tsc --init
```



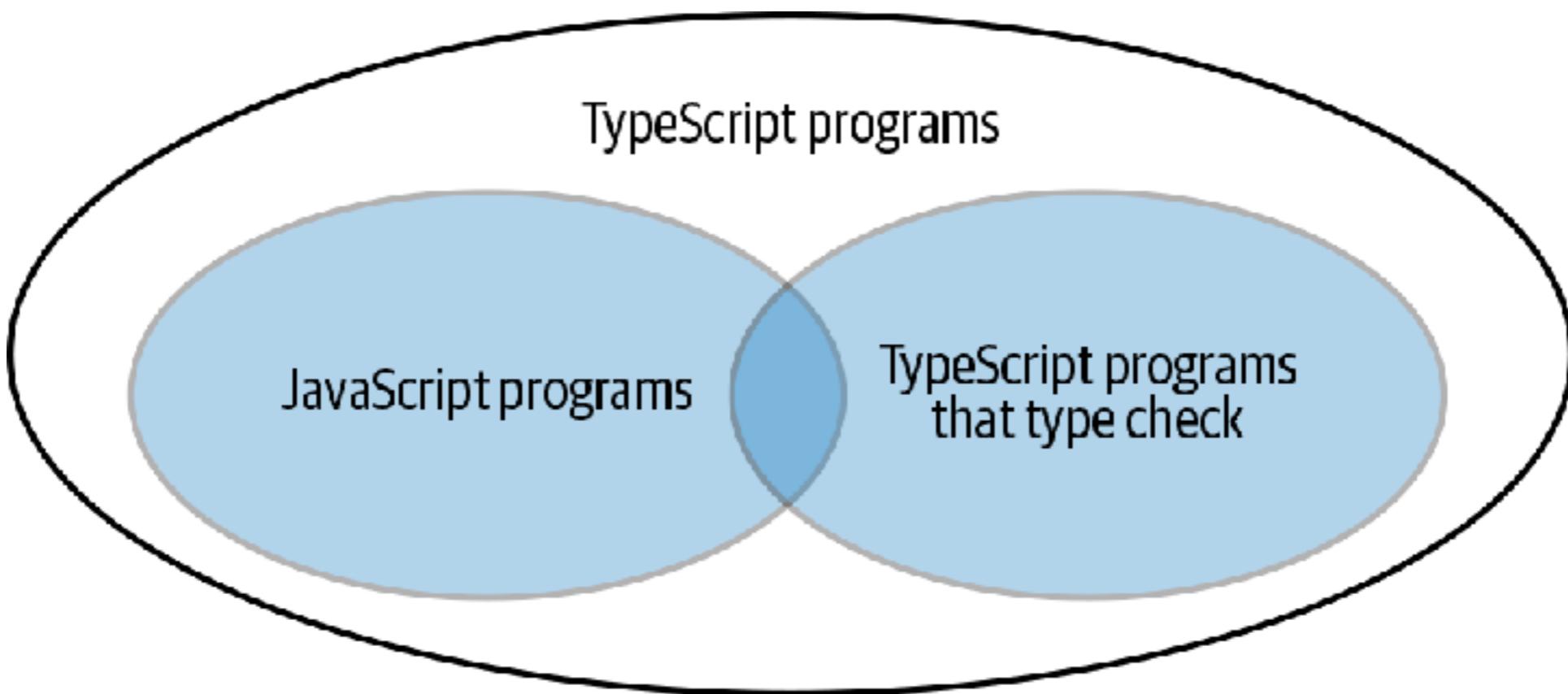
# Create project (2)

\$npm init

\$npm install typescript -S



# Type Checking

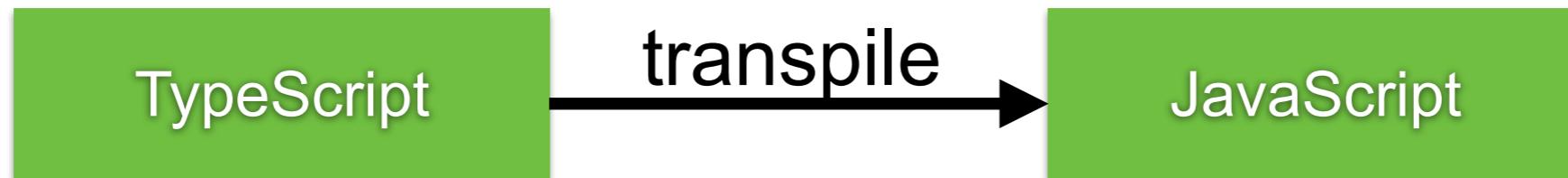


# Hello TS

\$tcs hello.ts

\$node hello.js

```
let message: string = "Hello, World!";
console.log(message);
```



# Variable Declarations

var , let, const

	var	let	const
Stored in Global Scope	✓	✗	✗
Function Scope	✓	✓	✓
Block Scope	✗	✓	✓
Can Be Reassigned?	✓	✓	✗
Can Be Redeclared?	✓	✗	✗
Can Be Hoisted?	✓	✗	✗



# Function declaration

```
// Simple function
function add(a: number, b: number): number {
    return a + b;
}

// Arrow function
const add2 = (a: number, b: number): number => a + b;
```



# Data Types

```
let isDone: boolean = false;
let decimal: number = 6;
let color: string = "blue";
let myTuple: [string, number] = ["hello", 10];
let myArray: number[] = [1, 2, 3];
let myObject: { name: string, age: number }
  = { name: "John", age: 30 };

let myAny: any = 4;
let nothing: undefined = undefined;
let myNull: null = null;
```



# Custom Object Types

```
// Type declaration
type Person = {name: string, age?: number, email?: string};

// Variable declaration
let p1: Person = {name: 'demo', };
let p2: Person = {name: 'demo', age: 23};
let p3: Person = {name: 'demo', age: 23, email: ''};

p1.name = "new name";
p1.age = 30;
```



# Object-Oriented Programming

## Class Interface



# Class

## Blueprint for creating objects

```
class Person {  
    name: string;  
    age: number;  
  
    constructor(name: string, age: number) {  
        this.name = name;  
        this.age = age;  
    }  
  
    greet() {  
        console.log(`Hello, my name is ${this.name} and I'm ${this.age} years old.`);  
    }  
}  
  
let john = new Person("John", 30);  
john.greet();
```



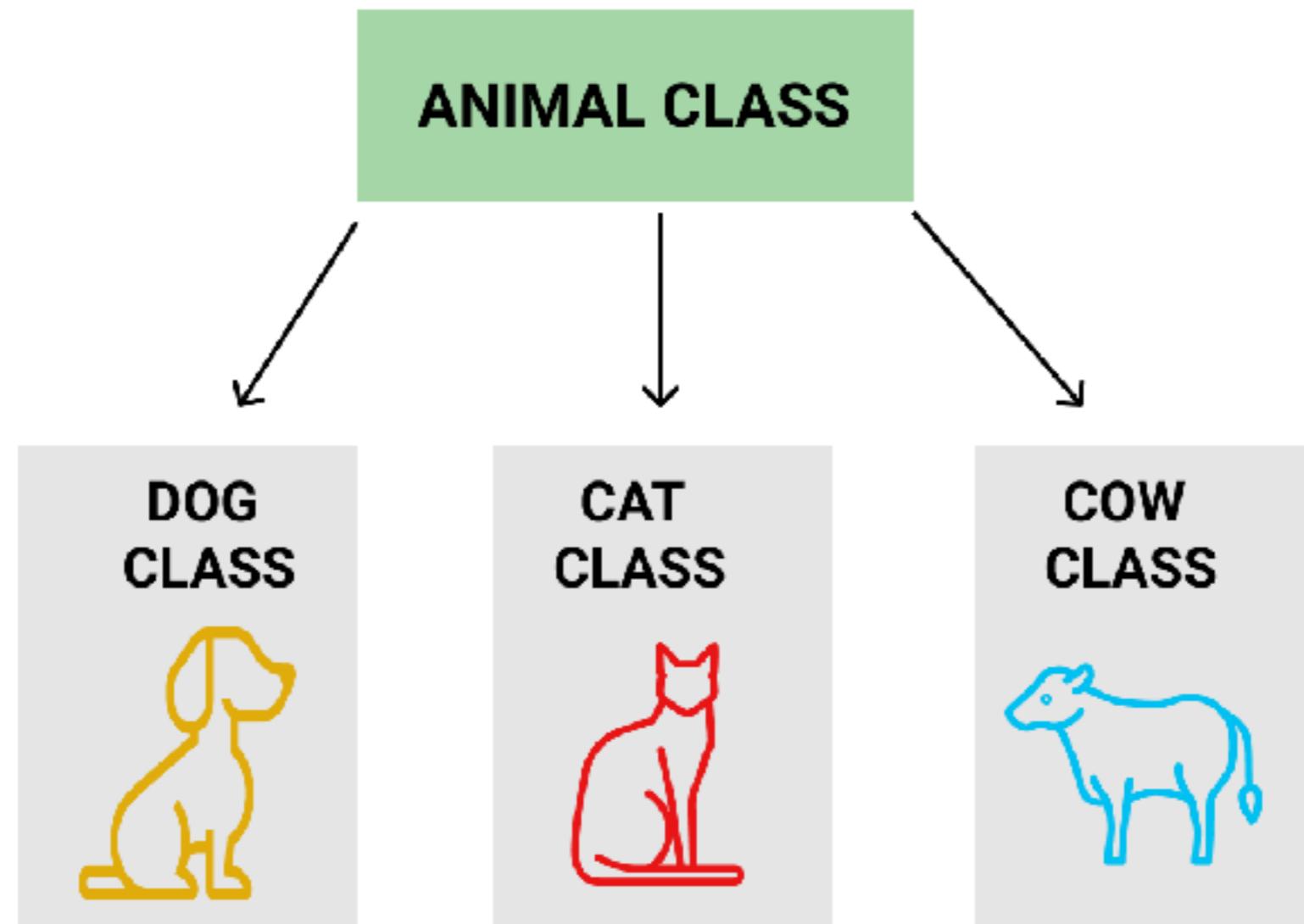
# Class

## Shorthand constructor

```
class Person {  
  constructor(public name: string, public age: number) {}  
  greet() {  
    console.log(`Hello, my name is ${this.name} and I'm ${this.age} years  
old.`);  
  }  
}  
  
let john = new Person("John", 30);  
john.greet();
```



# Inheritance



# Visibility of class member

Public (default)

Protected

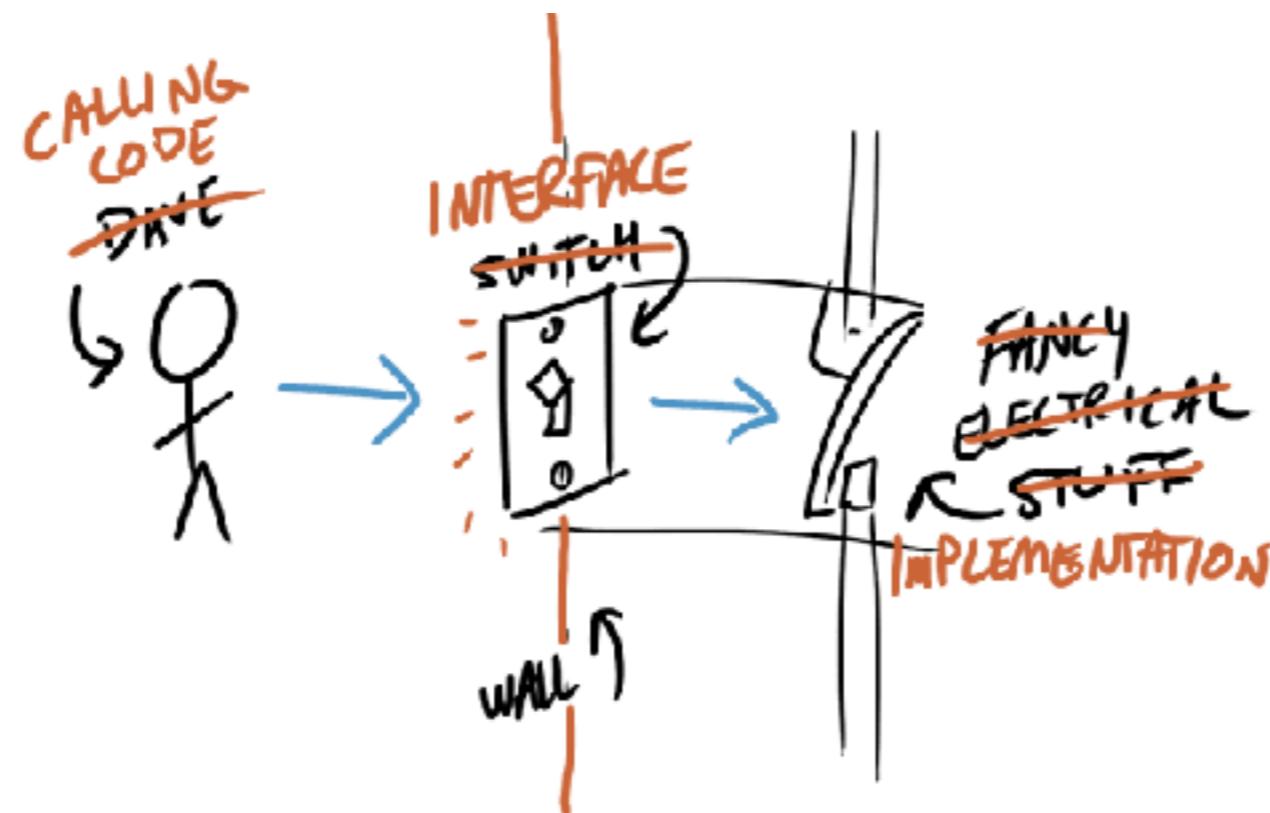
Private

Readonly



# Interface (Encapsulation)

Contract for a class or function  
Can't create an object/instance  
Used for type checking



# Interface

```
interface IPerson {  
    name: string;  
    age: number;  
    greet(): void;  
}
```

```
class Person implements IPerson {  
    name: string;  
    age: number;  
  
    constructor(name: string, age: number) {  
        this.name = name;  
        this.age = age;  
    }  
  
    greet() {  
        ...  
    }  
}
```



# Generic

Use in function and class  
Reusable and flexible !!

```
class QueueV1 {  
    private items: string[];  
  
    constructor() {  
        this.items = [];  
    }  
  
    push = (newItem: string) => this.items.push(newItem);  
    pop = () => this.items.shift();  
}
```



# Generic

Improve code with Generic way

```
class QueueWithGeneric<T> {
    private items: T[];

    constructor() {
        this.items = [];
    }

    push = (newItem: T) => this.items.push(newItem);
    pop = () => this.items.shift();
}
```



# Mapped types

Pick

Partial

Readonly

<https://www.typescriptlang.org/docs/handbook/2/mapped-types.html>



# Pick some properties

```
type User = {  
    id: string,  
    name: string,  
    age: number  
};  
  
type NewUserData = Pick<User, 'name'>;  
const user1: NewUserData = { name: 'John' };  
  
const user2: User = {  
    ...user1,  
    id: '1',  
    age: 20  
};
```

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread\\_syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax)



NodeJS

© 2017 - 2018 Siam Chamnkit Company Limited. All rights reserved.

274

# Partial

```
type User = {  
    id: string,  
    name: string,  
    age: number  
};  
  
type PartialUser = Partial<User>;  
const user1: PartialUser = {  
    name: 'John',  
};
```

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread\\_syntax](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax)



# ReadOnly

```
type ReadonlyUser = Readonly<User>;  
  
const user2: ReadonlyUser = {  
  id: '123',  
  name: 'John',  
  age: 30  
};  
  
user2.age = 31; // Error: Cannot assign to 'age'  
// because it is a read-only property.
```



# More configuration ...



# Configurations in project

tsconfig.json

Linting

Prettier

Husky



# tsconfig.json file

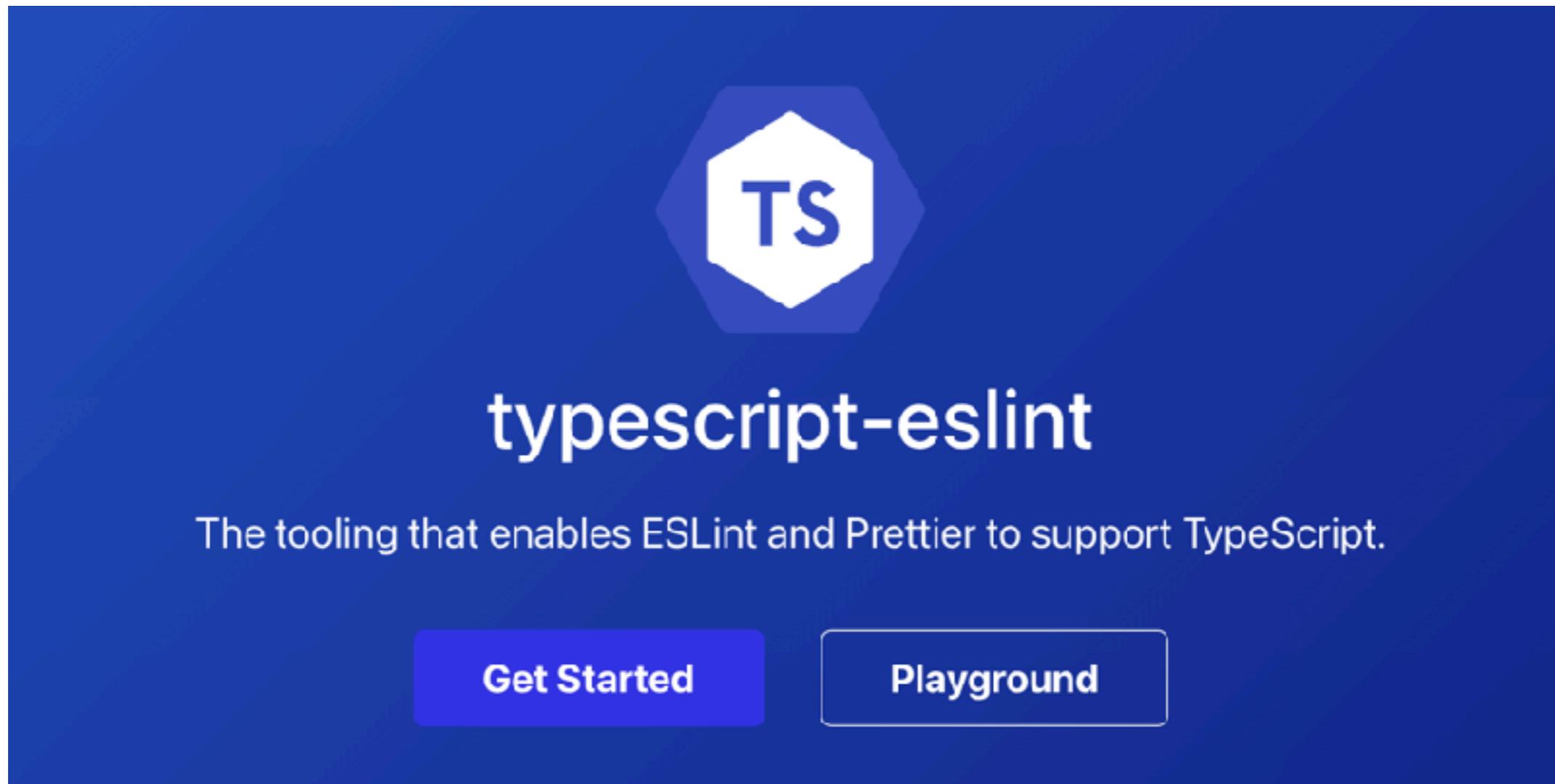
TypeScript configuration file  
Compiler settings

```
{  
  "include": ["src", "tests"],  
  "exclude": ["node_modules"],  
  
  "compilerOptions": {  
    "outDir": "build",  
    "target": "es6"  
  }  
}
```



# TSLint and ESLint

Enforce coding style



<https://typescript-eslint.io/>



# eslint.config.js file

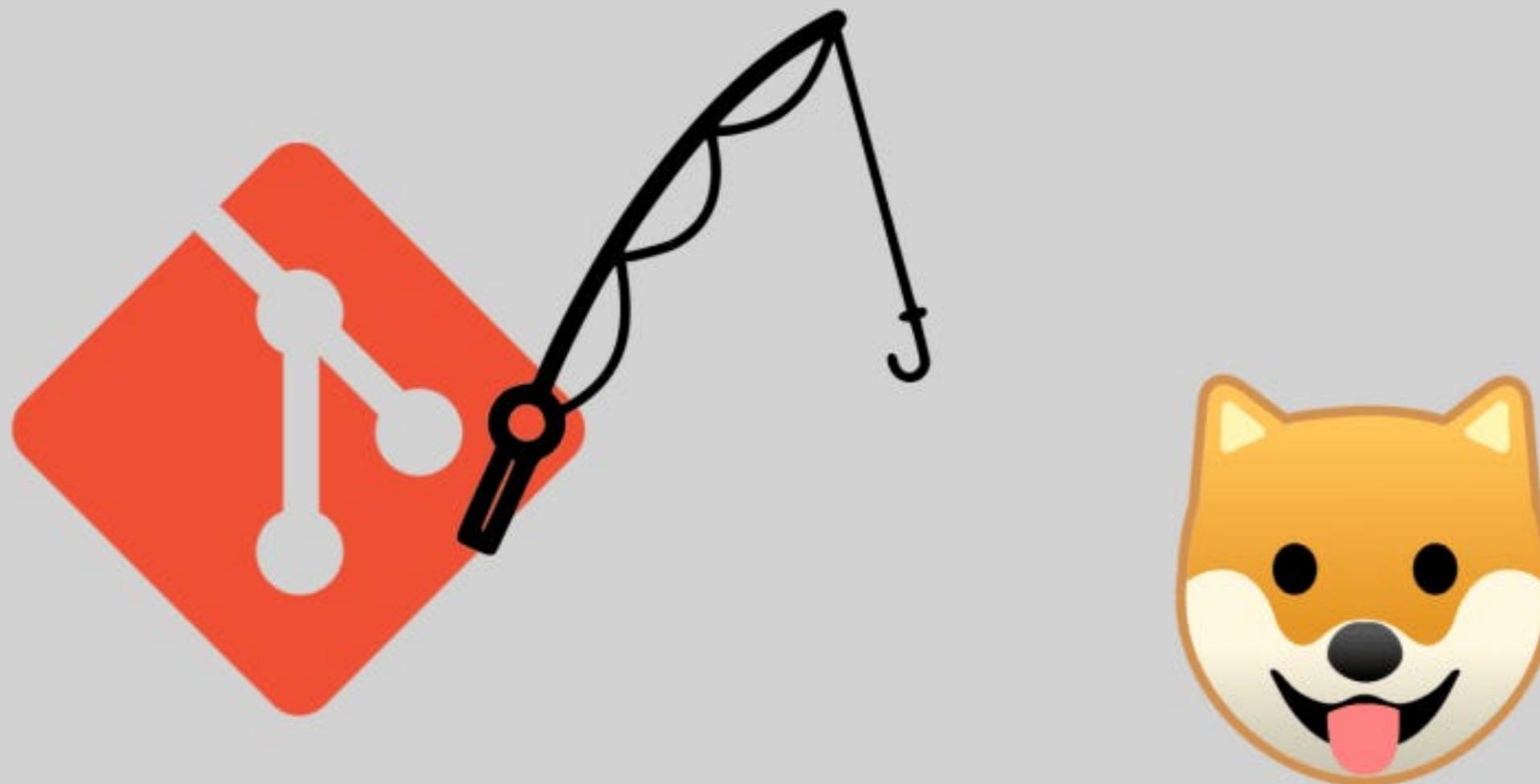
```
// @ts-check
import eslint from '@eslint/js';
import tseslint from 'typescript-eslint';

export default tseslint.config(
  eslint.configs.recommended,
  ...tseslint.configs.recommended,
);
```

\$npx eslint .



# GitHook with Husky



<https://typicode.github.io/husky/>



# Learning from Test



Jest



# Jest

A mockup of the Jest website landing page. At the top, there's a navigation bar with three cards: 'PASS' (green), 'PASS' (green), and 'JEST' (white). Below the navigation is a large green arrow pointing right. The main content area contains the following text:

Jest is a delightful JavaScript Testing Framework with a focus on simplicity.

It works with projects using: [Babel](#), [TypeScript](#), [Node](#), [React](#), [Angular](#), [Vue](#) and more!

<https://jestjs.io/>



# Install Jest in project

```
npm install -D jest @types/jest ts-jest ts-node  
npm test
```

```
"scripts": {  
  ...  
  "test": "jest"  
}
```

<https://jestjs.io/>



# Jest.config.ts

jest --init

```
import type {Config} from '@jest/types';
// Sync object
const config: Config.InitialOptions = {
  moduleDirectories: ['node_modules', 'src'],
  verbose: true,
  transform: {
    '^.+\\.tsx?$': 'ts-jest',
  }
};
export default config;
```

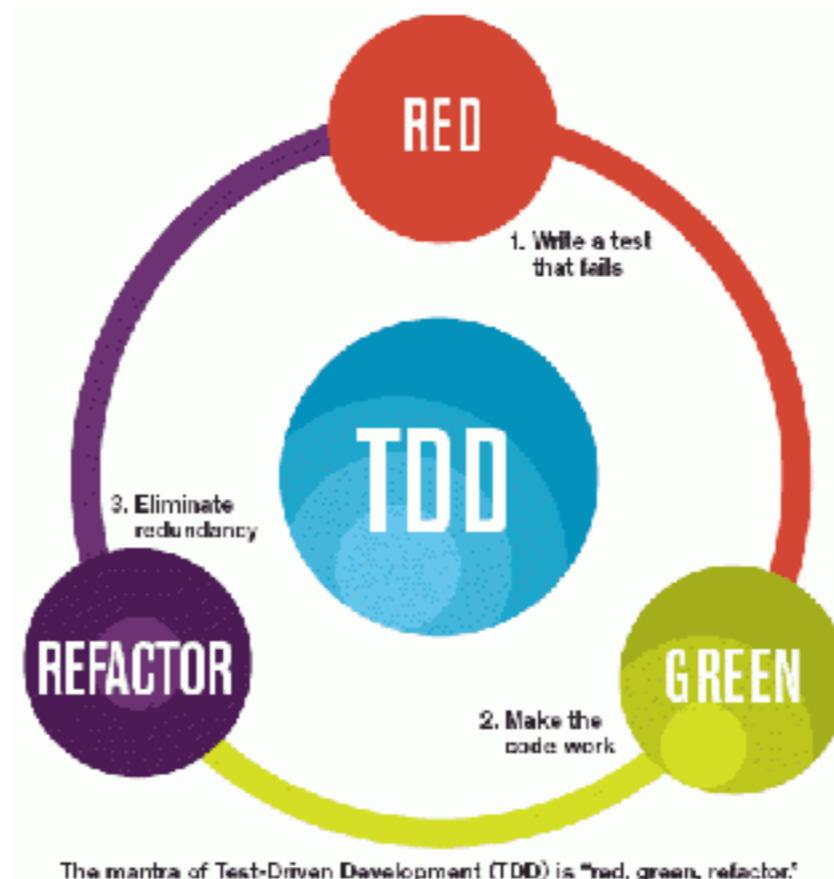
<https://jestjs.io/docs/configuration>



# Learn from Testing

Test  
code

Production  
code



# Hello with Test

## hello.test.ts

```
import { sayHi } from "../../src/hello";

test('Hello world with sayHi', () => {
  expect(sayHi()).toBe("Hello world!");
});
```

## hello.ts

```
export const sayHi = () => {
  return "Hello world!";
}
```



# Remove ../../ from code !!

## hello.test.ts

```
import { sayHi } from "../../src/hello";

test('Hello world with sayHi', () => {
  expect(sayHi()).toBe("Hello world!");
});
```

## jestconfig.json

```
{
  "compilerOptions": {
    "baseUrl": "./src",
    "paths": {
      "*": ["*"]
    }
  },
}
```

```
import { sayHi } from "hello";
test('Hello world with sayHi', () => {
  expect(sayHi()).toBe("Hello world!");
});
```



# Test Coverage



# Jest.config.ts

## Config test coverage report

```
const config: Config.InitialOptions = {  
  moduleDirectories: ['node_modules', 'src'],  
  verbose: true,  
  transform: {  
    '^.+\\.tsx?$': 'ts-jest',  
  },  
  collectCoverage: true,  
  coverageReporters: ['json', 'lcov', 'text', 'clover'],  
};
```

<https://jestjs.io/docs/configuration>



# Run test with coverage

## npm test

```
PASS  tests/unit/hello.test.ts
  ✓ Hello world with sayHi (1 ms)

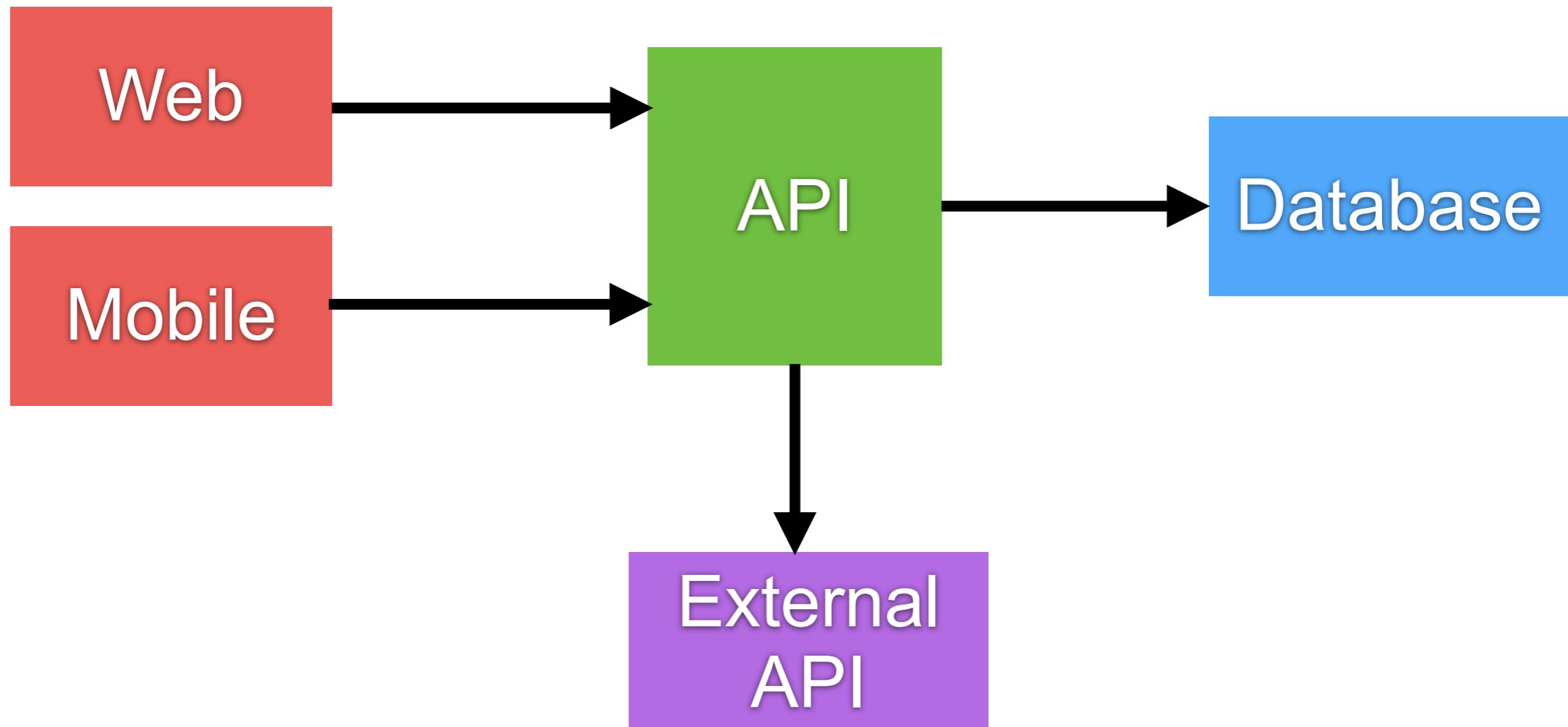
-----|-----|-----|-----|-----|-----|
File    | %Stmts | %Branch | %Funcs | %Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files | 100   | 100   | 100   | 100   |
hello.ts  | 100   | 100   | 100   | 100   |
-----|-----|-----|-----|-----|-----|
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.368 s, estimated 1 s
Ran all test suites.
```



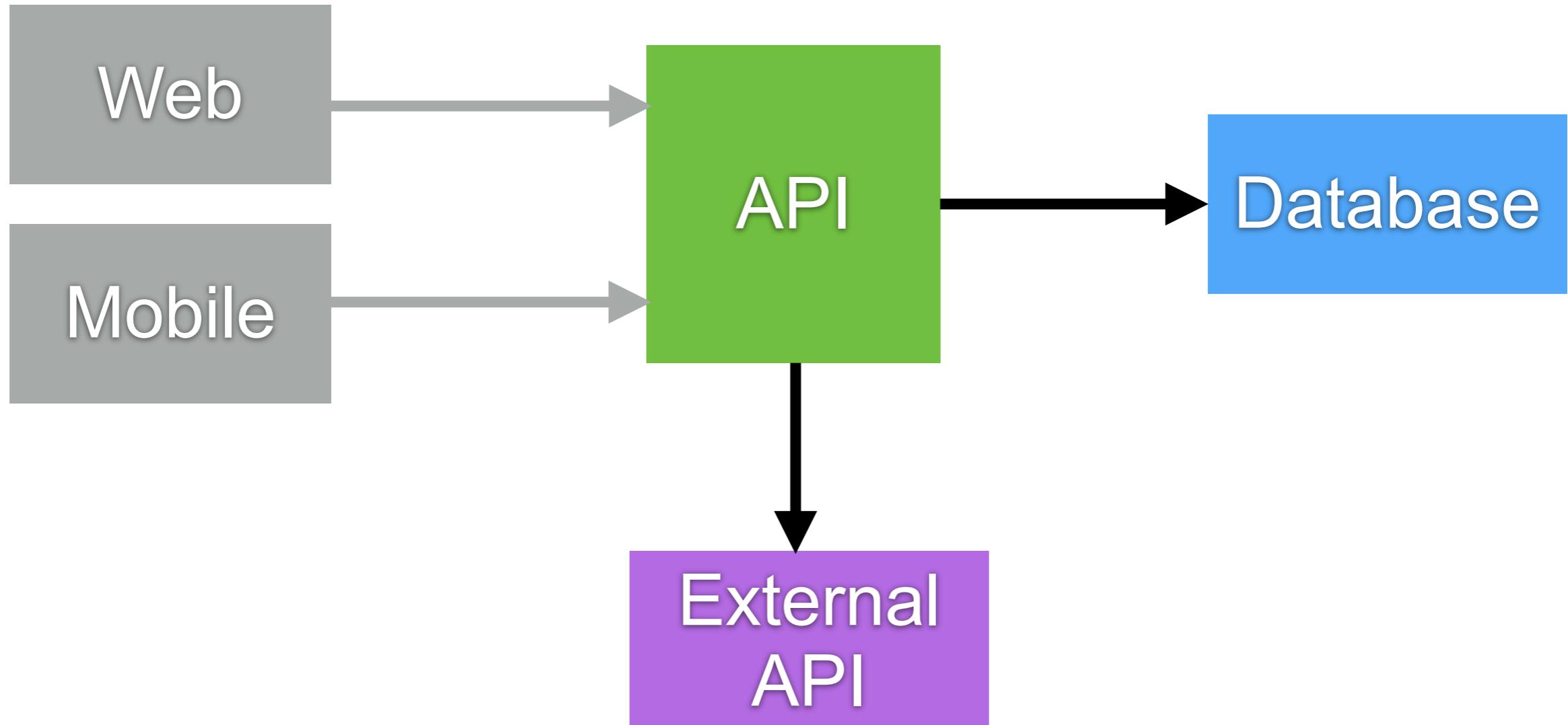
# Test Strategies



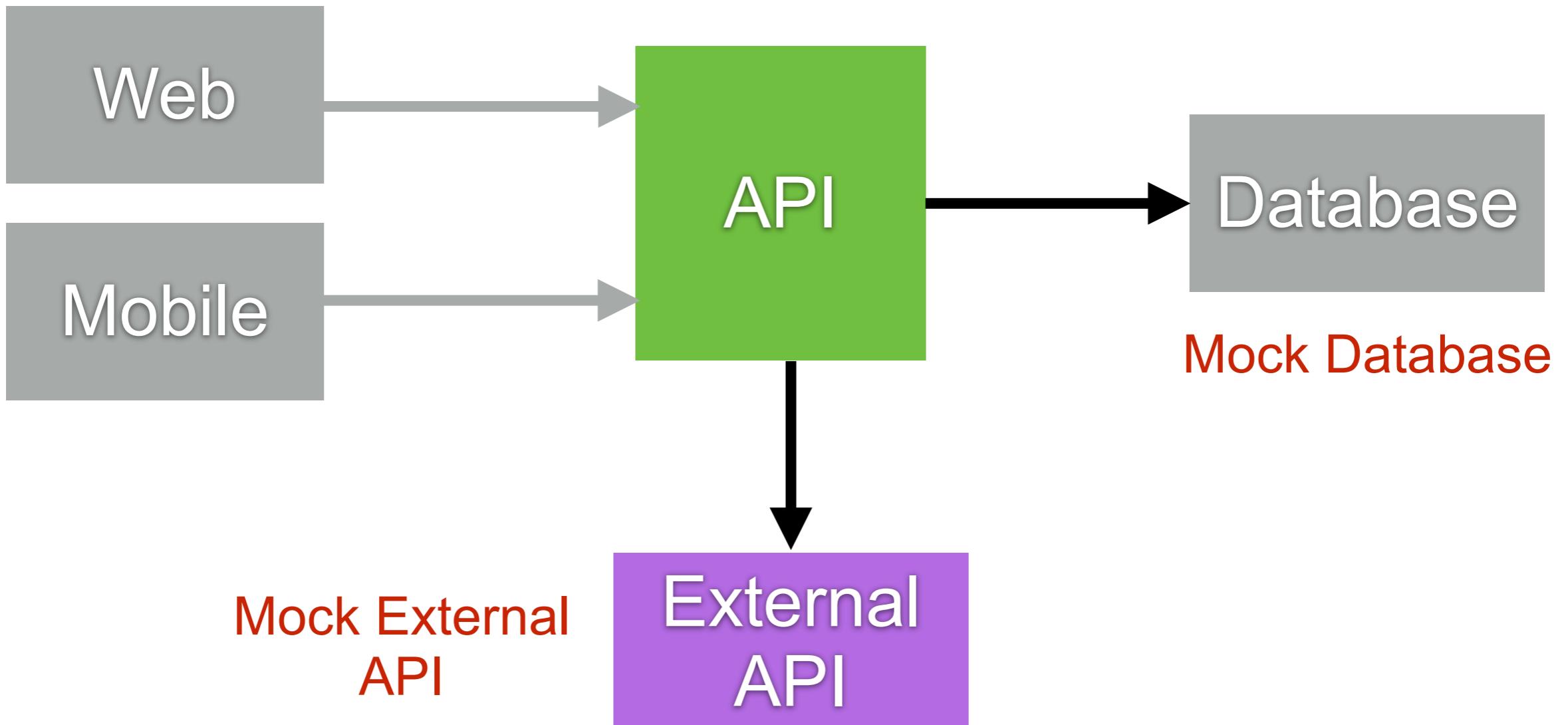
# Architecture



# API Testing ?



# API Testing ?

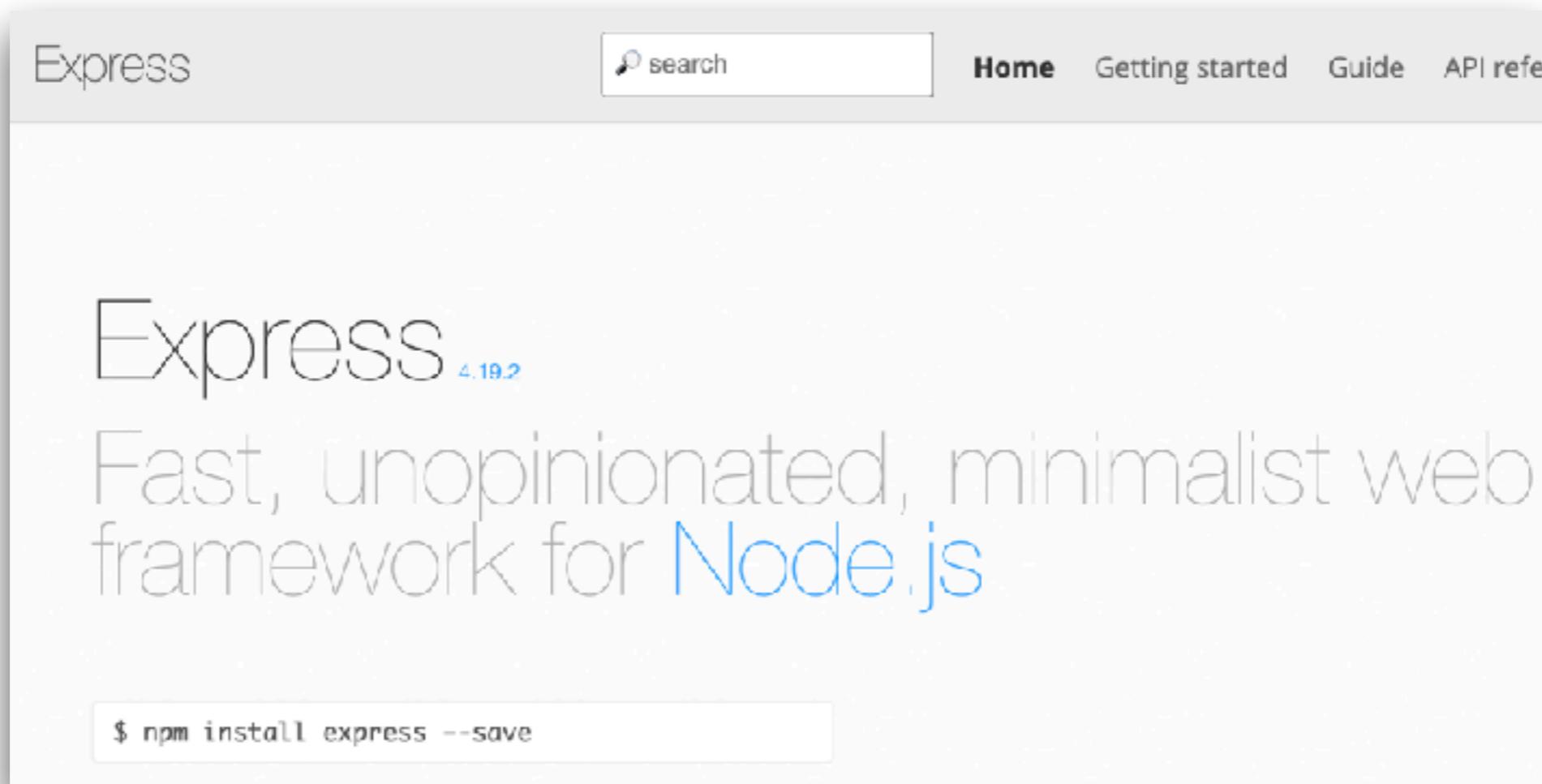


# How to design REST API ?



# Develop REST APIs

Use expressjs library



<https://expressjs.com/>



# Create project

```
npm init -y  
npx tsc --init  
npm install -S express  
npm install -D @types/express ts-node nodemon
```

## tsconfig.json

```
{  
  "include": ["src"],  
  "compilerOptions": {  
    "outDir": "./build",
```

<https://expressjs.com/>



# Hello API

<http://localhost:3000>

## server.ts

```
import express from "express";

const app = express();
const port = 3000;

app.get("/", (req, res) => {
  res.json({
    message: "Hello World from TypeScript!",
  });
});

app.listen(port, () => {
  console.log(`Server started at http://localhost:${port}`);
});
```

<https://expressjs.com/>



# How to run project

`npx ts-node src/server.ts`

<https://github.com/TypeStrong/ts-node>



# How to run project

Edit file **package.json**  
Working with nodemon

```
{  
  
  "scripts": {  
    "dev": "npx tsc && node build/server.js",  
    "serve": "nodemon --watch 'src/**/*.{ts,js}' --exec ts-node src/server.ts",  
  },  
}
```

<https://www.npmjs.com/package/nodemon>



# Dev Mode

npm run dev

```
> ts03-rest@1.0.0 dev
> tsc && nodemon build/server.js

[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node build/server.js`
Server started at http://localhost:3000
```



# Restart when changed

npm run serve

```
> ts03-rest@1.0.0 serve
> nodemon --watch 'src/server.ts' --exec ts-node src/server.ts

[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): src/server.ts
[nodemon] watching extensions: ts,json
[nodemon] starting `ts-node src/server.ts`
Server started at http://localhost:3000
```



# How to test your API ?



# How to test your API ?

External Testing

Internal Testing



POSTMAN

Coding with  
Jest, SuperTest, Nock



# Working with Jest and SuperTest

```
npm install -D jest @types/jest ts-jest ts-node
```

```
npm install -D supertest @types/supertest
```

<https://www.npmjs.com/package/supertest>



# Testable application

## server.ts

```
import app from "./app";
const port = 3000;
app.listen(port, () => {
  console.log(`Server started at http://localhost:${port}`);
});
```

## app.ts

```
import express from "express";
const app = express();

app.get("/", (req, res) => {
  res.json({
    message: "Hello World from TypeScript",
  });
});

export default app;
```



# Hello API Test

## hello.api.ts

```
// Test with supertest
import request from 'supertest';
import app from '../src/app';

test('GET /', async () => {
  const response = await request(app).get('/');
  expect(response.status).toBe(200);
  expect(response.body).toEqual({
    message: 'Hello World from TypeScript'
  });
});
```

<https://www.npmjs.com/package/supertest>



# Config jest.config.ts

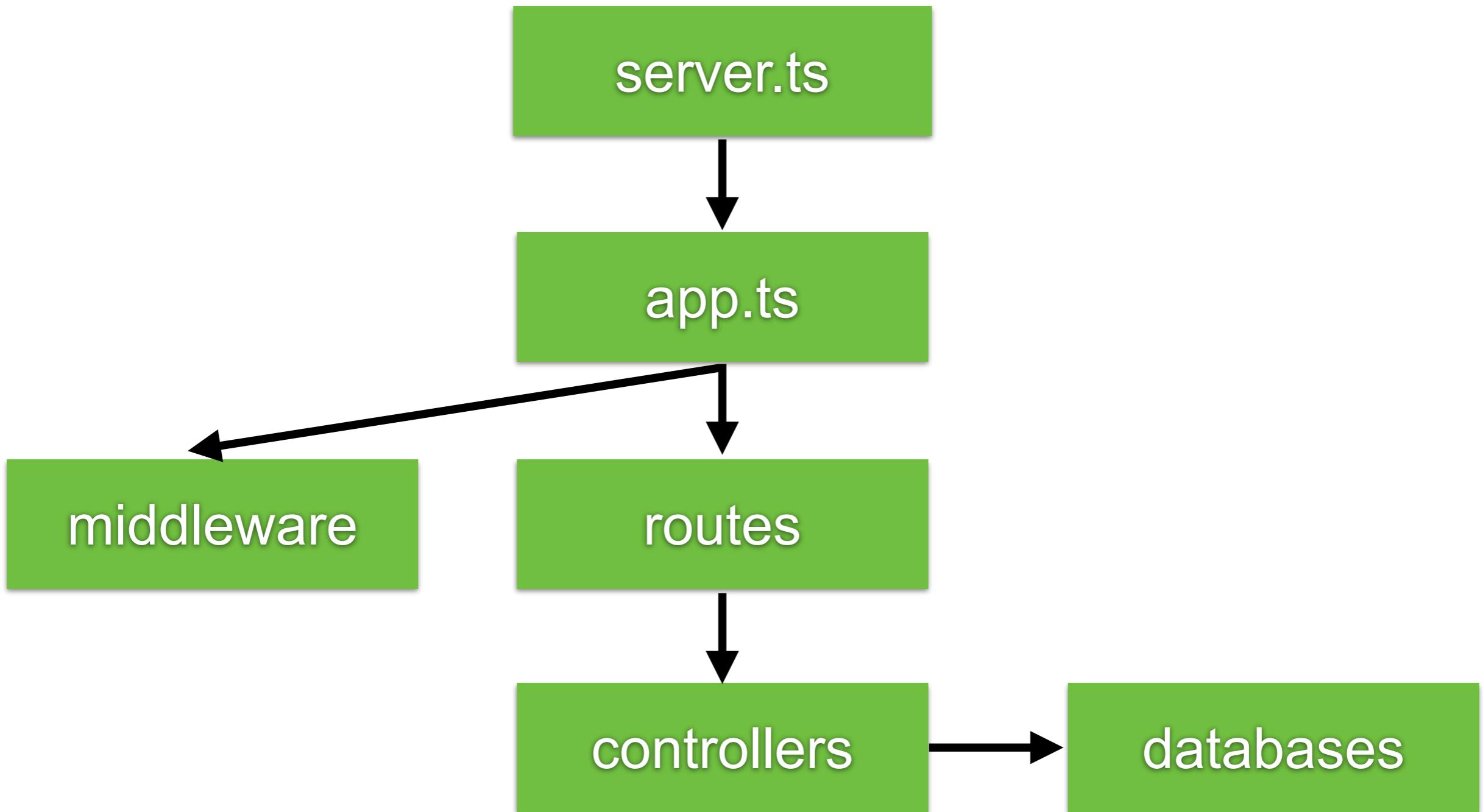
npm test

```
import type {Config} from '@jest/types';
// Sync object
const config: Config.InitialOptions = {
  verbose: true,
  roots: ['<rootDir>/tests'],
  transform: {
    '^.+\\.tsx?$': 'ts-jest',
  },
  collectCoverage: true
};
export default config;
```

<https://www.npmjs.com/package/supertest>



# Structure of project ?



# Routes and Controller

Authentication and Authorization  
Error handling  
Validation



# Controller

Receive request and return response to client

```
import { NextFunction, Request, Response } from "express";

export default async (req: Request, res: Response, next:
NextFunction) => {
  try {
    res.status(200).json({ message: "Hello World!" });
  } catch (error) {
    next(error);
  }
};
```



# Routes

Manage router for domain (group of function)

```
import { Router } from 'express';
import getHello from '../controllers/helloController/
getHello';

const router = Router();
router.get('/', getHello);

export default router;
```



# Middleware

Authentication and Authorization  
Error handling  
Validation



# Middleware

## Error handling

```
import { NextFunction, Request, Response } from "express";

export const errorHandler = (
  err: Error,
  req: Request,
  res: Response,
  next: NextFunction
) => {
  console.error(err);
  res.status(500).send({ errors: [{ message: "Something
went wrong" }] });
};
```



# App.ts

```
import express from "express";
import helloRouter from "./routes/hello";
import { errorHandler } from "./middlewares/errors";
const app = express();

app.use(express.json());
app.use(helloRouter);
app.use(errorHandler);

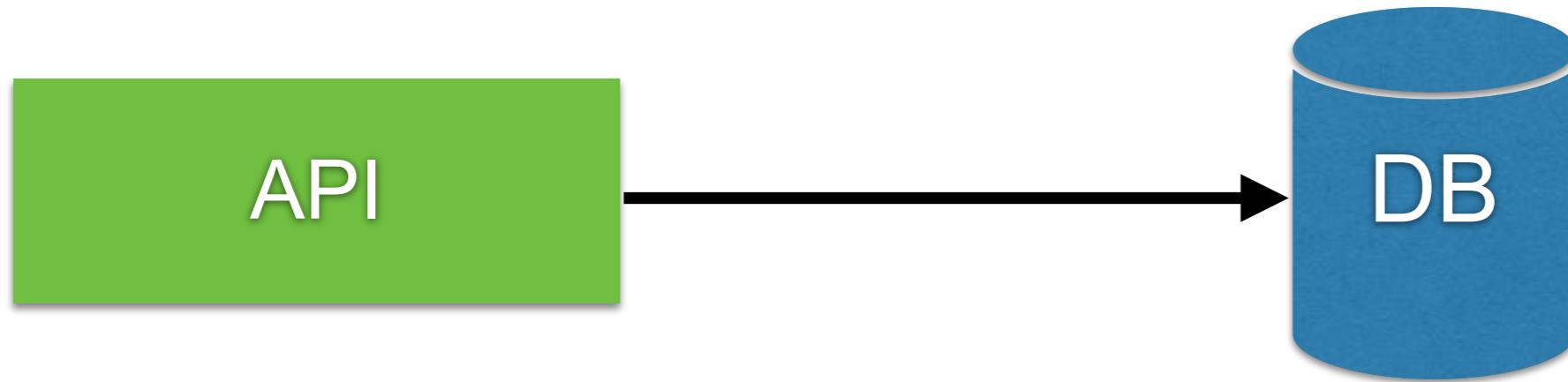
export default app;
```



# Working with Database



# Working with Database



SQL Native



# DBML (Database Markup Language)

## DBML - Database Markup Language

Open source



2.4k

### Intro

DBML (Database Markup Language) is an open-source DSL language designed to define and document database schemas and structures. It is designed to be simple, consistent and highly-readable.

It also comes with command-line tool and open-source module to help you convert between DBML and SQL.

<https://dbml.dbdiagram.io/>



© 2017 - 2018 Siam Chamnkit Company Limited. All rights reserved.

NodeJS

320

# DBML (Database Markup Language)

```
Table Post {
    id Int [pk, increment]
    title String [not null]
    content String [not null]
    likesCount Int [not null, default: 0]
    createdAt DateTime [default: `now()`], n
    updatedAt DateTime [not null]
    comments Comment [not null]
}

Table Comment {
    id Int [pk, increment]
    content String [not null]
    createdAt DateTime [default: `now()`], n
    updatedAt DateTime [not null]
    post Post [not null]
    postId Int [not null]
}
```

The diagram illustrates the DBML schema. It shows two tables: **Post** and **Comment**. The **Post** table contains fields: id (primary key, auto-increment), title, content, likesCount, createdAt, updatedAt, and comments (a many-to-one relationship to the Comment table). The **Comment** table contains fields: id (primary key, auto-increment), content, createdAt, updatedAt, post (a one-to-many relationship to the Post table), and postId (foreign key linking to the Post table's id).

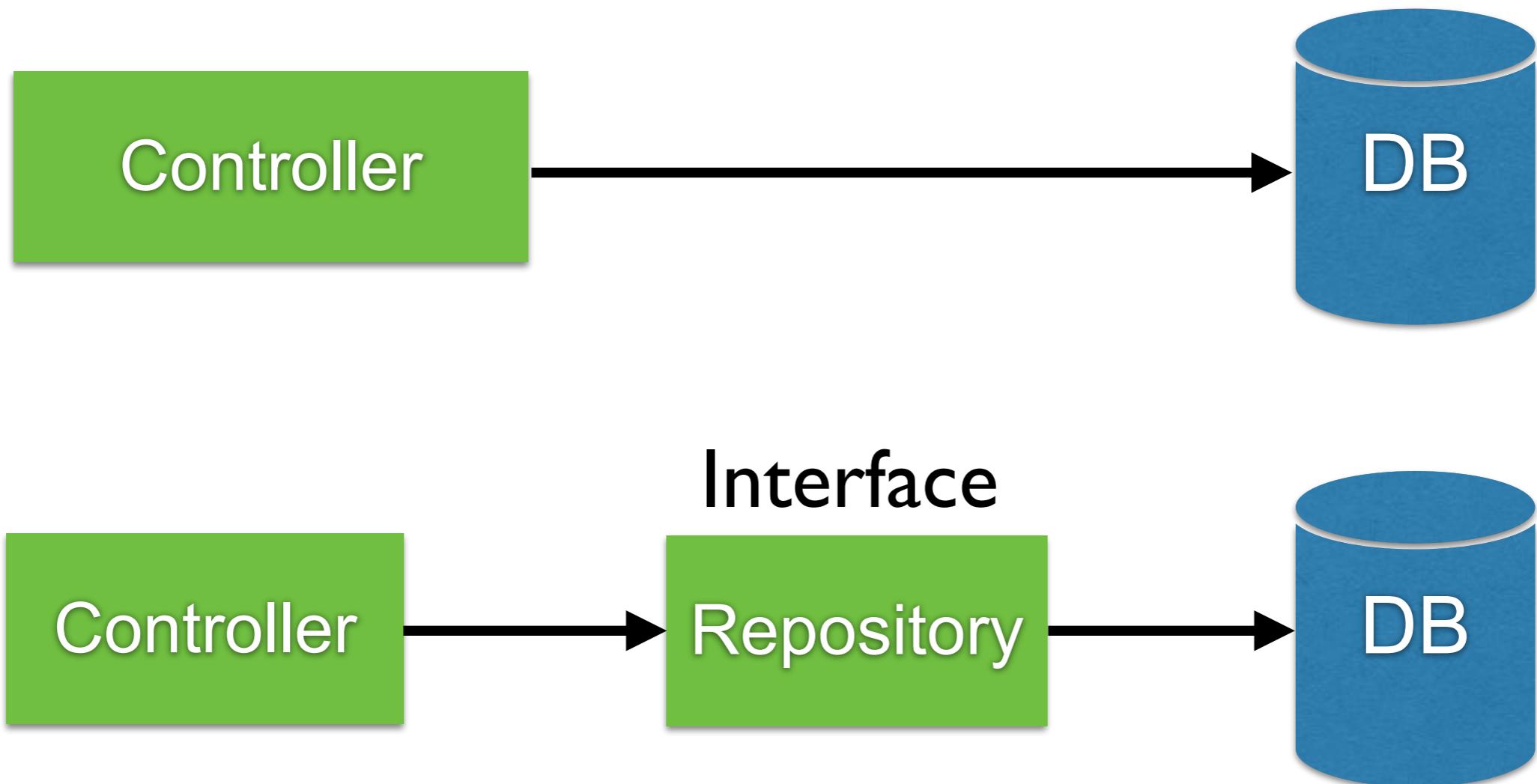
Post	
id	Int PK
title	String NN
content	String NN
likesCount	Int NN
createdAt	DateTime NN
updatedAt	DateTime NN
comments	Comment NN

Comment	
id	Int PK
content	String NN
createdAt	DateTime NN
updatedAt	DateTime NN
post	Post NN
postId	Int NN

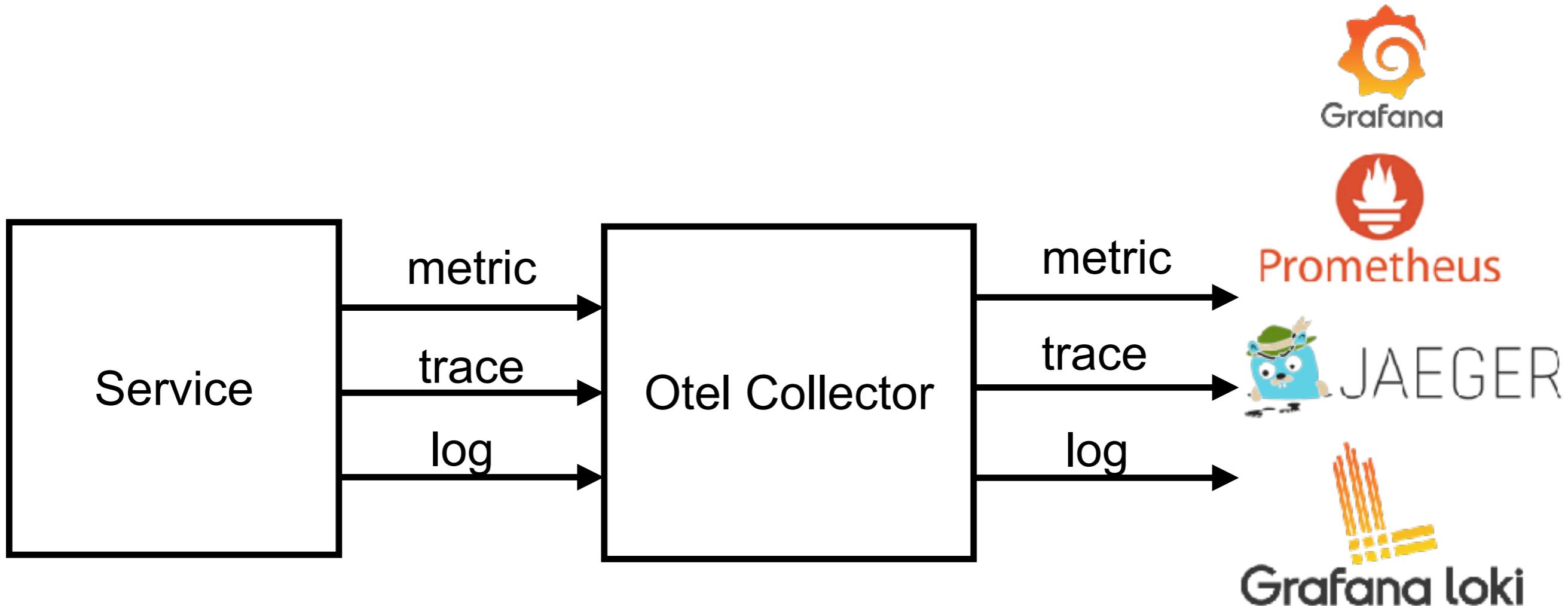
<https://dbml.dbdiagram.io/>



# Working with Database



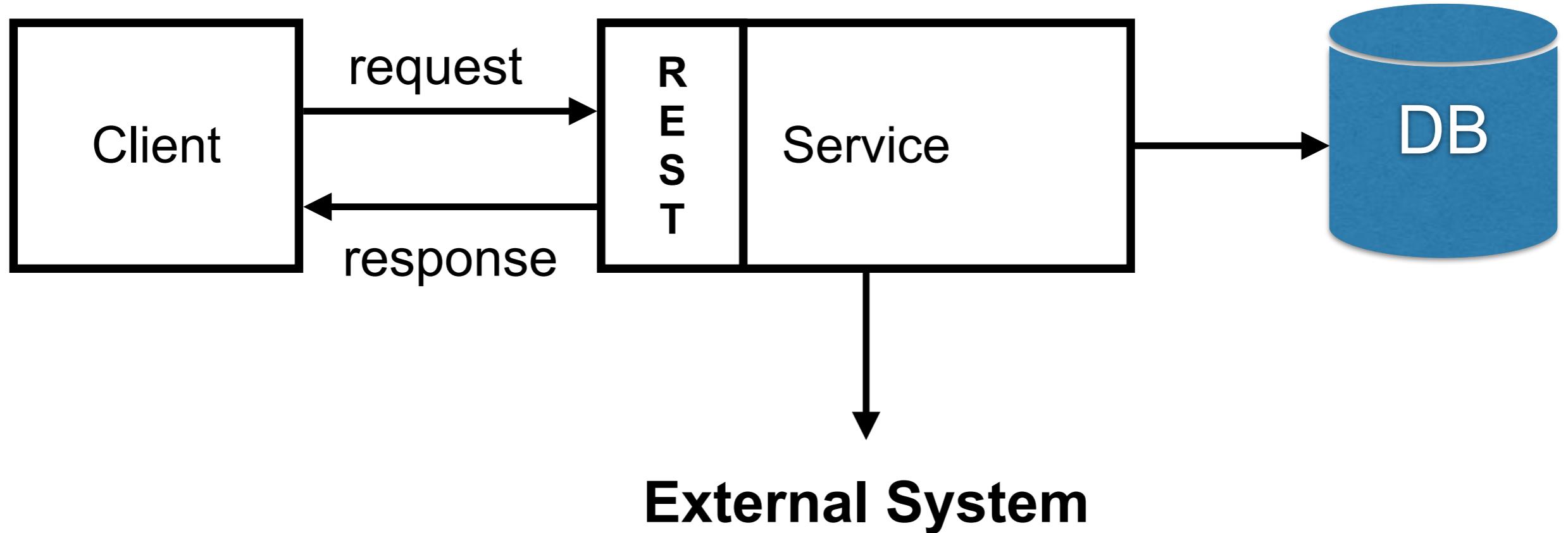
# Working with OpenTelemetry



# Develop RESTful API



# Structure of Service



# Topics (1)

Develop Project with Go

    Essential features

    Testing with Go

Testable project with Go

Develop REST API with gin

Working with Database (SQL, NoSQL)



# Topics (2)

## Authentication and Authorization API gateway



# Let's



[Why Go ▾](#)[Learn](#)[Docs ▾](#)[Packages](#)[Community ▾](#)

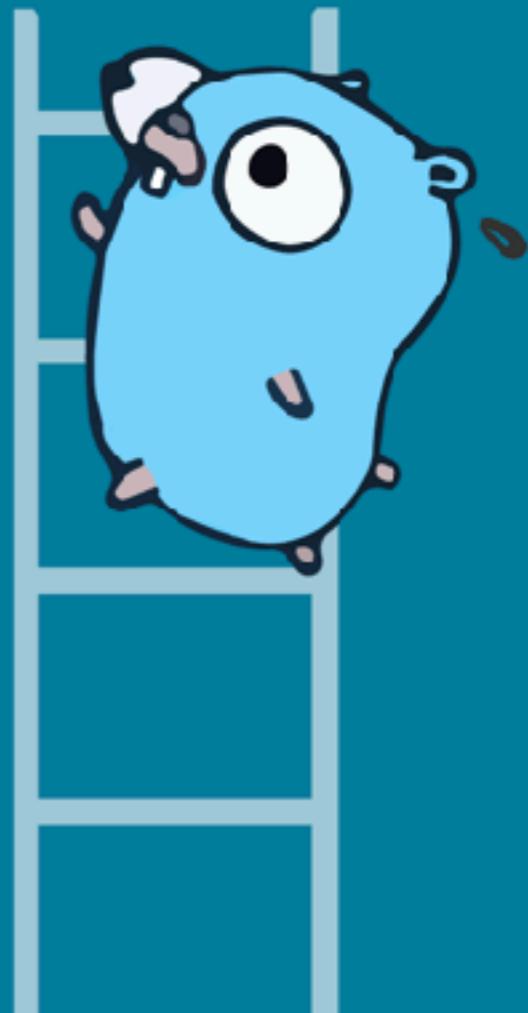
# Build simple, secure, scalable systems with Go

- ✓ An open-source programming language supported by Google
- ✓ Easy to learn and great for teams
- ✓ Built-in concurrency and a robust standard library
- ✓ Large ecosystem of partners, communities, and tools

[Get Started](#)[Download](#)

Download packages for [Windows 64-bit](#), [macOS](#), [Linux](#), and [more](#)

The go command by default downloads and authenticates modules using the Go module mirror and Go checksum database run by Google. [Learn more.](#)



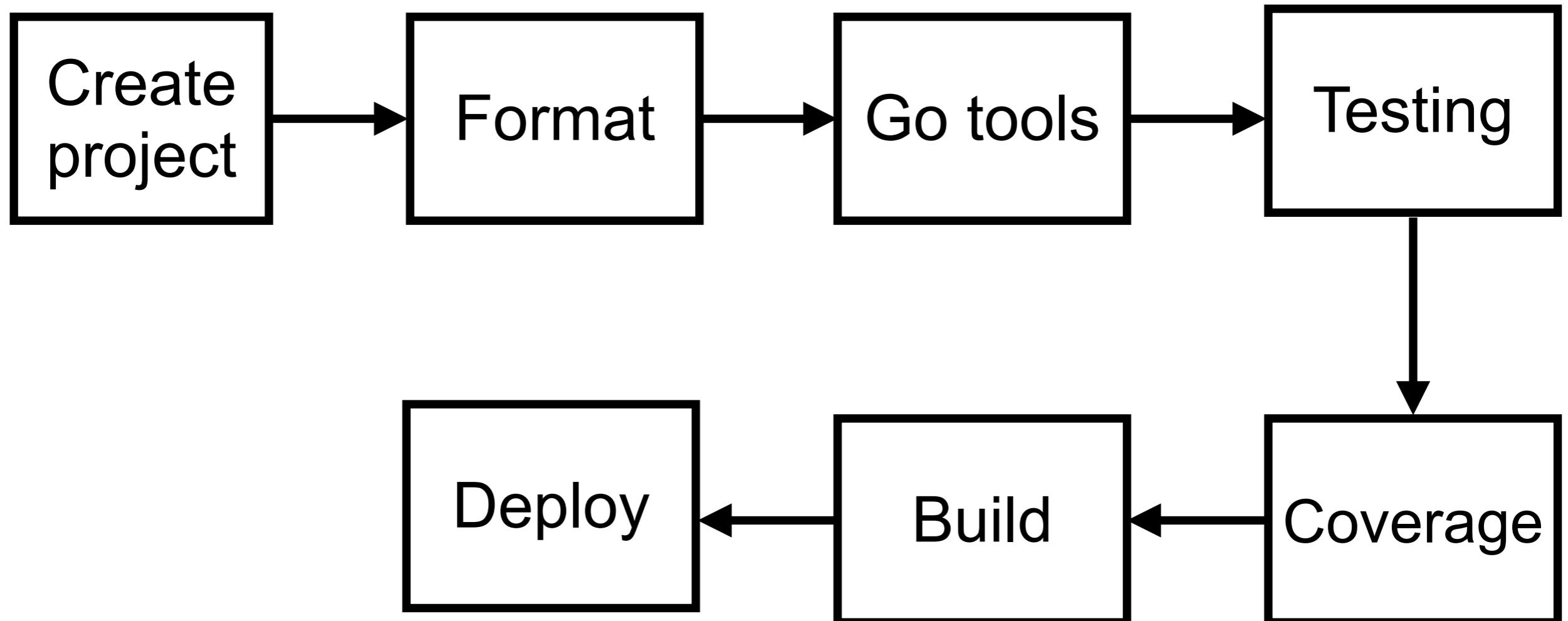
<https://go.dev/>



```
go env -w GOTOOLCHAIN=go1.24.1
go version
```



# Development workflow



# Hello Go !!

```
package main

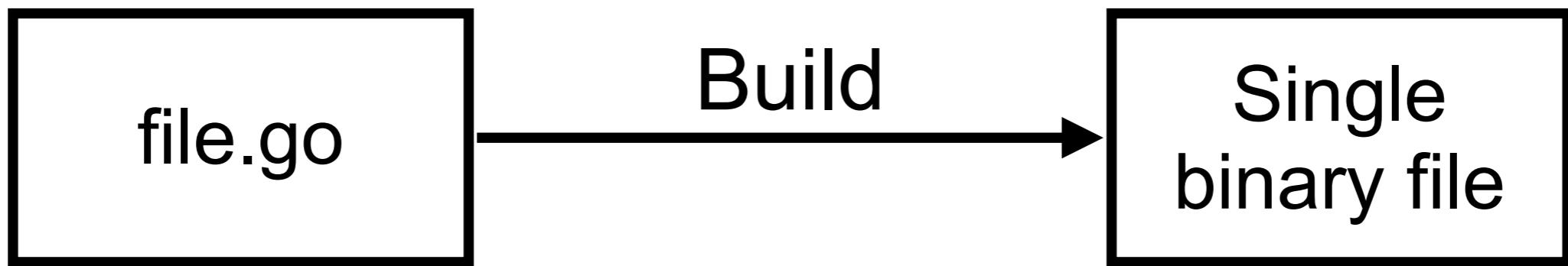
import "fmt"

func main() {
    fmt.Println("Hello, world 2024")
    print("Hello, world 2024")
}
```

```
$go run hello.go
```



# Basic Go



```
$go run file.go  
$go build -o <output>
```



# Building for architecture

**\$GOOS=?**  
**\$GOARCH=?**

GOOS	GOARCH
darwin	amd64 arm64
windows	amd64 arm64 arm
linux	amd64 arm64 arm

<https://go.dev/doc/install/source#environment>



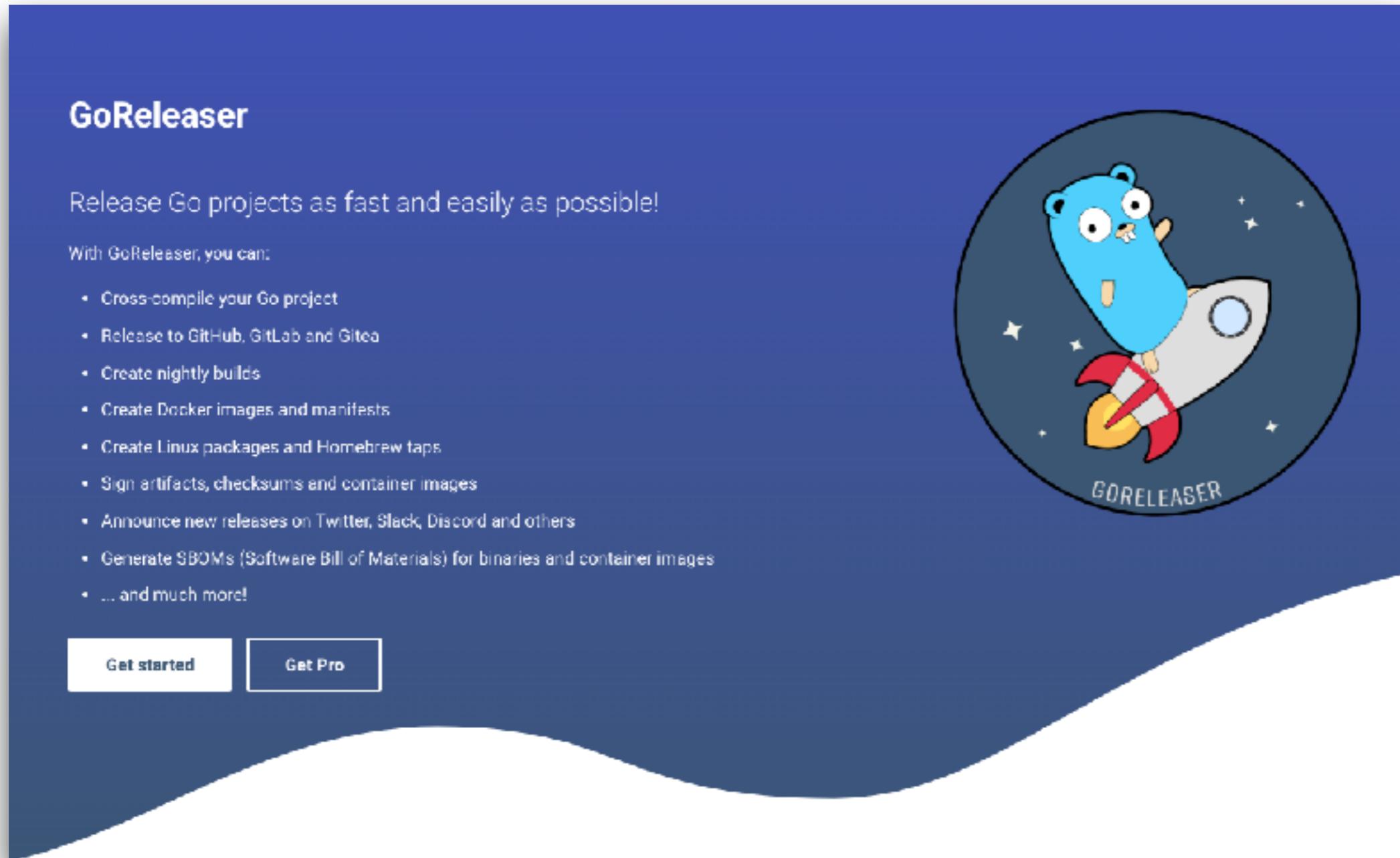
# Building for architecture

```
$GOOS=darwin GOARCH=amd64 go build  
$GOOS=darwin GOARCH=arm64 go build  
$GOOS=windows GOARCH=amd64 go build  
$GOOS=windows GOARCH=arm64 go build  
$GOOS=linux GOARCH=amd64 go build  
$GOOS=linux GOARCH=arm64 go build
```

<https://go.dev/doc/install/source#environment>



# GoReleaser



The screenshot shows the GoReleaser landing page. At the top left is the title "GoReleaser". Below it is the tagline "Release Go projects as fast and easily as possible!". A list of features follows: "With GoReleaser, you can:"

- Cross-compile your Go project
- Release to GitHub, GitLab and Gitea
- Create nightly builds
- Create Docker images and manifests
- Create Linux packages and Homebrew taps
- Sign artifacts, checksums and container images
- Announce new releases on Twitter, Slack, Discord and others
- Generate SBOMs (Software Bill of Materials) for binaries and container images
- ... and much more!

At the bottom left are two buttons: "Get started" and "Get Pro". To the right is a circular logo featuring a blue Go gopher character riding a white rocket ship against a dark blue background with stars. The word "GORELEASER" is written in white at the bottom of the circle.

<https://goreleaser.com/>



# Create go project

```
$go mod init demo
```

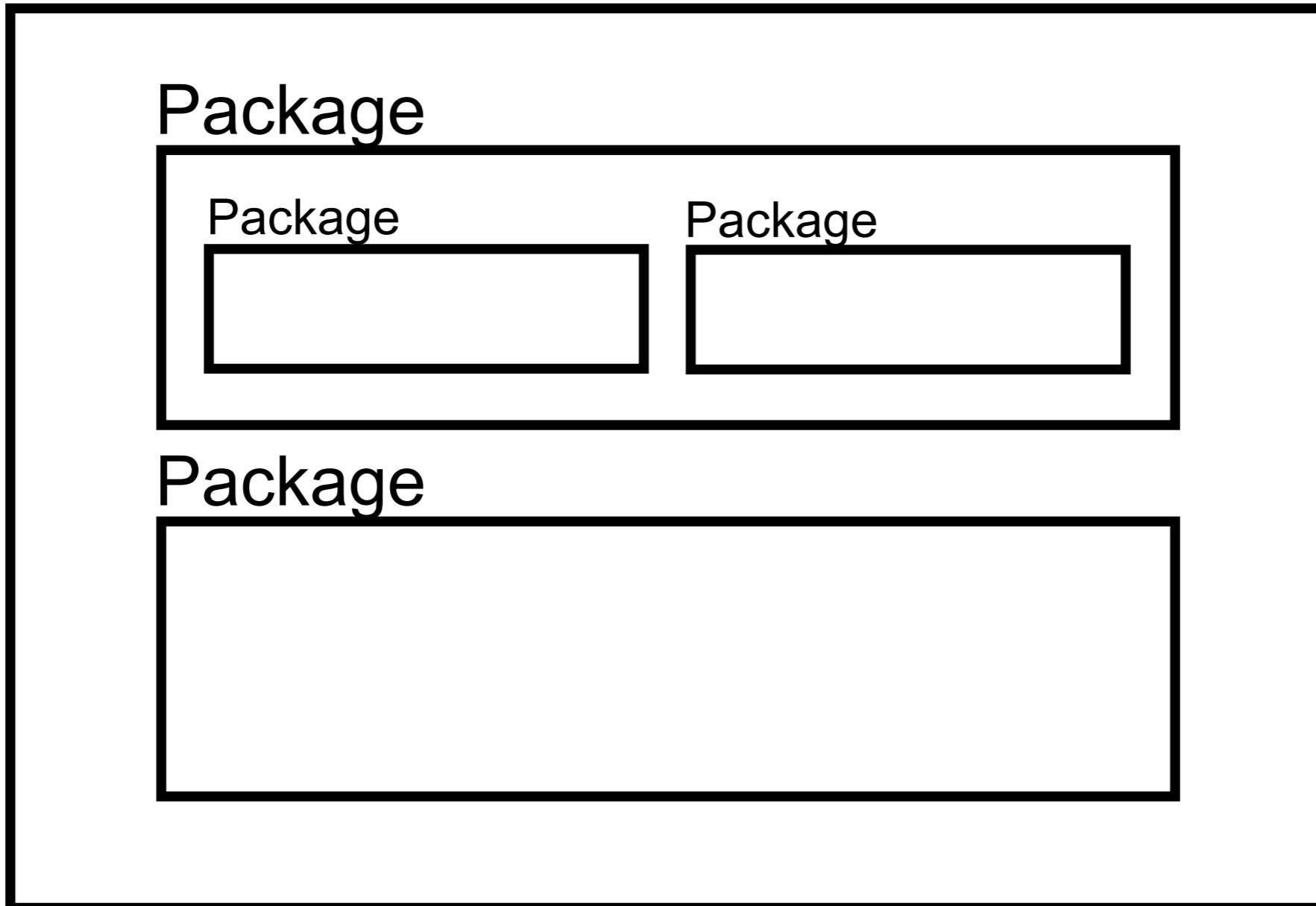
```
$go mod tidy
```

<https://go.dev/blog/using-go-modules>



# Create go project

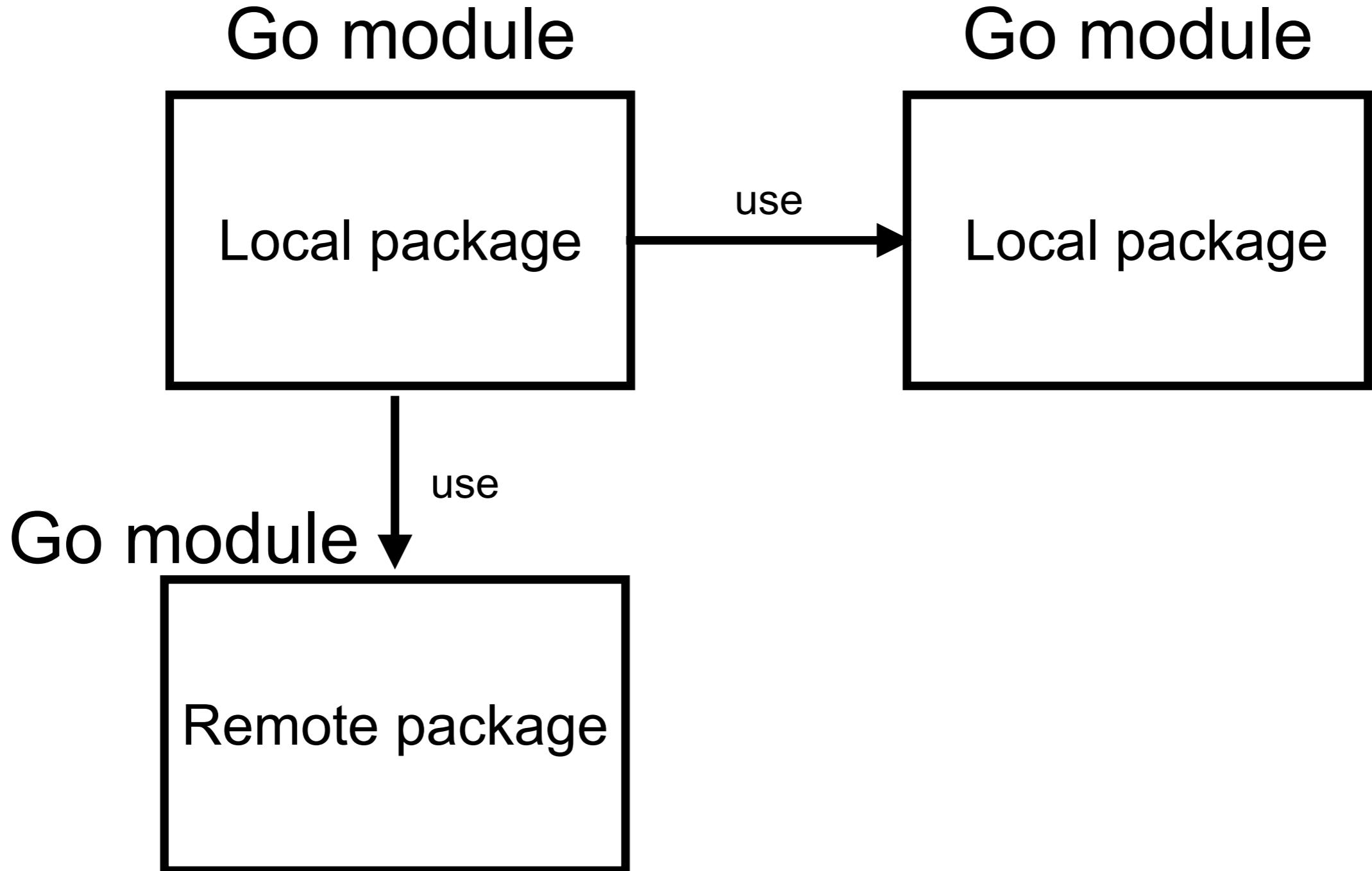
## Go module



<https://go.dev/blog/using-go-modules>



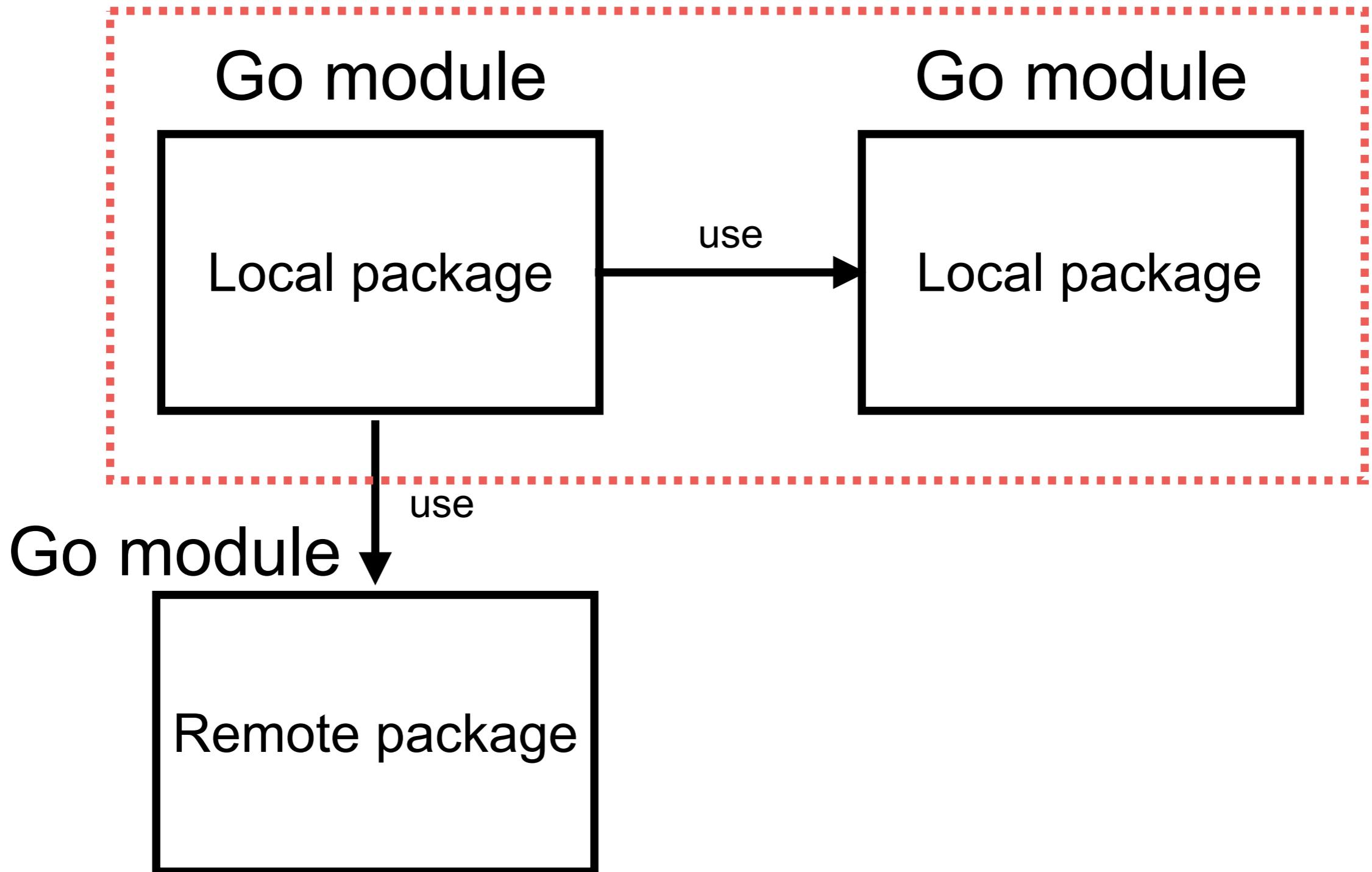
# Go Workspace



<https://go.dev/doc/tutorial/workspaces>



# Go Workspace



<https://go.dev/doc/tutorial/workspaces>



# Go workspace

\$go work init

\$go work use ./m1

\$go work use ./m2

<https://go.dev/blog/using-go-modules>



# Formatting

\$go fmt



# Go tools

\$go tool

\$go tool cover

\$go tool doc

\$go tool vet

\$go tool pprof

<https://pkg.go.dev/golang.org/x/tools>



# Go Doc

```
$go install -v golang.org/x/tools/cmd/godoc@latest
```

```
$godoc -http=:6060
```

<https://pkg.go.dev/golang.org/x/tools>



# Go Doc

The screenshot shows a web browser window displaying the Go Documentation Server at `localhost:6060/pkg/`. The title bar says "Go Documentation Server". A search bar is at the top right. On the left, there's a sidebar with links: "Standard library", "Other packages", "Sub-repositories", and "Community". On the right, there's a cartoon character of a penguin wearing a hard hat. The main content area has a blue header "Standard library ▾". Below it is a table:

Name	Synopsis
<a href="#">archive</a>	
<a href="#">tar</a>	Package tar implements access to tar archives.
<a href="#">zip</a>	Package zip provides support for reading and writing ZIP archives.
<a href="#">arena</a>	The arena package provides the ability to allocate memory for a collection of Go values and free that space manually all at once, safely.
<a href="#">bufio</a>	Package bufio implements buffered I/O. It wraps an io.Reader or io.Writer object, creating another object (Reader or Writer) that also implements the interface but provides buffering and some help for textual I/O.
<a href="#">builtin</a>	Package builtin provides documentation for Go's predeclared identifiers.
<a href="#">bytes</a>	Package bytes implements functions for the manipulation of byte slices.
<a href="#">compress</a>	

<https://pkg.go.dev/golang.org/x/tools>



# Linting and Static Analysis

## GolangCI Lint



<https://github.com/golangci/golangci-lint>



# Security Scanning

## GitLeaks

SAST tool for detect and prevent code !!  
Secrets, API keys and tokens

<https://github.com/gitleaks/gitleaks>



# Testing with Go



# Learning by test !!



RED GREEN REFACTOR

<https://quii.gitbook.io/learn-go-with-tests/>



# Testing with Go

## testing package

Test

Benchmark

Example

Fuzzing

<https://pkg.go.dev/testing>



# Testing with Go

hello.go

```
package main

import "fmt"

func Hello() string {
    return "Hello, world 2023"
}

func main() {
    fmt.Println(Hello())
}
```

hello\_test.go

```
package main

import "testing"

func TestHello(t *testing.T) {
    got := Hello()
    want := "Hello, world 2023"

    if got != want {
        t.Errorf("got %q want %q", got, want)
    }
}
```

```
$go test -v
```



# Testing

\$go test ./...

\$go test ./... -count=1



[https://pkg.go.dev/cmd/go#hdr-Testing\\_flags](https://pkg.go.dev/cmd/go#hdr-Testing_flags)



# Test Coverage with Go



# Test Coverage

\$go test -v -cover

```
==== RUN    TestBankingDeposit
---- PASS: TestBankingDeposit (0.00s)
==== RUN    TestBankingWithdraw
---- PASS: TestBankingWithdraw (0.00s)
PASS
      demo      coverage: 75.0% of statements
ok     demo      0.093s
```



# Test Coverage

Generate coverage report (HTML)

```
$go test -coverprofile=coverage.out  
$go tool cover -html=coverage.out
```



# Test Coverage

## Generate coverage report (HTML)

```
demo/banking.go (75.0%) ▾ not tracked  not covered  covered

package main

import "fmt"

type THB int

func (thb THB) String() string {
    return fmt.Sprintf("%d THB", thb)
}

type Banking struct {
    balance THB
}

func (b *Banking) Balance() THB {
    return b.balance
}

func (b *Banking) Deposit(amount THB) {
    b.balance += amount
}

func (b *Banking) Withdraw(amount THB) {
    b.balance -= amount
}
```



# Benchmark

```
$go test --bench .
```

```
$go test --bench . -count 5
```

<https://pkg.go.dev/testing#hdr-Benchmarks>



# Fuzzing

\$go test -fuzz .

<https://go.dev/doc/tutorial/fuzz>



# Essential Features

Struct and Pointer

Interface

Error

Defer, panic, recover

Function as a parameter

Go routine and channel



# No private/public in Go !!



**Data = public or exported**  
**data = private or unexported**



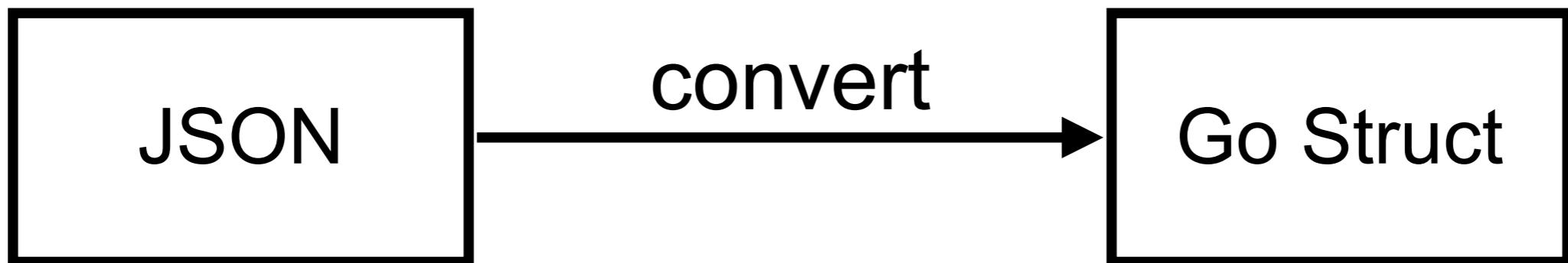
# Struct and Pointer

Named collection of fields  
Use to store data

```
type User struct {
    ID      int     `json:"id"`
    Name    string  `json:"name"`
    Username string  `json:"username"`
    Email   string  `json:"email"`
    Address struct {
        Street  string `json:"street"`
        Suite   string `json:"suite"`
        City    string `json:"city"`
        Zipcode string `json:"zipcode"`
        Geo     struct {
            Lat    string `json:"lat"`
            Lng    string `json:"lng"`
        } `json:"geo"`
    } `json:"address"`
}
```



# Convert JSON to Go Struct



# Convert JSON to Go Struct



## Paste JSON as Code v12.0.46

quicktype | ⚡ 1,684,428 | ★★★★★ (44)

Copy JSON, paste as Go, TypeScript, C#, C++ and more.

[Disable](#) | ▾

[Uninstall](#) | ▾



This extension is enabled globally.

<https://marketplace.visualstudio.com/items?itemName=quicktype.quicktype>



Go workshop

© 2022 - 2024 Siam Chamnankit Company Limited. All rights reserved.

364

# Convert JSON to Go Struct

The screenshot shows the quicktype.io interface. On the left, there's a sidebar with a 'Name' input field containing 'Welcome' and a 'Source type' dropdown set to 'JSON'. Below this is a text area with a JSON sample:

```
[{"greeting": "Welcome to quicktype!", "instructions": ["Type or paste JSON here", "Or choose a sample above", "quicktype will generate code in your chosen language to parse the sample data"]}]
```

The main panel displays the generated Go code:

```
// This file was generated from JSON Schema using quicktype, do not modify it directly.
// To parse and unparse this JSON data, add this code to your project.
// welcome, err := UnmarshalWelcome(bytes)
// bytes, err = welcome.Marshal()

package main

import "encoding/json"

func UnmarshalWelcome(data []byte) (Welcome, error) {
    var r Welcome
    err := json.Unmarshal(data, &r)
    return r, err
}

func (r *Welcome) Marshal() ([]byte, error) {
    return json.Marshal(r)
}

type Welcome struct {
    Greeting     string `json:"greeting"`
    Instructions []string `json:"instructions"`
}
```

On the right, there are settings for the generated package: 'Language' set to 'Go', 'Generated package name' set to 'main', and several optional checkboxes: 'Plain types only', 'Renders each top-level object in its own Go file', 'Plain types with package only', and 'Make all properties optional'. A 'Copy Code' button is also present.

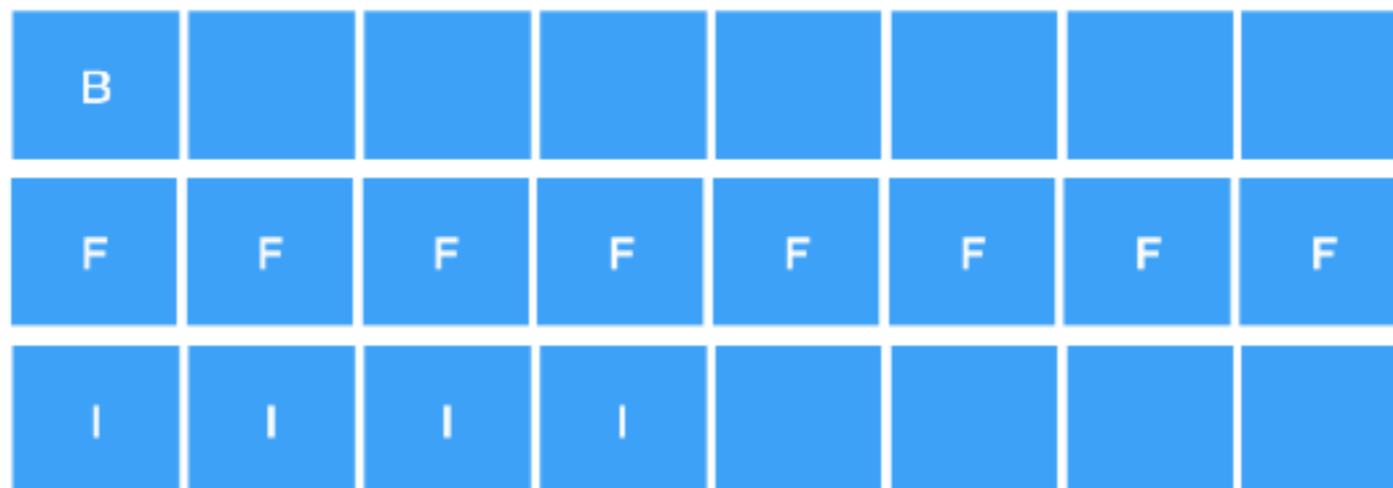
<https://app.quicktype.io/>



# Sequence of fields in struct

```
func main() {
    type first struct {
        b bool    // 1 byte
        f float64 // 8 bytes
        i int32   // 4 bytes
    }
    a := first{}
    fmt.Println(unsafe.Sizeof(a)) // 24 bytes
}
```

## Memory padding



<https://www.somkiat.cc/golang-memory-of-struct/>



# Sequence of fields in struct

```
func main() {
    type first struct {
        f float64 // 8 bytes
        b bool     // 1 byte
        i int32    // 4 bytes
    }
    a := first{}
    fmt.Println(unsafe.Sizeof(a)) // 16 bytes
}
```

## Memory padding



<https://www.somkiat.cc/golang-memory-of-struct/>



Go workshop

© 2022 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Tool :: Field alignment

```
$go install golang.org/x/tools/go/analysis/passes/  
fieldalignment/cmd/fieldalignment@latest
```

```
$fieldalignment -fix demo.go
```

<https://www.somkiat.cc/go-struct-field-alignment/>



# Workshop with Struct

Banking

+ balance

Balance():int

Deposit(int)

Withdraw(int)



# Workshop with Struct

```
package main

import "fmt"

type THB int

func (thb THB) String() string {
    return fmt.Sprintf("%d THB", thb)
}

type Banking struct {
    balance THB
}

func (b *Banking) Balance() THB {
    return b.balance
}

func (b *Banking) Deposit(amount THB) {
    b.balance += amount
}

func (b *Banking) Withdraw(amount THB) {
    b.balance -= amount
}
```



# Workshop with Struct

## Create custom type from integer

```
package main

import "fmt"

type THB int

func (thb THB) String() string {
    return fmt.Sprintf("%d THB", thb)
}
```



# Workshop with Struct

## Implement from Stringer interface

```
package main

import "fmt"

type THB int

func (thb THB) String() string {
    return fmt.Sprintf("%d THB", thb)
}
```

<https://pkg.go.dev/fmt#Stringer>



# Workshop with Struct

## Create struct and receiver

```
type Banking struct {
    balance THB
}

func (b *Banking) Balance() THB {
    return b.balance
}

func (b *Banking) Deposit(amount THB) {
    b.balance += amount
}

func (b *Banking) Withdraw(amount THB) {
    b.balance -= amount
}
```



# How to testing ?



# Testing with Deposit

```
func TestBankingDeposit(t *testing.T) {
    // Arrange
    banking := Banking{}
    // Act
    banking.Deposit(THB(1000))
    got := banking.Balance()
    // Assert
    want := THB(1000)
    if got != want {
        t.Errorf("got %s want %s", got, want)
    }
}
```

```
$go test -v
```



# Testing with Withdraw

```
func TestBankingWithdraw(t *testing.T) {
    // Arrange
    banking := Banking{
        balance: 5000,
    }
    // Act
    banking.Withdraw(THB(1000))
    got := banking.Balance()
    // Assert
    want := THB(4000)
    if got != want {
        t.Errorf("got %s want %s", got, want)
    }
}
```

```
$go test -v
```



# Try by yourself

## Remove pointer (\*)

```
type Banking struct {
    balance THB
}

func (b Banking) Balance() THB {
    return b.balance
}

func (b Banking) Deposit(amount THB) {
    b.balance += amount
}

func (b Banking) Withdraw(amount THB) {
    b.balance -= amount
}
```

```
$go test -v
```



# Result !!

```
---- FAIL: TestBankingDeposit (0.00s)
  banking_test.go:16: got 0 THB want 1000 THB
---- FAIL: TestBankingWithdraw (0.00s)
  banking_test.go:31: got 5000 THB want 4000 THB
FAIL
exit status 1
FAIL      demo      0.460s
```

## Why ?



# Print memory address

```
func (b Banking) Deposit(amount THB) {
    fmt.Printf("Address of balance in Deposit is %p \n", &b.balance)
    b.balance += amount
}
```

```
func TestBankingDeposit(t *testing.T) {
    // Arrange
    banking := Banking{}
    // Act
    ...
    fmt.Printf("Address of balance in Deposit is %p \n", &banking.balance)
    ...
}
```



# Result

```
Address of balance in Deposit is 0x14000016128
Address of balance in Deposit is 0x14000016120
--- FAIL: TestBankingDeposit (0.00s)
    banking_test.go:18: got 0 THB want 1000 THB
--- FAIL: TestBankingWithdraw (0.00s)
    banking_test.go:33: got 5000 THB want 4000 THB
FAIL
exit status 1
FAIL      demo      0.457s
```



# Improve your test



# Improve your test

Use sub-test  
Refactor assertion



# Improve your test

```
func TestBanking(t *testing.T) {  
  
    assertBalance := func(t testing.TB, b Banking, want THB) {  
        t.Helper()  
        got := b.Balance()  
        if got != want {  
            t.Errorf("got %s want %s", got, want)  
        }  
    }  
}  
}
```

Create custom assertion :: easy to read



# Improve your test

```
func TestBanking(t *testing.T) {
    assertBalance := func(t testing.TB, b Banking, want THB) {
        t.Helper()
        got := b.Balance()
        if got != want {
            t.Errorf("got %s want %s", got, want)
        }
    }

    t.Run("deposit", func(t *testing.T) {
        b := Banking{}
        b.Deposit(THB(1000))
        assertBalance(t, b, THB(1000))
    })

    t.Run("withdraw", func(t *testing.T) {
        b := Banking{balance: 5000}
        b.Withdraw(THB(1000))
        assertBalance(t, b, THB(4000))
    })
}
```

## Sub-test

<https://go.dev/blog/subtests>



Go workshop

© 2022 - 2024 Siam Chamnkit Company Limited. All rights reserved.

# Add new feature with test ?



# Withdraw :: not enough money

```
func TestBanking(t *testing.T) {  
    t.Run("withdraw not enough money", func(t *testing.T) {  
        startingBalance := THB(1000)  
        b := Banking{startingBalance}  
        err := b.Withdraw(THB(2000))  
        assertBalance(t, b, startingBalance)  
        if err == nil {  
            t.Error("wanted an error but didn't get one")  
        }  
    })  
}
```

Return error !!



# Withdraw()

## Use errors package

```
func (b *Banking) Withdraw(amount THB) error {  
    if amount > b.balance {  
        return errors.New("Money not enough")  
    }  
  
    b.balance -= amount  
    return nil  
}
```

Return error

<https://pkg.go.dev/errors>



# Run test :: passed



# Refactor code



# Production code

## Define error into variable

```
var ErrNotEnoughMoney = errors.New("Cannot withdraw, money not enough")

func (b *Banking) Withdraw(amount THB) error {
    if amount > b.balance {
        return ErrNotEnoughMoney
    }

    b.balance -= amount
    return nil
}
```

<https://github.com/uber-go/guide/blob/master/style.md#errors>



# Test code

```
func TestBanking(t *testing.T) {  
    t.Run("withdraw not enough money", func(t *testing.T) {  
        startingBalance := THB(1000)  
        b := Banking{startingBalance}  
  
        err := b.Withdraw(THB(2000))  
  
        assertBalance(t, b, startingBalance)  
        if err == nil {  
            t.Error("wanted an error but didn't get one")  
        }  
    })  
}
```



# Assert with Error

```
func assertBalance(t testing.TB, b Banking, want THB) {
    t.Helper()
    got := b.Balance()

    if got != want {
        t.Errorf("got %q want %q", got, want)
    }
}

func assertError(t testing.TB, got, want error) {
    t.Helper()
    if got == nil {
        t.Fatal("didn't get an error but wanted one")
    }

    if got != want {
        t.Errorf("got %q, want %q", got, want)
    }
}
```



# Test code

```
t.Run("withdraw", func(t *testing.T) {
    b := Banking{balance: 5000}
    err := b.Withdraw(THB(1000))

    assertBalance(t, b, THB(4000))
    assertNoError(t, err)
})

t.Run("withdraw not enough money", func(t *testing.T) {
    startingBalance := THB(1000)
    b := Banking{startingBalance}
    err := b.Withdraw(THB(2000))

    assertBalance(t, b, startingBalance)
    assertError(t, err, ErrNotEnoughMoney)
})
```



# More testing library

<https://github.com/stretchr/testify>



# Error handling with Go

<https://go.dev/blog/error-handling-and-go>



# Build-in error type

```
func doSth() (result string, err error) {  
    return "", fmt.Errorf("Error")  
}
```

```
func main() {  
    result, err := doSth()  
    if err != nil {  
        fmt.Println(err)  
    }  
    fmt.Println("Result=", result)  
}
```



# Error handling

```
func doSth() (result string, err error) {  
    return "", fmt.Errorf("Error")  
}
```

```
func main() {  
    result, err := doSth()  
    if err != nil {  
        fmt.Println(err)  
    }  
    fmt.Println("Result=", result)  
}
```



# Error interface

```
type error interface {
    Error() string
}
```

```
type errorString struct {
    s string
}

func (e *errorString) Error() string {
    return e.s
}
```



# Custom error

```
type error interface {
    Error() string
}
```

```
type errorString struct {
    s string
}

func (e *errorString) Error() string {
    return e.s
}
```



# Errors package

Provide function to manipulate errors  
Use New() to create error

```
var ErrNotEnoughMoney = errors.New("Cannot withdraw, money not enough")

func (b *Banking) Withdraw(amount THB) error {
    if amount > b.balance {
        return ErrNotEnoughMoney
    }

    b.balance -= amount
    return nil
}
```

<https://pkg.go.dev/errors>



# Best practices for Error handling

Use `errors.New()` to create error

Wrap error with `fmt.Errorf()`

Return `nil` if everything is ok

Create **custom type** for error

Don't use `panic()` and `recover()`

**Check error immediately** after call function

Log error before return

<https://climbtheladder.com/10-golang-error-handling-best-practices/>



Go workshop

© 2022 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Example with error

```
func main() {
    f, err := os.Open("filename.ext")
    if err != nil {
        log.Fatal(err)
    }
    defer f.Close()

    // do something with the open file

}
```

<https://climbtheladder.com/10-golang-error-handling-best-practices/>



# Working with multiple errors

```
func step1() error {
    return errors.New("step1 error")
}

func step2() error {
    return errors.New("step2 error")
}
```

```
func main() {

    err1 := step1()
    err2 := step2()
    err := errors.Join(err1, err2)

    if err != nil {
        fmt.Printf("some error: \n%v", err)
    }
}
```

<https://go.dev/play/p/fKVMnn9NFEK>

<https://github.com/hashicorp/go-multierror>



# How to checking for unchecked errors in Go code ?



# Use errcheck

```
$go install github.com/kisielk/errcheck@latest  
$errcheck .
```

<https://github.com/kisielk/errcheck>



**Don't just check errors,  
handle them gracefully** 

– Go Proverb

<https://dave.cheney.net/2016/04/27/dont-just-check-errors-handle-them-gracefully>



# Defer, Panic and Recover

try/catch/finally

<https://go.dev/blog/defer-panic-and-recover>



# Defer, Panic and Recover

**Defer =>** put function call into a stack

**Panic =>** Stop the normal execution flow

**Recover =>** Handle panic, use with defer



```
func A() {
    defer fmt.Println("Called A")
    B()
}

func B() {
    defer fmt.Println("Called B")
    C()
}

func C() {
    defer fmt.Println("Called C")
    Break()
}
```

```
func Break() {
    defer fmt.Println("Called Break")
    panic(errors.New("My Error with panic"))

}

func main() {
    A()
}
```



# Working with Recover()

```
func A() {
    defer fmt.Println("Called A")

    defer func() {
        if x := recover(); x != nil {
            fmt.Printf("Panic: %+v\n", x)
        }
    }()
}

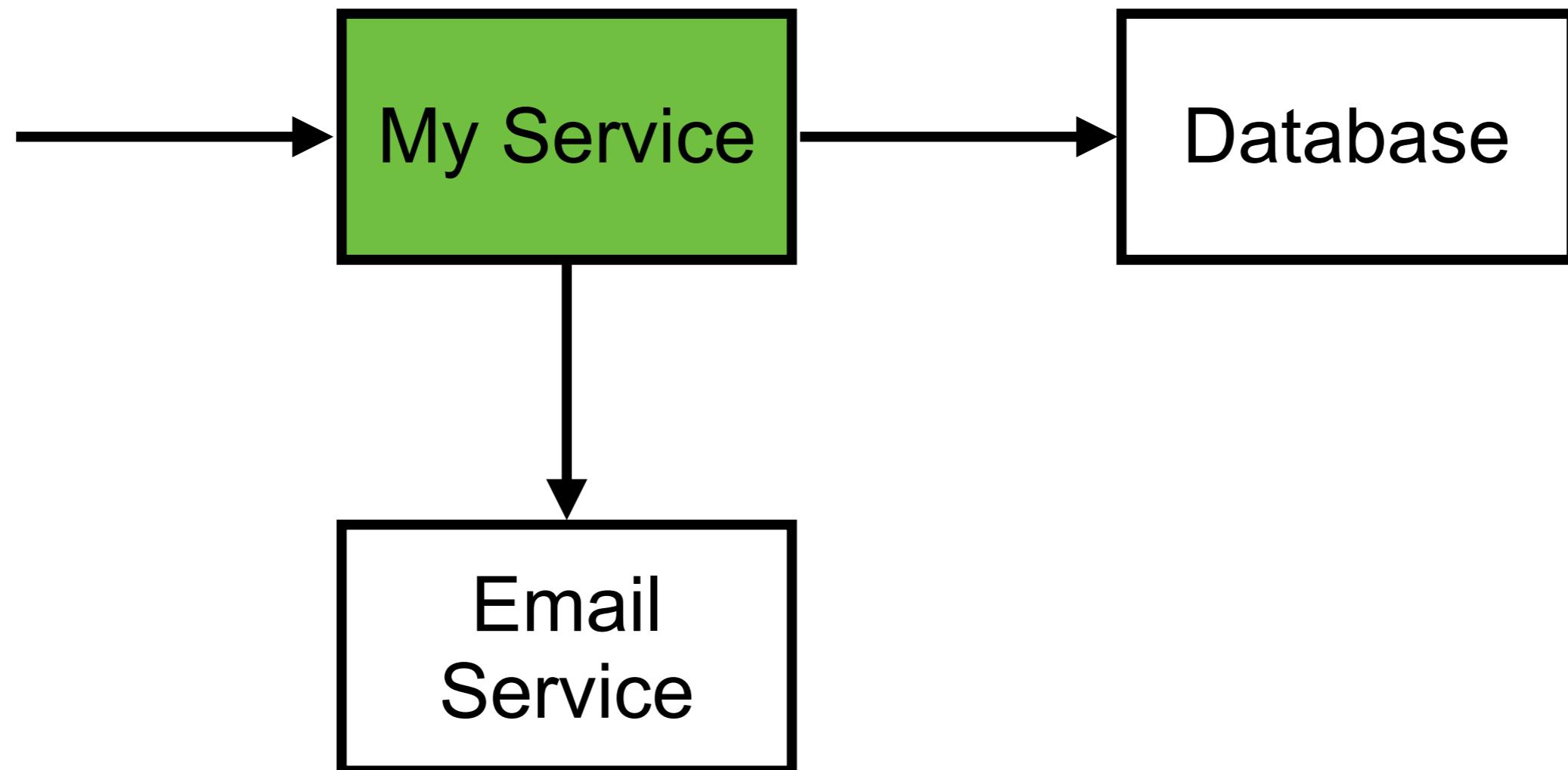
B()
}
```



# **Project structure with Testable !!**



# Structure of project



# Let's Start

```
type MyService struct {  
}  
  
func (s *MyService) Process() error {  
    // 1. Create database connection  
    // 2. Insert data into database  
    // 3. Create email service  
    // 4. Send email  
}
```



# Design with struct

```
type Database struct {  
}  
  
func (d *Database) connect() error {  
    return nil  
}  
func (d *Database) insertData() error {  
    return nil  
}
```

```
type EmailService struct {  
}  
  
func (e *EmailService) connect() error {  
    return nil  
}  
  
func (e *EmailService) send() error {  
    return nil  
}
```

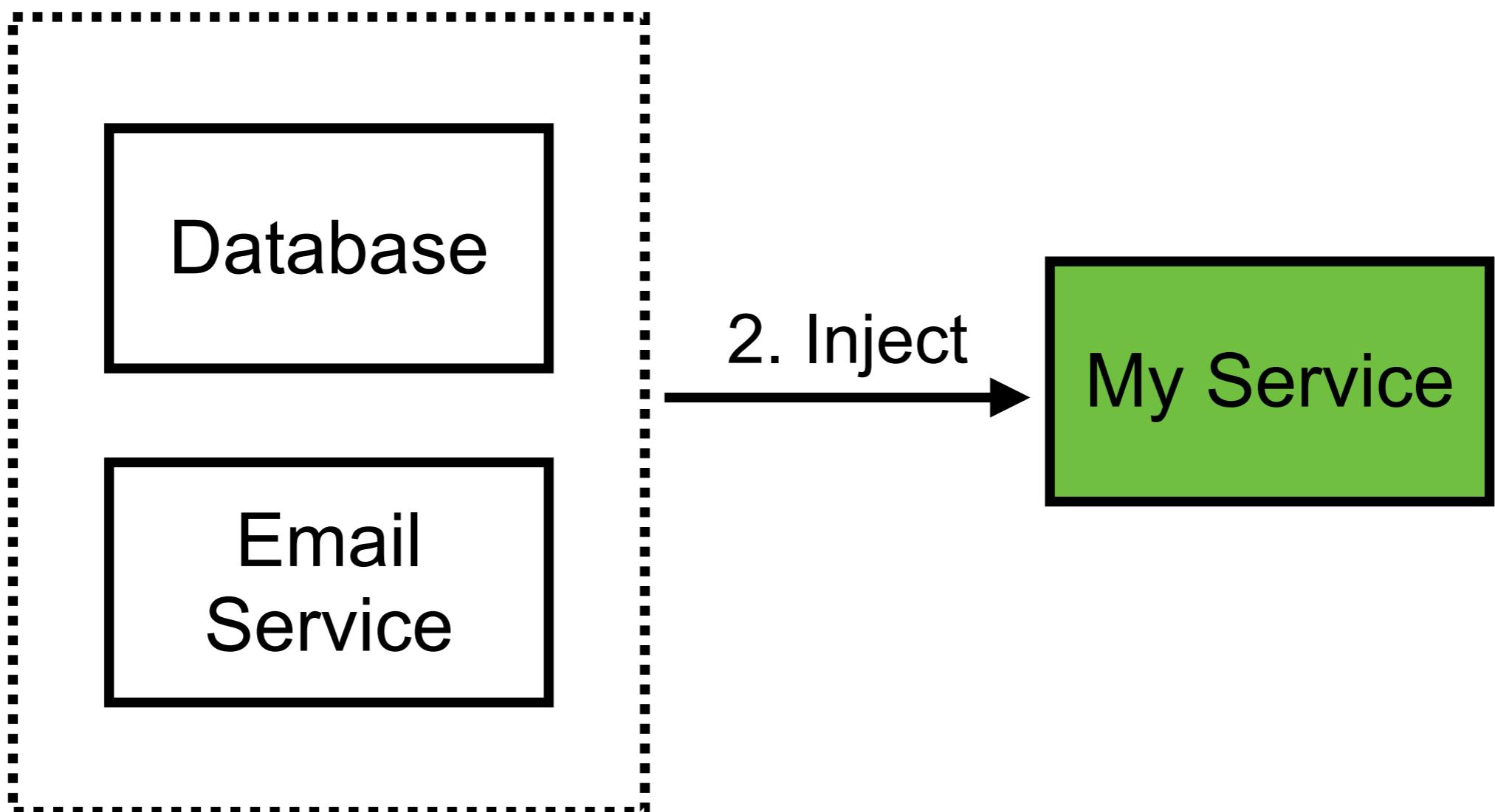


# **How to create database and email service ?**



# Use dependency injection

1. Create object of dependencies



# Use dependency injection

```
type MyService struct {
    Db      *Database
    Email   *EmailService
}

func (s *MyService) Process() error {
    // 1. Create database connection
    s.Db.connect()
    // 2. Insert data into database
    s.Db.insertData()
    // 3. Create email service
    s.Email.connect()
    // 4. Send email
    s.Email.send()
}
```



# Create MyService with dependencies

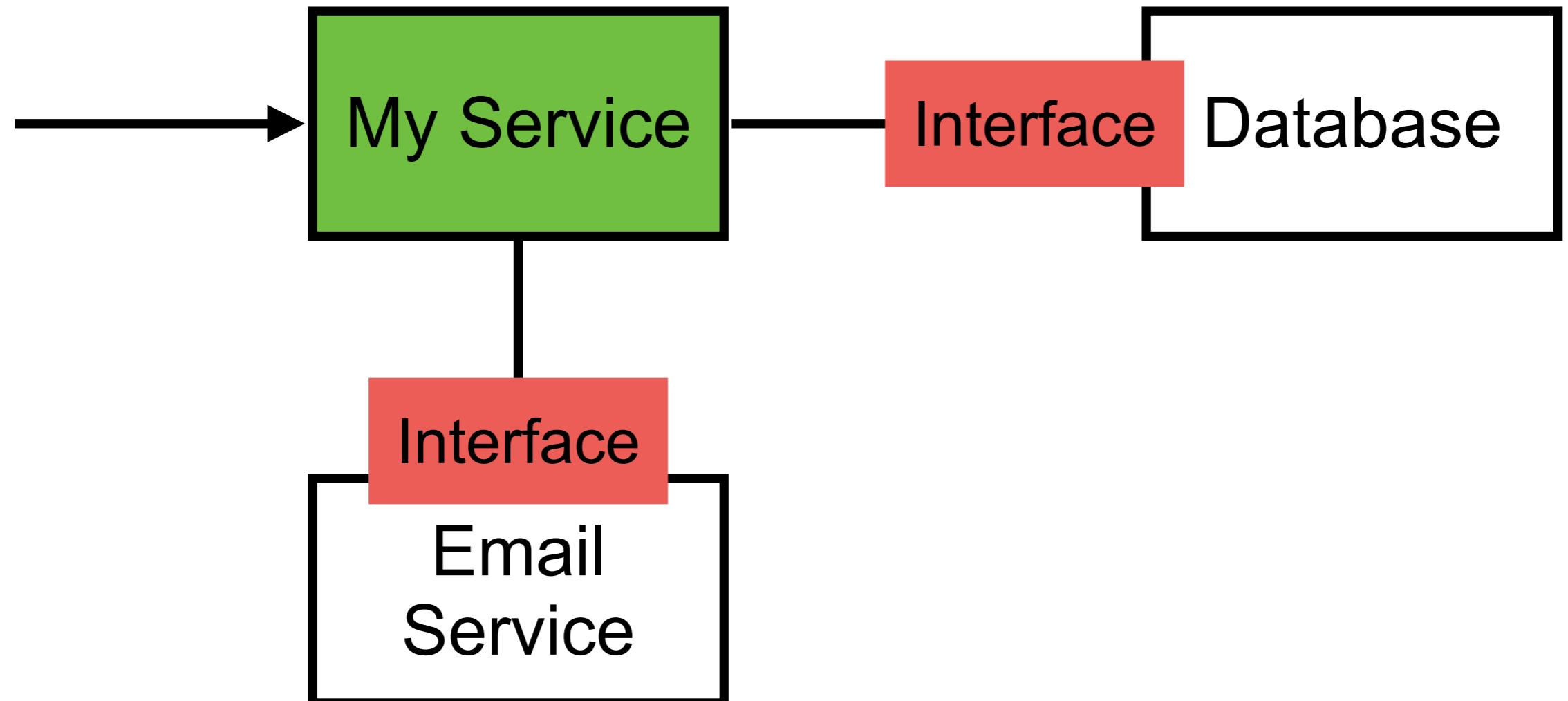
```
func NewMyService(db *Database, email *EmailService) *MyService
{
    return &MyService{
        Db:    db,
        Email: email,
    }
}
```



**How to test ?  
Easy to test ?  
Testable code ?**



# Working with interface ?



# Working with interface ?

```
type IDatabase interface {
    connect() error
    insertData() error
}

type IEmailService interface {
    connect() error
    send() error
}
```

```
type MyService struct {
    Db    IDatabase
    Email IEmailService
}

func NewMyService(db IDatabase, email IEmailService) *MyService {
    return &MyService{
        Db:    db,
        Email: email,
    }
}
```



# Try to test with Unit test

```
type MockDatabase struct{}

func (d *MockDatabase) connect() error {
    return nil
}

func (d *MockDatabase) insertData() error {
    return nil
}

type MockEmailService struct{}

func (e *MockEmailService) connect() error {
    return nil
}

func (e *MockEmailService) send() error {
    return nil
}
```



# Try to test with Unit test

```
func TestDI(t *testing.T) {  
  
    db := &MockDatabase{}  
    email := &MockEmailService{}  
    service := NewMyService(db, email)  
    service.Process()  
  
}
```

```
$go test -v
```



# Common mistakes with Interface

Create too many interfaces (pollution)

Don't design with interfaces, discover them

Create too many methods

Don't write behavior-driven interface

Create interface on the producer side

Create interface for testing



# Go Proverbs

Simple, Poetic, Pithy

Don't communicate by sharing memory, share memory by communicating.

Concurrency is not parallelism.

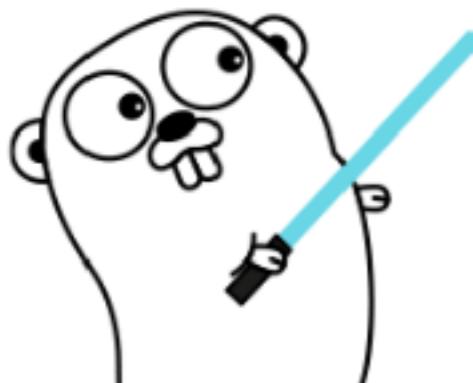
Channels orchestrate; mutexes serialize.

The bigger the interface, the weaker the abstraction.

Make the zero value useful.

`interface{}` says nothing.

Gofmt's style is no one's favorite, yet gofmt is everyone's favorite.



A little copying is better than a little dependency.

Syscall must always be guarded with build tags.

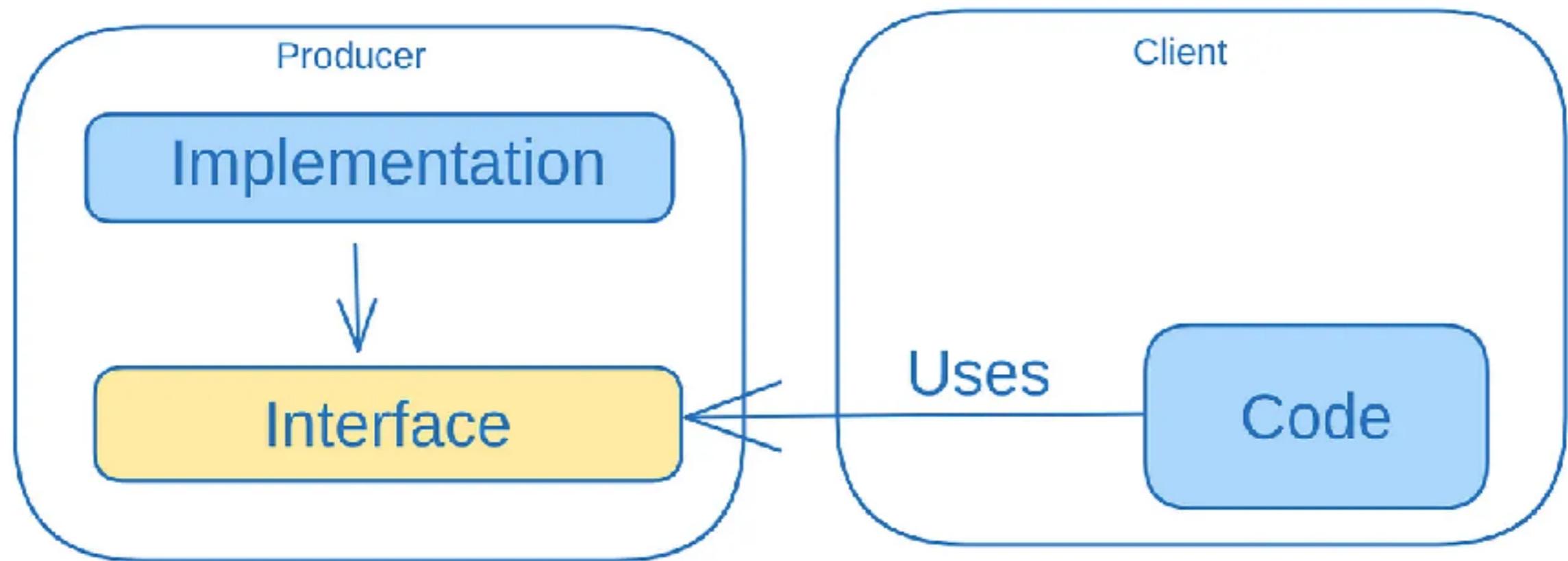
Cgo must always be guarded with build tags.

Cgo is not Go.

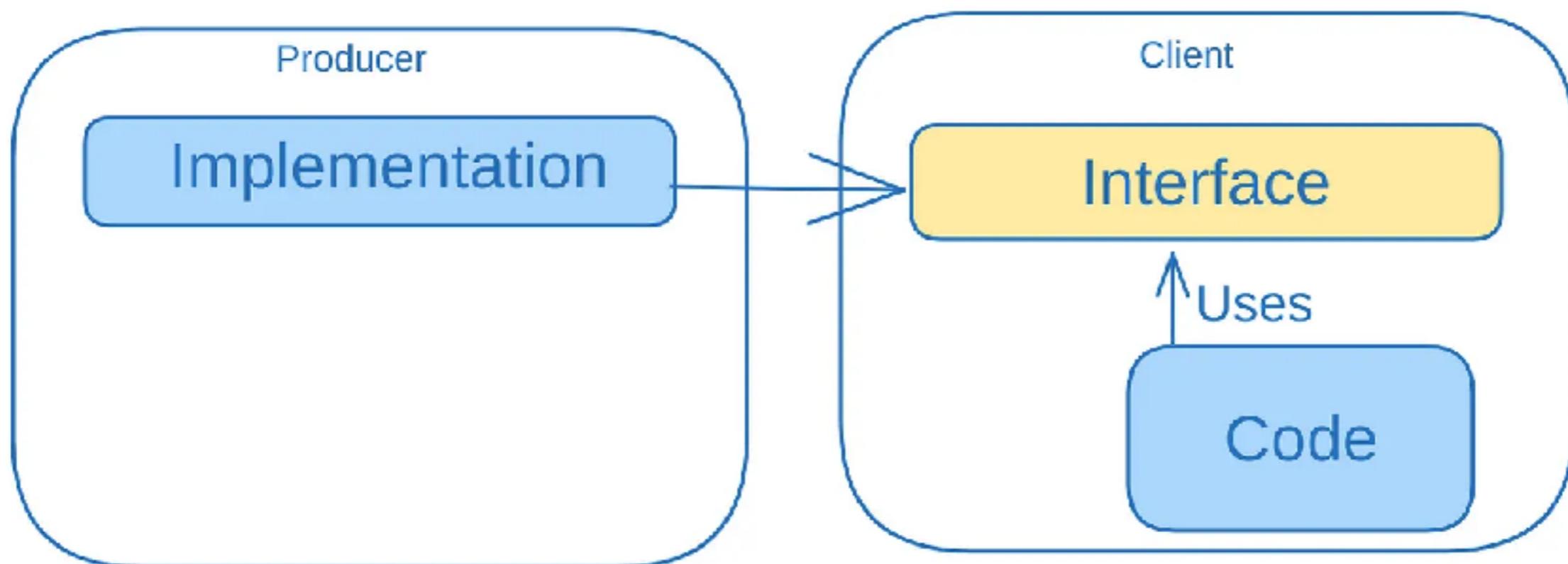
<https://dave.cheney.net/2016/04/27/dont-just-check-errors-handle-them-gracefully>



# Create interface on producer side



# Create interface on consumer side



# Better interface !!

```
type UsersRepository interface {  
    GetAllUsers()  
    GetUser(id string)  
}  
  
type UserRepository struct {  
}  
  
func (UserRepository) GetAllUsers() {}  
func (UserRepository) GetUser(id string) {}
```

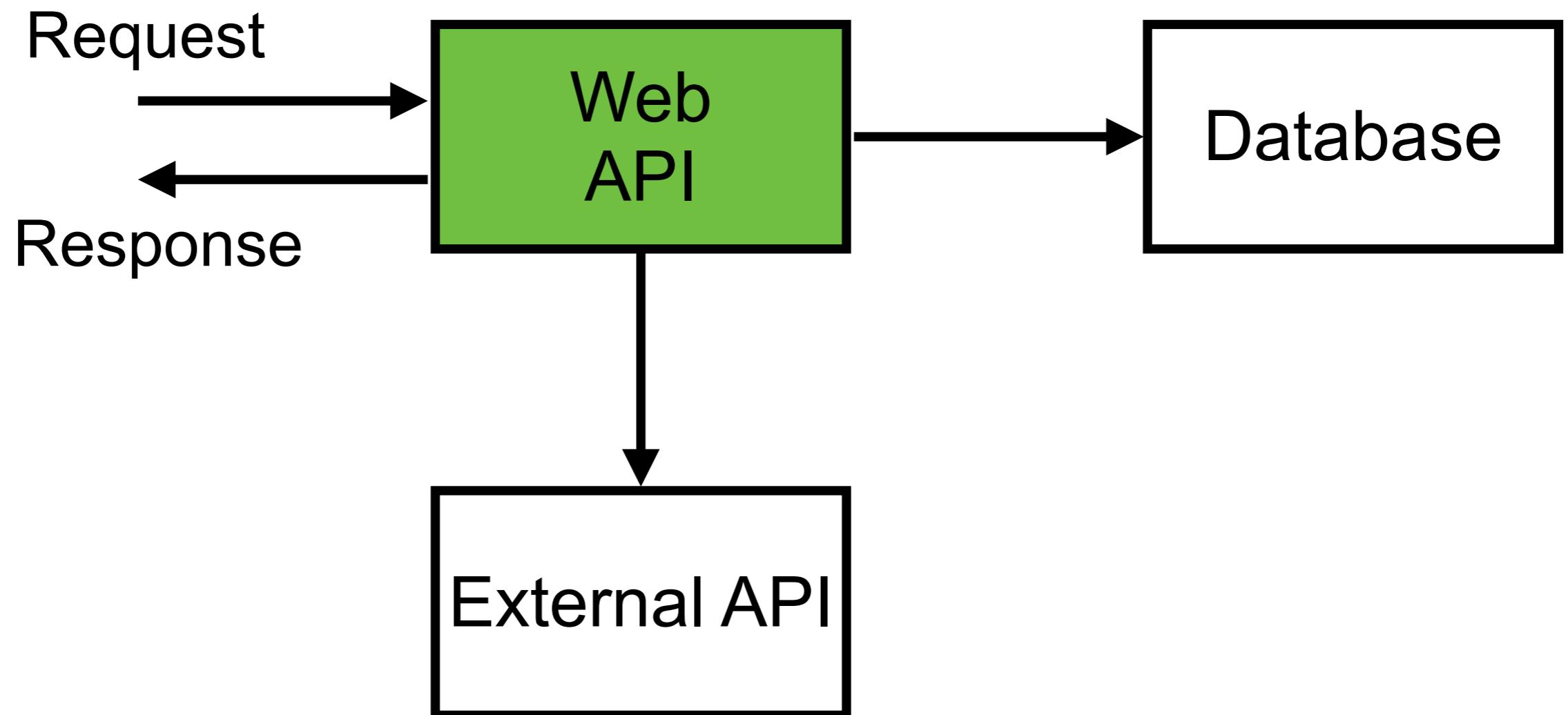
```
type UserGetter interface {  
    GetUser(id string)  
}
```



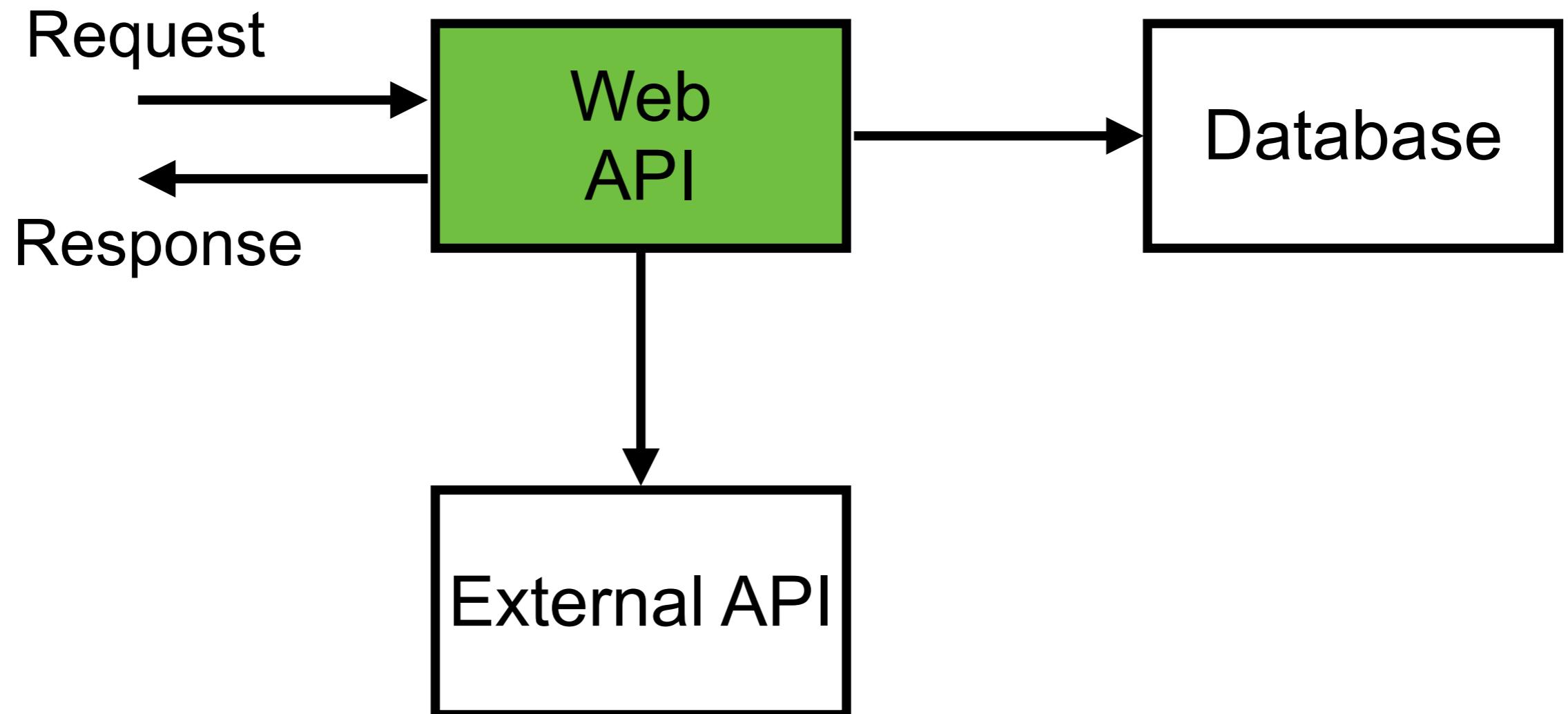
# Test Strategies



# Architecture



# What to Test ?



# Component and Contract Testing

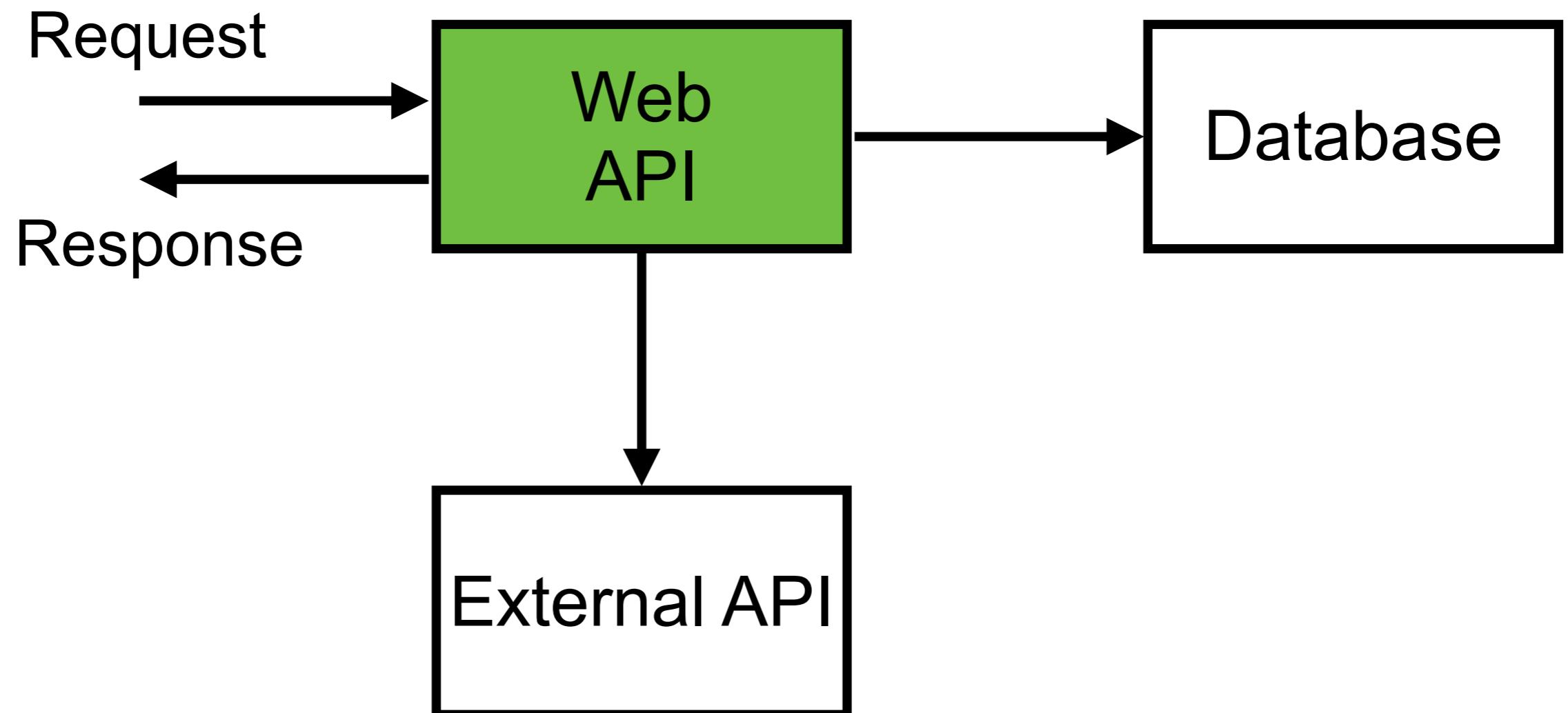
<https://github.com/up1/course-contract-testing>



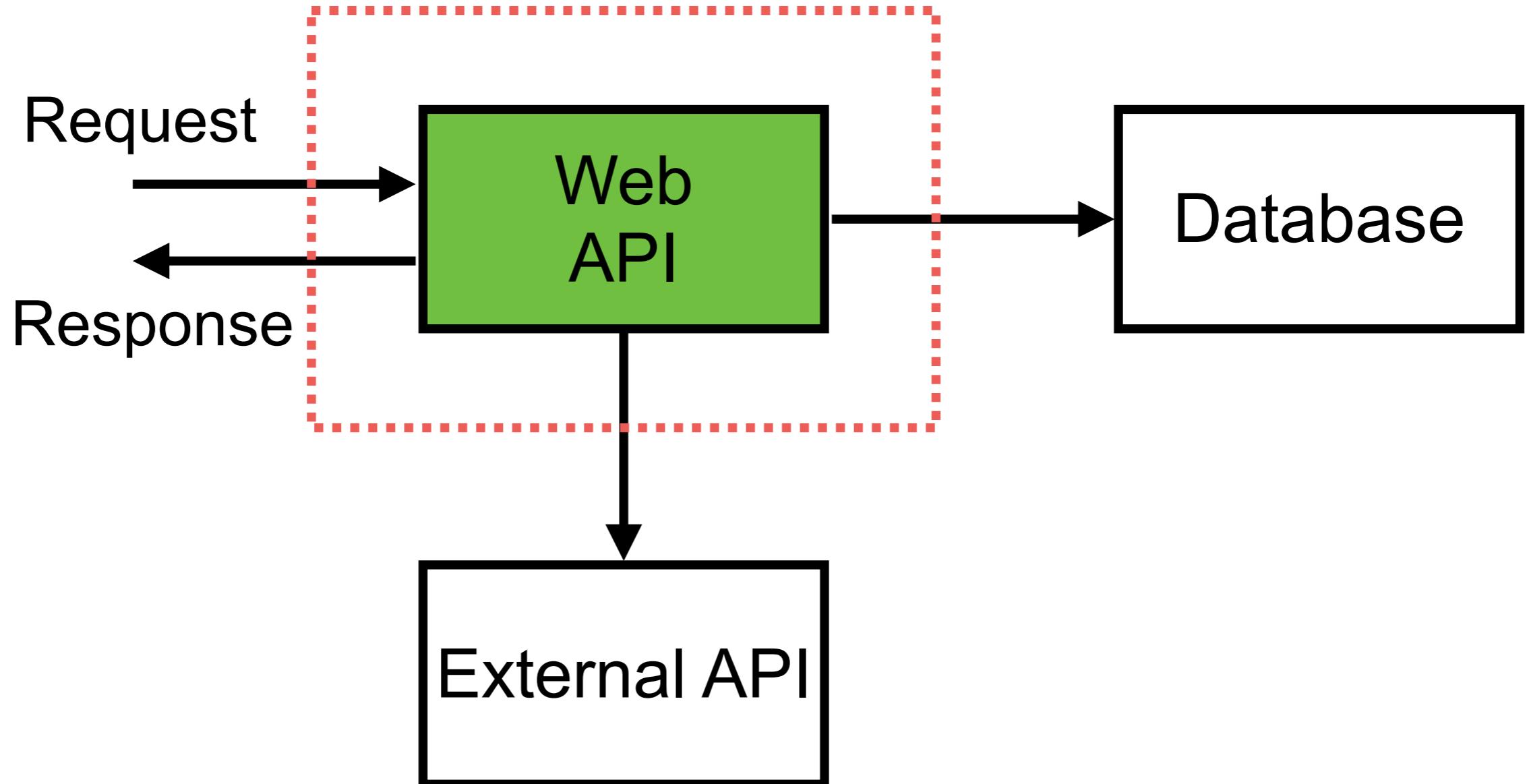
# Component Testing



# Component Testing



# Component Testing

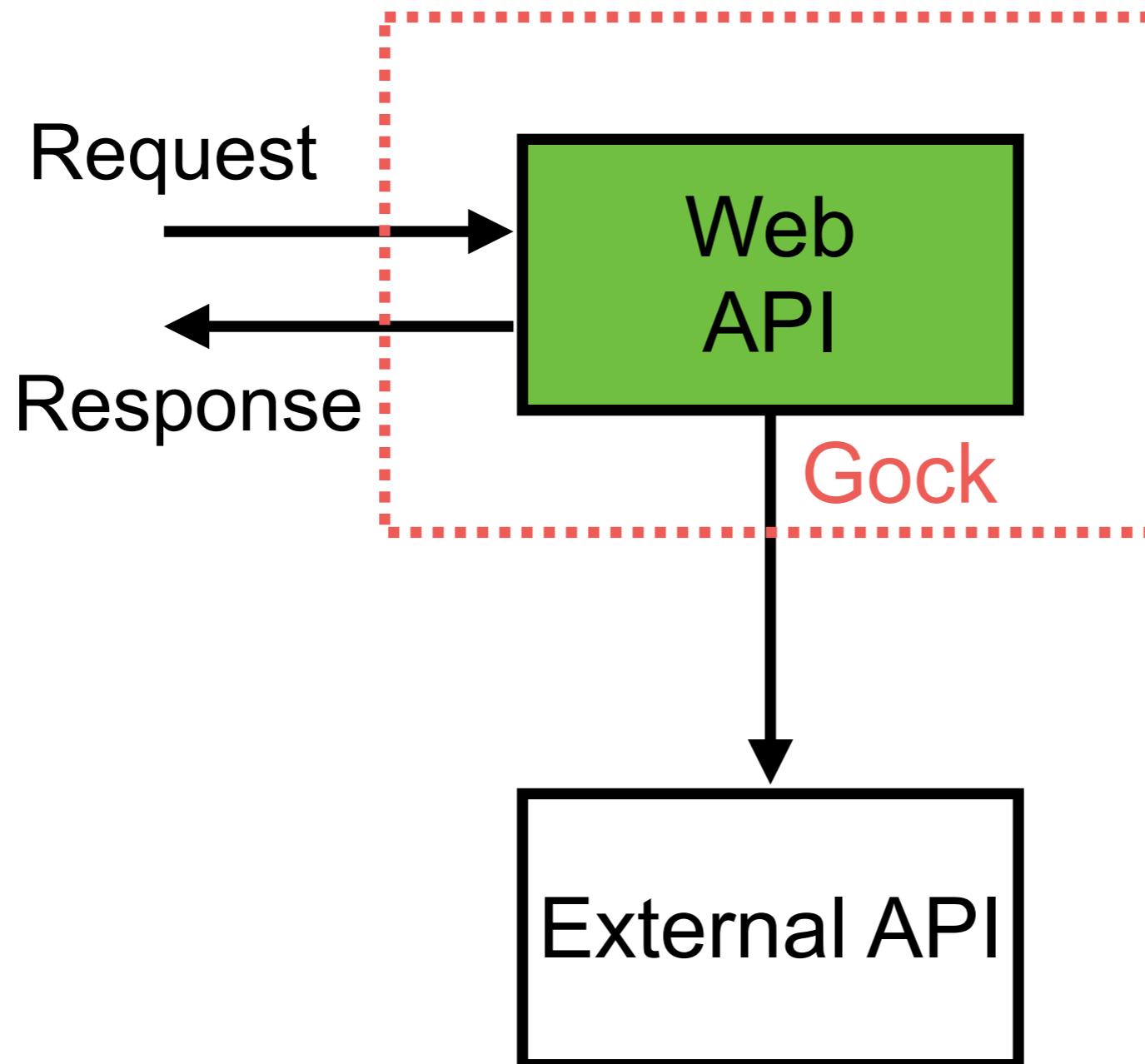


# Component testing

In-process  
Separate process



# In-process with Gock



<https://github.com/h2non/gock>



# Try to test with Gock

```
func TestSimple(t *testing.T) {
    defer gock.Off()

    // Arrange
    gock.New("http://facebook.com").
        Get("/bar").
        Reply(200).
        JSON(map[string]string{"foo": "bar"})

    // Act
    res, err := http.Get("http://facebook.com/bar")

    // Assert
    st.Expect(t, err, nil)
    st.Expect(t, res.StatusCode, 200)

    body, _ := ioutil.ReadAll(res.Body)
    st.Expect(t, string(body)[:13], `{"foo":"bar"}`)

    // Verify that we don't have pending mocks
    st.Expect(t, gock.IsDone(), true)
}
```

```
$go test -v
```

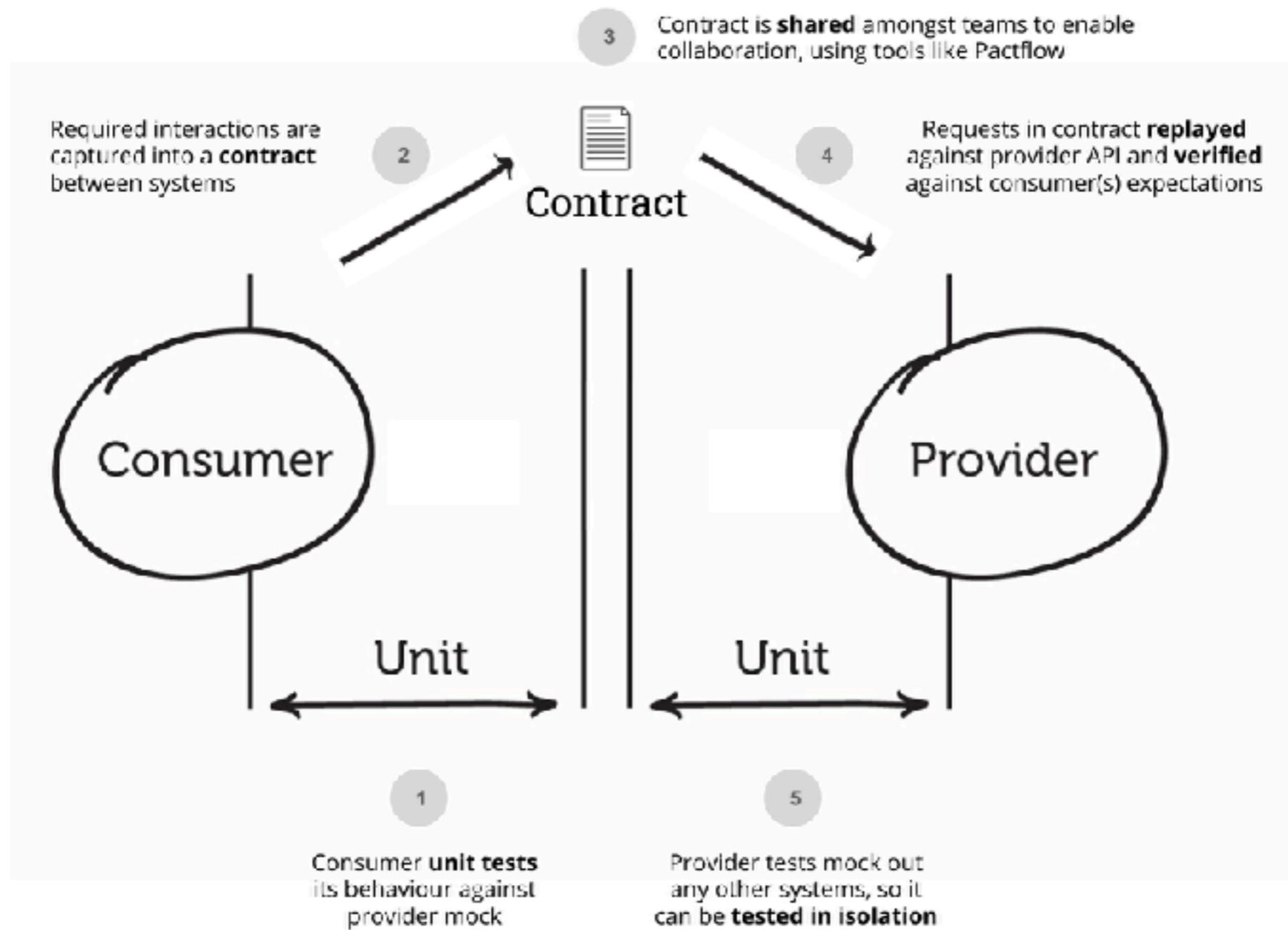


# Contract Testing

<https://github.com/up1/course-contract-testing>



# Contract Testing



<https://docs.pact.io/>



Go workshop

© 2022 - 2024 Siam Chamnkit Company Limited. All rights reserved.

# Pact Broker

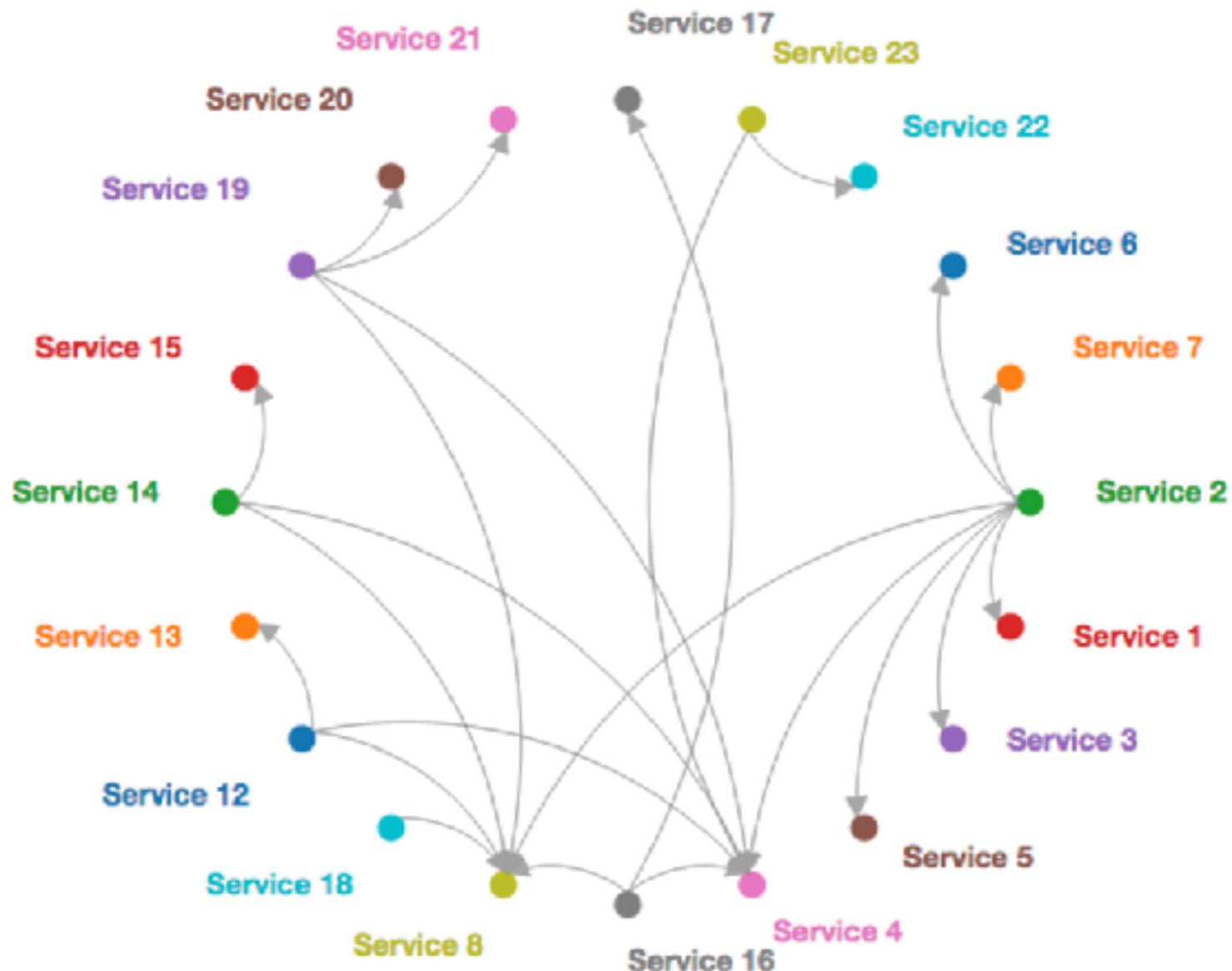
## Pacts

Consumer ↓↑	Provider ↓↑	Latest pact published	Last verified
Foo	Animals	2 minutes ago	2 days ago
Foo	Bar	7 days ago	15 days ago ⚡
Foo	Hello World App	1 day ago	
Foo	Wiffles	less than a minute ago	7 days ago
Some other app	A service	26 days ago	less than a minute ago
The Android App	The back end	less than a minute ago	

<https://docs.pact.io/>



# Pact Broker



<https://docs.pact.io/>



Go workshop

© 2022 - 2024 Siam Chamnkit Company Limited. All rights reserved.

# Project Structure



# Good structure ?

Consistent

Easy to understand, navigate and reason

Easy to change, loosely coupled

**Easy to test**

Simple as possible

Structure reflects the design !!



# Project Structure

Flat structure

Layer (group of function)

Group by module

Group by context

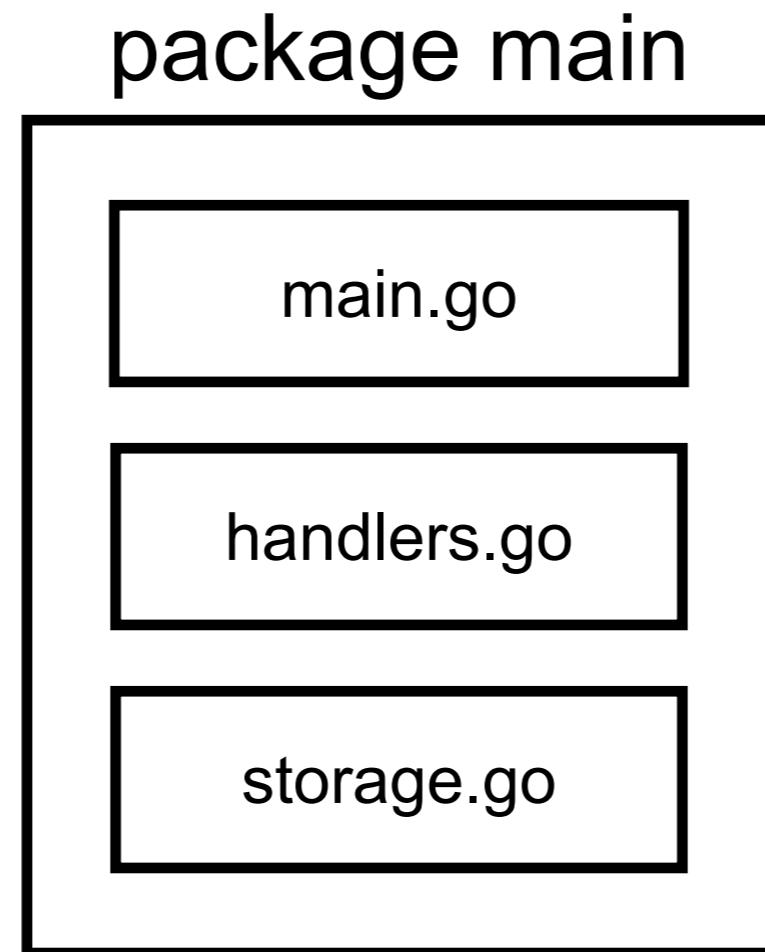
Clean architecture

Hexagonal architecture



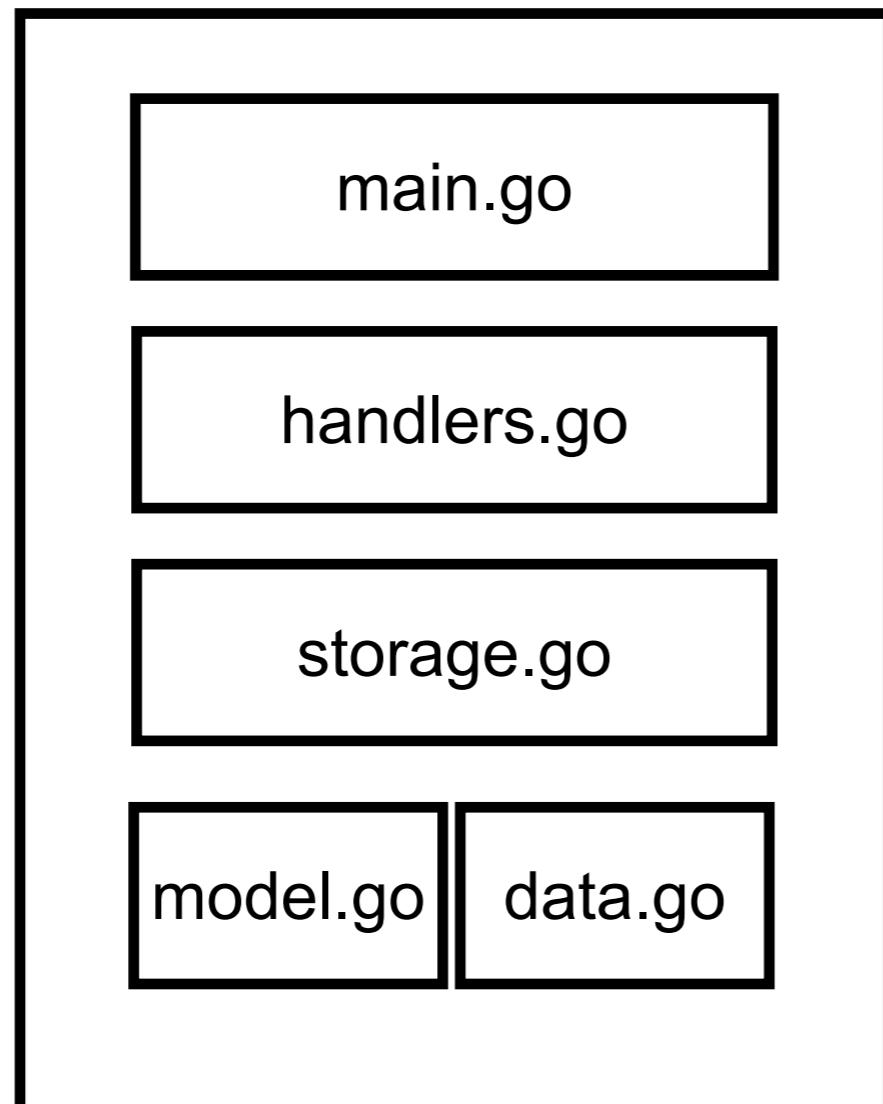
# Flat structure

Easy to start  
Only main package



# Beer !!

package main

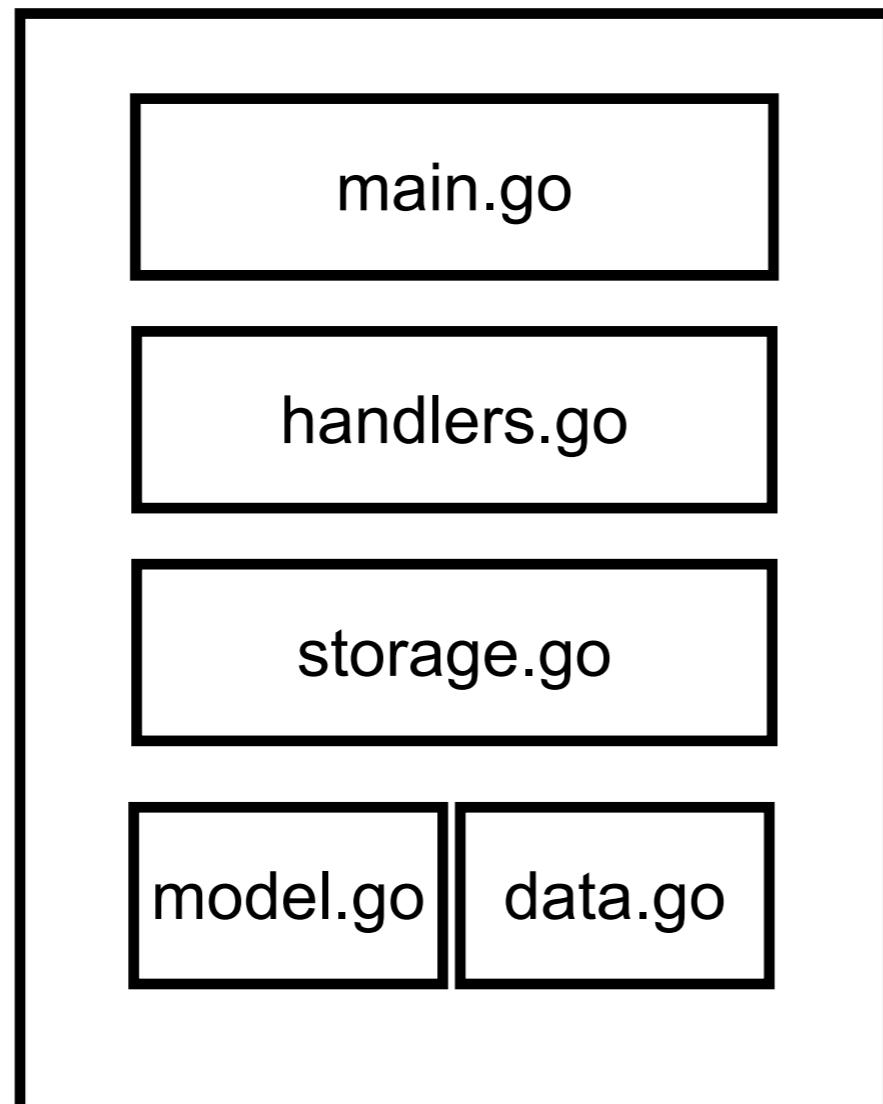


Create Storage as global variable  
Start Web server



# Beer !!

package main



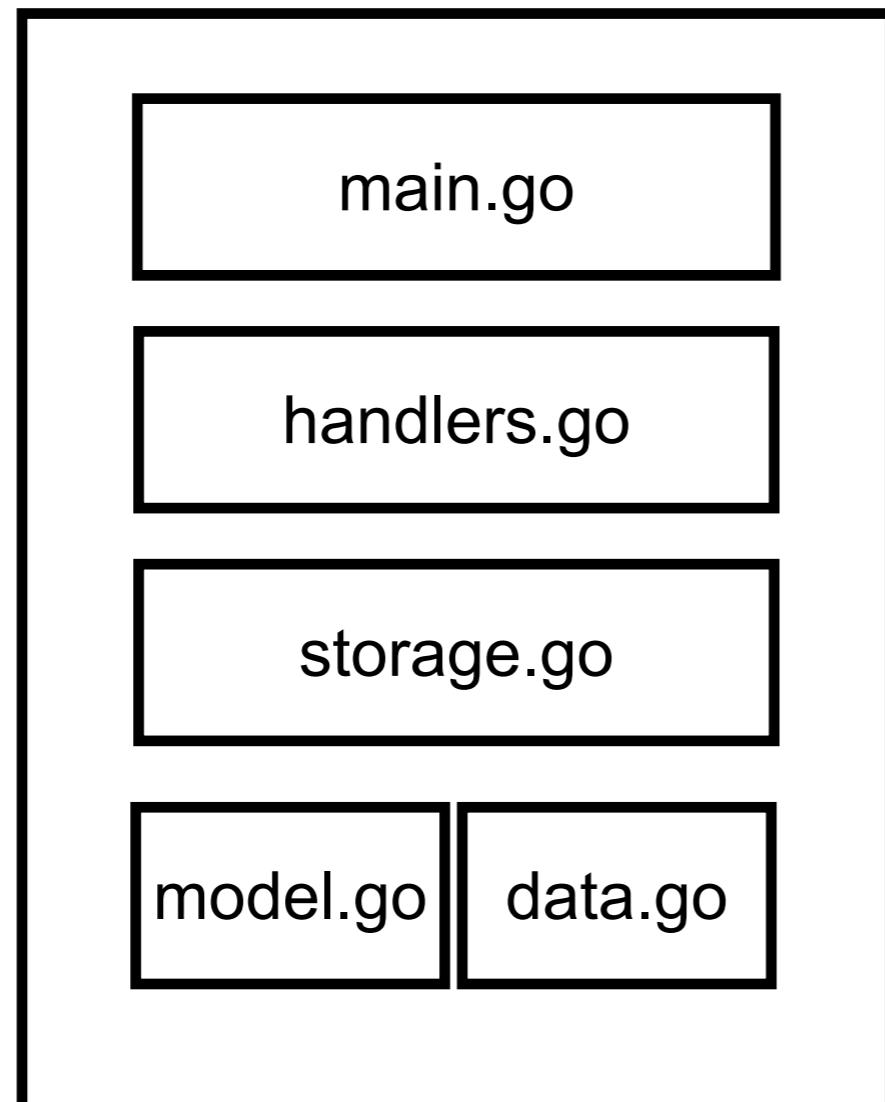
Create Storage as global variable  
Start Web server

GET /beers  
POST /beers



# Beer !!

package main



Create Storage as global variable

Start Web server

GET /beers

POST /beers

GetBeers() from memory

SaveBeer(beer) into memory



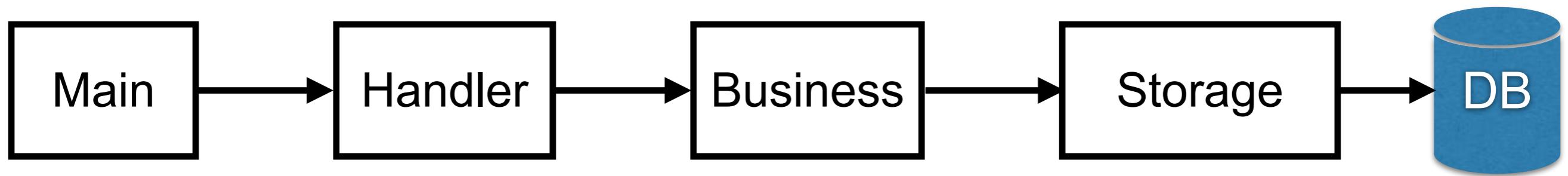
# How to test ?

Storage  
Handler (integration)  
Handler (mock)



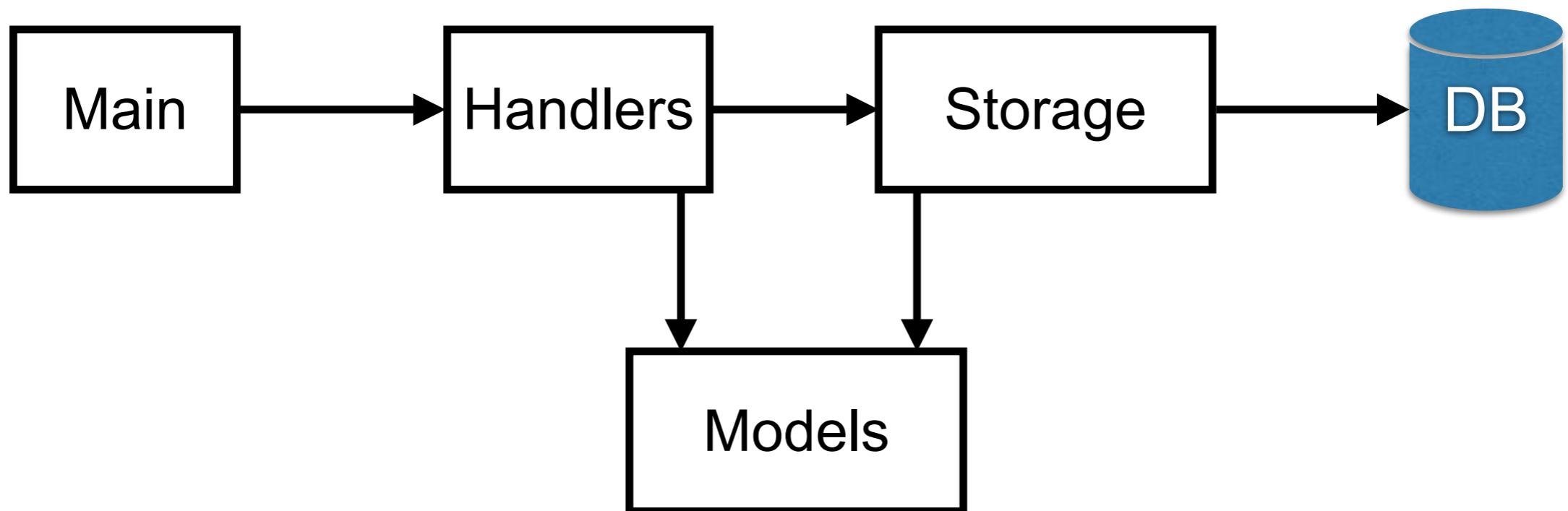
# Layer

Grouping by function



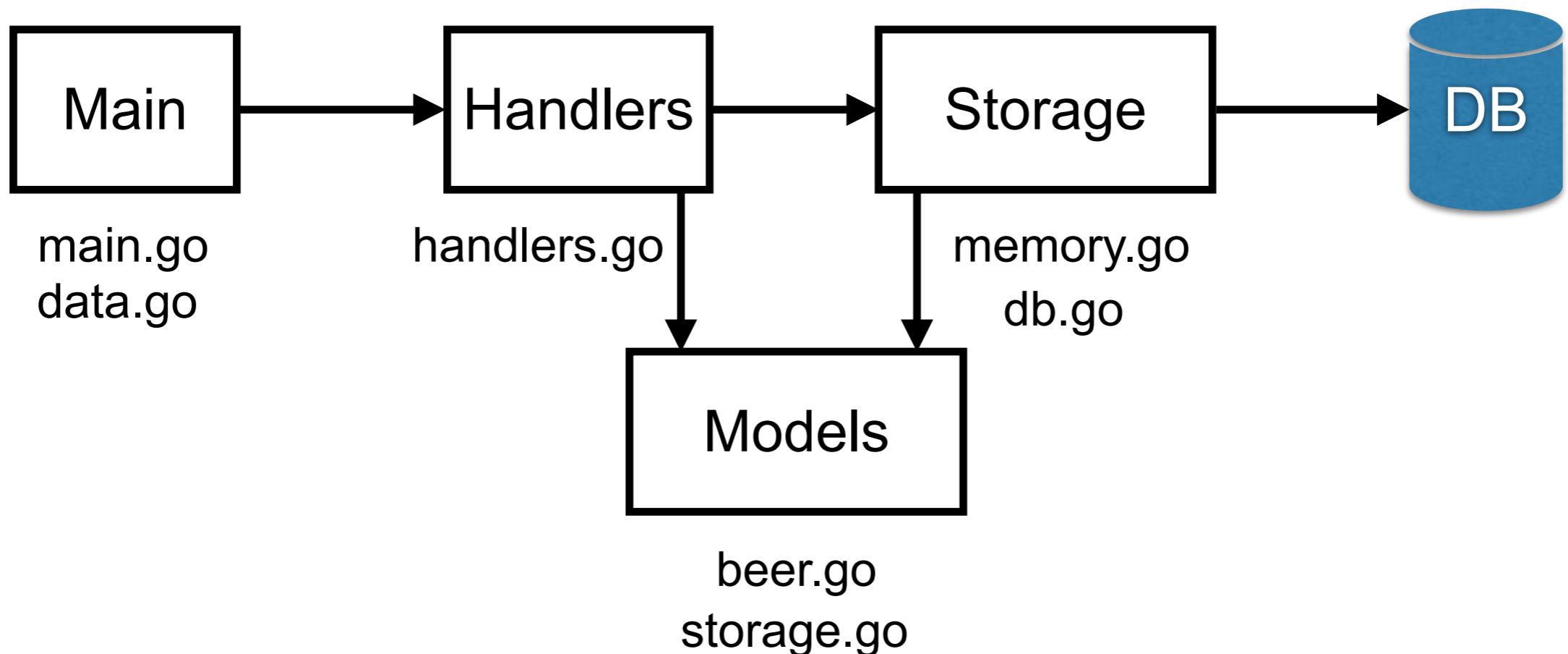
# Layer

Grouping by function



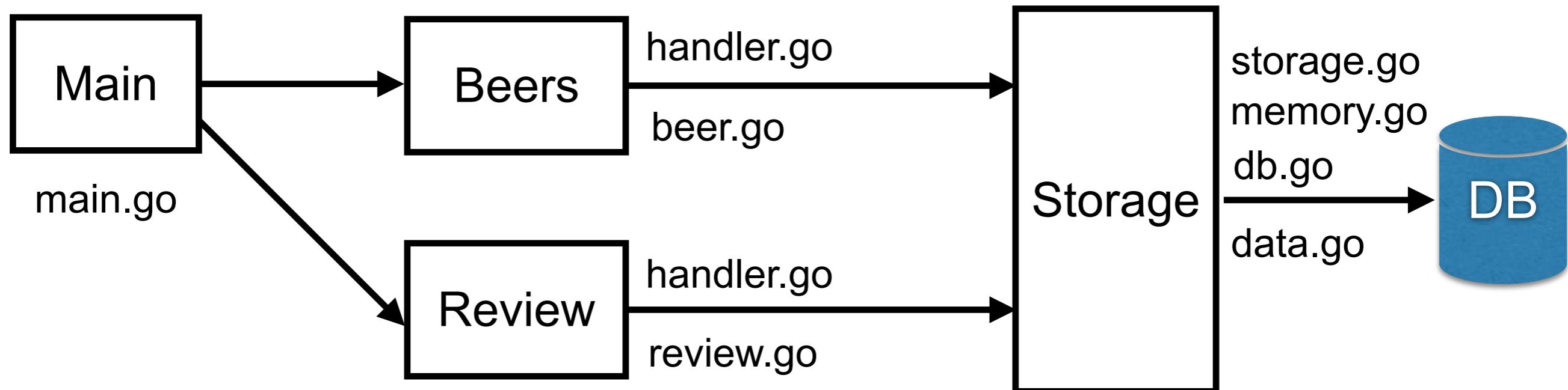
# Layer

## Grouping by function



# Modular

## Grouping by module



# Manage Dependency



# Manage Dependency ?



# Manage Dependency ?

## Global variable



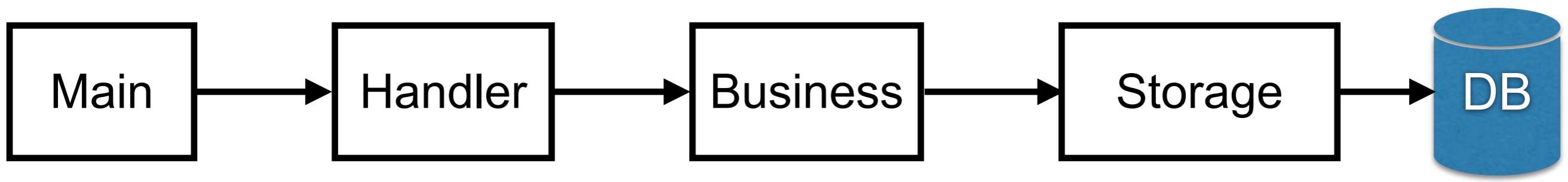
```
// Global variable
var db Storage

func setupRouter() *gin.Engine {
    var err error
    db, err = NewStorage()
    if err != nil {
        log.Fatal(err)
    }
}
```



# Manage Dependency ?

Dependency with Struct



```
type MyConfig struct {
    Db Storage
}

func setupRouter() *gin.Engine {
    db, _ := NewStorage()
    config := &MyConfig{Db: db}

    PopulateBeers(config)

    r := gin.New()
    r.GET("/beers", config.GetBeers)
    r.POST("/beers", config.AddBeer)
    return r
}
```

## 1. Create and initial

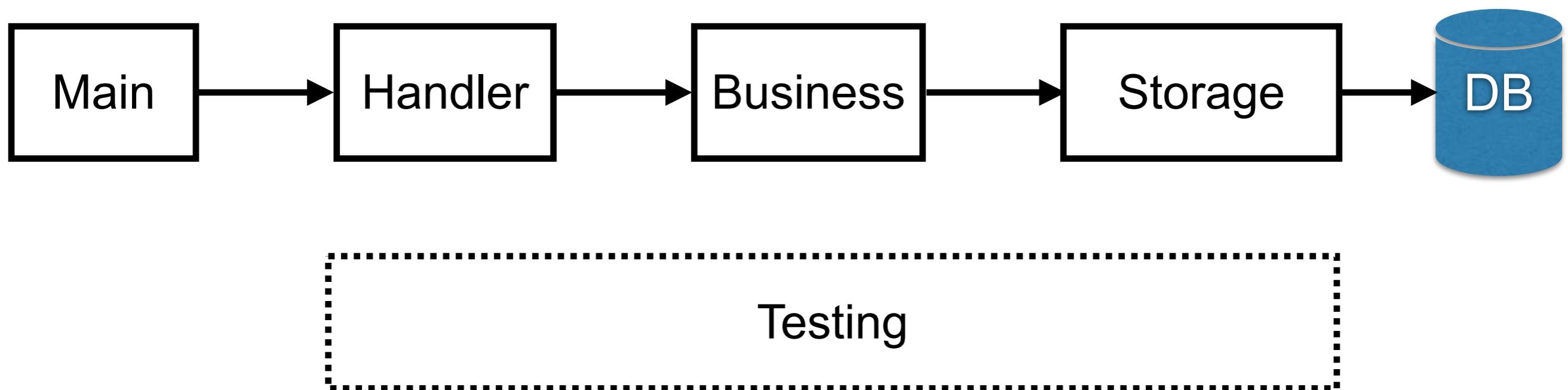
```
func (config *MyConfig) GetBeers(c *gin.Context) {
    beers, err := config.Db.FindBeers()
    if err != nil {
        c.AbortWithStatus(http.StatusBadRequest)
        return
    }
    c.JSON(http.StatusOK, beers)
}
```

## 2. Add receiver



# Manage Dependency ?

Dependency with closure



```

type MyConfig struct {
    Db Storage
}

func setupRouter() *gin.Engine {
    db, _ := NewStorage()
    config := &MyConfig{Db: db}

    PopulateBeers(config)

    r := gin.New()
    r.GET("/beers", GetBeers(config))
    r.POST("/beers", AddBeer(config))
    return r
}

```

## 1. Create and initial

```

func GetBeers(config *MyConfig) gin.HandlerFunc {
    return func(c *gin.Context) {
        beers, err := config.Db.FindBeers()
        if err != nil {
            c.AbortWithStatus(http.StatusBadRequest)
            return
        }
        c.JSON(http.StatusOK, beers)
    }
}

```

## 2. Use closure



# RESTful API with Go

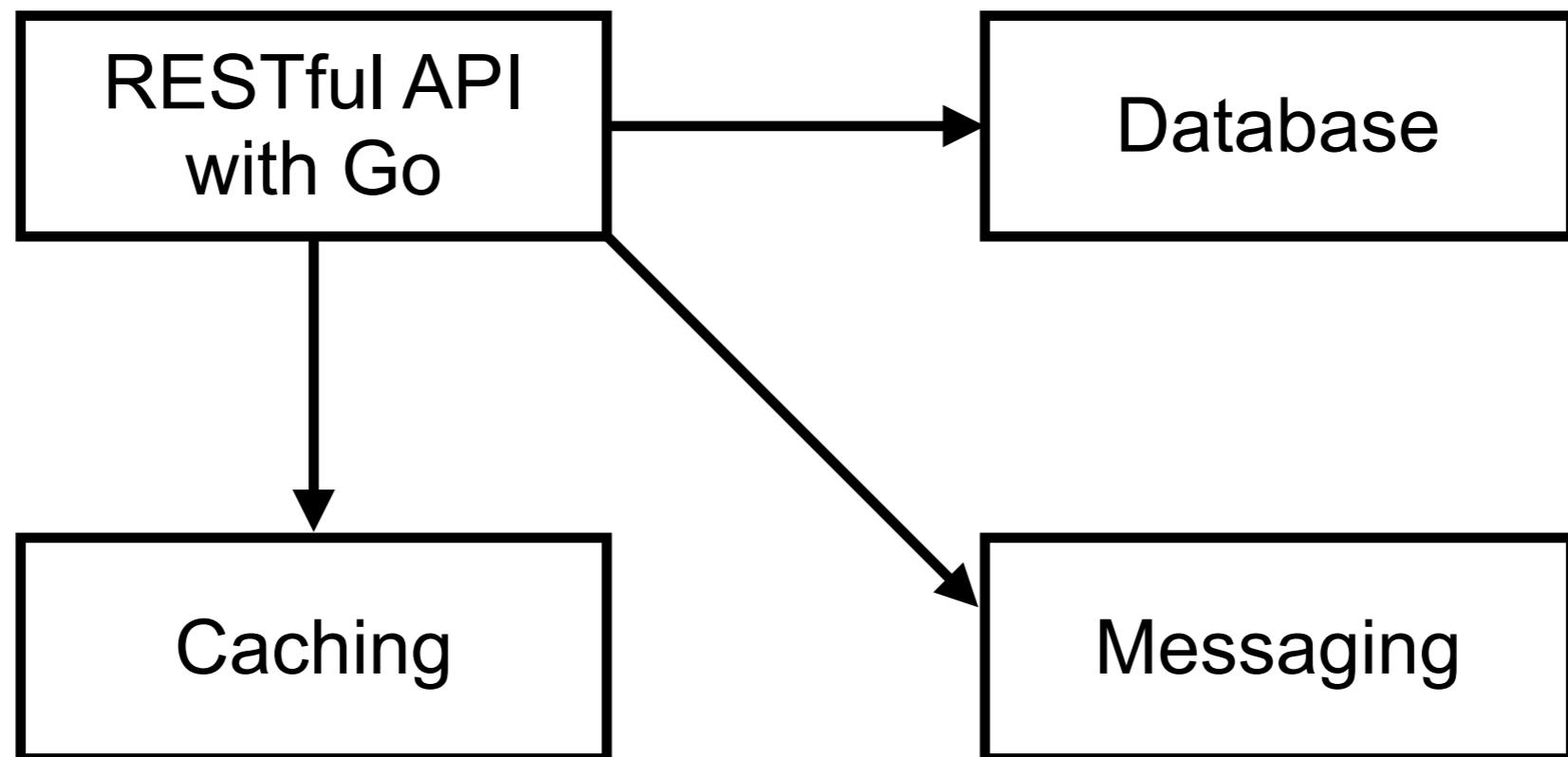


# Workshop

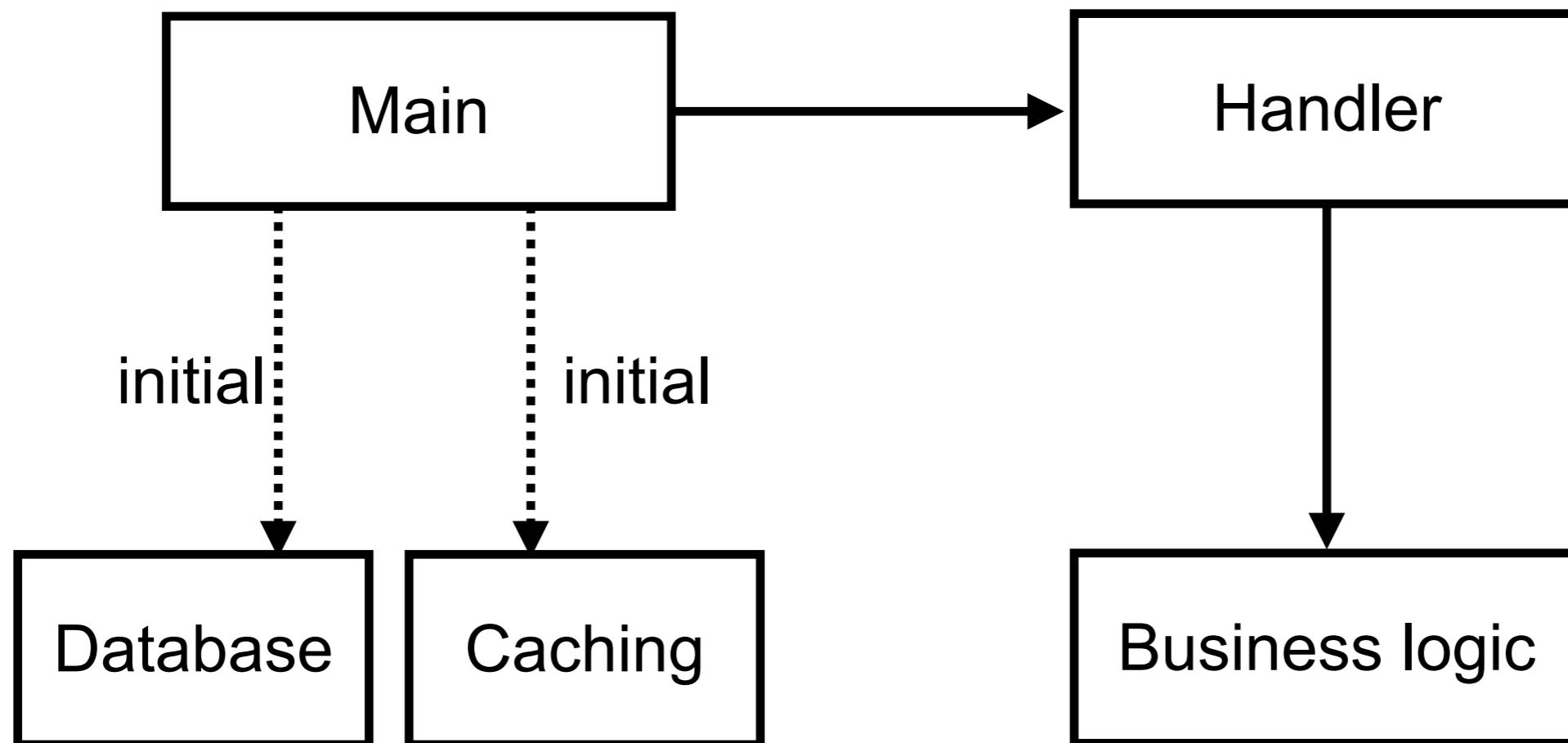
<https://github.com/up1/course-go-2024/wiki/REST-API>



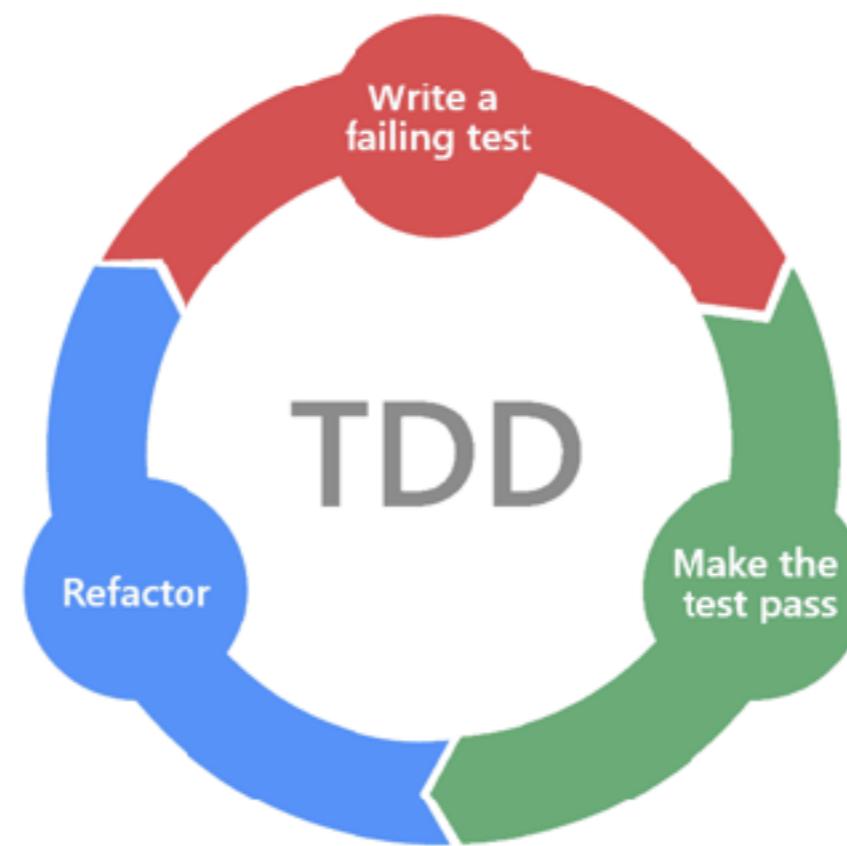
# Architecture



# Structure for Go project ?



# Try to refactor with safety

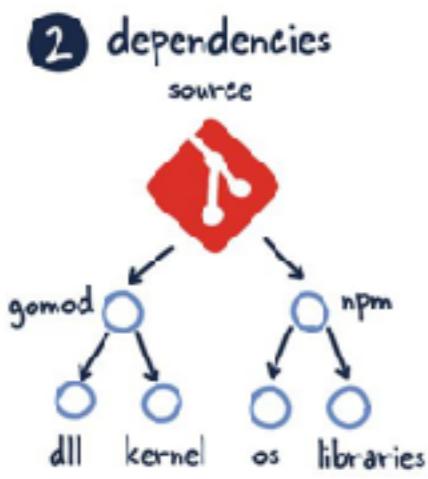
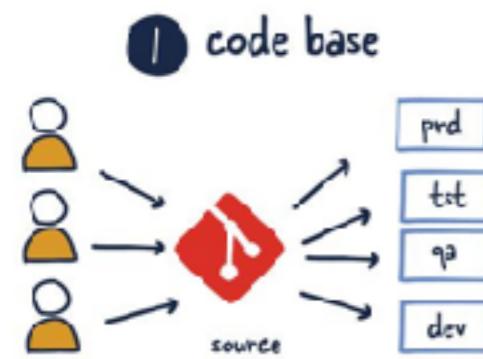


# Twelve factors

<https://12factor.net/>



# 12 FACTOR APP REVISITED



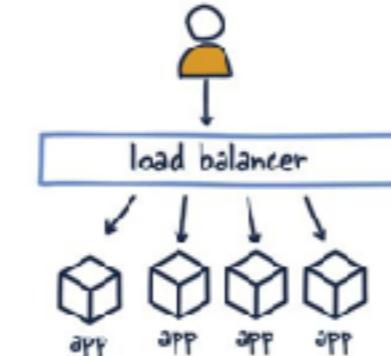
## 4 backing services



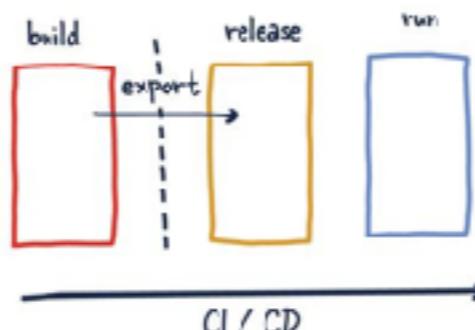
## 6 Processes



## 8 concurrency



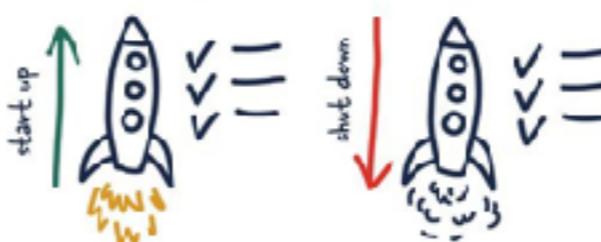
## 5 build, release, run



## 7 port binding



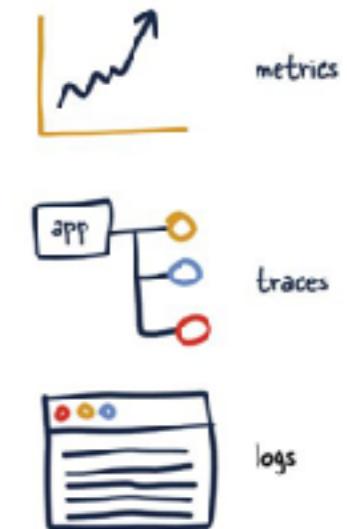
## 9 disposability



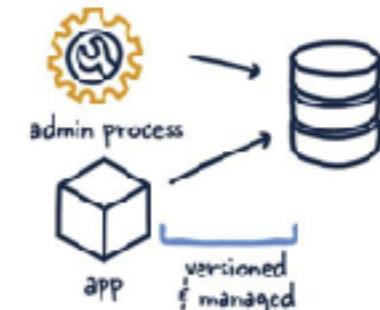
## 10 dev/prod parity



## 11 logs



## 12 admin processes



<https://architecturenotes.co/p/12-factor-app-revisited>

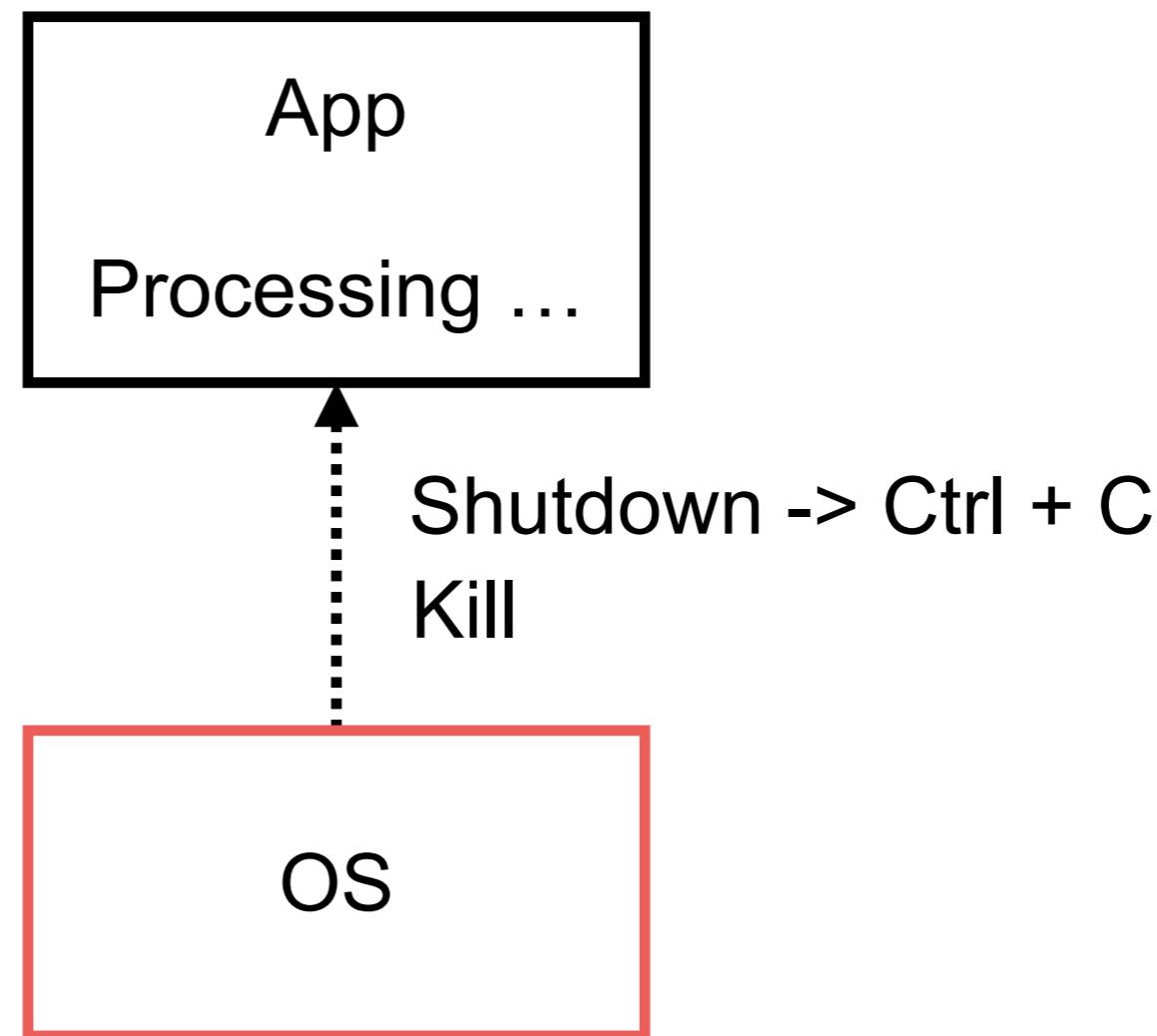


Go workshop

© 2022 - 2024 Siam Chamnkit Company Limited. All rights reserved.

# Graceful Shutdown

How to make sure, existing requests are successfully responded before shutdown ?



# Signals

Signal	Action	
SIGTERM	Kill command	kill <process id>
SIGINT	CTRL + C	Terminate foreground process
SIGKILL SIGSTOP		Can't handle from code



# Structured Logging

<https://go.dev/blog/slog>



# Log Format ?

## Structured

- Usually JSON or LogFmt.
- Least human-readable.
- Easy to parse, search and filter.
- Consistent formatting.
- Faster troubleshooting due to added context.
- Automated reporting and analysis.



RECOMMENDED

## Semi-structured

- Mix of structured and unstructured data.
- Human-readable.
- Formatting variability leads to inconsistency.
- Limited automation possibilities.
- Requires a more complex parsing logic.

## Unstructured

- Free-form plain text messages.
- Human-readable.
- Challenging to parse.
- Requires manual interpretation.
- Suitable only for quick ad-hoc logging in development environments.



NOT RECOMMENDED

<https://betterstack.com/community/guides/logging/structured-logging/>



Go workshop

© 2022 - 2024 Siam Chamnankit Company Limited. All rights reserved.

# Log Format ?

```
2024-06-17T08:55:00Z [INFO] User 'user123' logged in from 192.168.0.1
2024-06-17T08:55:01Z [ERROR] Database connection failed: Connection refused
2024-06-17T08:55:00Z [INFO] User 'user123' logged in from 192.168.0.1
2024-06-17T08:55:01Z [ERROR] Database connection failed: Connection refused
```

```
{
  "timestamp": "2024-06-17T08:55:00Z",
  "level": "INFO",
  "message": "User 'user123' logged in",
  "user": "user123",
  "ip_address": "192.168.0.1"
},
{
  "timestamp": "2024-06-17T08:55:01Z",
  "level": "ERROR",
  "message": "Database connection failed",
  "error": {
    "code": "DB_CONNECT_ERROR",
    "message": "Connection refused"
  }
}
```



# Structured Log

```
{  
  "timestamp": "2024-06-17T08:55:00Z",  
  "level": "INFO",  
  "message": "User 'user123' logged in",  
  "user": "user123",  
  "ip_address": "192.168.0.1"  
},  
{  
  "timestamp": "2024-06-17T08:55:01Z",  
  "level": "ERROR",  
  "message": "Database connection failed",  
  "error": {  
    "code": "DB_CONNECT_ERROR",  
    "message": "Connection refused"  
  }  
}
```

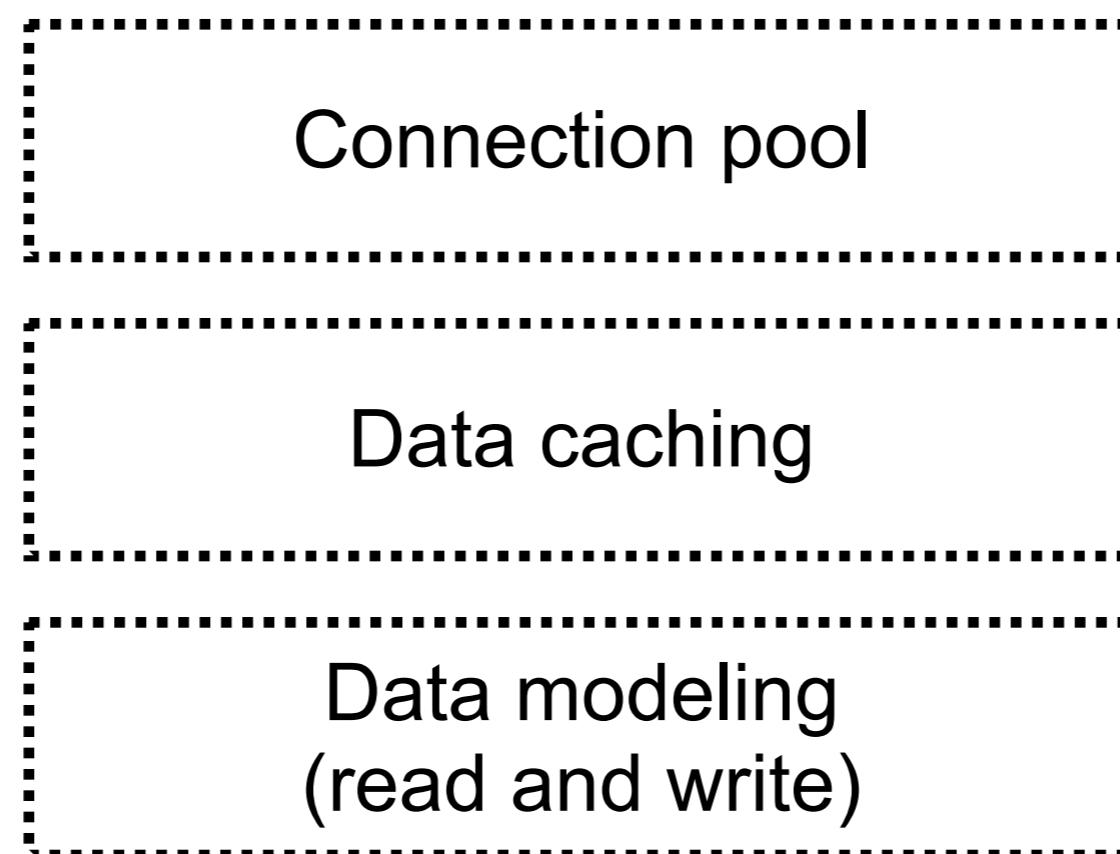
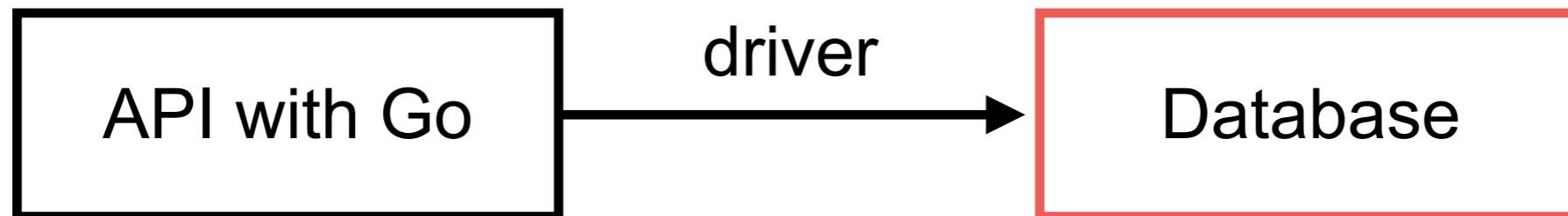
Machine readable format  
Key-value pairs  
Contextual information  
Hierarchical structure



# Database



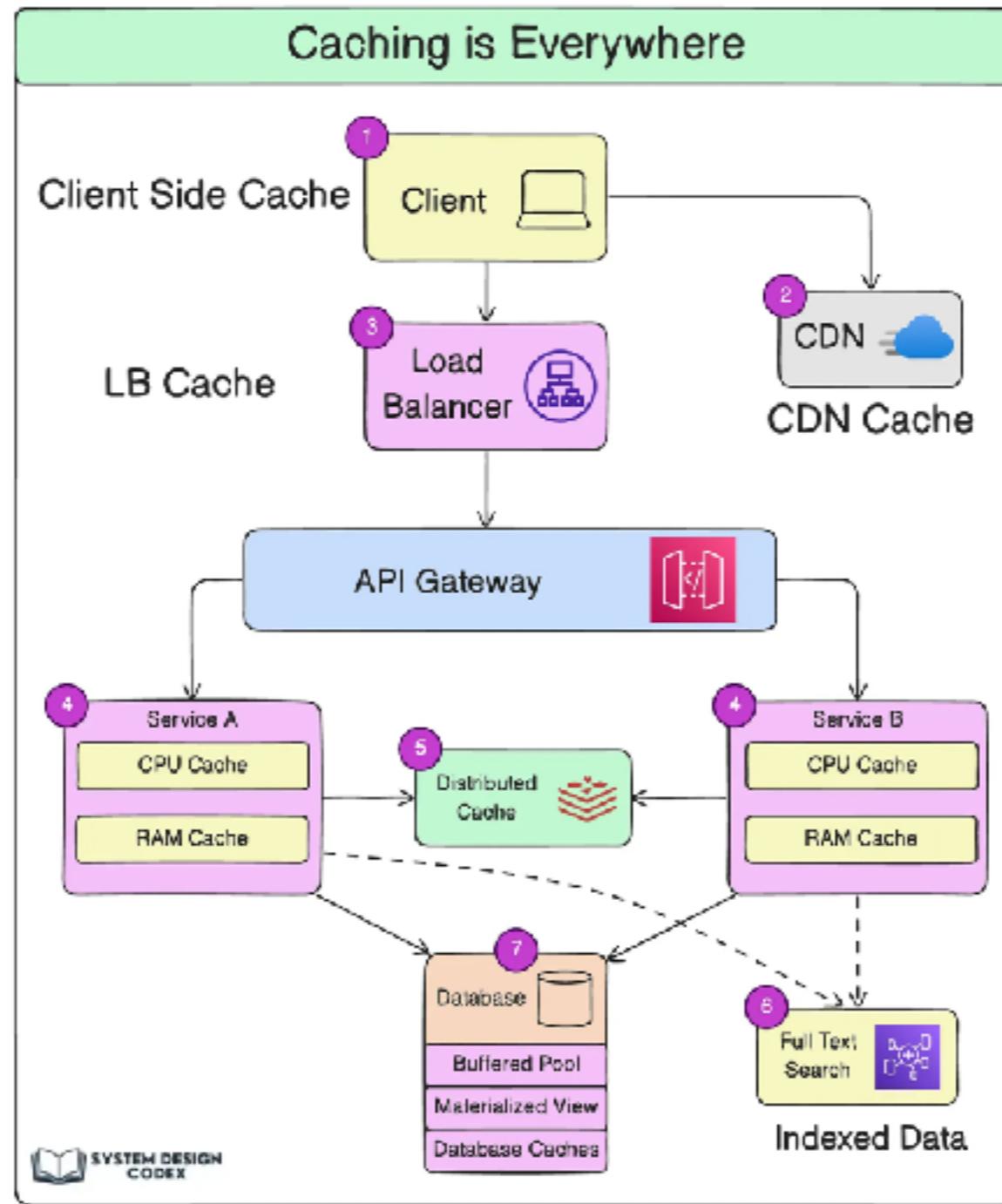
# Project Structure



# Data Caching



# All about caching



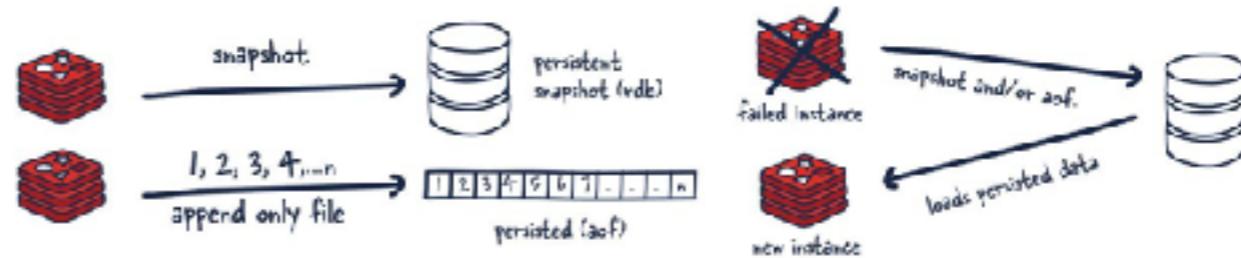
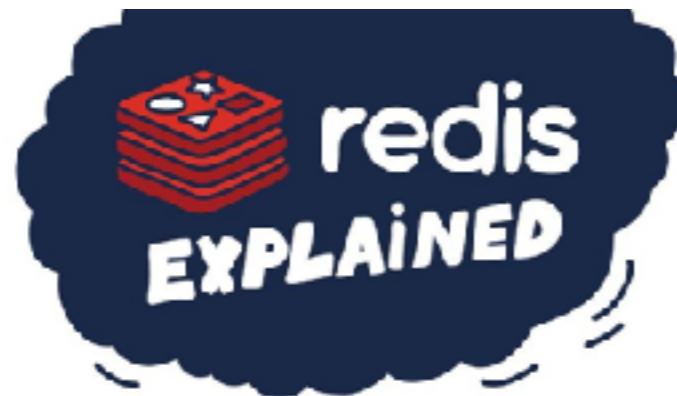
<https://newsletter.systemdesigncodex.com/p/caching-at-multiple-levels>



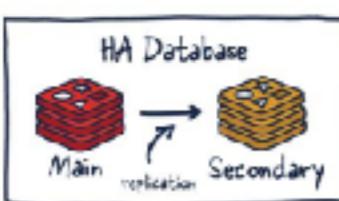
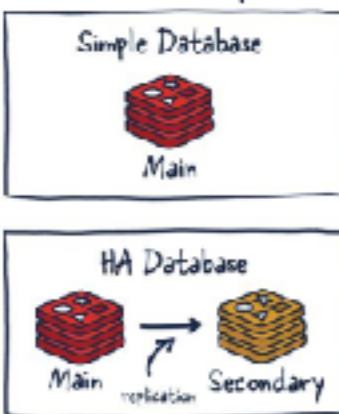
Go workshop

© 2022 - 2024 Siam Chamnkit Company Limited. All rights reserved.

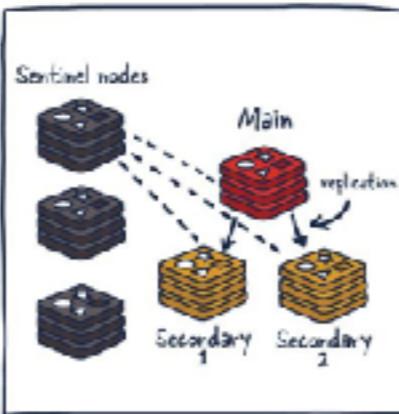
# Redis



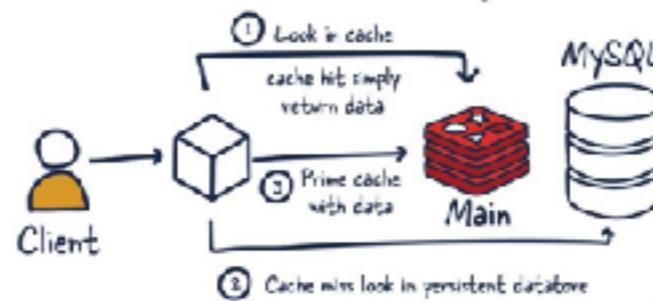
Redis setup



Redis sentinel

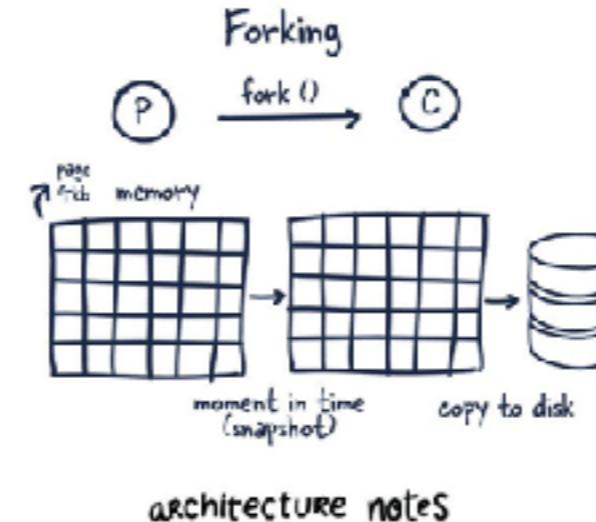
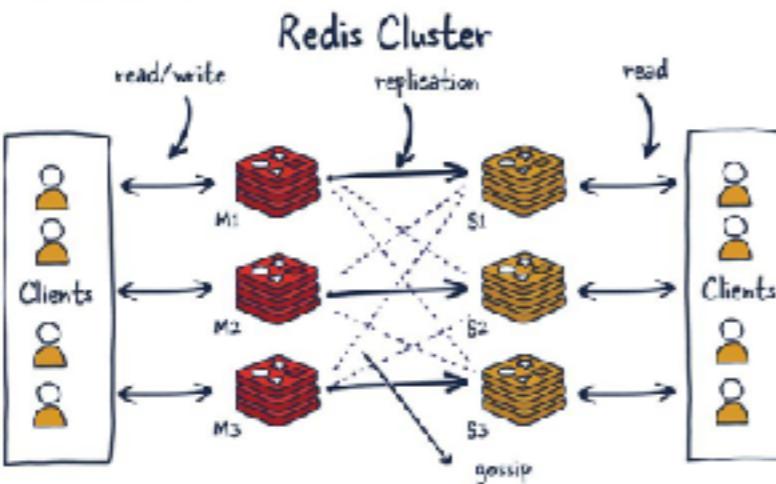


How is redis traditionally used



**KEY**  
foo  
**VALUE**  
bar

hello world	String
010101010101010101	Bitmap
{23334}{6634728}{916}	Bitfield
{a: "Hello", b: "World"}	Hash
[A>B>C>C]	List
{A<B<C}	Set
[A1, B2, C3]	Sorted set
{A: (50.1, 0.5)}	Geospatial
010101010101010101	Hyperlog
[id=blue, self = "foo", a: "bar"]	Stream



architecture notes

<https://architecturenotes.co/p/redis>



Go workshop

© 2022 - 2024 Siam Chamnkit Company Limited. All rights reserved.

# Memcached vs Redis

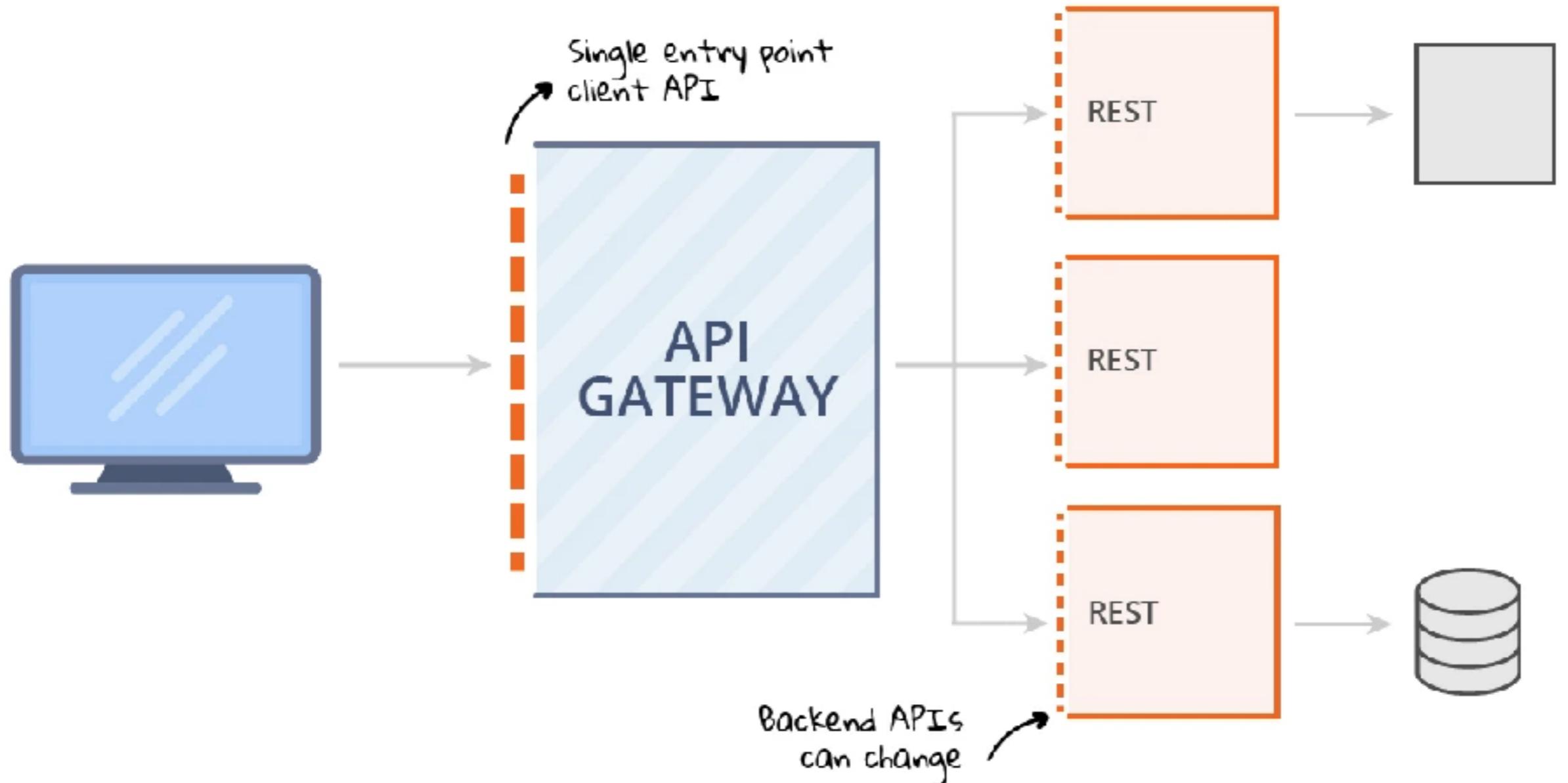
Features	Memcached	Redis
Store key-value In memory	Yes	Yes
Opensource	Yes	Yes
Support difference data types	No (String)	Yes (String, list, set)
Persistence	No	Yes
Replication	No	Yes
Cluster	No	Yes
Support multi-thread	Yes	Yes
Eviction policy	Yes (LRU only)	Yes



# API Gateway



# API Gateway

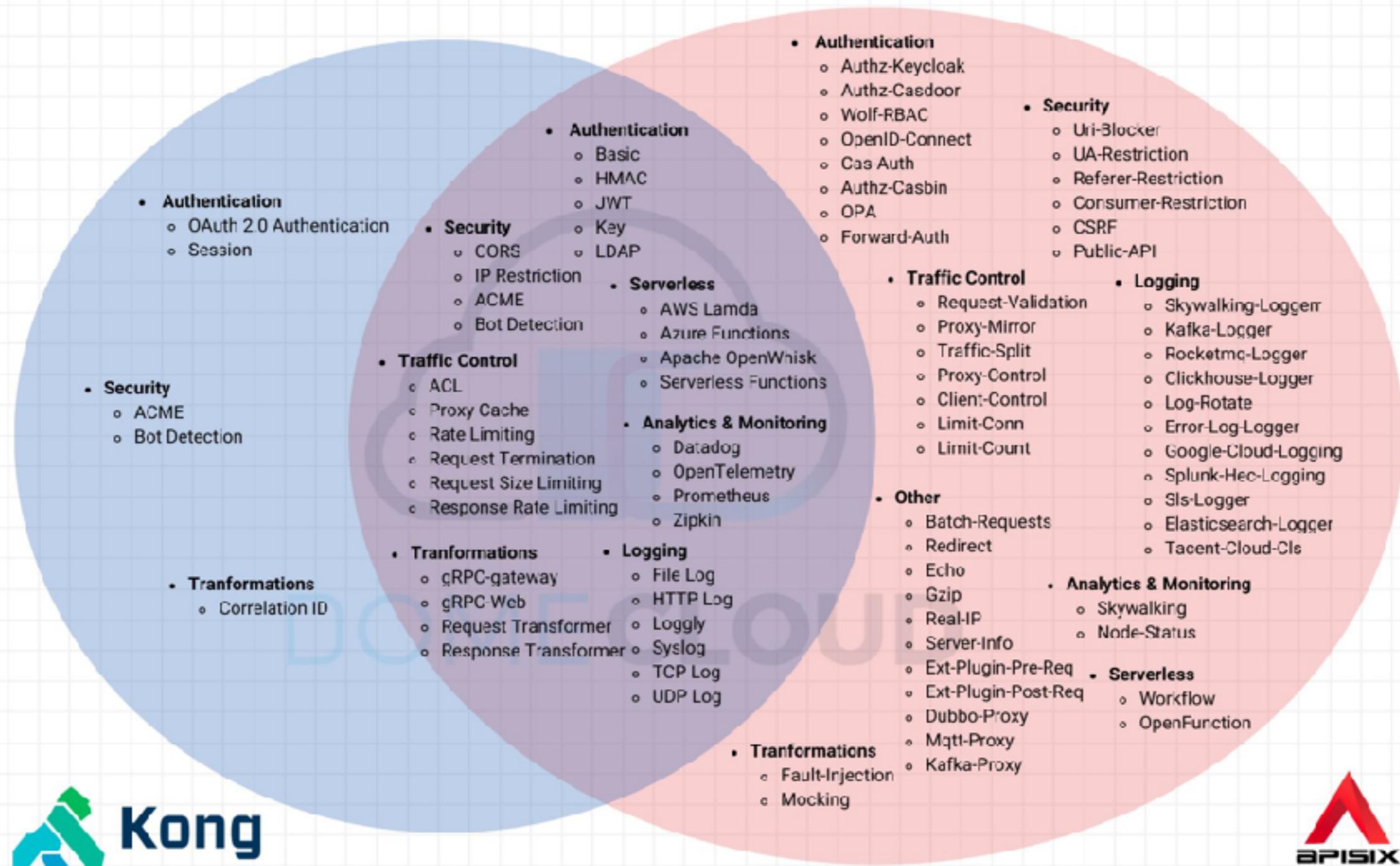


# API Gateway

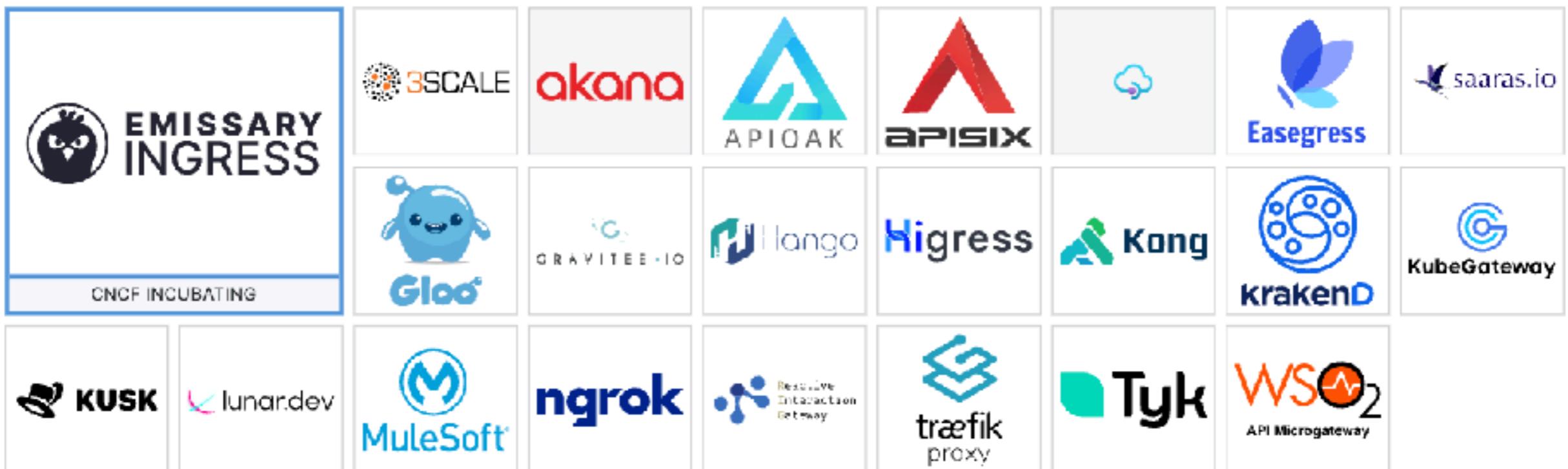
Authentication and Authorization  
Rate limit  
Circuit breaker  
Metric, Tracing, Logging  
Refactor assertion g



# Kong vs. APISIX Plugins comparison



# API Gateway



<https://landscape.cncf.io/guide#orchestration-management--api-gateway>



# Security APIs

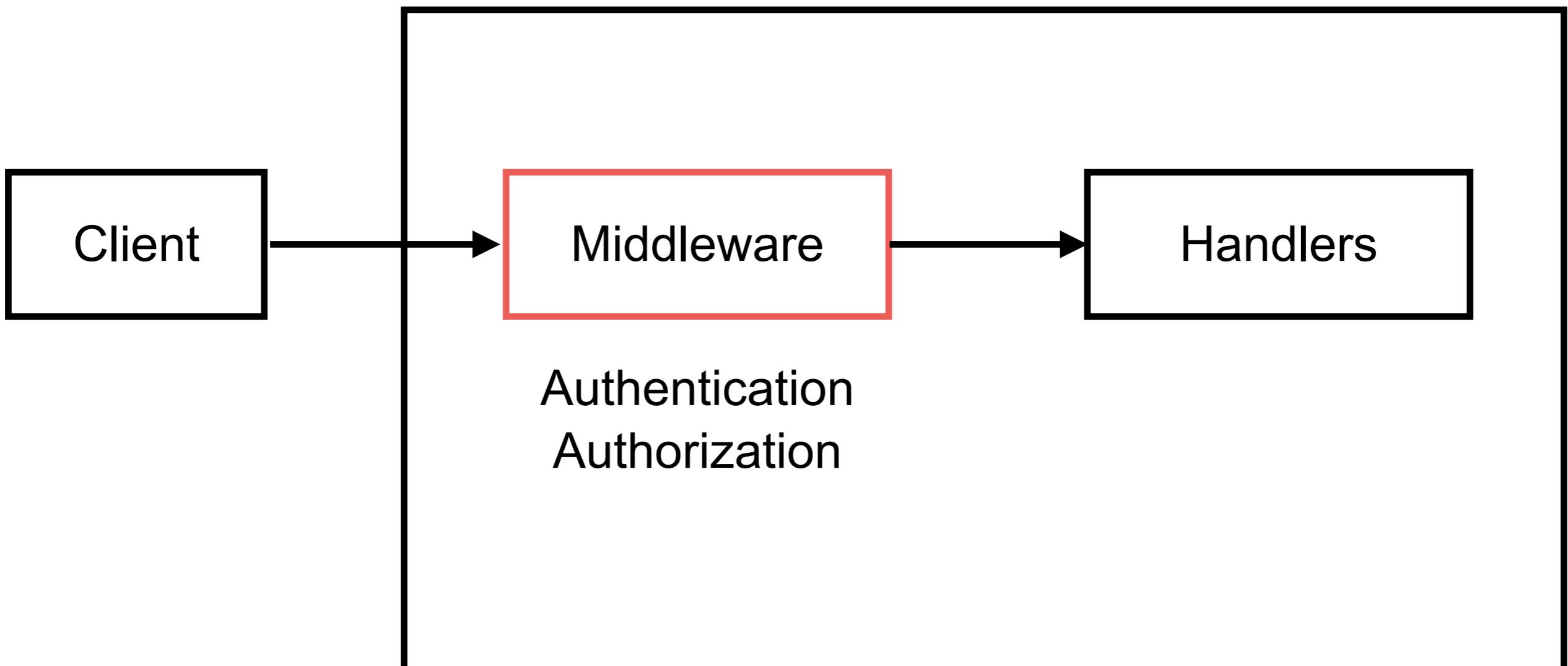
Authentication

Authorization

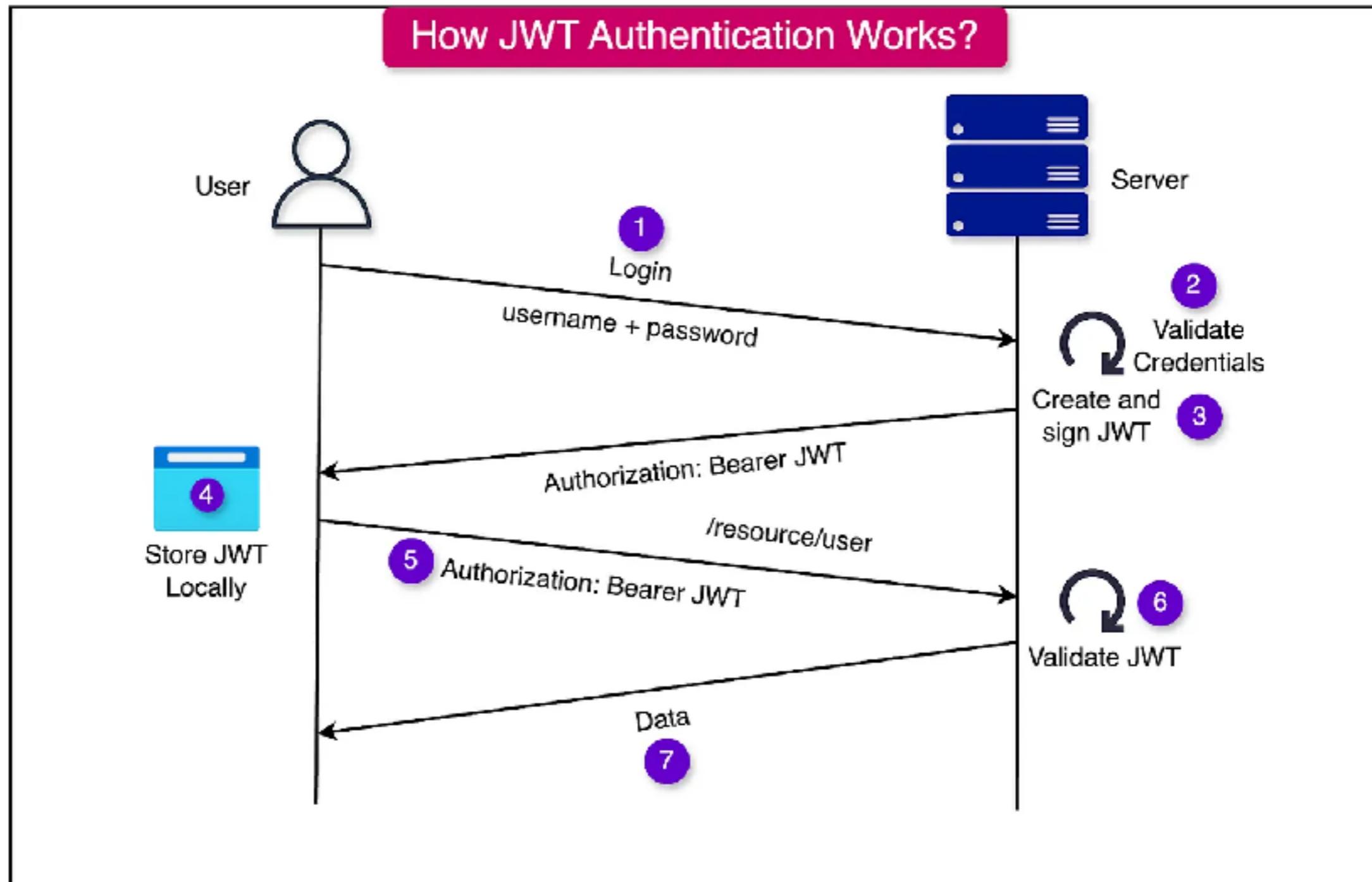


# Project Structure

## RESTFul API with Go



# Working with JWT



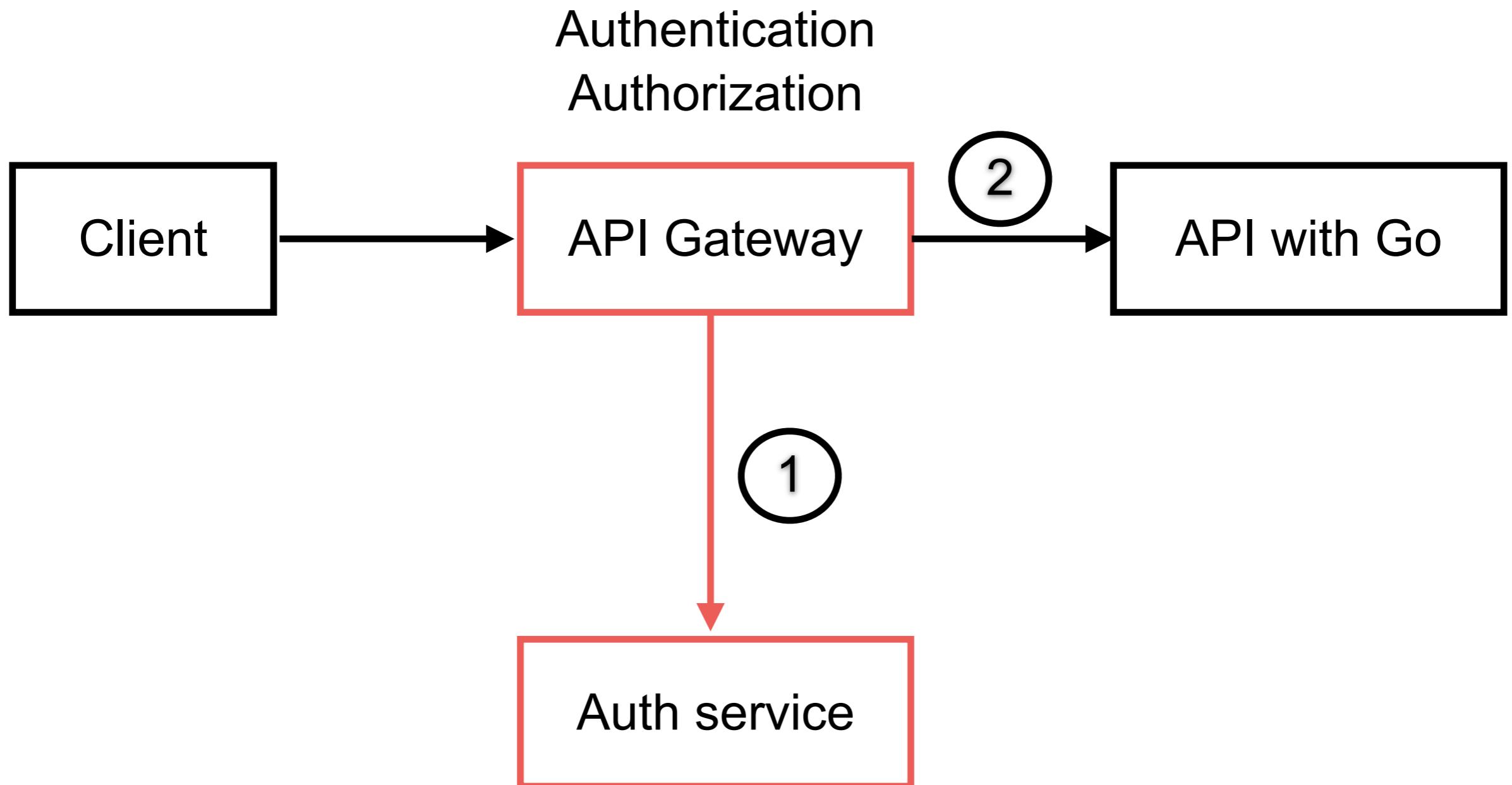
<https://blog.bytebytogo.com/p/mastering-modern-authentication-cookies>



Go workshop

© 2022 - 2024 Siam Chamnkit Company Limited. All rights reserved.

# Project Structure



# Observable Service



# Observable Service

Application  
metric

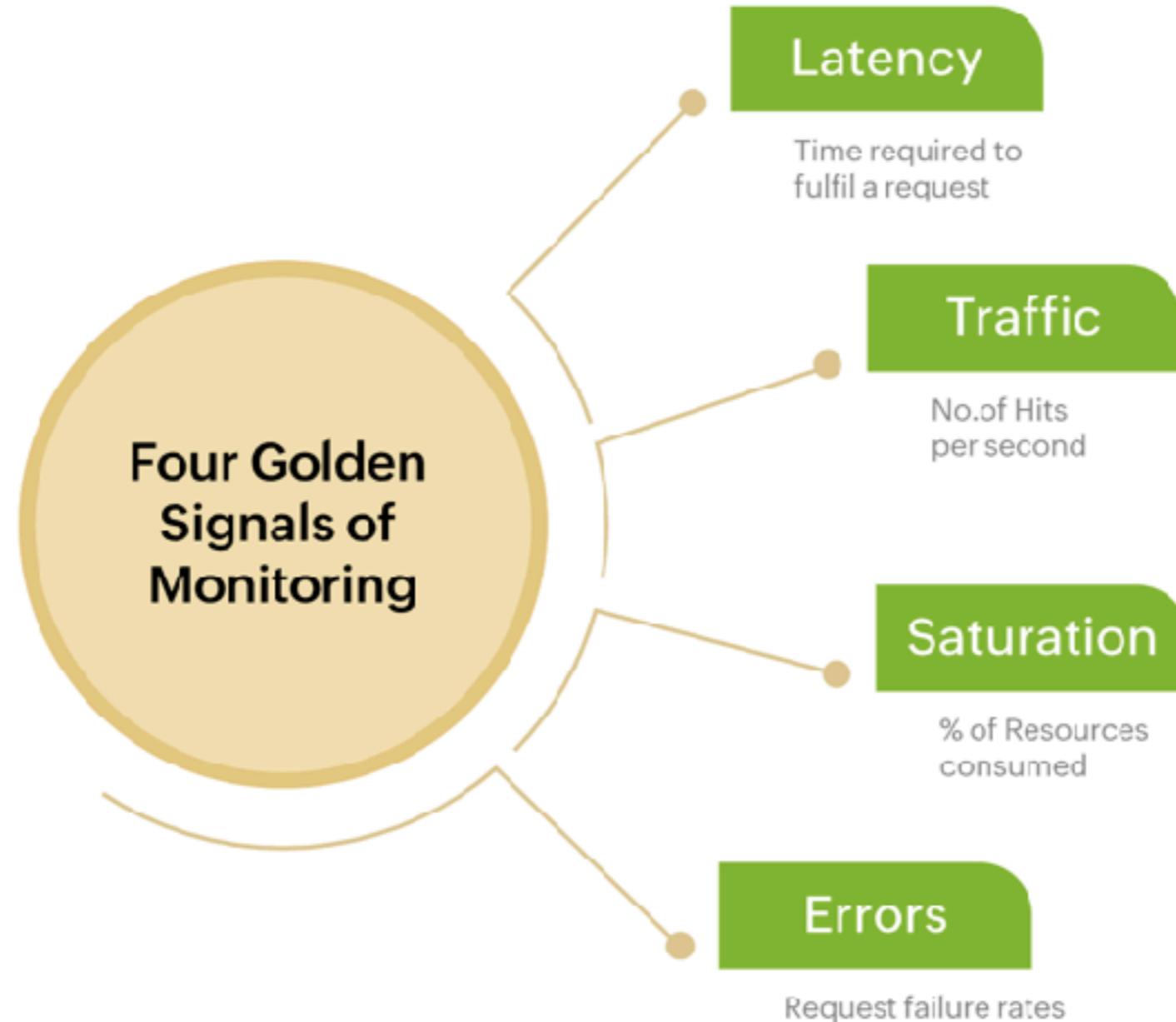
Distributed  
tracing

Log aggregation

Exception  
tracking



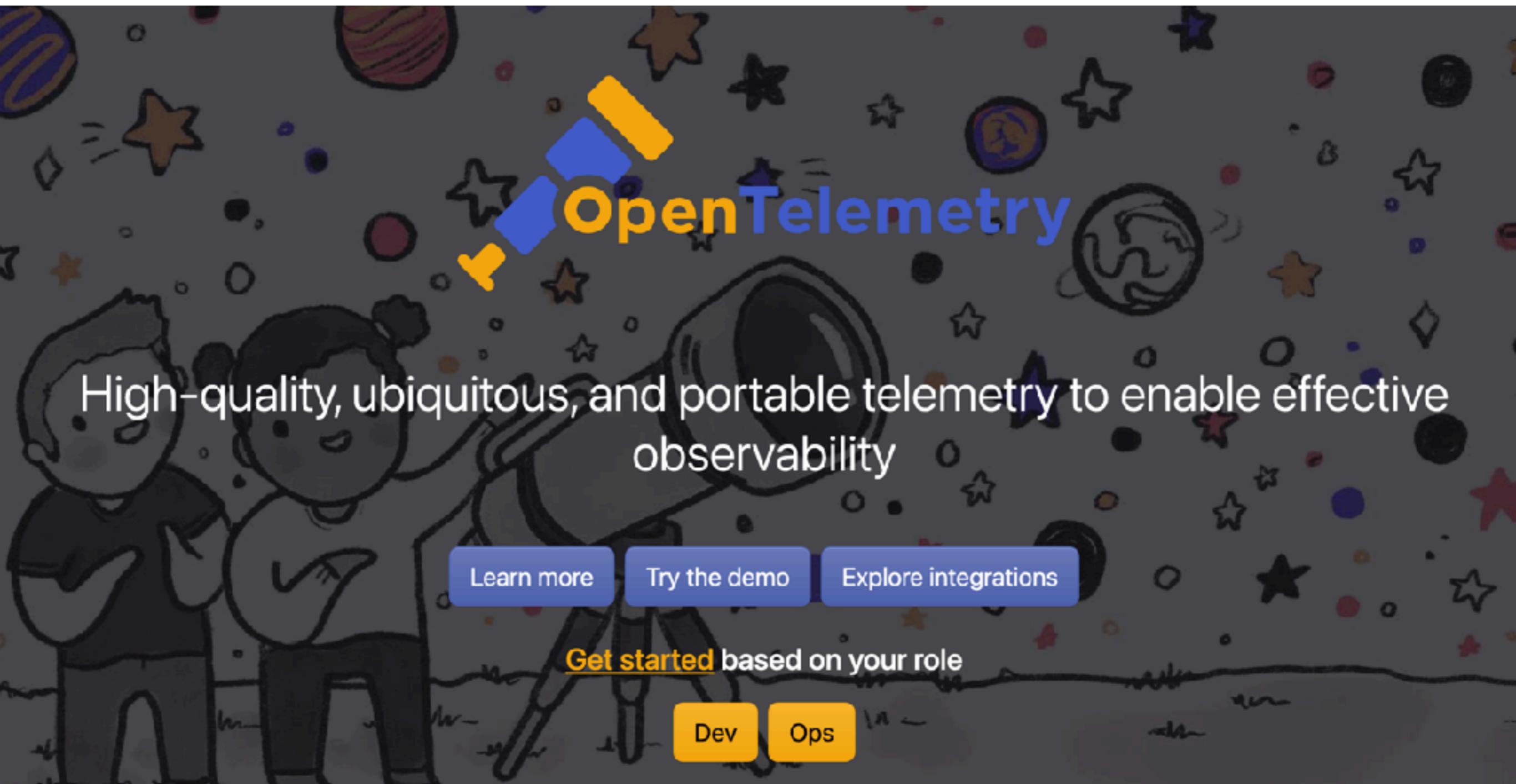
# The 4 Golden Signals



<https://sre.google/sre-book/monitoring-distributed-systems/>



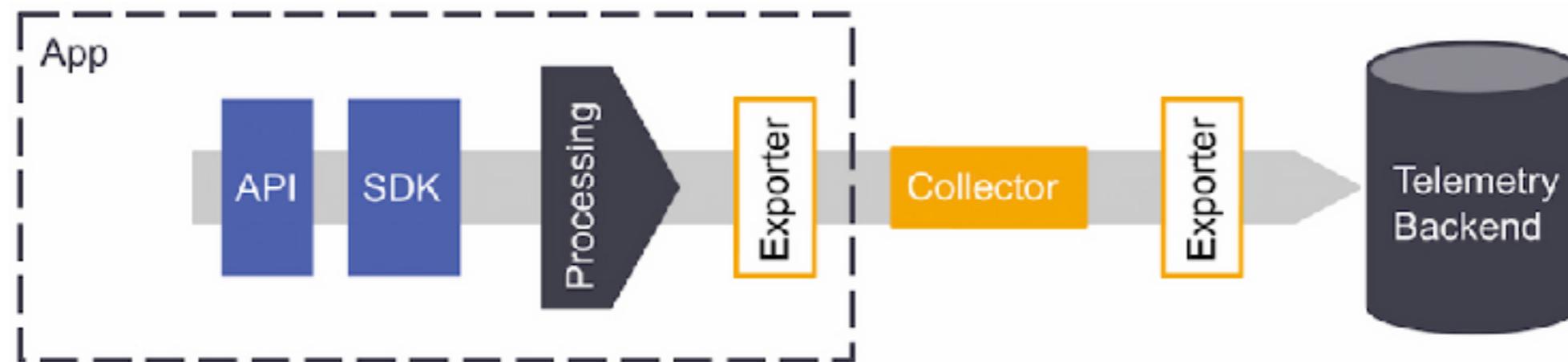
# OpenTelemetry



<https://opentelemetry.io/>



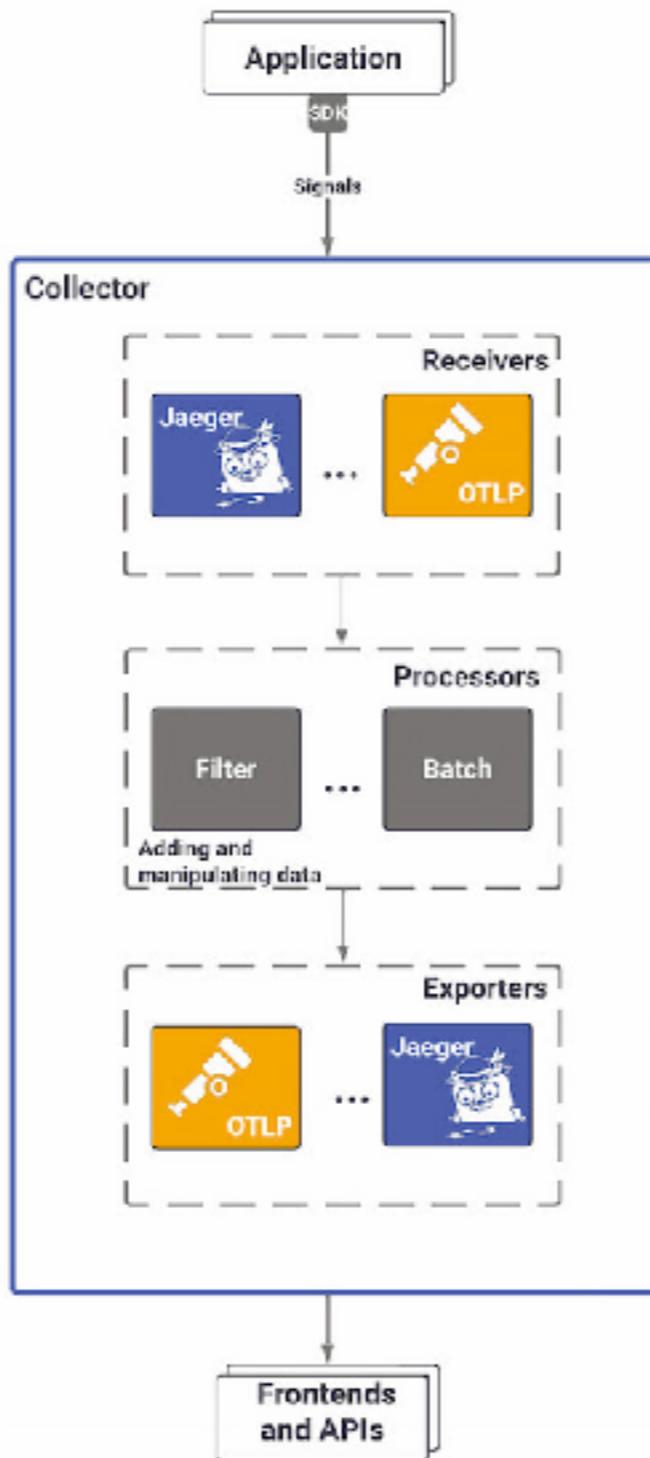
# OpenTelemetry Structure



<https://dzone.com/refcardz/getting-started-with-opentelemetry>



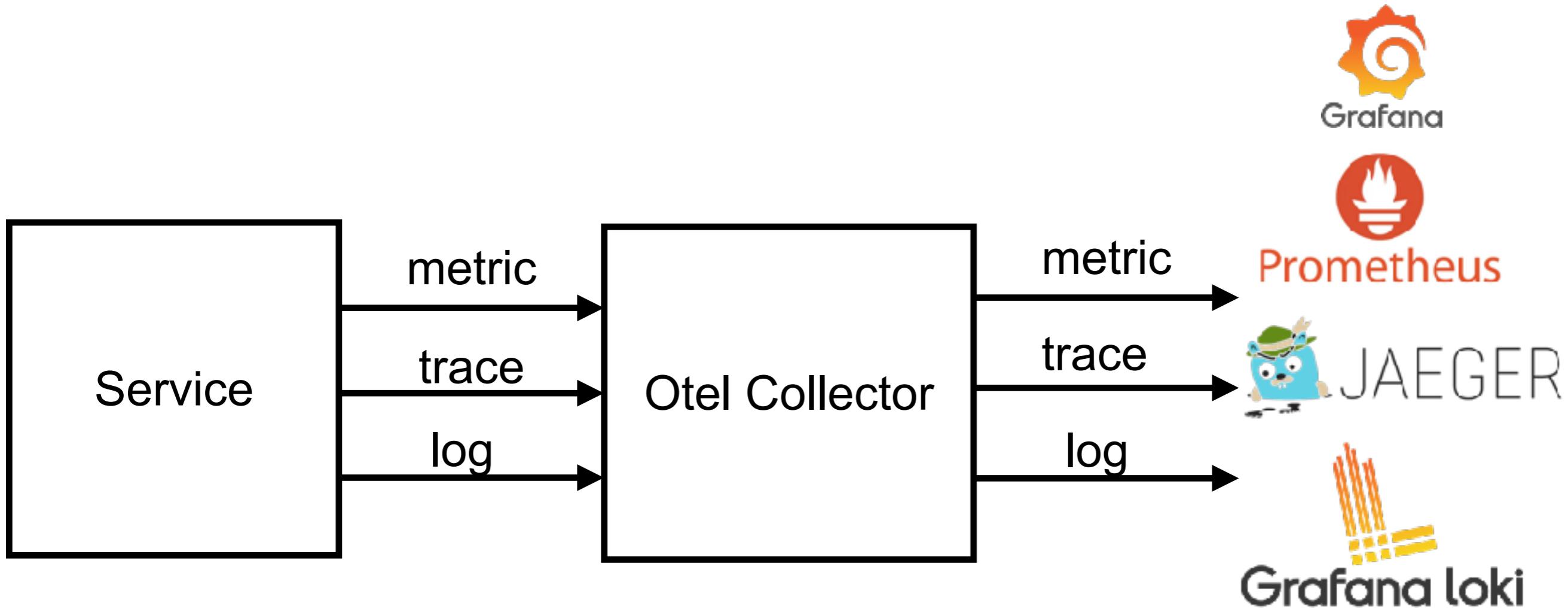
# OpenTelemetry Structure



<https://dzone.com/refcardz/getting-started-with-opentelemetry>



# Working with OpenTelemetry



# Q/A

