Spring Cloud Contract在微服务中的应用

彭金虎

ABOUT ME

ThoughtWorks developer

专注于后端开发和DevOps

Github: https://github.com/pjhu

CONTENTS

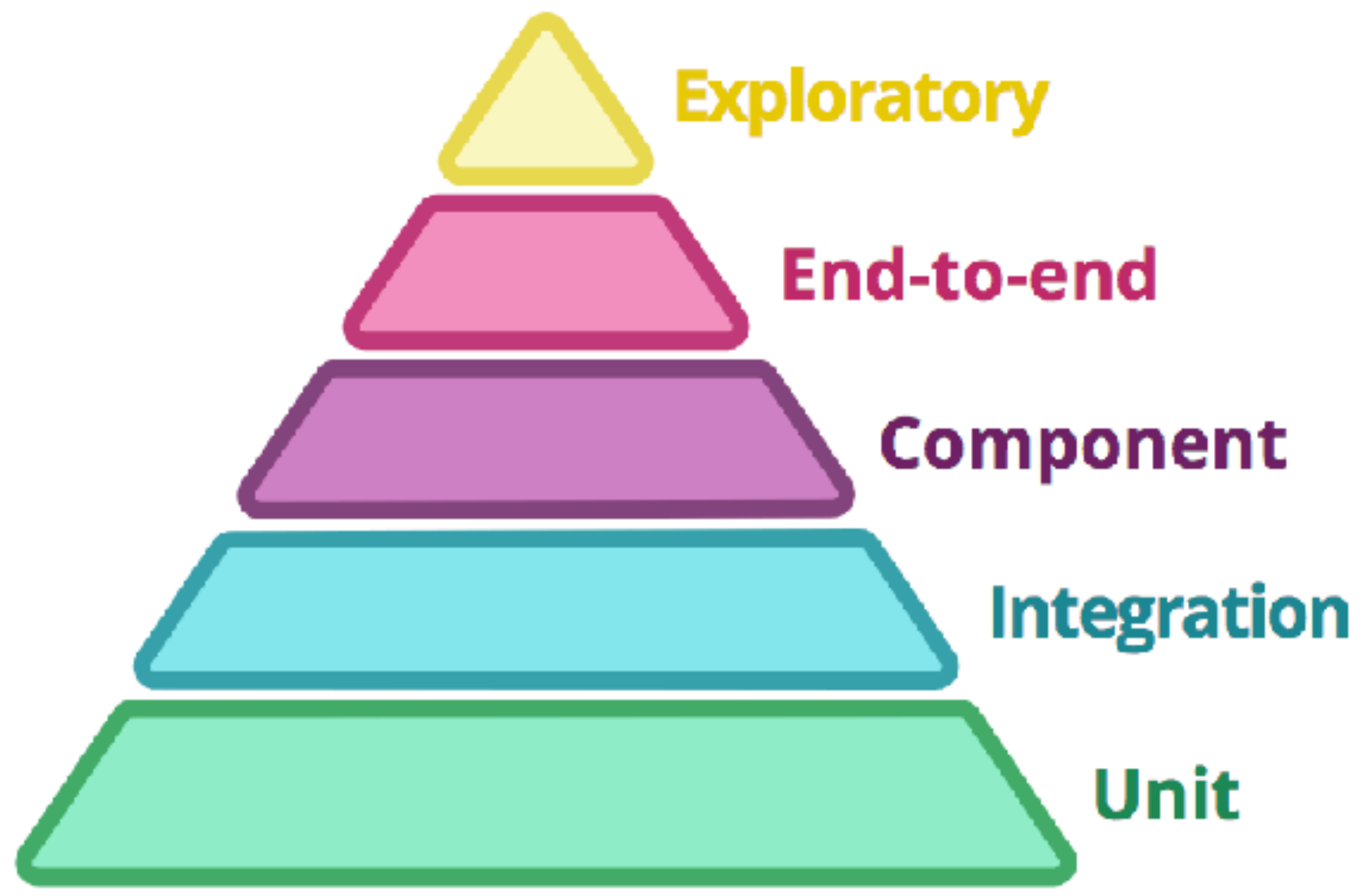
微服务

微服务

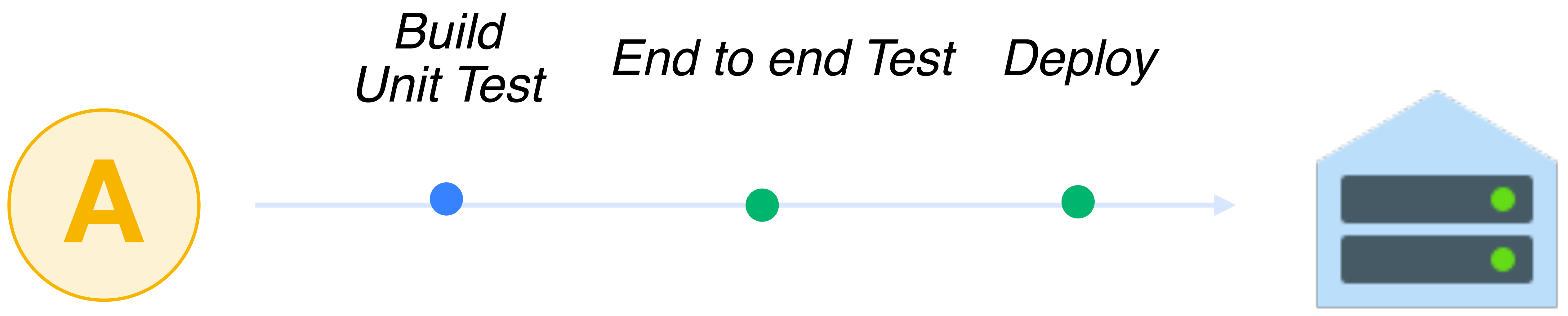
1. 一种架构风格，专注于将复杂的、多功能的系统拆解为单一的、简单的多服务系统
2. 鼓励每个服务独立演进、应对变化
3. 服务之间通过HTTP API方式通信
4. **服务独立部署**

https://martinfowler.com/articles/microservices.html

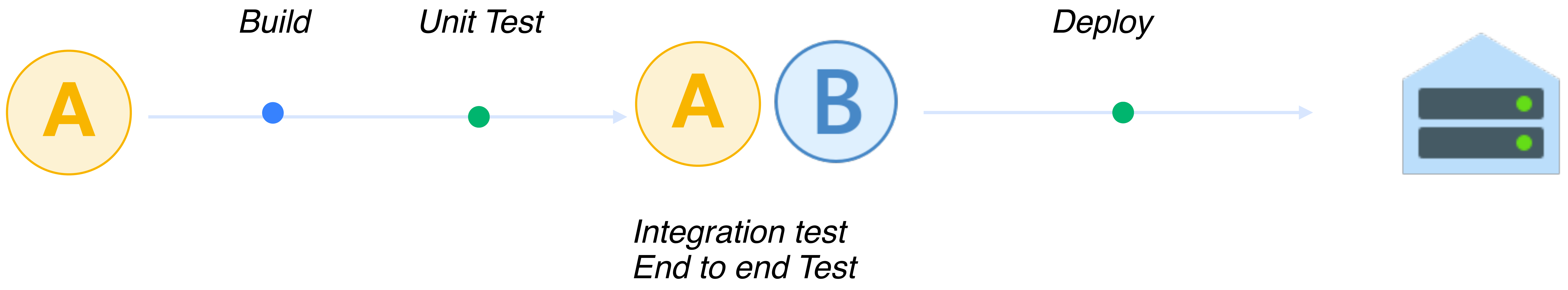
单体应用PIPELINE



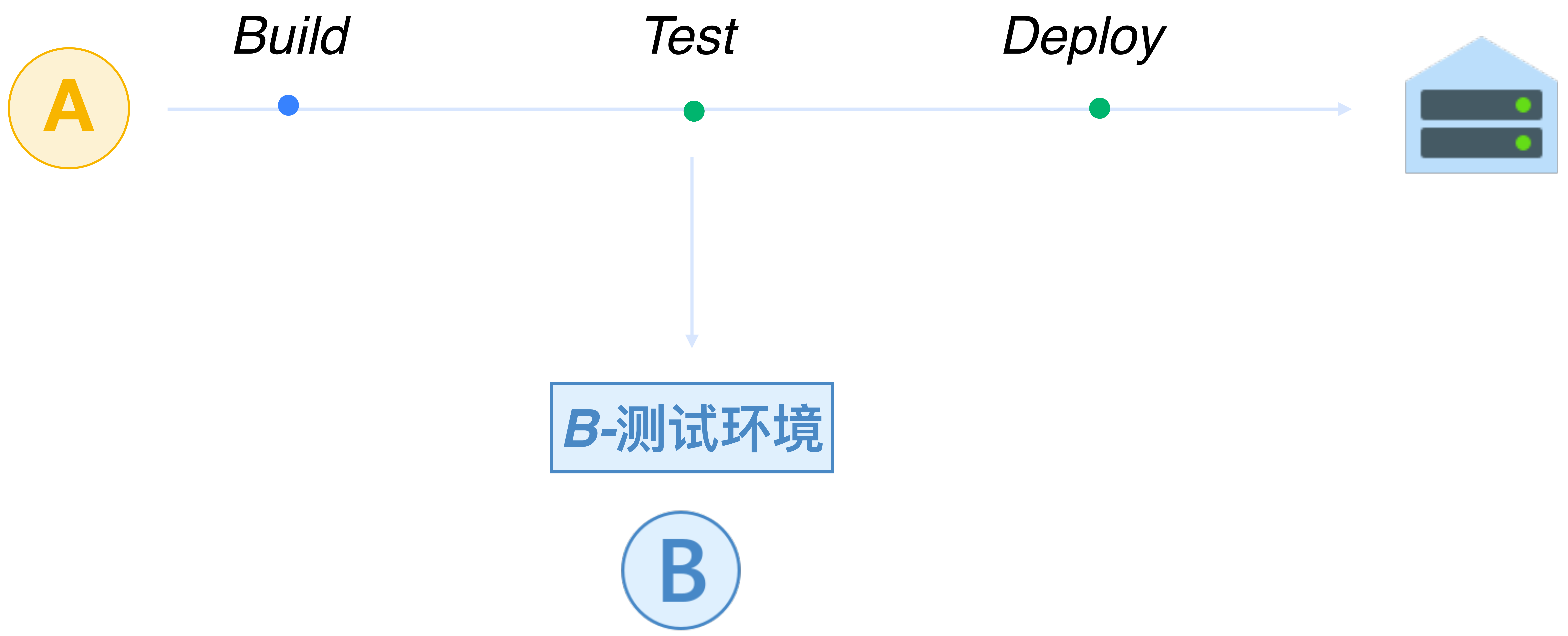
https://martinfowler.com/articles/microservice-testing/#conclusion-test-pyramid



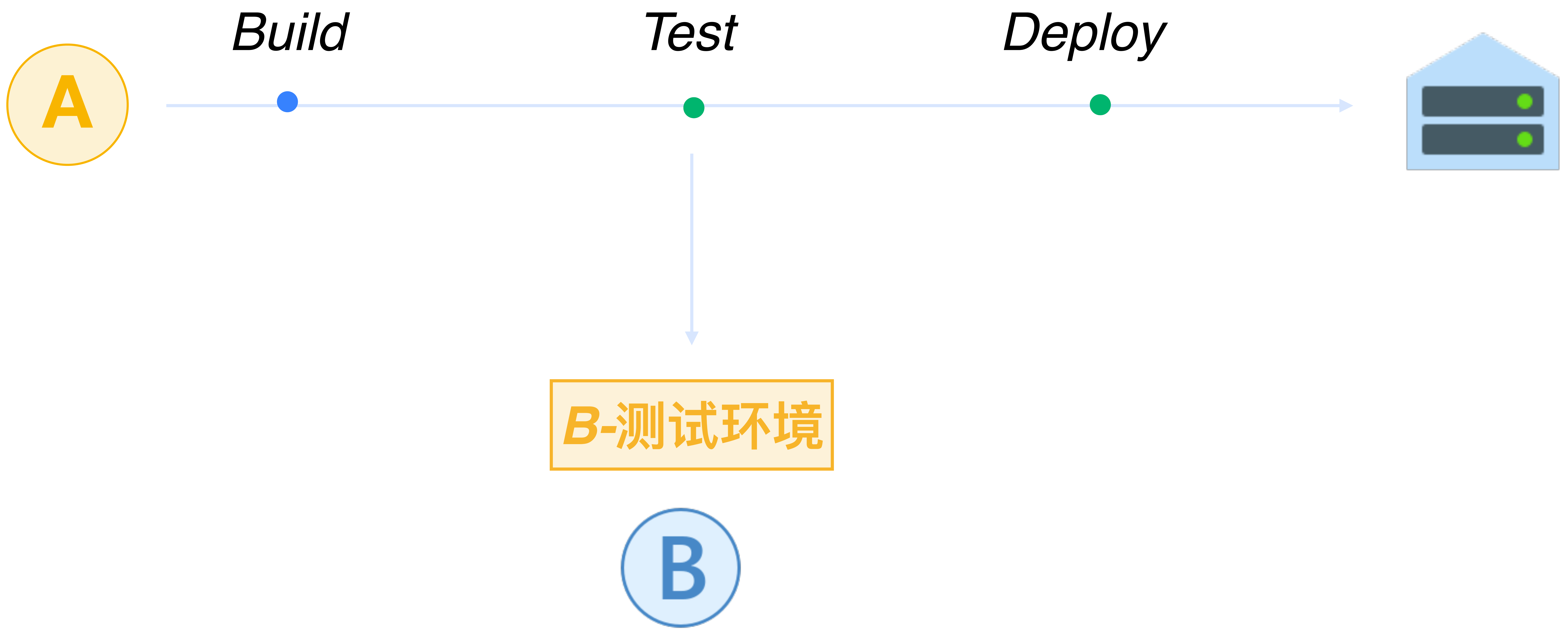
单体应用PIPELINE



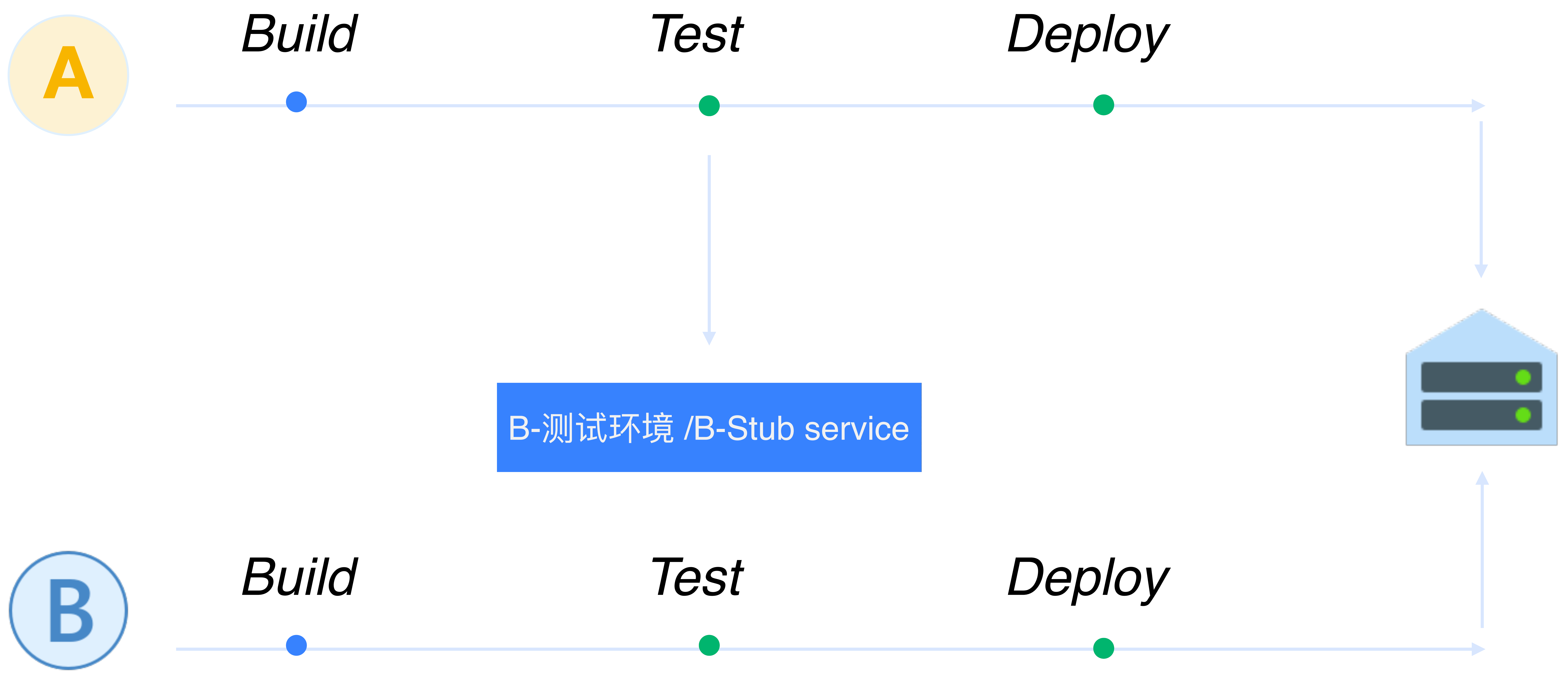
微服务PIPELINE



微服务PIPELINE



微服务PIPELINE

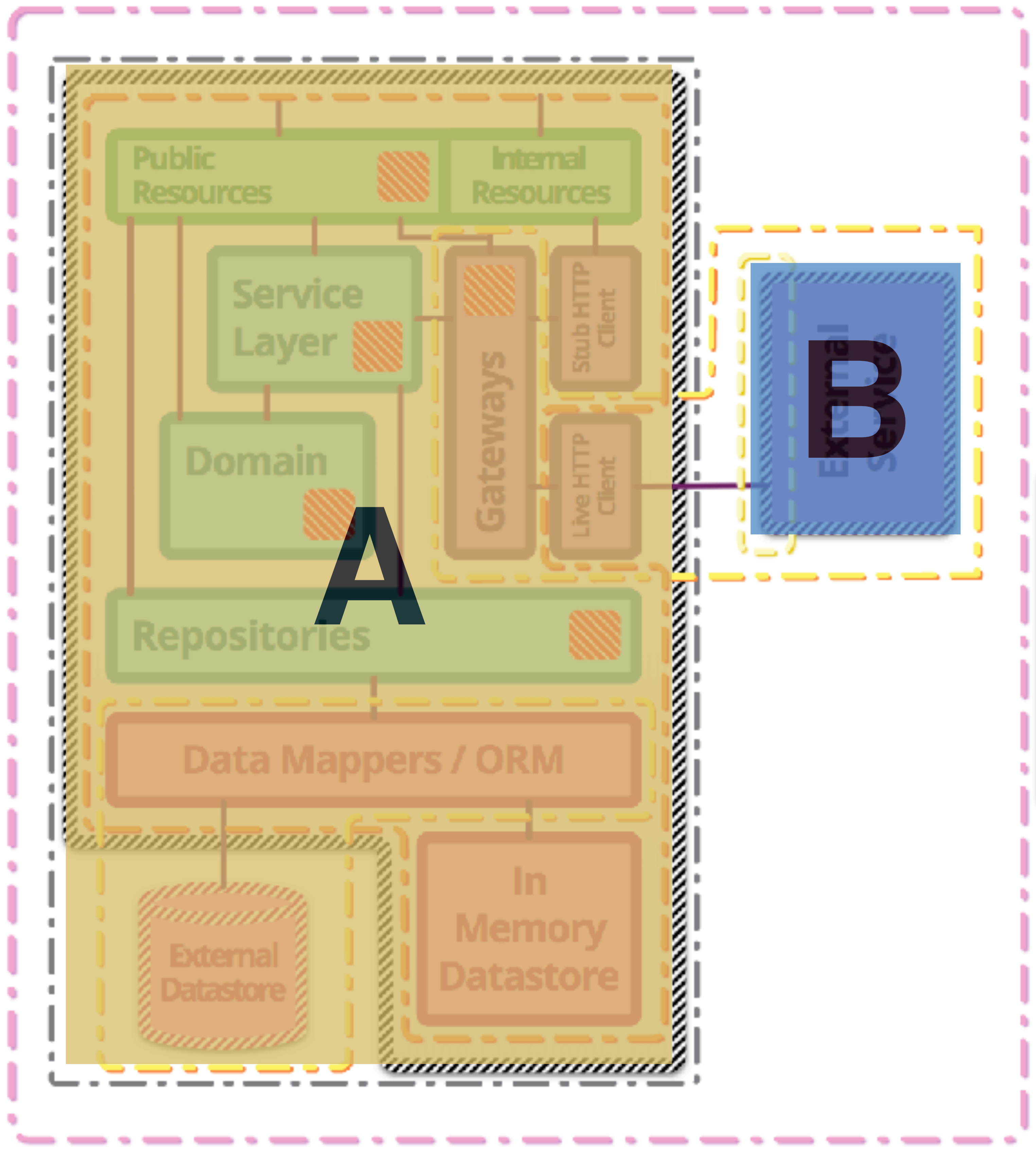


契约测试

测试策略

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

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



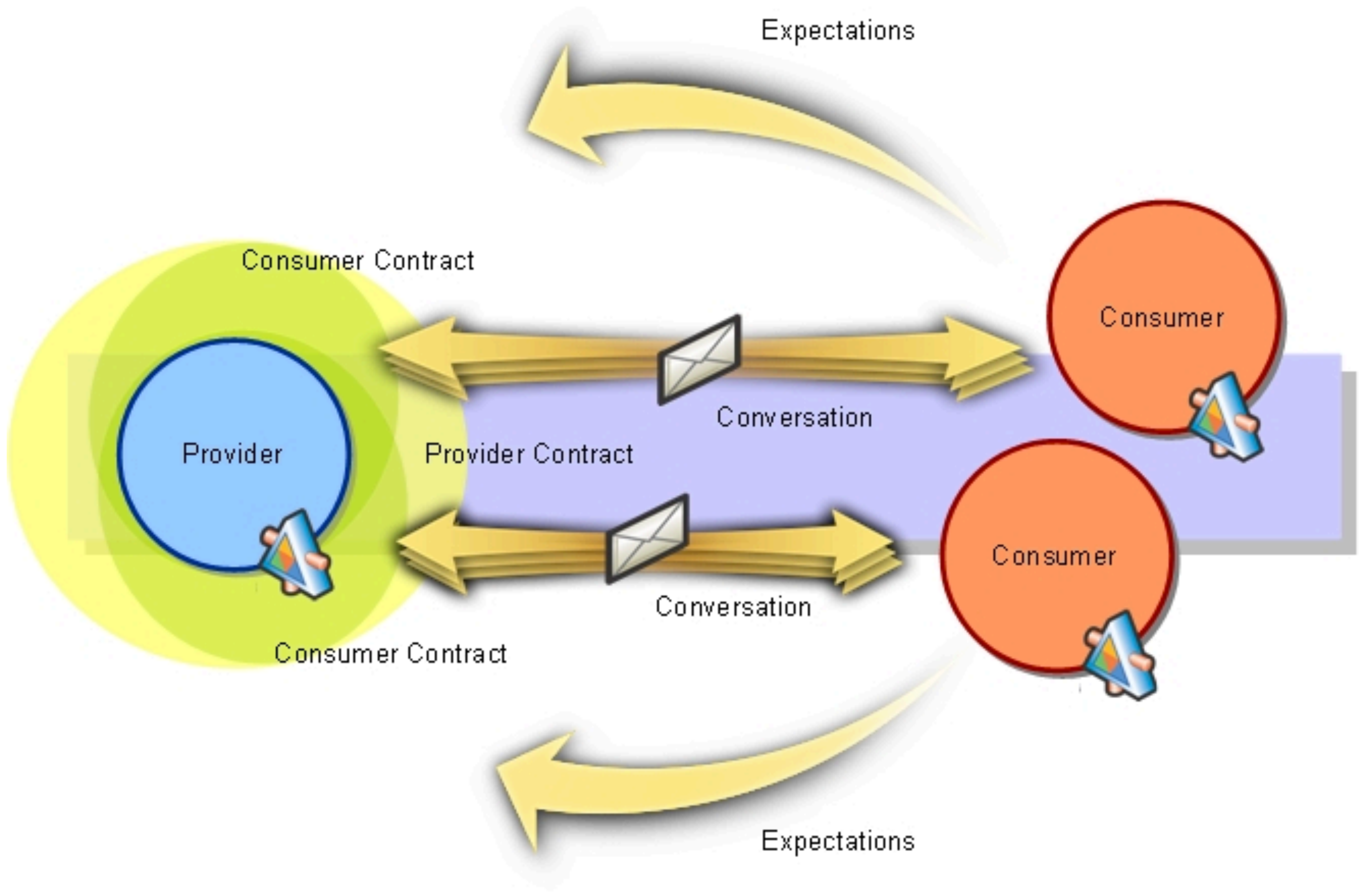
▨ **Component tests** : limit the scope of the exercised software to a portion of the system under test, manipulating the system through internal code interfaces and using test doubles to isolate the code under test from other components.

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

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

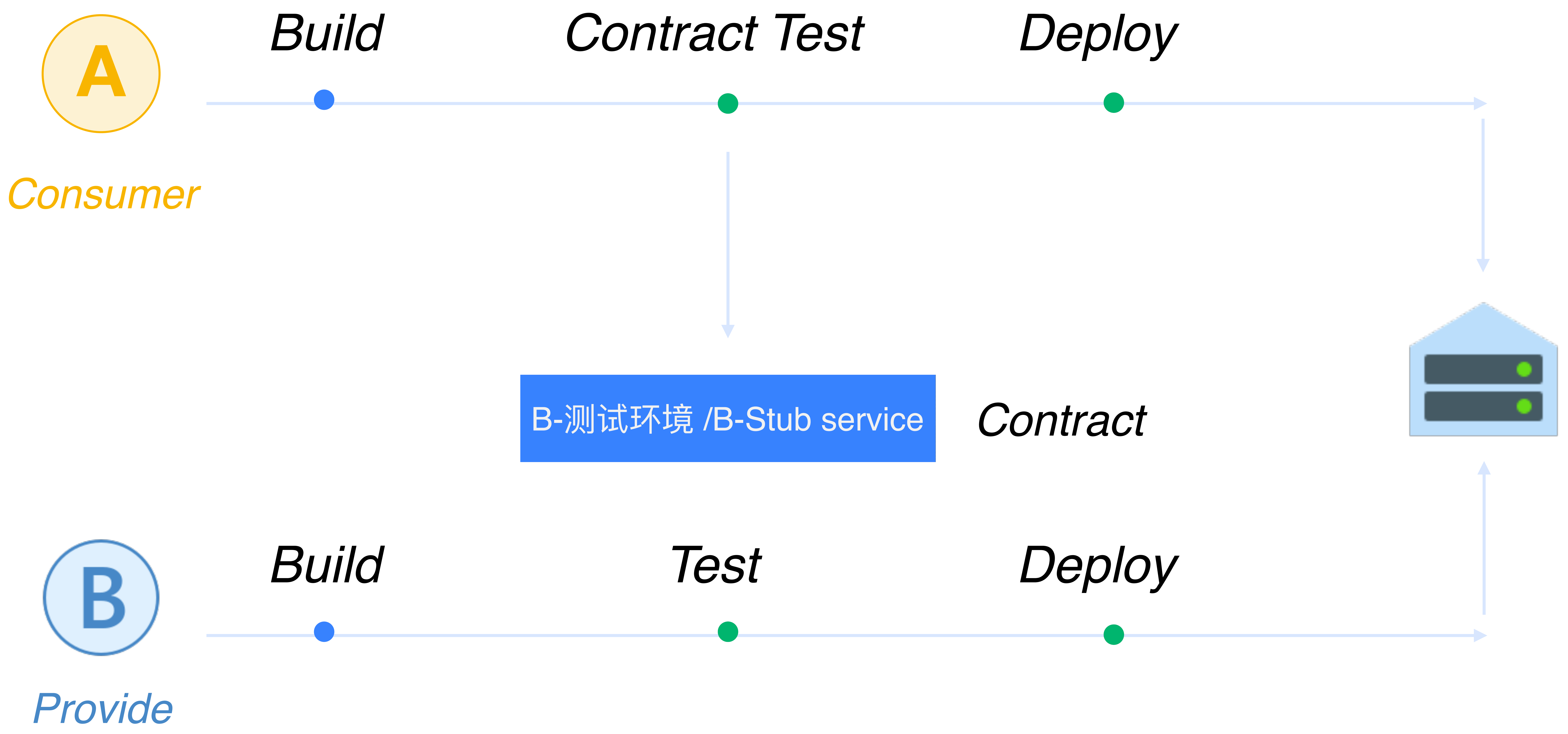
https://martinfowler.com/articles/microservice-testing/#conclusion-summary

什么是契约测试



https://martinfowler.com/articles/consumerDrivenContracts.html

微服务PIPELINE



契约测试工具

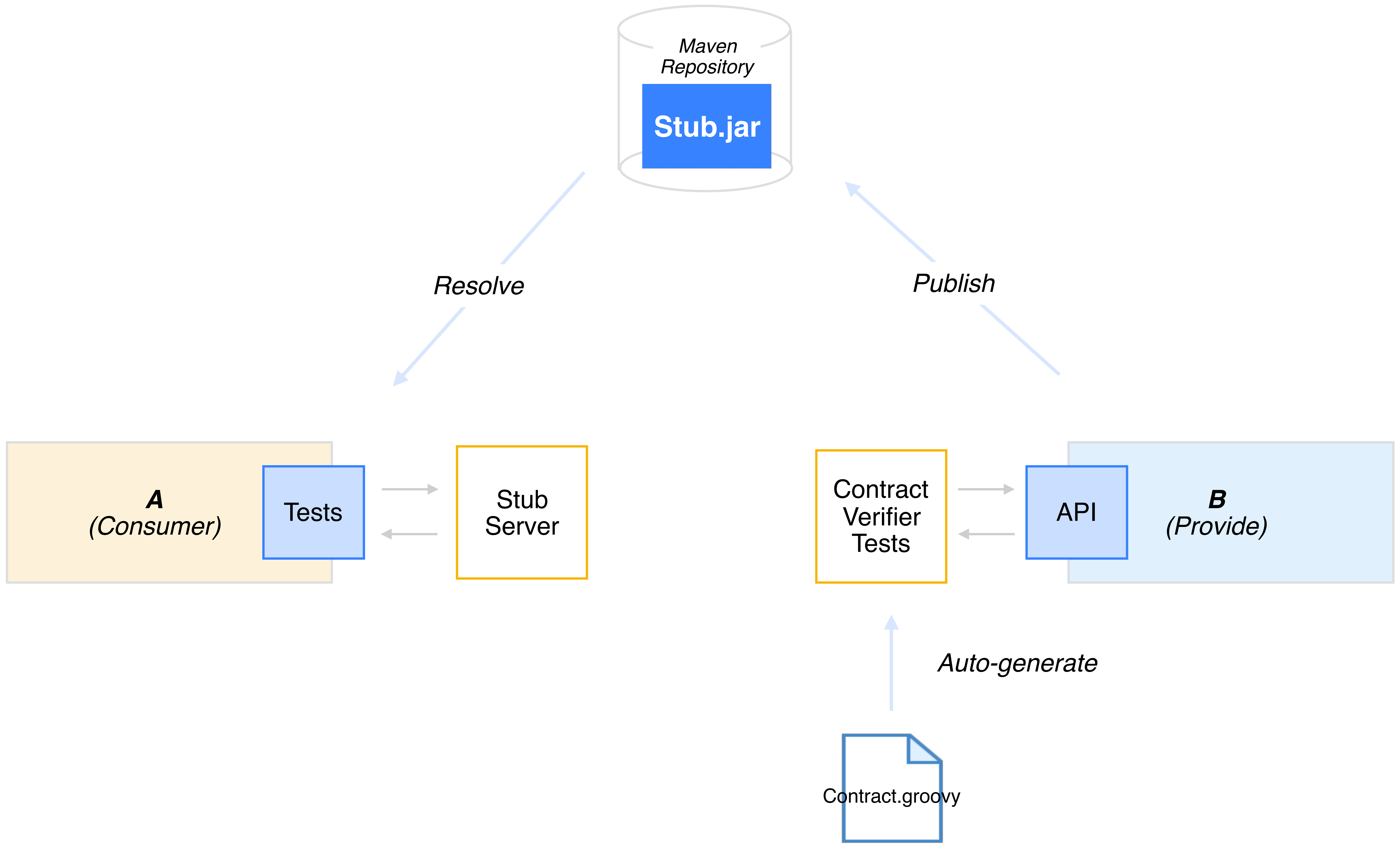
01 Spring Cloud Contract

02 Pact

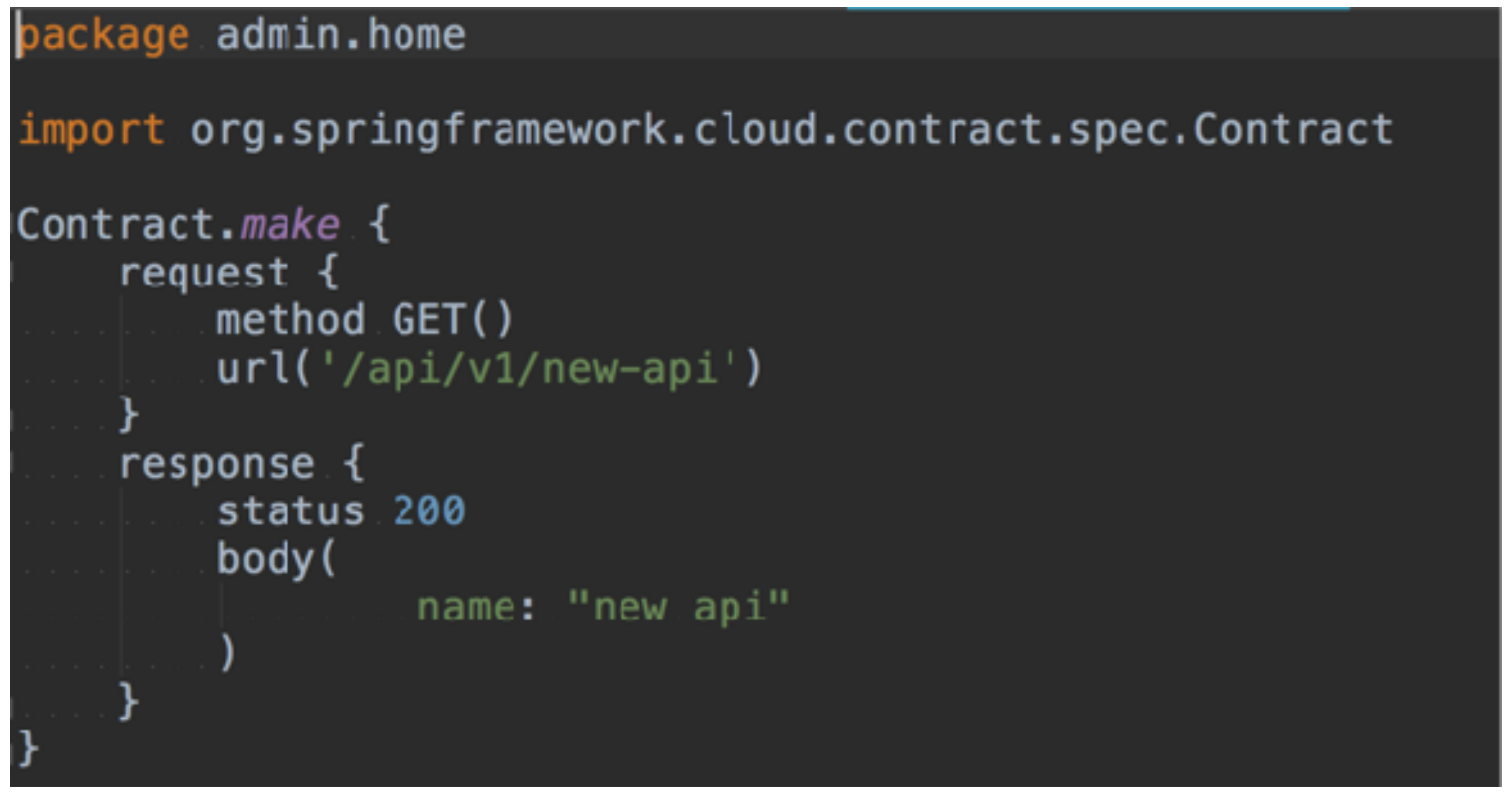
03 Mountebank

Spring Cloud Contract

SPRING CLOUD CONTRACT工作模式

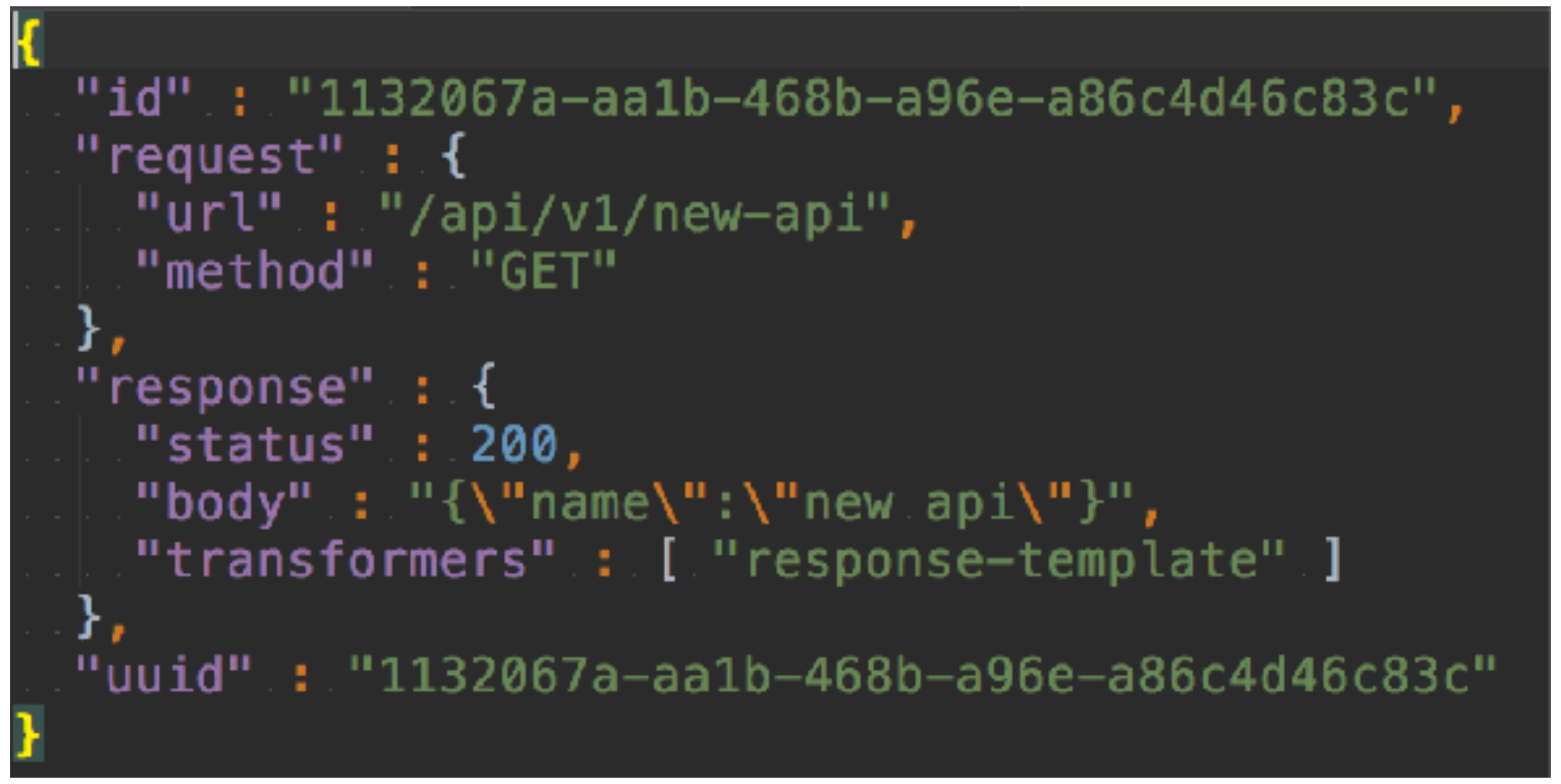


CONTRACT STUB



```
package admin.home

import org.springframework.cloud.contract.spec.Contract

Contract.make {
    request {
        method GET()
        url('/api/v1/new-api')
    }
    response {
        status 200
        body(
                name: "new api"
        )
    }
}
```

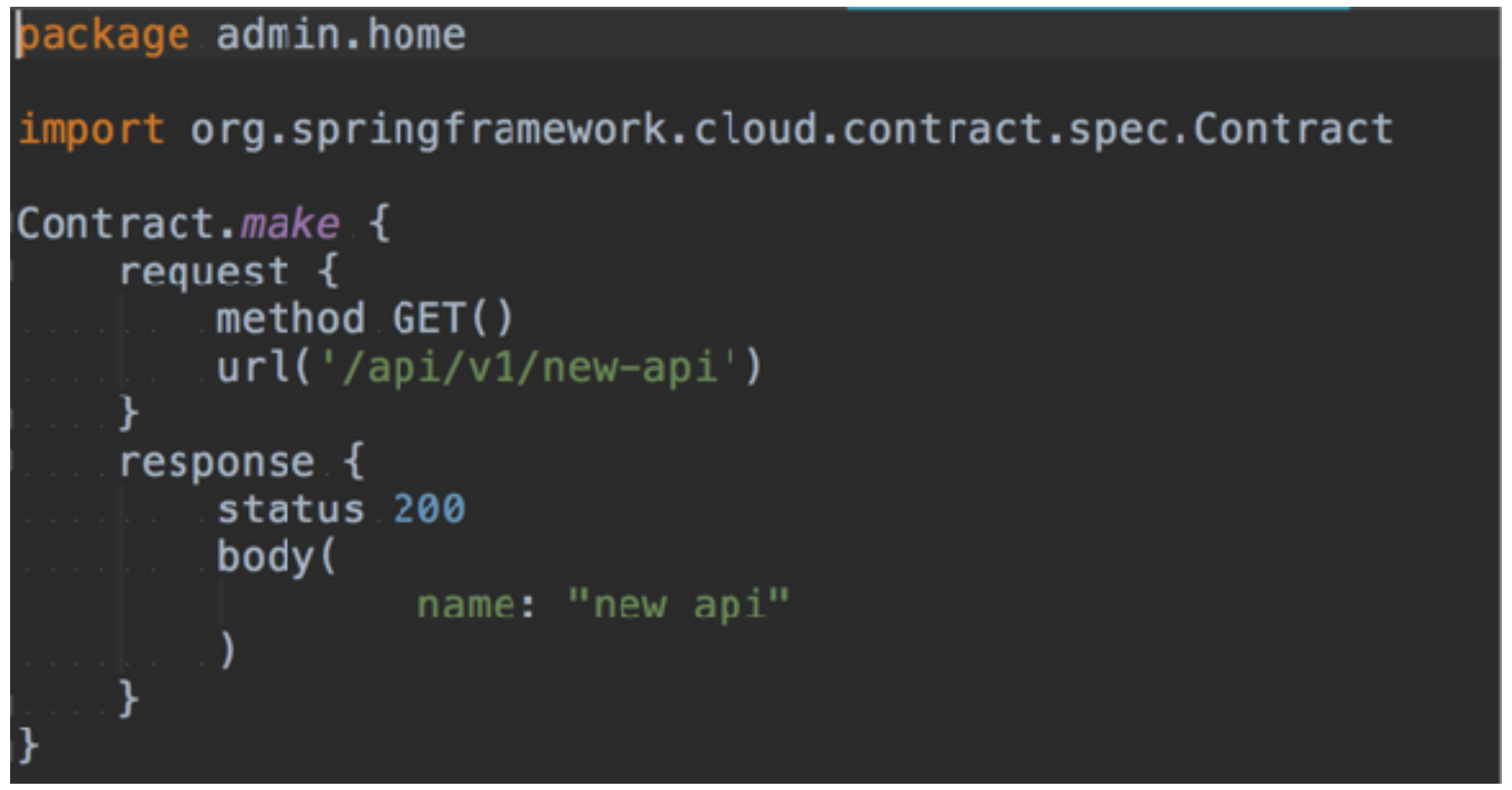
Contract



```
{
  "id" : "1132067a-aa1b-468b-a96e-a86c4d46c83c",
  "request" : {
    "url" : "/api/v1/new-api",
    "method" : "GET"
  },
  "response" : {
    "status" : 200,
    "body" : "{\"name\":\"new api\"}",
    "transformers" : [ "response-template" ]
  },
  "uuid" : "1132067a-aa1b-468b-a96e-a86c4d46c83c"
}
```

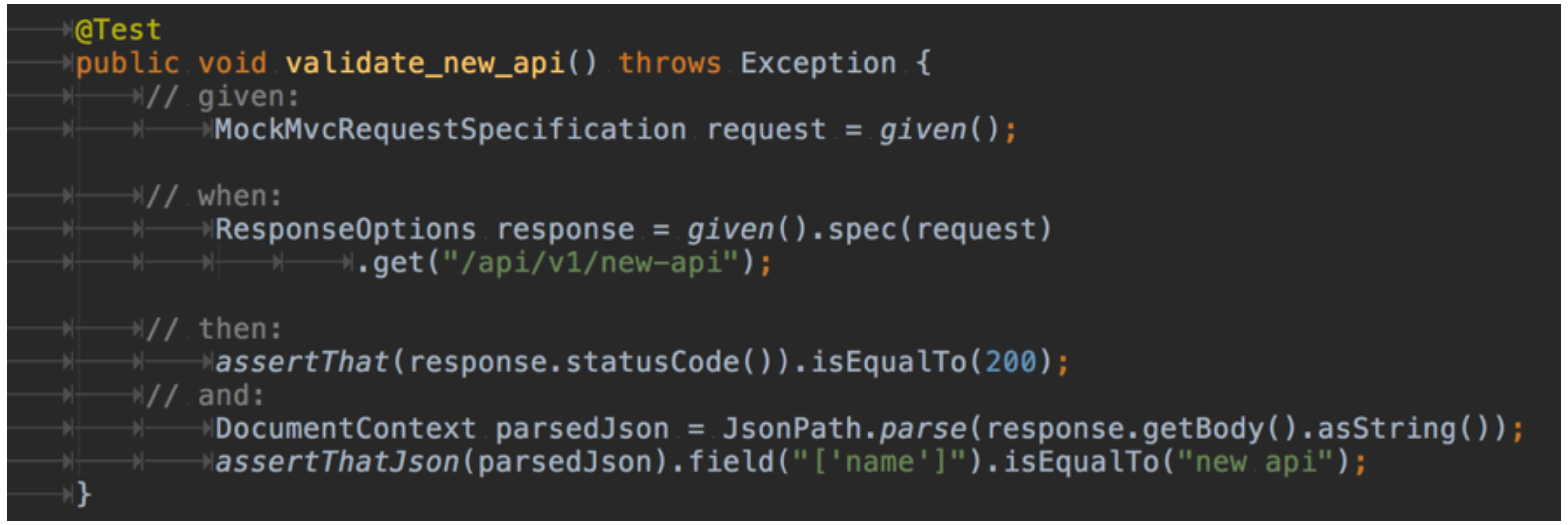
Stub Mappings

CONTRACT VERIFIER TESTS



```
package admin.home

import org.springframework.cloud.contract.spec.Contract

Contract.make {
    request {
        method GET()
        url('/api/v1/new-api')
    }
    response {
        status 200
        body(
                name: "new api"
        )
    }
}
```

Contract

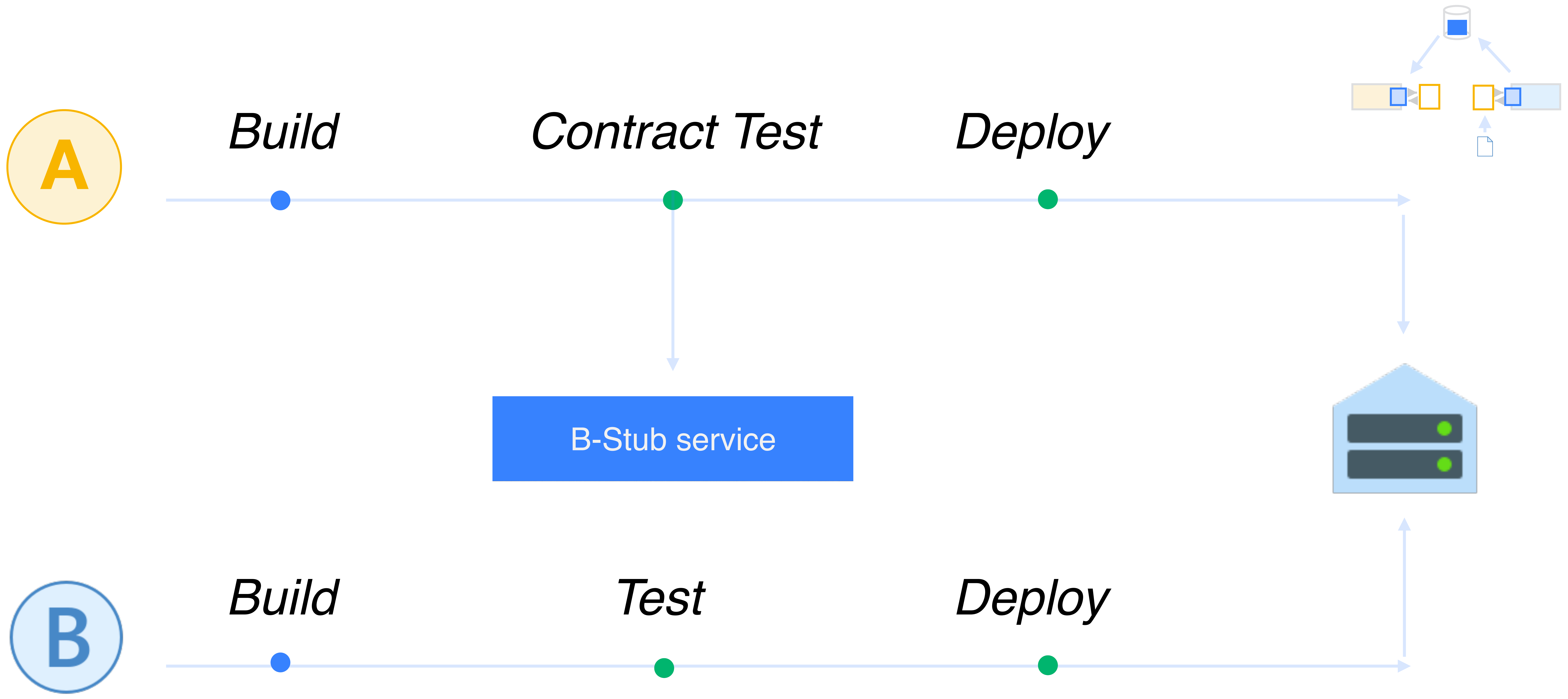


```
@Test
public void validate_new_api() throws Exception {
    // given:
        MockMvcRequestSpecification request = given();

    // when:
        ResponseOptions response = given().spec(request)
                .get("/api/v1/new-api");

    // then:
        assertThat(response.statusCode()).isEqualTo(200);
    // and:
        DocumentContext parsedJson = JsonPath.parse(response.getBody().asString());
        assertThatJson(parsedJson).field("['name']").isEqualTo("new api");
}
```

Contract Verifier Tests

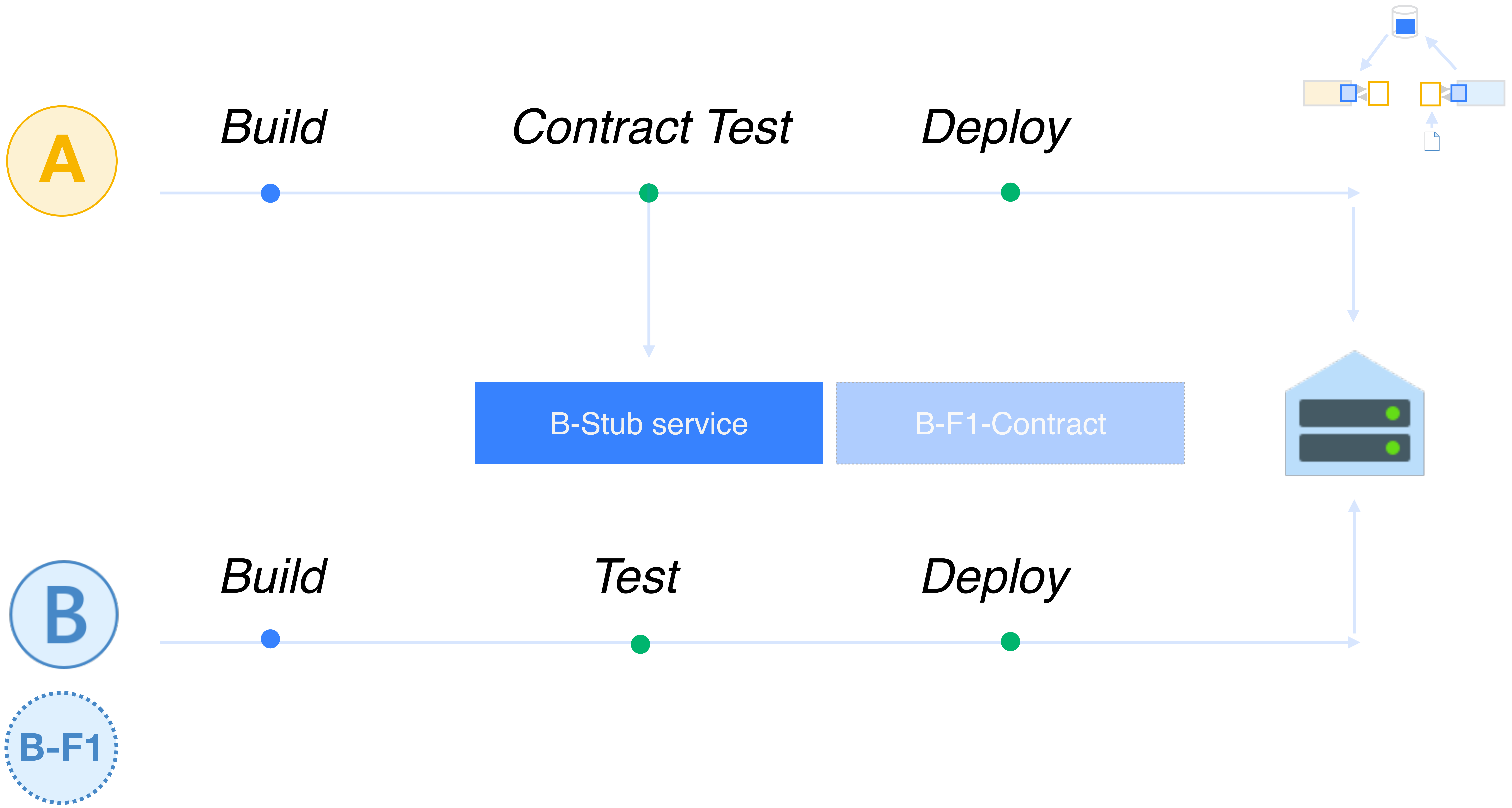
微服务PIPELINE

服务A为Consumer, 服务B为Provide



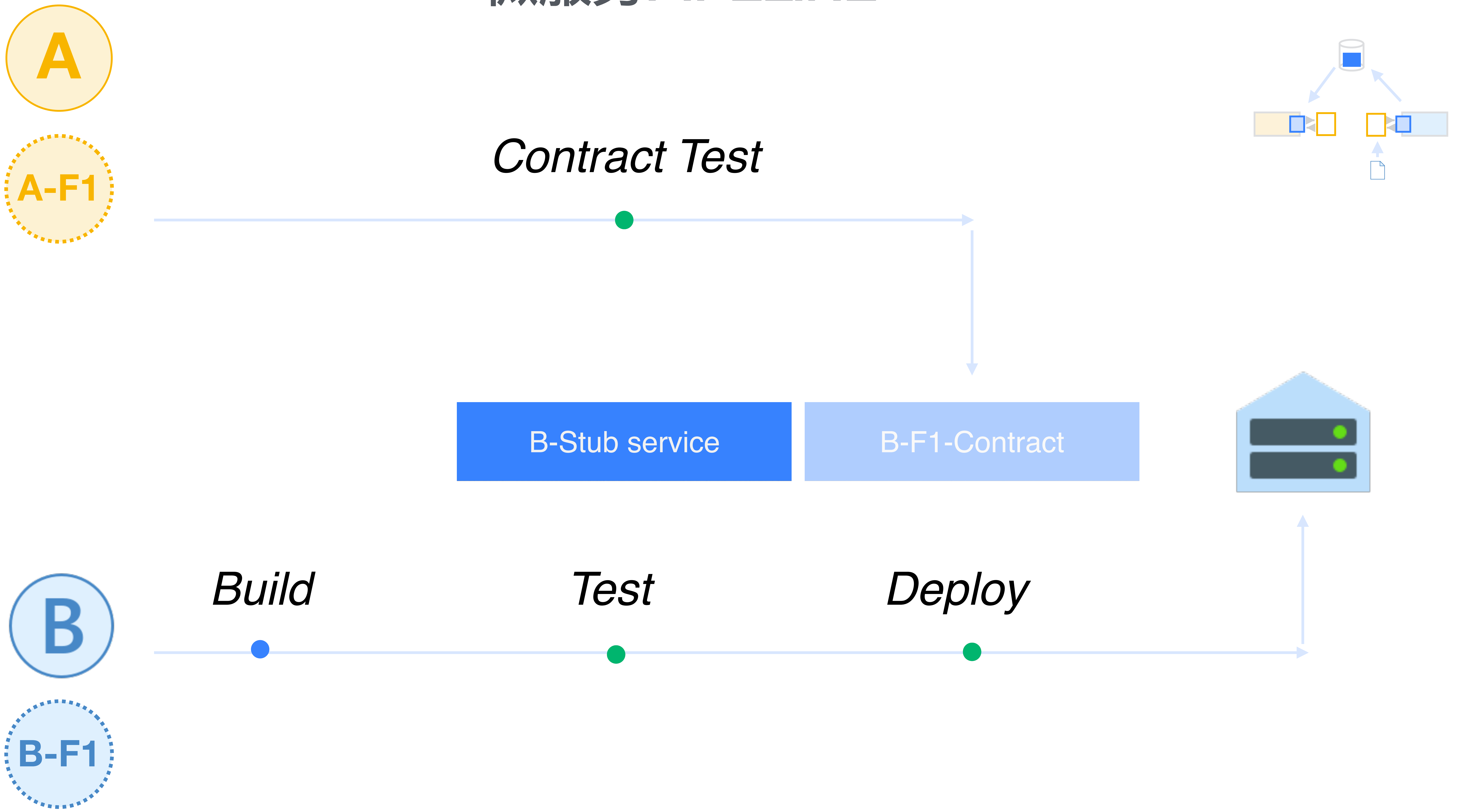
微服务PIPELINE

服务A为Consumer, 服务B为Provide



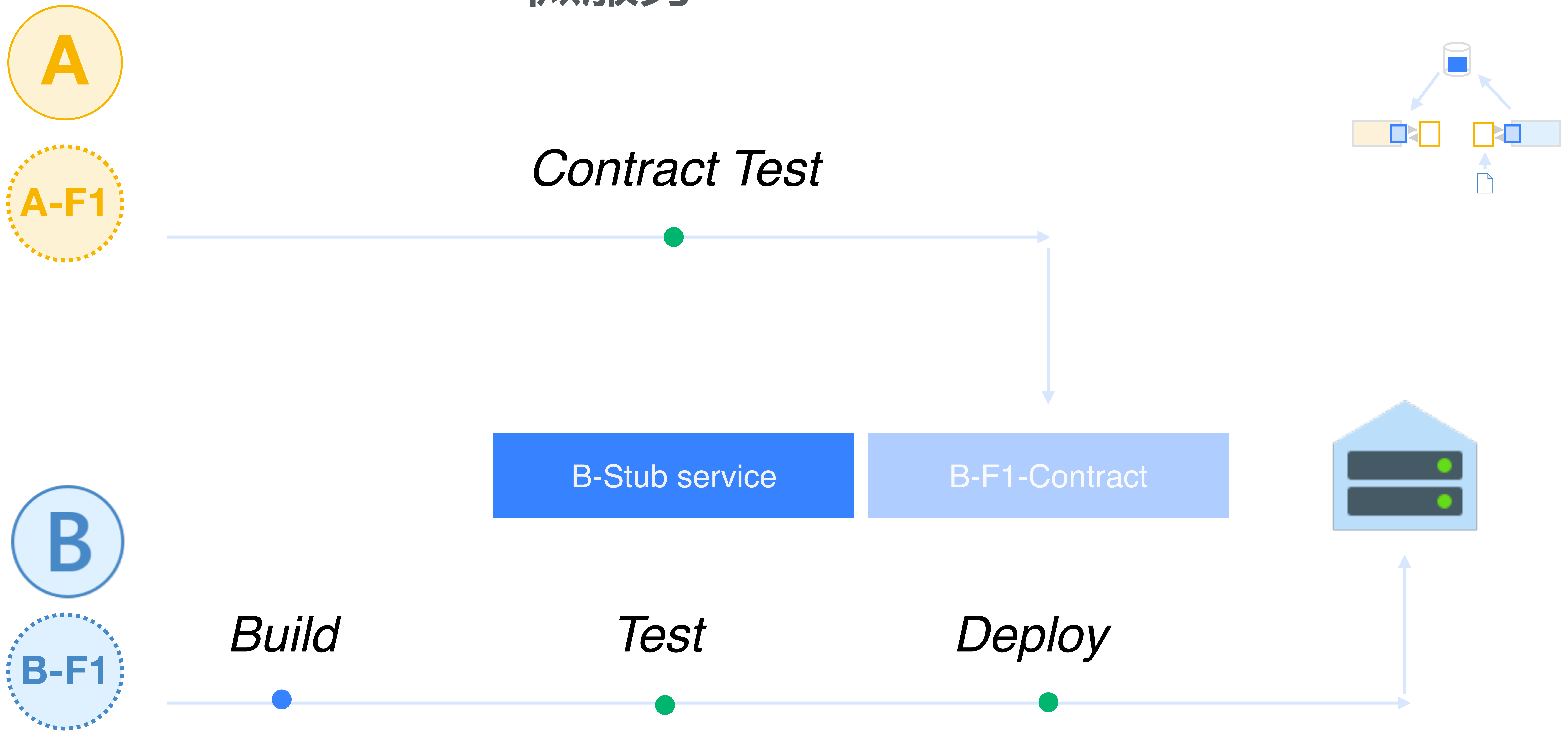
微服务PIPELINE

服务A为Consumer, 服务B为Provide



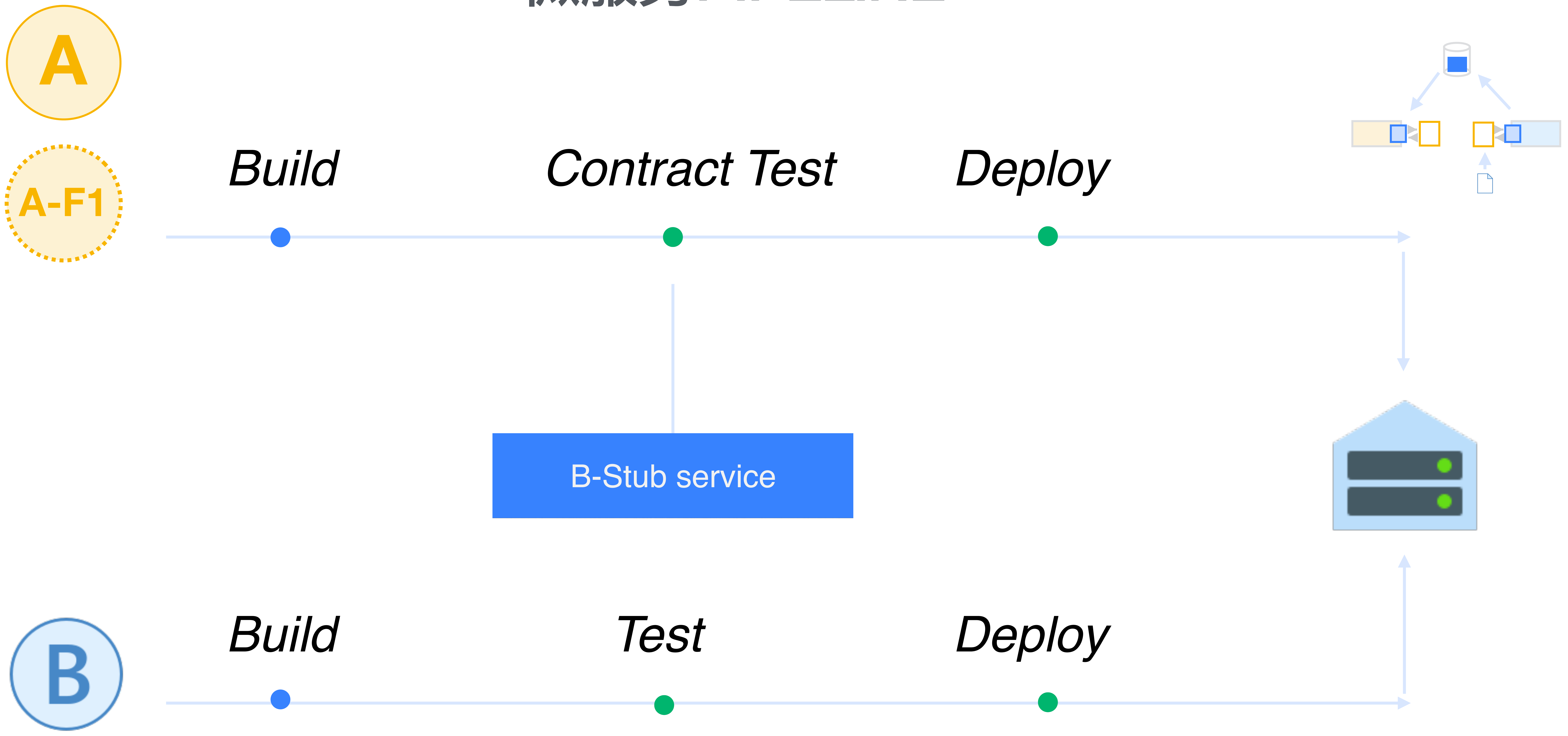
微服务PIPELINE

服务A为Consumer, 服务B为Provide



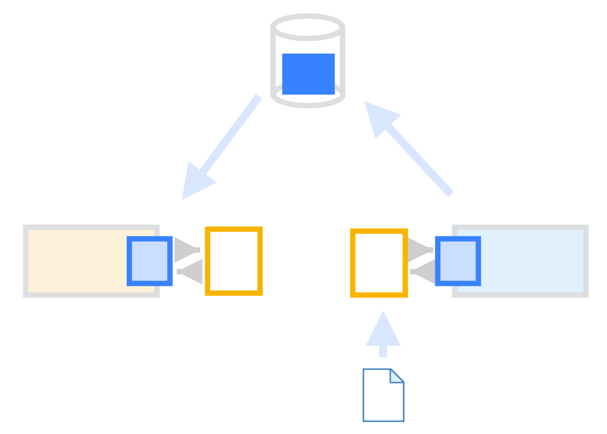
微服务PIPELINE

服务A为Consumer, 服务B为Provide



WORKFLOW

服务A为Consumer, 服务B为Provide



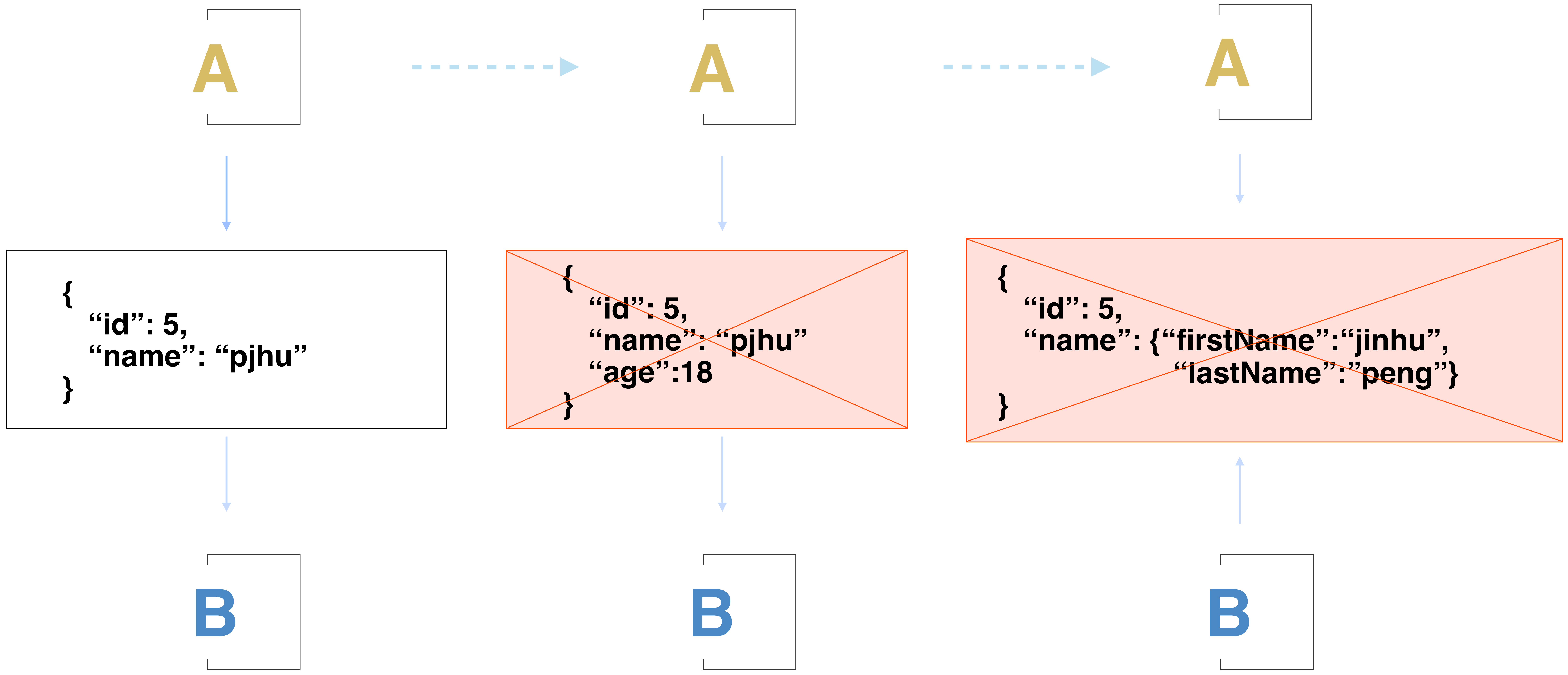
A端Dev将B端代码Clone到本地
创建分支B-F1,
添加契约并提交到分支B-F1

A端Dev在B项目安装到Local Maven Repository,启动Stub runner
A端checkout创建分支A-F1

B端checkout到B-F1代码，开发新功能
通过Contract Verifier Tests
合并B-F1到B，提交并处触发CI
发布新版本的Stub service

A端checkout创建分支A-F1
安装新契约到Local Maven Repository
连接B-Stub service测试
合并A-F1到A，提交并触发CI

CONSUMER-DRIVEN



```
{
  "id": 5,
  "name": "pjhu"
}
```

```
{
  "id": 5,
  "name": "pjhu"
  "age":18
}
```

```
{
  "id": 5,
  "name": {"firstName":"jinhu",
            "lastName":"peng"}
}
```

谢谢观看

ThoughtWorks®