Cloud Native Java Applications with Spring Boot

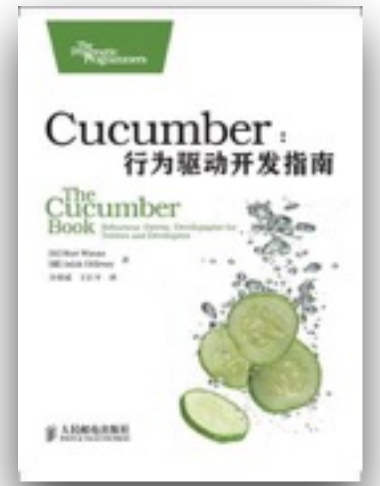
许晓斌
阿里巴巴高级技术专家

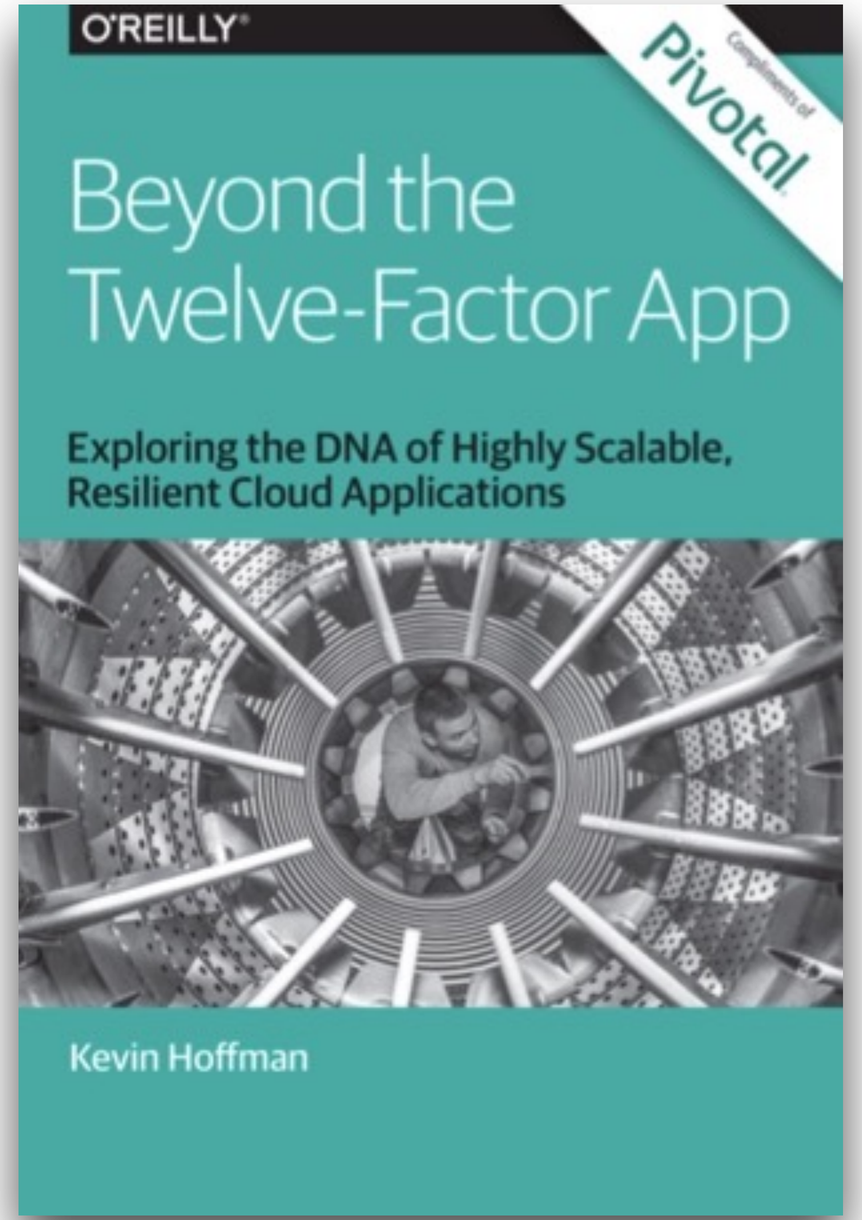
CNUTCon2016 全球容器技术大会

Alibaba Group 阿里巴巴集团

个人简介

- 程序员，Agile, DevOps
- 目前在 AliExpress 从事微服务实施及推广工作
- juven.xuxb@alibaba-inc.com




O'REILLY
Compliments of Pivotal
Beyond the Twelve-Factor App
Exploring the DNA of Highly Scalable, Resilient Cloud Applications
Kevin Hoffman

What is Cloud Native?

"A cloud-native application is an application that has been designed and implemented to run on a **Platform-as-a-Service** installation and to embrace **horizontal elastic scaling.** "

The Twelve-Factor App

https://12factor.net/

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

Beyond

https://pivotal.io/beyond-the-twelve-factor-app

1. API First
2. Telemetry
3. Authentication and Authorization

AliExpress™

Smarter Shopping, Better Living!

- Alibaba 集团旗下2C跨境电商网站
- 几乎是 100% Java
- 最早的代码来自 Alibaba B2B，有10年以上历史
- 有数百工程师，推崇 DevOps
- 已经有数百可独立发布的应用，数字还在不断增长

Codebase

- 100% Git
- 服务端代码 100% Maven
- 使用一个 Gitlab Group 来组织一条产品线代码
- 严格区分应用代码和共享代码
- 应用代码可以被 build, release, run，不进 maven 仓库
- 共享代码主要是 service api，可被发布进 maven 仓库
- 没有完全强制一个应用对应一个Git Project，同一产品线内可以适当共享服务内代码

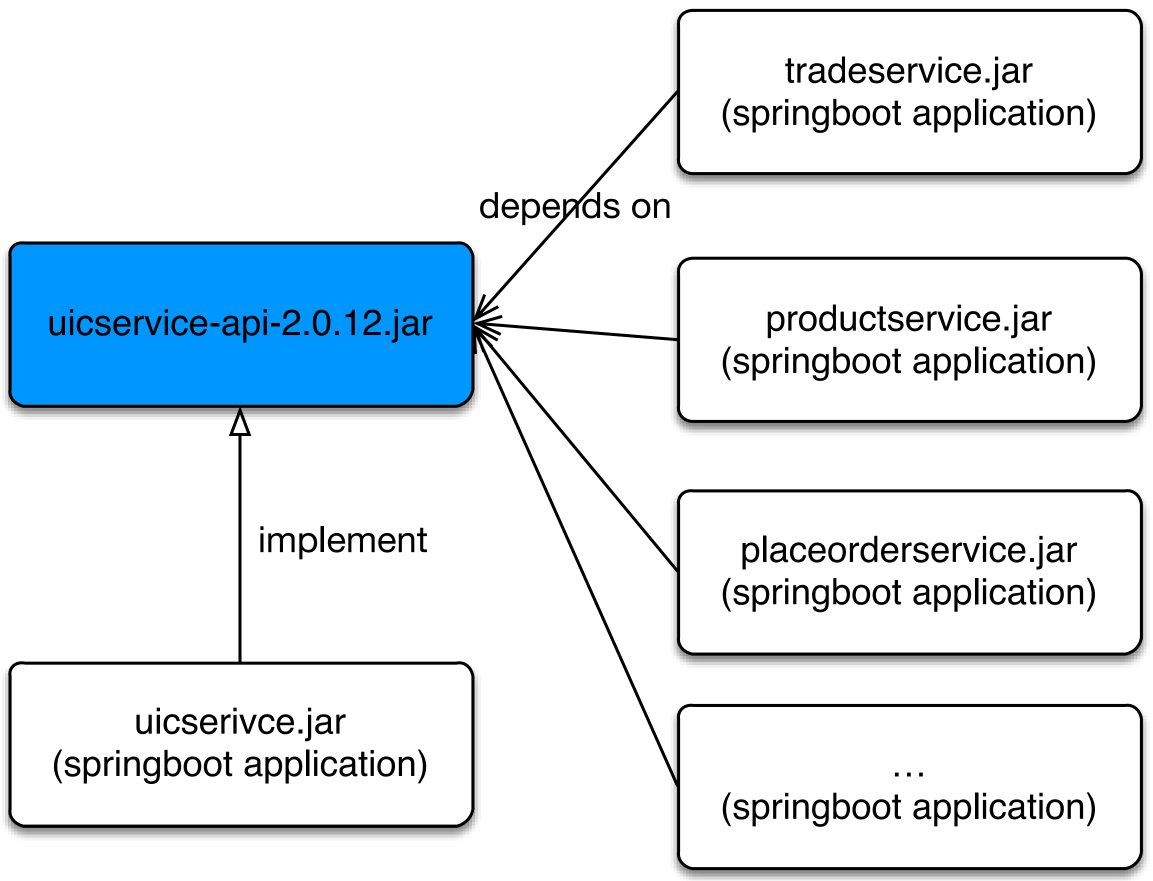
Codebase

```xml
<!-- generate git.properties -->
<plugin>
    <groupId>pl.project13.maven</groupId>
    <artifactId>git-commit-id-plugin</artifactId>
    <version>2.2.0</version>
    <executions>
        <execution>
            <goals>
                <goal>revision</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <verbose>true</verbose>
        <dateFormat>yyyy-MM-dd'T'HH:mm:ssZ</dateFormat>
        <generateGitPropertiesFile>true</generateGitPropertiesFile>
        <failOnNoGitDirectory>false</failOnNoGitDirectory>
    </configuration>
</plugin>
```

```json
{
    "team": {
        …
    },
    "service": {
        …
    },
    "scm": {
        "url": "http://gitlab.alibaba-inc.com/
fileserver2/fileserver-fileserver"
    },
    "git": {
        "branch": "master",
        "commit": {
            "id": "98ca8b9",
            "time": "2016-08-12T11:55:10+0800"
        }
    }
}
```

API First

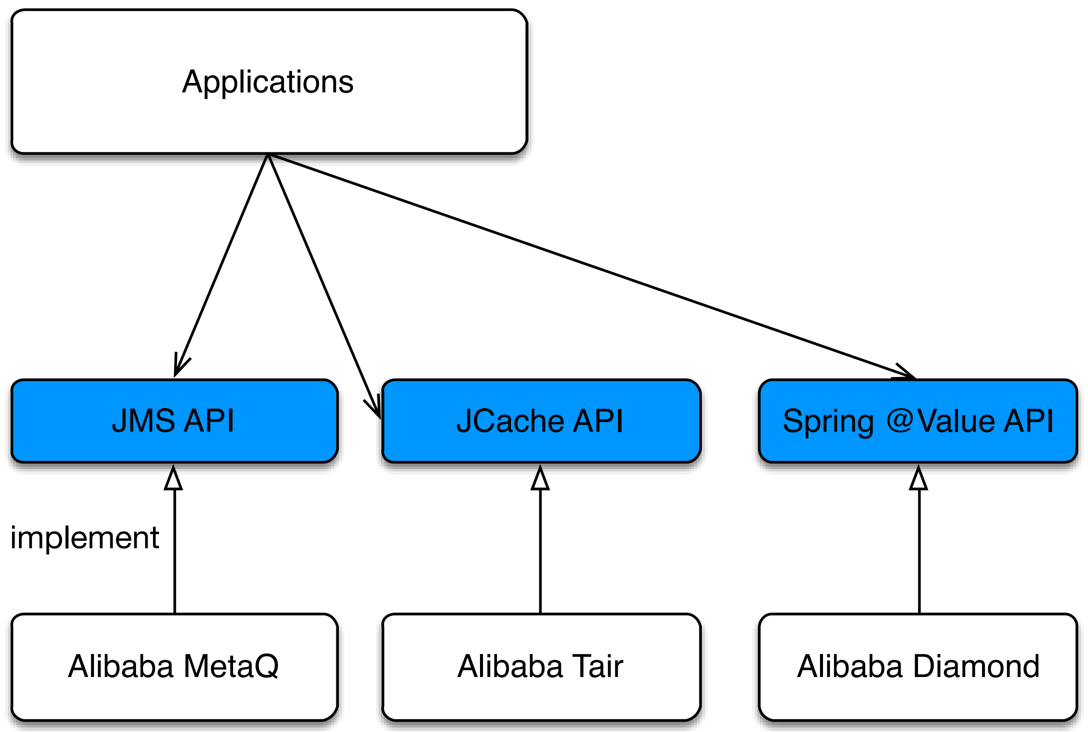
- 大多数下层服务 API 都是 Java Interface & DTO
- API 的升级成本非常得高，定义的时候需要和团队及用户 Review
- 独立的版本（Release Notes），独立的 Git Repo



tradeservice.jar (springboot application)
depends on
uicservice-api-2.0.12.jar
productservice.jar (springboot application)
implement
placeorderservice.jar (springboot application)
uicserivce.jar (springboot application)
… (springboot application)

API First

- 基础服务如消息、缓存、配置，使用业界API，结合阿里巴巴中间件实现
  - API 更易学习、理解、沟通
  - 实现有极高的性能和可靠性



Dependencies

- 对于大多数程序员来说，需要关心的系统依赖只有 Java 8 & SpringBoot
- Java 依赖使用 Maven 管理，内部有统一的 Nexus 存储 Java 依赖
- 使用 SNPASHOT parent pom (小心！)和 maven-enforcer-plugin 做强制
- 和 SpringBoot 一样，使用 Maven BOM 管理版本
- 自己实现了 spring-boot-starter-maven 方便查看 java 依赖

Dependencies - Enforcer

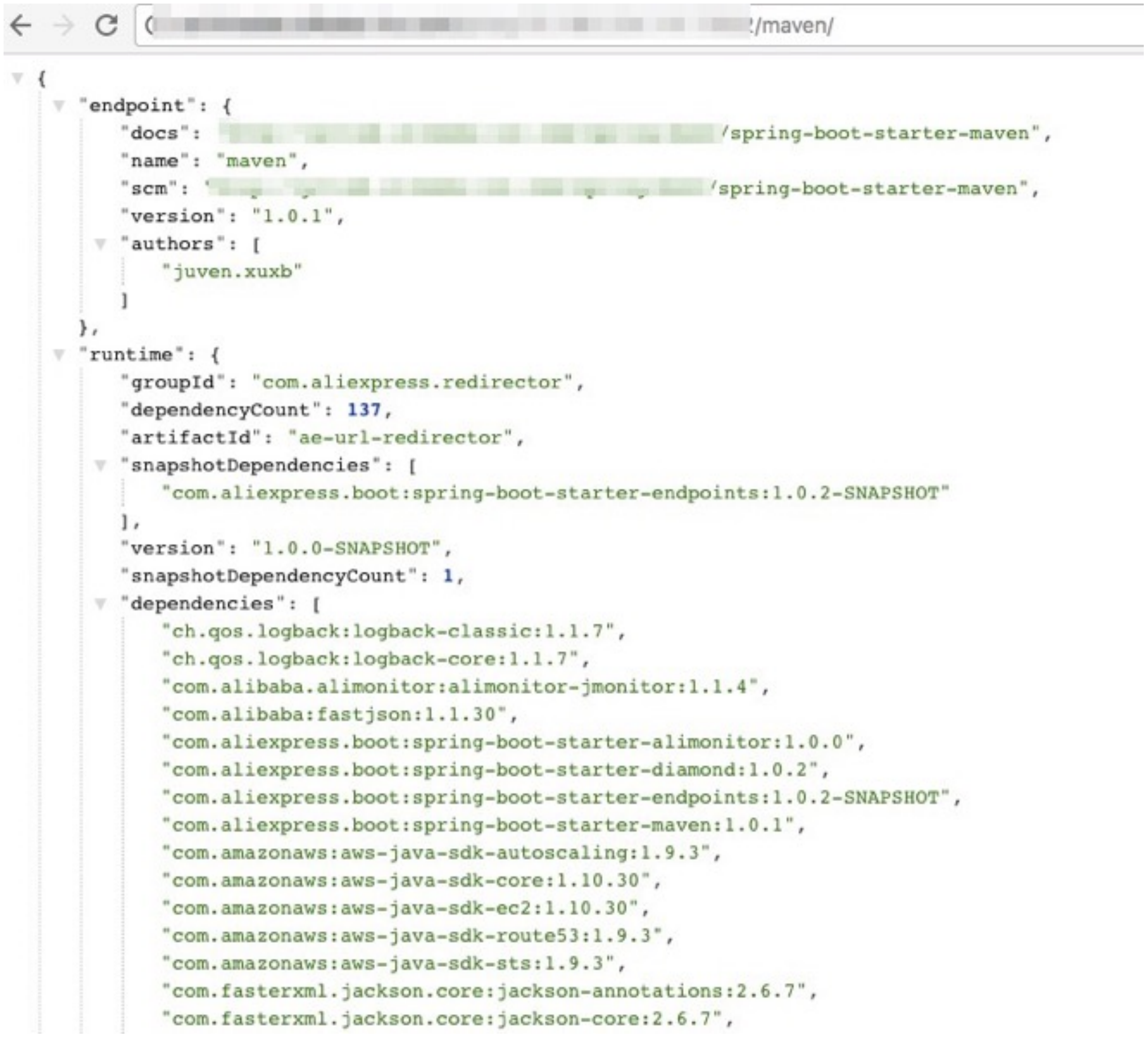
```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-enforcer-plugin</artifactId>
    <version>1.4.1</version>
    <executions>
        <execution>
            <id>enforce-maven3</id>
            <goals>
                <goal>enforce</goal>
            </goals>
            <configuration>
                <rules><requireMavenVersion><version>[3.2.5,)</versio
            </configuration>
        </execution>
        <execution>
            <id>enforce-java8</id>
            <goals>
                <goal>enforce</goal>
            </goals>
            <configuration>
                <rules><requireJavaVersion><version>[1.8.0,)</version
                </rules>
            </configuration>
        </execution>
        <execution>
            <id>enforce-src-main-java</id>
            <goals>
                <goal>enforce</goal>
            </goals>
            <configuration>
                <fail>true</fail>
                <rules>
                    <requireFilesDontExist>
                        <files>
                            <file>src/java</file>
                            <file>src/java.test</file>
                       </files>
                    </requireFilesDontExist>
                </rules>
            </configuration>
        </execution>
```

```
<parent>
    <groupId>com.aliexpress</groupId>
    <artifactId>aliexpress-parent</artifactId>
    <version>4-SNAPSHOT</version>
</parent>
```

```
<execution>
    <id>enforce-banned-dependencies</id>
    <goals>
        <goal>enforce</goal>
    </goals>
    <configuration>
        <fail>true</fail>
        <rules>
            <bannedDependencies>
                <excludes>
                    <exclude>javax.servlet:servlet-api</exclude>
                    <exclude>org.springframework.spring</exclude>
                    <exclude>org.mortbay.jetty:servlet-api-2.5</exclude>
                    ...
                </excludes>
            </bannedDependencies>
        </rules>
    </configuration>
</execution>
```

Dependencies - BOM

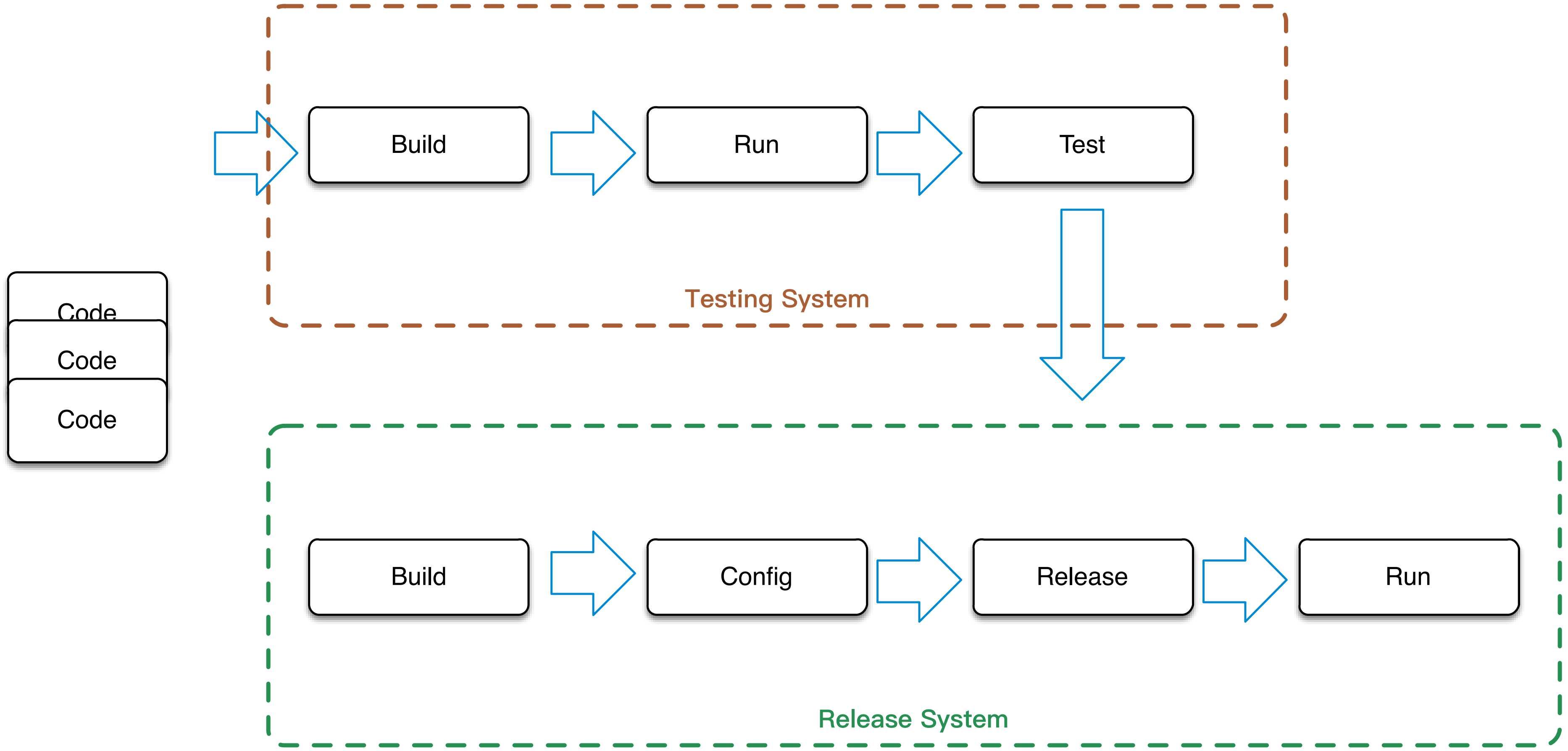
```xml
<properties>
    <spring-boot-starter-bom.version>1.0.2</spring-boot-starter-bom.version>
    <spring-boot.version>1.3.5.RELEASE</spring-boot.version>
    <spring-cloud.version>Brixton.RELEASE</spring-cloud.version>
    <spring-platform-bom.version>2.0.5.RELEASE</spring-platform-bom.version>
</properties>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.aliexpress.boot</groupId>
            <artifactId>spring-boot-starter-bom</artifactId>
            <version>${spring-boot-starter-bom.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>${spring-boot.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>${spring-cloud.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
            <groupId>io.spring.platform</groupId>
            <artifactId>platform-bom</artifactId>
            <version>${spring-platform-bom.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```



```
{
  "endpoint": {
    "docs": [redacted]/spring-boot-starter-maven",
    "name": "maven",
    "scm": "[redacted]/spring-boot-starter-maven",
    "version": "1.0.1",
    "authors": [
      "juven.xuxb"
    ]
  },
  "runtime": {
    "groupId": "com.aliexpress.redirector",
    "dependencyCount": 137,
    "artifactId": "ae-url-redirector",
    "snapshotDependencies": [
      "com.aliexpress.boot:spring-boot-starter-endpoints:1.0.2-SNAPSHOT"
    ],
    "version": "1.0.0-SNAPSHOT",
    "snapshotDependencyCount": 1,
    "dependencies": [
      "ch.qos.logback:logback-classic:1.1.7",
      "ch.qos.logback:logback-core:1.1.7",
      "com.alibaba.alimonitor:alimonitor-jmonitor:1.1.4",
      "com.alibaba:fastjson:1.1.30",
      "com.aliexpress.boot:spring-boot-starter-alimonitor:1.0.0",
      "com.aliexpress.boot:spring-boot-starter-diamond:1.0.2",
      "com.aliexpress.boot:spring-boot-starter-endpoints:1.0.2-SNAPSHOT",
      "com.aliexpress.boot:spring-boot-starter-maven:1.0.1",
      "com.amazonaws:aws-java-sdk-autoscaling:1.9.3",
      "com.amazonaws:aws-java-sdk-core:1.10.30",
      "com.amazonaws:aws-java-sdk-ec2:1.10.30",
      "com.amazonaws:aws-java-sdk-route53:1.9.3",
      "com.amazonaws:aws-java-sdk-sts:1.9.3",
      "com.fasterxml.jackson.core:jackson-annotations:2.6.7",
      "com.fasterxml.jackson.core:jackson-core:2.6.7",
```

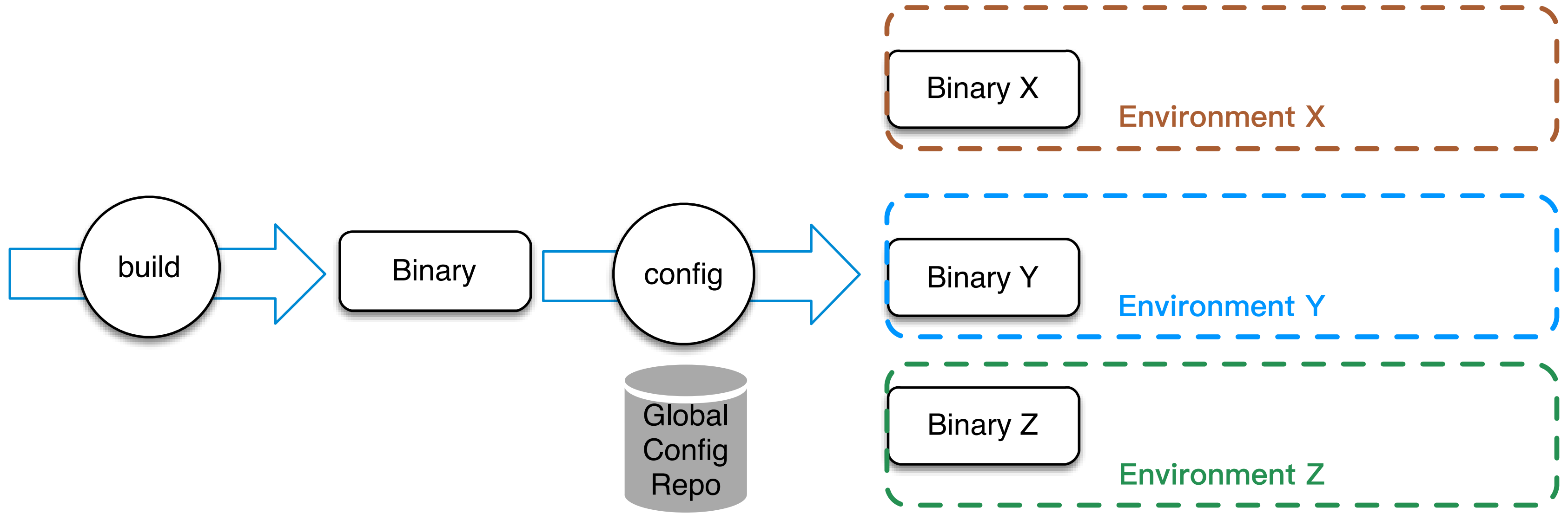
Build, Release, Run

- 独立的测试和发布系统，高度自动化
- 由于历史原因，系统并非最优的，例如 Build 可以优化、配置可以优化
- 系统应该尽可能保持简单
- 发布流程较复杂（考虑多套环境、Beta发布、回滚…）

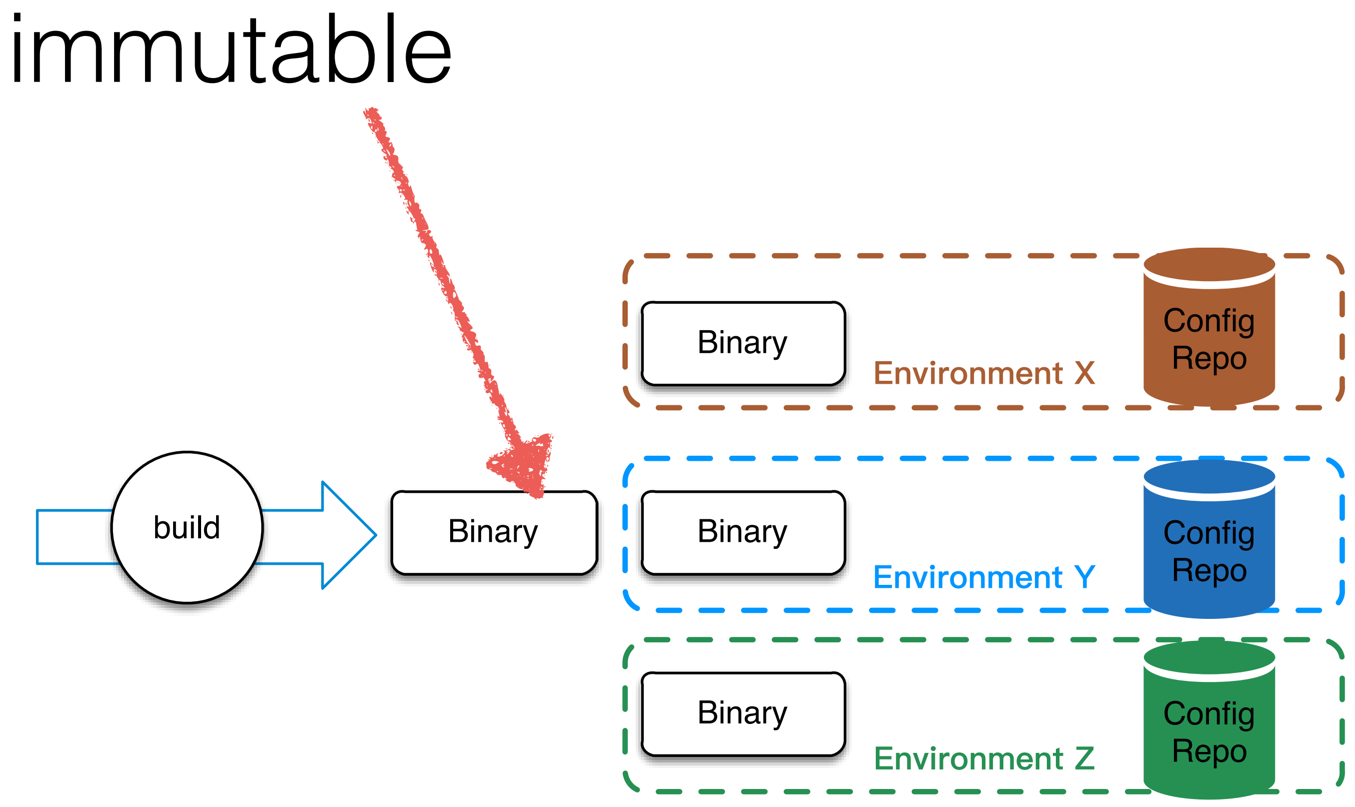


Configuration

- Before using Spring Boot:



- After using Spring Boot:



Configuration

- 构建时配置 v.s. 启动时配置
- immutable binary
  - 面向 Docker
  - 新增环境成本更低
- 基于 Spring Cloud Config 和 Alibaba Diamond 实现

Configuration

```java
import org.springframework.cloud.bootstrap.config.PropertySourceLocator;
import org.springframework.core.env.CompositePropertySource;
import org.springframework.core.env.Environment;
import org.springframework.core.env.PropertySource;

public class DiamondPropertySourceLocator implements PropertySourceLocator {

    @Override
    public PropertySource<?> locate(Environment environment) {
        String applicationName = environment.getProperty("spring.application.name");
        String applicationGroup = environment.getProperty("spring.application.group");

        CompositePropertySource compositePropertySource = new CompositePropertySource(DIAMOND_PROPERTY_SOURCE_NAME);

        loadGroupConfigurationRecursively(compositePropertySource, applicationGroup);

        loadApplicationConfiguration(compositePropertySource, environment, applicationGroup, applicationName);

        return compositePropertySource;
    }

}
```

Log

- SpringBoot 应用往本地磁盘写日志
- Agent 收集日志，传送到日志分析系统
- 独立的日志系统做分析和展现

Log

commons-logging

log4j

logback

slf4j

log4j-api

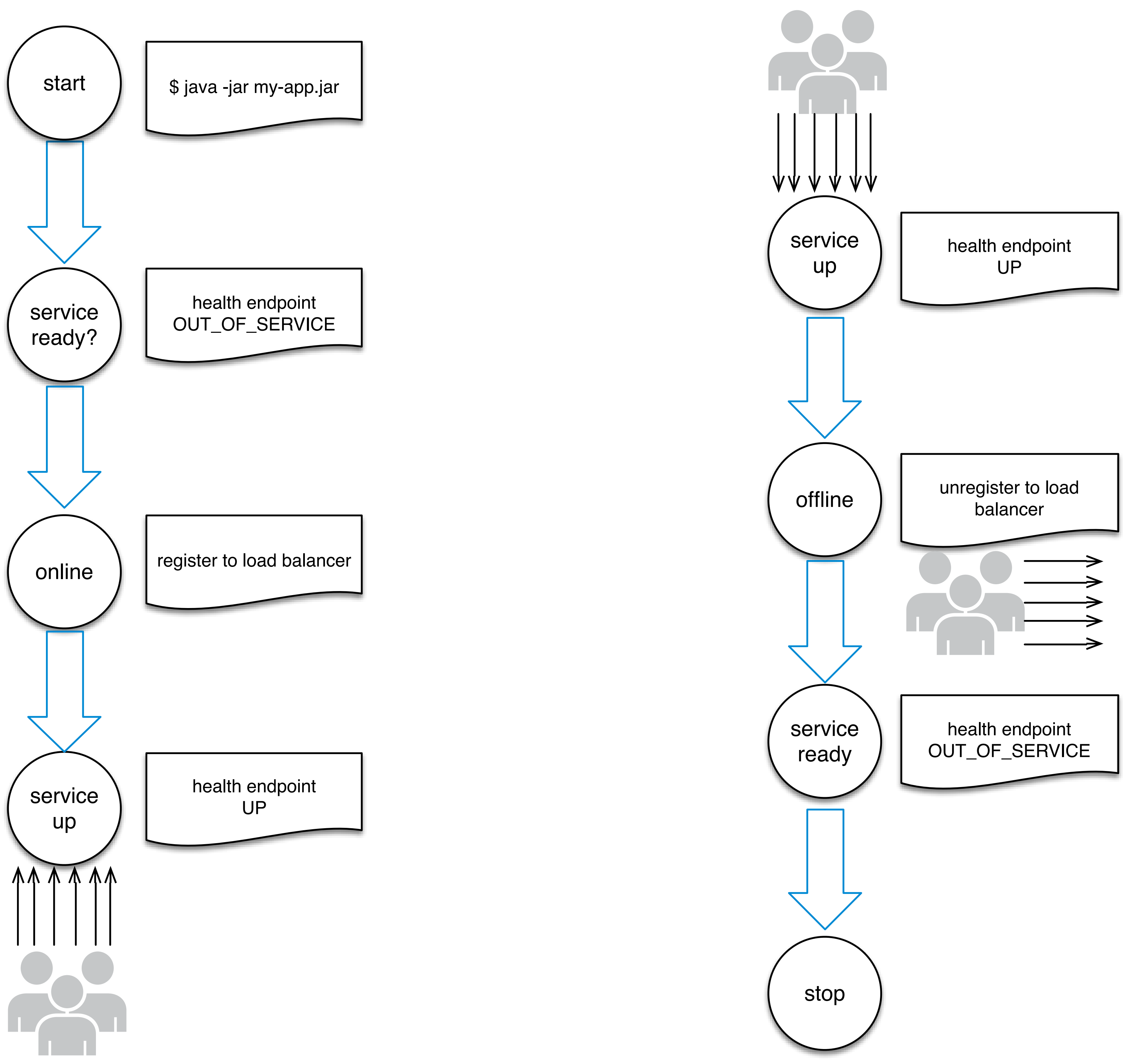
jcl-over-slf4j

log4j-over-slf4j

Disposability

- 应用要能够快速启动完毕，快速关闭
  - 应用启动时间是可以用来衡量微服务的参考指标
- 应用的启动和关闭过程不应该对用户产生影响

Disposability - 优雅上下线



Backing Services

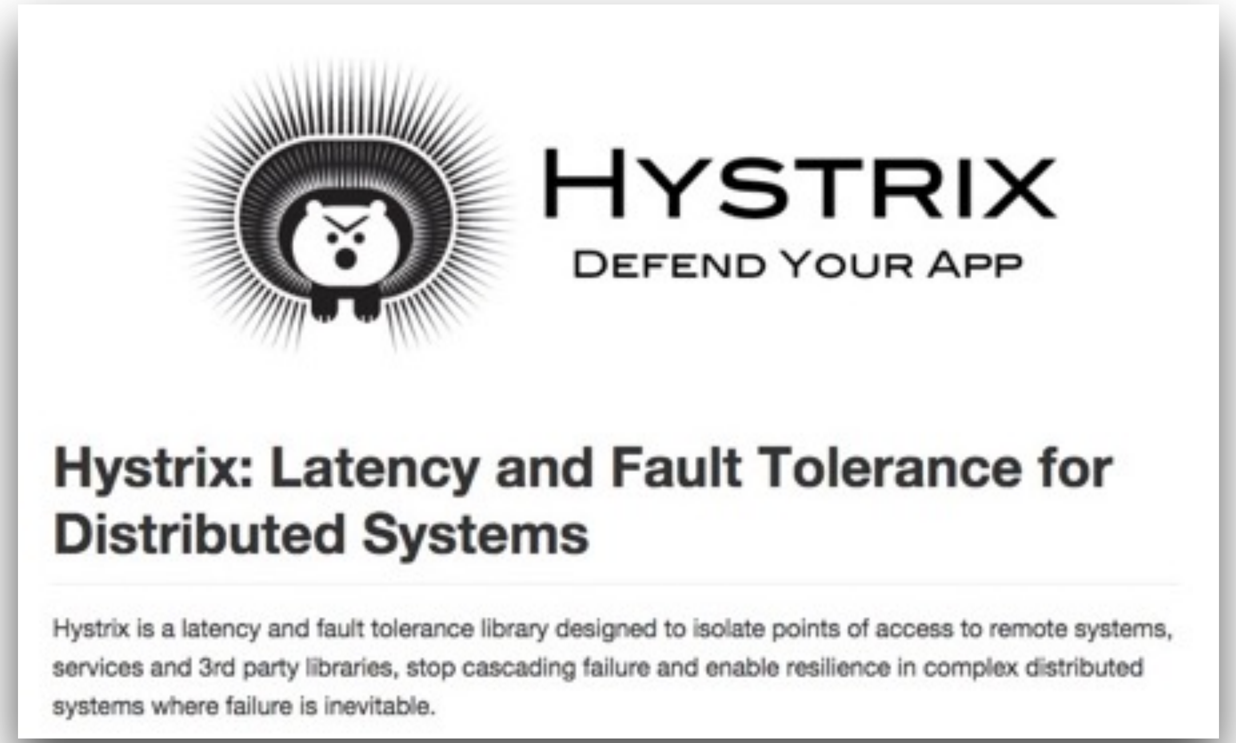
- 在 application.properties 中声明后端服务的 uri

```
spring.tddl.app = SPRING_BOOT_DEMO_APP
spring.tair.uri = diamond:///xxx
spring.hbase.uri = diamond://xxxx
spring.metaq.uri= diamond://xxx?producerId=spring-boot-demo-producer
```

- @HSFConsumer

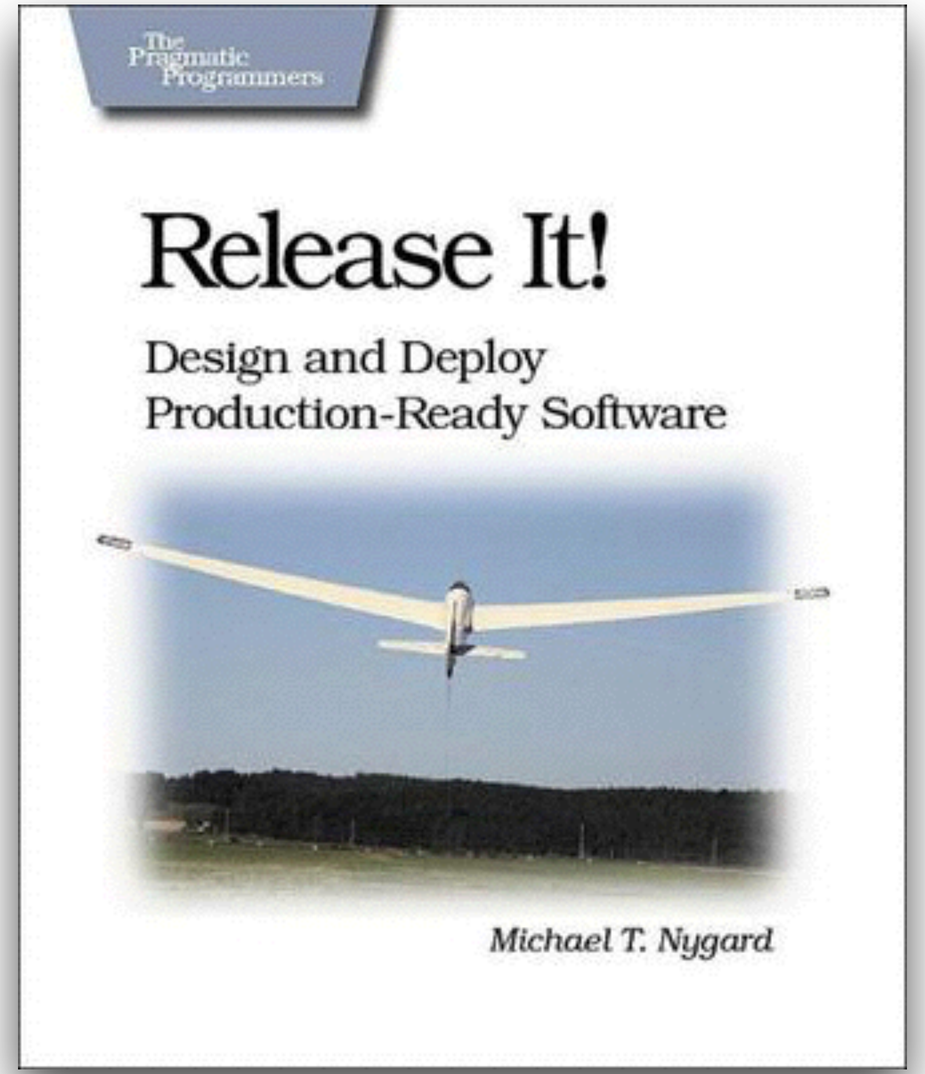
```java
@Component
public class Demo3Client {

    @HSFConsumer(serviceGroup = "HSF", serviceVersion = "1.0.0")
    private UserService userService;

    public String run() {
        return userService.getNick(123L);
    }
}
```

Backing Services - Circuit Breaker



Hystrix: Latency and Fault Tolerance for Distributed Systems

Hystrix is a latency and fault tolerance library designed to isolate points of access to remote systems, services and 3rd party libraries, stop cascading failure and enable resilience in complex distributed systems where failure is inevitable.


The Pragmatic Programmers
Release It!
Design and Deploy Production-Ready Software
Michael T. Nygard

Environment Parity

- 环境：笔记本/测试/预发布/正式
- 开发可以访问所有环境（正式环境通常 read only）
- 很高的发布频率（每天几十次）
- 配置点集中到 application.properties 中

Port Binding

- 目前所有应用使用同样端口，后续可能会改变
- 同一个应用的服务接口和 SpringBoot 管理借口使用不同端口

```
SPRINGBOOT_OPTS="--server.port=8080 --management.port=9090"
```

Stateless

- 所有状态存储在后端服务
  - MySql
  - Tair (Caching)
  - Hbase
  - ZooKeeper

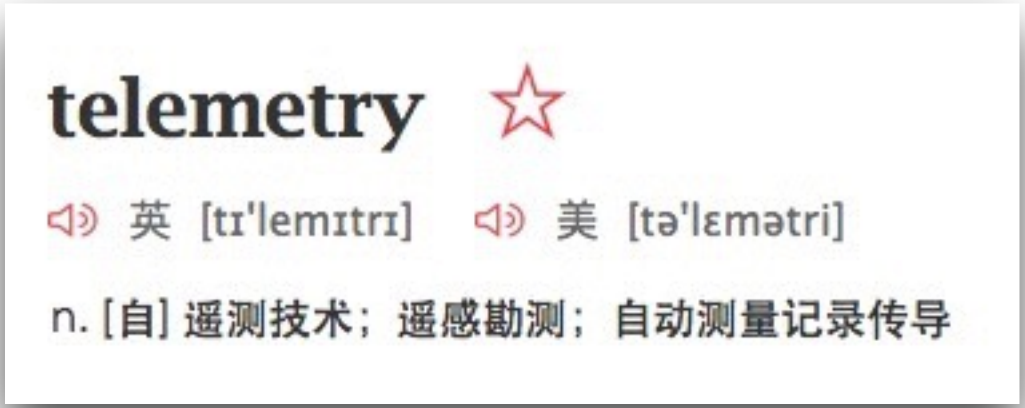
Concurrency

- 基于进程做水平扩展
- 前提：

Disposability

Stateless

Telemetry



telemetry ☆

英 [tɪ'lemɪtrɪ] 美 [tə'lɛmətri]

n. [自] 遥测技术；遥感勘测；自动测量记录传导

Telemetry - Endpoint

- http://localhost:9090/health/
- http://localhost:9090/info/
- http://localhost:9090/beans/
- http://localhost:9090/maven/
- http://localhost:9090/hsf/
- http://localhost:9090/metaq/
- …

Telemetry - Endpoint - health

```json
{
    "status": "UP",
    "discoveryComposite": {
        "description": "Spring Cloud Eureka Discovery Client",
        "status": "UP",
        "discoveryClient": {
            "description": "Spring Cloud Eureka Discovery Client",
            "status": "UP"
        },
        "eureka": {
            "description": "Remote status from Eureka server",
            "status": "UP"
        }
    },
    "diamond": {
        "status": "UP",
        "dataIds": [
            "com.aliexpress.archimedes:archimedes-master.properties",
            "com.aliexpress:application.properties"
        ]
    },
    "jms": {
        "status": "UP",
        "metaq.producer.connection.clientId": "xxxx4-b5f4-cf061b89cb53",
        "metaq.producer.connection.provider": "Alibaba-ONS"
    },
    "diskSpace": {
        "status": "UP",
        "total": 64424509440,
        "free": 50193530880,
        "threshold": 10485760
    },
    "db": {
        "status": "UP",
        "database": "H2",
        "hello": 1
    },
    "hystrix": {
        "status": "UP"
    }
}
```

Telemetry - Endpoint - metaq

```
{
    "endpoint": {
        "docs": "http:/xxx/spring-boot/spring-boot-starter-metaq",
        "name": "metaq",
        "scm": "http://xxx/spring-boot/spring-boot-starter-metaq",
        "version": "1.0.0",
        "authors": [
            "juven.xuxb"
        ]
    },
    "runtime": {
        "lastSent.timestamp": 1464117003593,
        "lastSent.messageId": "ID:xxxx"
    },
    "metrics": {
        "metaq.archimedes_requests_queue:baseType.totalSent": {
            "count": 11837524
        },
        "metaq.archimedes_requests_queue:baseType.sentPerSecond": {
            "count": 11837524,
            "oneMinuteRate": 66.90813307115646,
            "fiveMinuteRate": 61.75123069534975,
            "fifteenMinuteRate": 60.04974740983519,
            "meanRate": 55.769492957319144
        },
        "metaq.archimedes_requests_queue:baseType.sentFailuresPerSecond": {
            "count": 0,
            "oneMinuteRate": 0.0,
            "fiveMinuteRate": 0.0,
            "fifteenMinuteRate": 0.0,
            "meanRate": 0.0
        },
        "metaq.archimedes_requests_queue:baseType.totalSentFailures": {
            "count": 0
        }
    },
    "config": {
        "producerId": "archimedes_requests_producer_master"
    }
}
```

Q & A

许晓斌
juven.xuxb@alibaba-inc.com

CNUTCon2016 全球容器技术大会

Alibaba Group 阿里巴巴集团