Java 微服务实践

Spring Cloud 原生云应用

小马哥

segmentfault 讲堂

Java 微服务实战系列课堂

- 讲师信息

小马哥，一线互联网技术专家，国内微服务技术客串讲师，目前主要负责微服务技术推广、架构设计、基础设施、迁移等。重点关注云计算、微服务以及软件架构等领域。从事十余年Java EE 开发，期间通过SUN Java（SCJP、SCWCD、SCBCD）以及Oracle OCA等的认证。

- 微信 / SF ID：mercyblitz
- 课程详情：http://sf.gg/n/1330000009887617
- 课件资源：https://github.com/mercyblitz/segmentfault-lessons/
- JSR资源：https://github.com/mercyblitz/jsr

segmentfault 讲堂

主要议题

- 系列介绍
- 背景知识
- 12-Factor 应用
- Bootstrap 上下文
- Actuator Endpoints
- 问答互动

系列介绍

系列介绍

- 课程特点

Spring Cloud 系列课程致力于以实战的方式覆盖 Spring Cloud 的功能特性，更为重要的是，小马哥希望通过“授人以渔”的方式，不仅让小伙伴们能够认识到技术的衍进并非凭空遐想，而是在其特定的场景下“生根发芽”，并且结合自身十余年的学习方法和工作经验，将技术的发展脉络贯穿其中。循序渐进式地引导朋友们，站在哲学的高度，体会 Spring Cloud 的作者设计意图。同时，结合 Spring Cloud 的源码加深理解，最终达到形成系统性的知识和技术体系的目的。

系列介绍

- 讲授方式
  - 核心理念：介绍 Spring Cloud 技术背后的核心概念、架构以及模式
  - 使用方法：掌握 Spring Cloud 功能组件基本使用方法
  - 经验分享：分享真实的生产经验，学以致用
  - 源码分析：通过核心组件源码，加深理解

系列介绍

- 什么是 Spring Cloud?

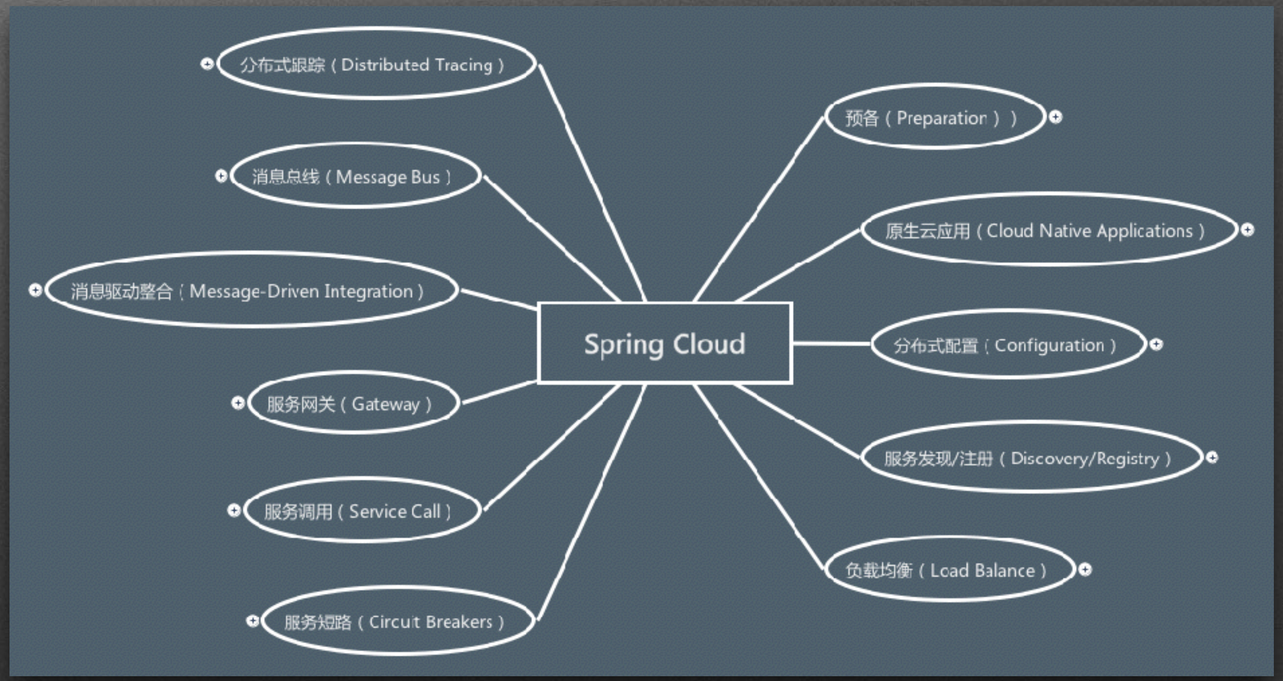
Spring Cloud 为开发人员提供快速构建分布式系统的一些通用模式，其中包括：配置管理、服务发现、服务短路、智能路由、微型网关、控制总线、一次性令牌、全局锁、领导选举、分布式会话和集群状态。分布式系统间的协调导向样板模式，并且使用 Spring Cloud 的开发人员能够快速地构建实现这些模式的服务和应用。这些服务和应用也将在任何环境下工作良好，无论是开发者的笔记本、还是数据中心裸机或者管控平台。

参考资源：http://cloud.spring.io/spring-cloud-static/current/

背景知识

Spring Cloud

Spring Cloud 为开发人员提供快速构建分布式系统的一些通用模式，其中包括：配置管理、服务发现、服务短路、智能路由、微型网关、控制总线、一次性令牌、全局锁、领导选举、分布式会话和集群状态。分布式系统间的协调导向样板模式，并且使用 Spring Cloud 的开发人员能够快速地构建实现这些模式的服务和应用。这些服务和应用也将在任何环境下工作良好，无论是开发者的笔记本、还是数据中心裸机或者管控平台。

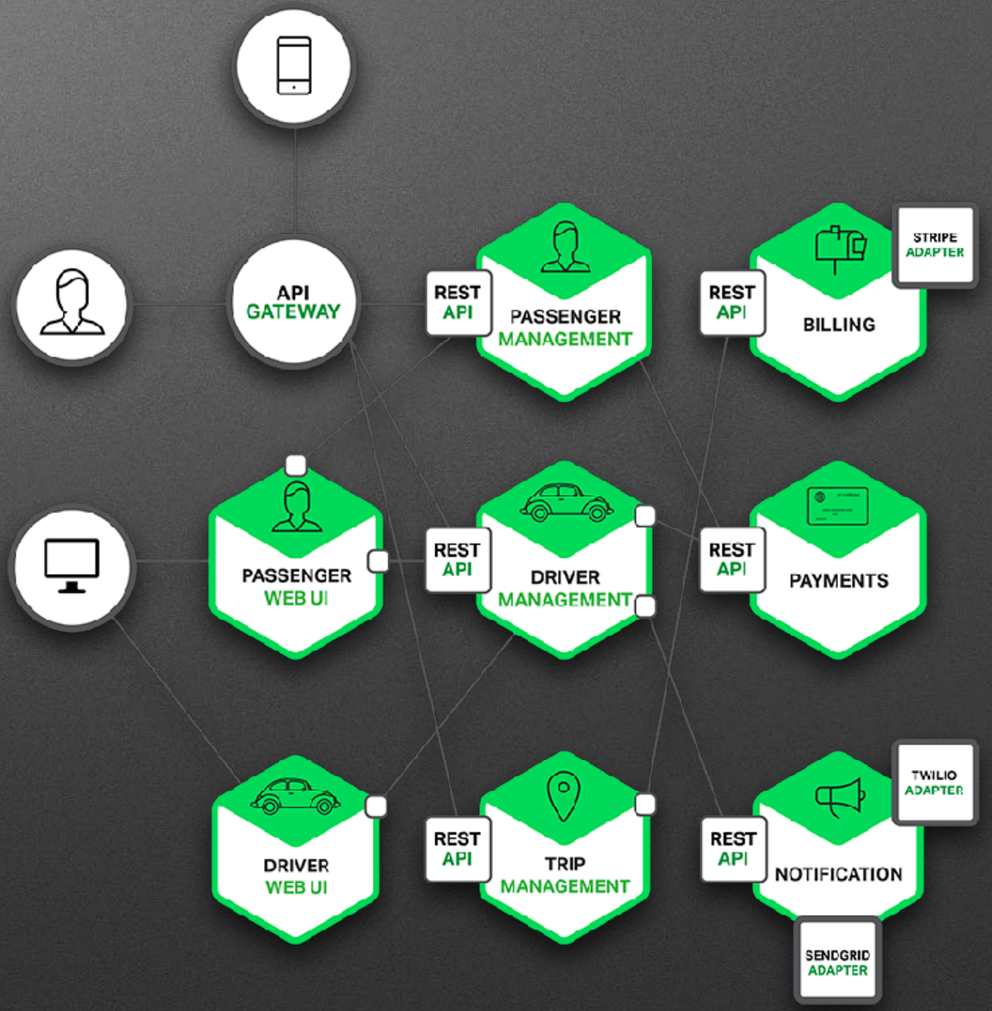


面向服务架构（SOA）

A service-oriented architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. The basic principles of service-oriented architecture are independent of vendors, products and technologies. A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online.

微服务架构（MSA）

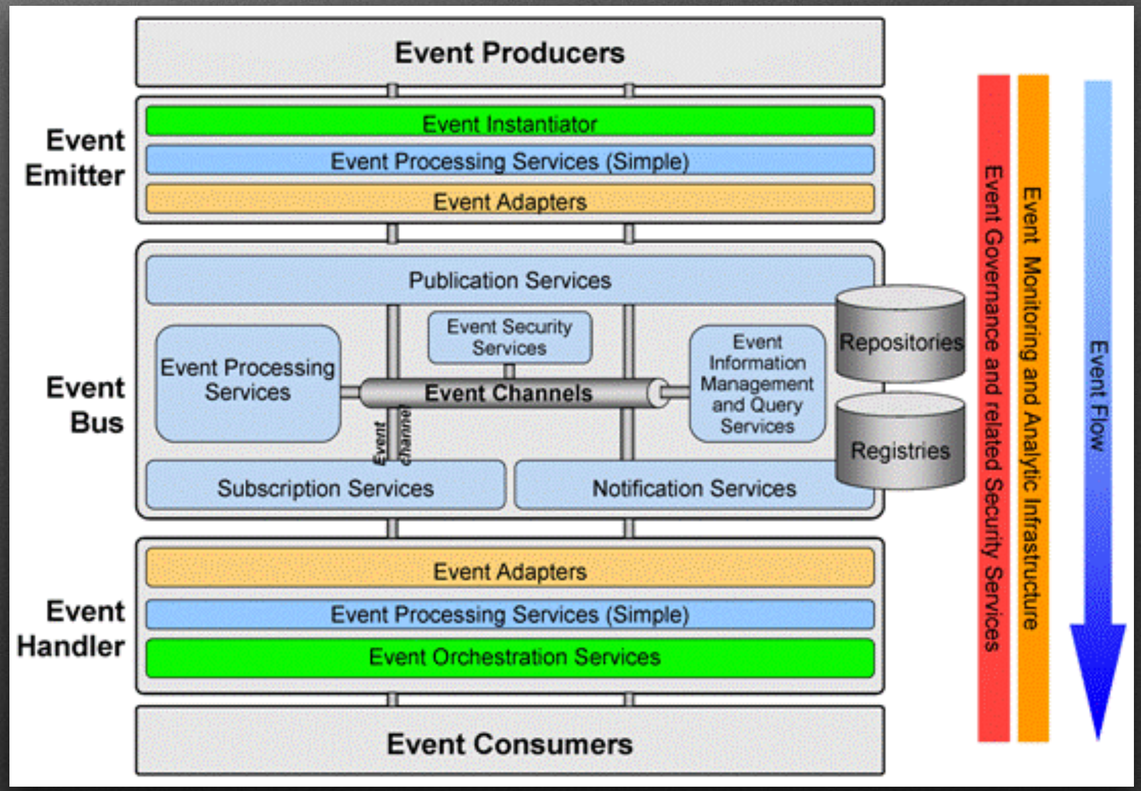
Microservices is a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services should be fine-grained and the protocols should be lightweight. The benefit of decomposing an application into different smaller services is that it improves modularity and makes the application easier to understand, develop and test.



事件驱动架构（EDA）

Event-driven architecture (EDA), is a software architecture pattern promoting the production, detection, consumption of, and reaction to events.

This architectural pattern may be applied by the design and implementation of applications and systems that transmit events among loosely coupled software components and services. An event-driven system typically consists of event emitters (or agents), event consumers (or sinks), and event channels.



12-Factor 应用

I. Codebase

II. Dependencies

III. Config

IV. Backing services

V. Build, release, run

VI. Processes

VII. Port binding

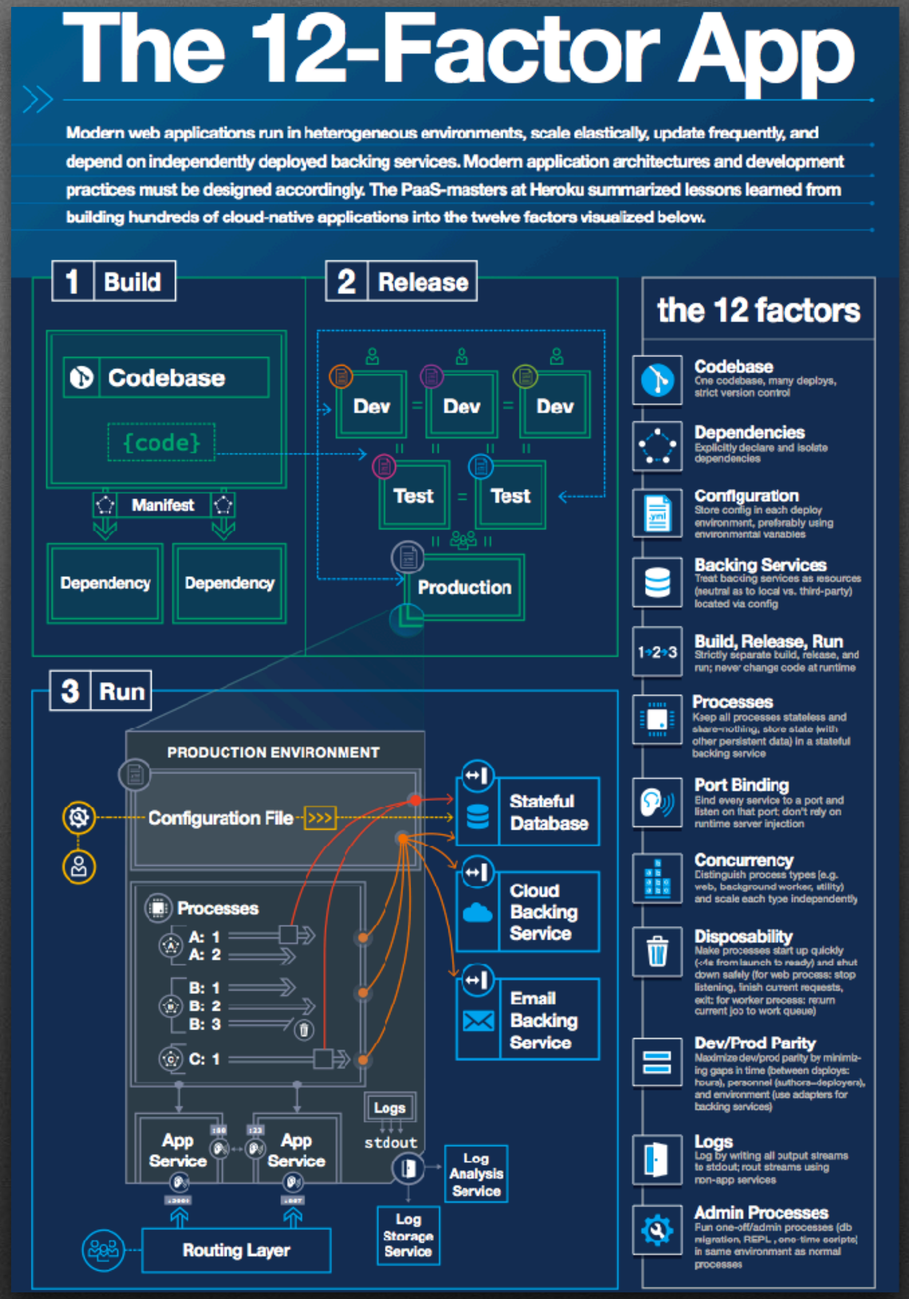
VIII. Concurrency

IX. Disposability

X. Dev/prod parity

XI. Logs

XII. Admin processes



Bootstrap 上下文

Bootstrap 上下文

- 理解 Bootstrap 上下文

Bootstrap 上下文是 Spring Cloud 新引入的，与传统 Spring 上下文相同，系 **ConfigurableApplicationContext** 实例，由 BootstrapApplicationListener 在监听 **ApplicationEnvironmentPreparedEvent** 时创建。

- Spring 事件/监听器模式
  - ApplicationEvent / ApplicationListener

Bootstrap 上下文

- 理解 SpringApplication

SpringApplication 是 Spring Boot 引导启动类，与 Spring 上下文、事件、监听器以及环境等组件关系紧密，其中提供了控制 Spring Boot 应用特征的行为方法。

- Spring Boot 应用运行监听器
  - SpringApplicationRunListener

Bootstrap 上下文

- 理解 Spring Boot 事件
  - 事件触发器：EventPublishingRunListener

ApplicationStartedEvent

ApplicationEnvironmentPreparedEvent

ApplicationPreparedEvent

ApplicationReadyEvent / ApplicationFailedEvent

Bootstrap 上下文

- 理解 Spring Boot / Spring Cloud 上下文层次关系
  - Spring Boot 上下文
    - 非 Web 应用：AnnotationConfigApplicationContext
    - Web 应用：AnnotationConfigEmbeddedWebApplicationContext
  - Spring Cloud 上下文：Bootstrap （父）

Actuator Endpoints

Actuator Endpoints

- 理解 Actuator Endpoints

Actuator 中文直译为“传动装置”，在 Spring Boot 使用场景中表示为“生产而准备的特性”（Production-ready features），这些特性通过 HTTP 端口的形式，帮助相关人员管理和监控应用。大致上可以归类为：

监控类：“端点信息”、“应用信息”、“外部化配置信息”、“指标信息”、“健康检查”、“Bean 管理”、“Web URL 映射管理”、“Web URL 跟踪”

管理类：“外部化配置”、“日志配置”、“线程dump”、“堆dump”、“关闭应用”

注意：Spring Boot 1.5 开始 Actuator 增强了安全能力

Actuator Endpoints

- Spring Cloud 扩展 Actuator Endpoints

上下文重启：/restart

暂定：/pause

恢复：/resume

注意：Spring Boot 1.5 开始 Actuator 增强了安全能力

问答互动

下期预告

配置服务器（*Spring Cloud Config Server*）

谢谢观看