

# Spring Cloud 云原生应用



扫码试看/订阅  
《玩转 Spring 全家桶》

# Spring Cloud 及 Cloud Native 概述

# 简单理解微服务

“微服务就是一些协同工作的小而自治的服务。”

– *Sam Newman*

# 微服务的优点

## 异构性

- 语言、存储.....

## 弹性

- 一个组件不可用，不会导致级联故障

## 扩展

- 单体服务不易扩展，多个较小的服务可以按需扩展

# 微服务的优点

- 易于部署
- 与组织结构对齐
- 可组合性
- 可替代性

# 实施微服务的代价

- 没有银弹！！！！
- 分布式系统的复杂性
- 开发、测试等诸多研发过程中的复杂性
- 部署、监控等诸多运维复杂性
- .....

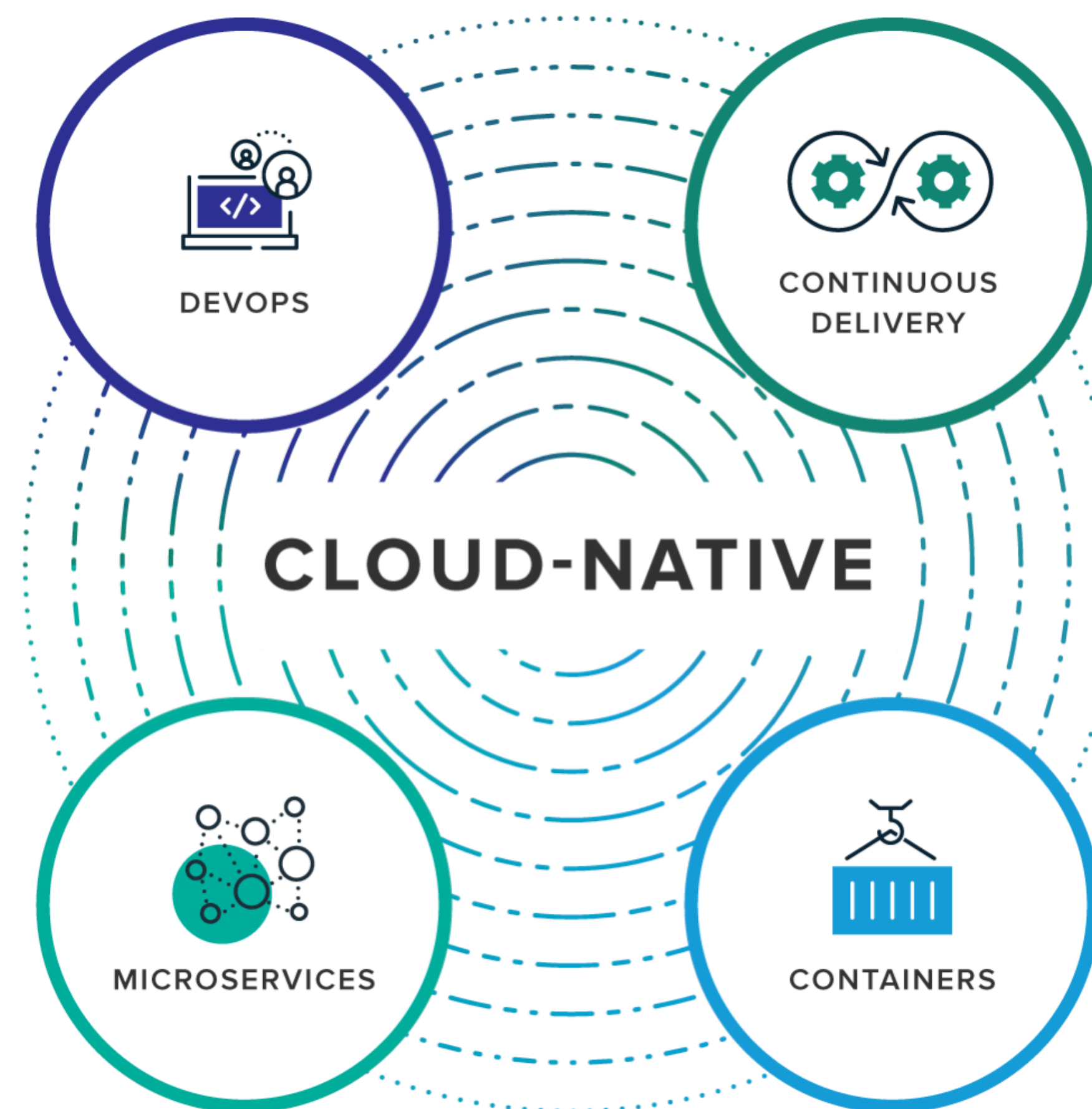


# 如何理解云原生

“云原生技术有利于各组织在公有云、私有云和混合云等新型动态环境中，构建和运行可弹性扩展的应用。”

– *CNCF Cloud Native Definition v1.0*

# 云原生应用要求.....



# 云原生应用要求.....

## DevOps

- 开发与运维一同致力于交付高品质的软件服务于客户

## 持续交付

- 软件的构建、测试和发布，要更快、更频繁、更稳定

## 微服务

- 以一组小型服务的形式来部署应用

## 容器

- 提供比传统虚拟机更高的效率

# Cloud Native Computing Foundation



Kubernetes

Orchestration



Prometheus

Monitoring



Envoy

Service Proxy



CoreDNS

Service Discovery



containerd

Container Runtime



# 12-Factor App

# THE TWELVE-FACTOR APP

## 目的

- 为构建 SaaS 应用提供行之有效的方法论
- 适用于任意语言和后端服务的开发的应用程序
- [https://12factor.net/zh\\_cn/](https://12factor.net/zh_cn/)

# 了解 12-Factors

## 基准代码 (Codebase)

- 一份基准代码，多份部署

## 依赖 (Dependencies)

- 显式声明依赖关系

## 配置 (Config)

- 在环境中存储配置

## 后端服务 (Backing services)

- 把后端服务当作附加资源



# 了解 12-Factors

## 构建，发布，运行 (Build, release, run)

- 严格分离构建和运行

## 进程 (Processes)

- 以一个或多个无状态进程运行应用

## 端口绑定 (Port Binding)

- 通过端口绑定提供服务

## 并发 (Concurrency)

- 通过进程模型进行扩展

# 了解 12-Factors

## 易处理 (Disposability)

- 快速启动和优雅终止可最大化健壮性

## 开发环境与线上环境等价 (Dev / Prod parity)

- 尽可能的保持开发，预发布，线上环境相同

## 日志 (Logs)

- 把日志当作事件流

## 管理进程 (Admin processes)

- 后台管理任务当作一次性进程运行

# 一份基准代码，多份部署

- 使用版本控制系统加以管理
- 基准代码与应用保持一一对应的关系
- 尽管每个应用只对应一份基准代码，但可以同时存在多份部署

# 显式声明依赖关系

- 12-Factor 的应用程序不会隐式依赖系统级的类库
- 它一定通过依赖清单，确切地声明所有依赖项 对于Java来说，maven程序所有依赖都显示定义groupID,artifactID,version
- 在运行过程中，通过**依赖隔离工具**来确保程序不会调用系统中存在但清单中未声明的依赖项

有些包可能goupId换了，或者artifactId换了，就会有不同版本，建议使用扫描工具例如duplicate-finder plugin扫描库里面有没有同名类

服务治理：  
定期对依赖扫描，看看依赖的东西有无变化；  
运行时候通过接口获取maven依赖库的信息，集中管理

# 严格分离构建和运行

构建时候：  
代码扫描；  
加入自动化测试运行；  
加入安全扫描

- 12-Factor 应用严格区分构建、发布、运行三个步骤
- 部署工具通常都提供了发布管理工具
- 每一个发布版本必须对应一个唯一的发布 ID

每次都用完整的包进行发布，便于追溯和管理

发布的过程使用自动化过程

# 以一个或多个无状态进程运行应用

所有的进程都是一样的

- 12-Factor 应用的进程必须**无状态且无共享** 便于扩展
- **任何需要持久化的数据都要存储在后端服务内**

非常昂贵的资源，可以适当的加入一点状态

比如缓存中用户信息很大，序列化开销很大。

可以在JVM内部做一个缓存，而且仅存在特定进程里面。

前端请求根据一定策略做一致性Hash选择。比如根据用户ID进行一致性Hash。

用户请求就会指定落到特定机器上面。这些用户相关的大对象，信息不再需要向后端缓存加载。

一旦这个进程重启或者变化，下一次请求会落到新的机器上，在这台机器上才进行向后端加载

# 快速启动和优雅终止可最大化健壮性

- 进程应当追求最小启动时间
- 进程一旦接收终止信号就会优雅的终止
- 进程应当在面对突然死亡时保持健壮

比如数据库中间件，代理所有SQL请求。  
重启的时候让当前跑的SQL顺利提交，不再建立新的连接。等连接都是0再关闭。  
同样web服务器等到没有请求再下线

## 尽可能的保持开发，预发布，线上环境相同

- 想要做到持续部署就必须缩小本地与线上差异
- 后端服务是保持开发与线上等价的重要部分
- 应该反对在不同环境间使用不同的后端服务

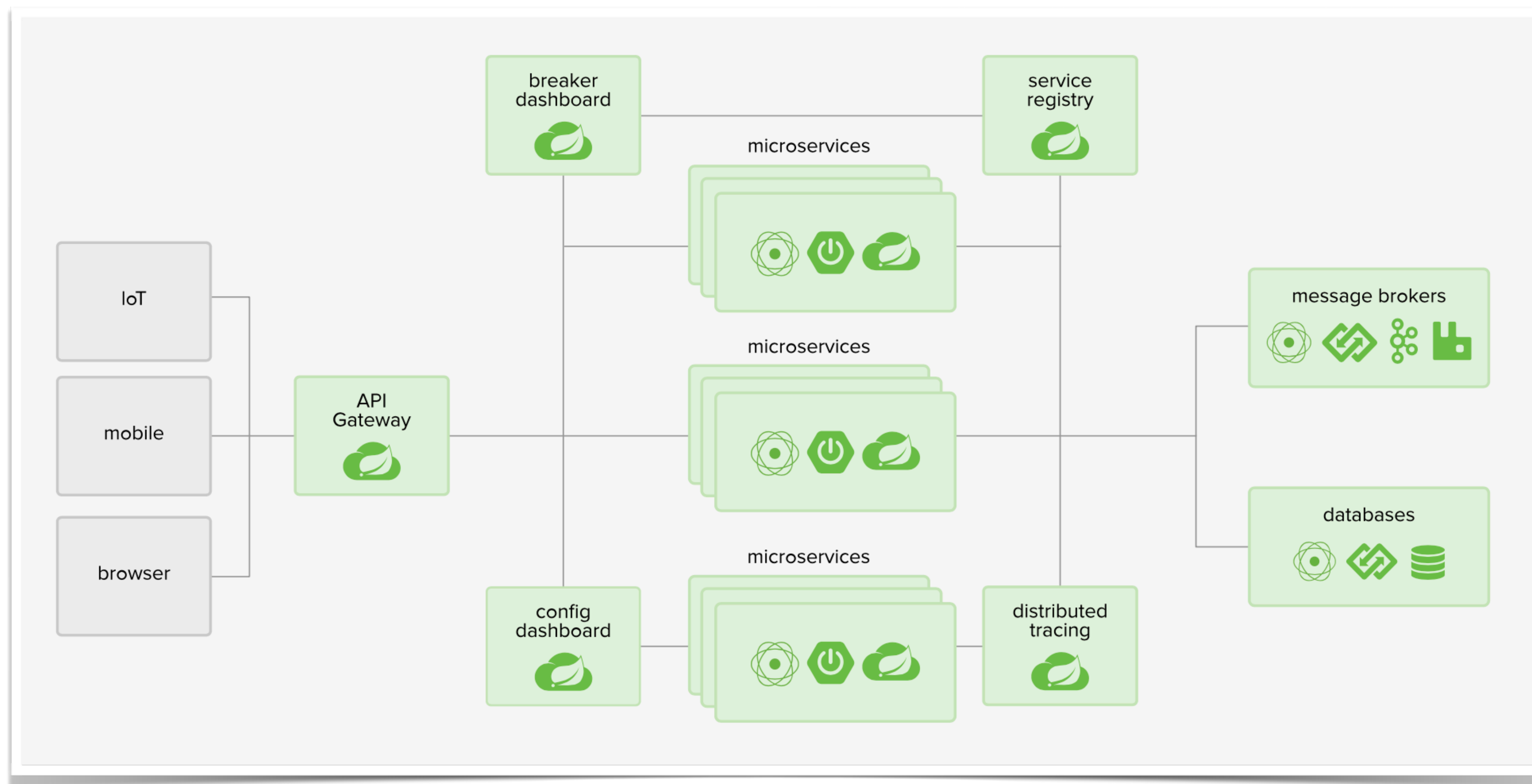


# 认识 Spring Cloud 的组成部分

**“Spring Cloud offers a simple and accessible programming model to the most common distributed system patterns, helping developers build resilient, reliable, and coordinated applications.”**

– *Spring* 官网

# Spring Cloud



# Spring Cloud 的主要功能

- 服务发现
- 服务熔断
- 配置服务
- 服务安全
- 服务网关
- 分布式消息
- 分布式跟踪
- 各种云平台支持

# Spring Cloud 的版本号规则

- Spring Cloud 是个大工程，其中包含多个独立项目
- BOM - Release Train
  - London Tube Stations
  - 字母序排列
    - Greenwich, Finchley, Edgware ...
- SR - Service Release



扫码试看/订阅  
《玩转 Spring 全家桶》