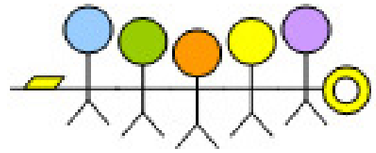
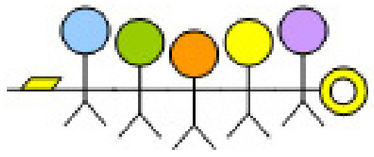


火龙果•整理
uml.org.cn

微服务架构分享

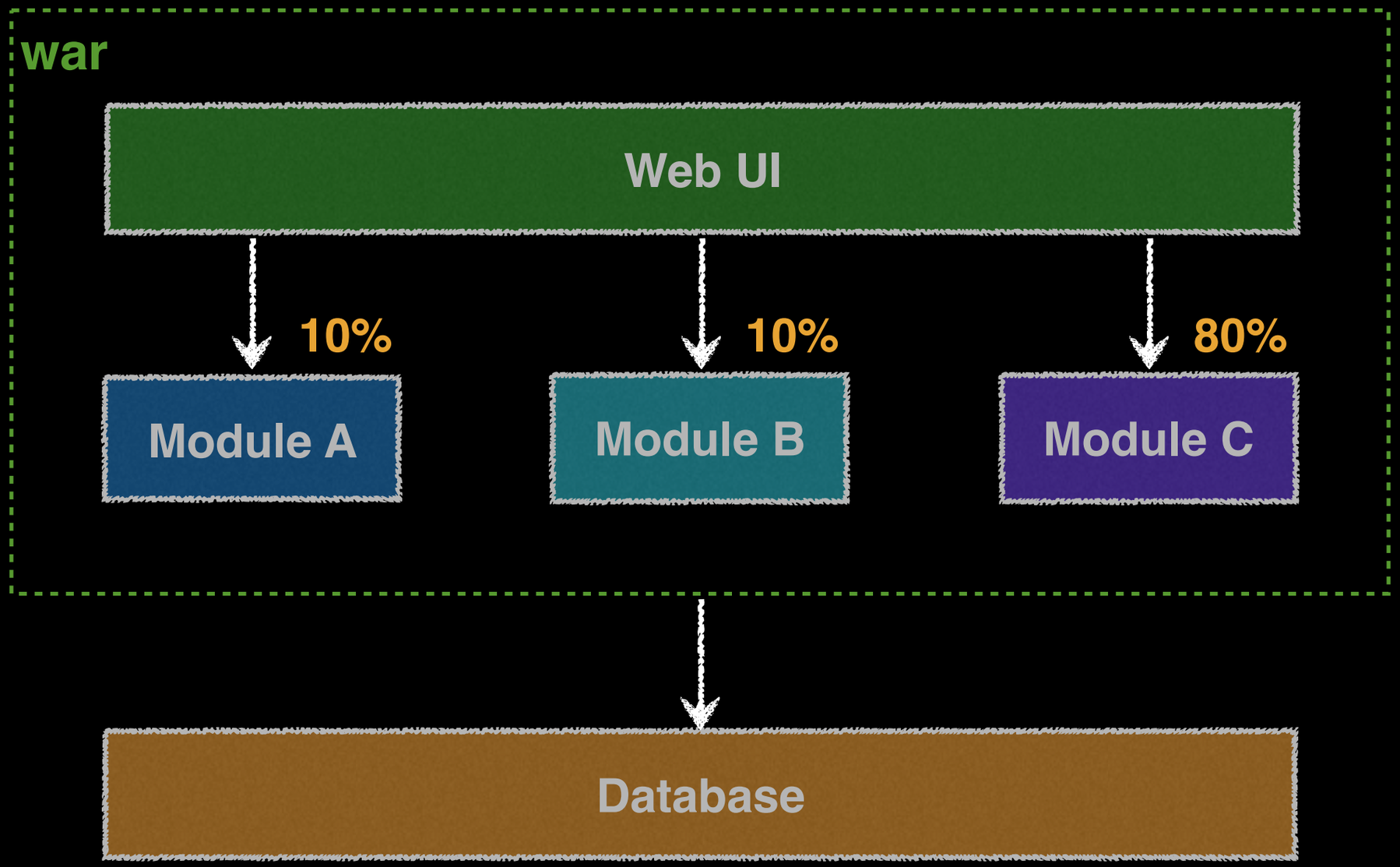
内容简介


微服务架构分享
1 为什么需要微服务架构?
2 微服务架构是什么?
3 微服务架构有哪些特点?
4 如何搭建微服务架构?
5 我开源的微服务框架
6 微服务架构应用案例



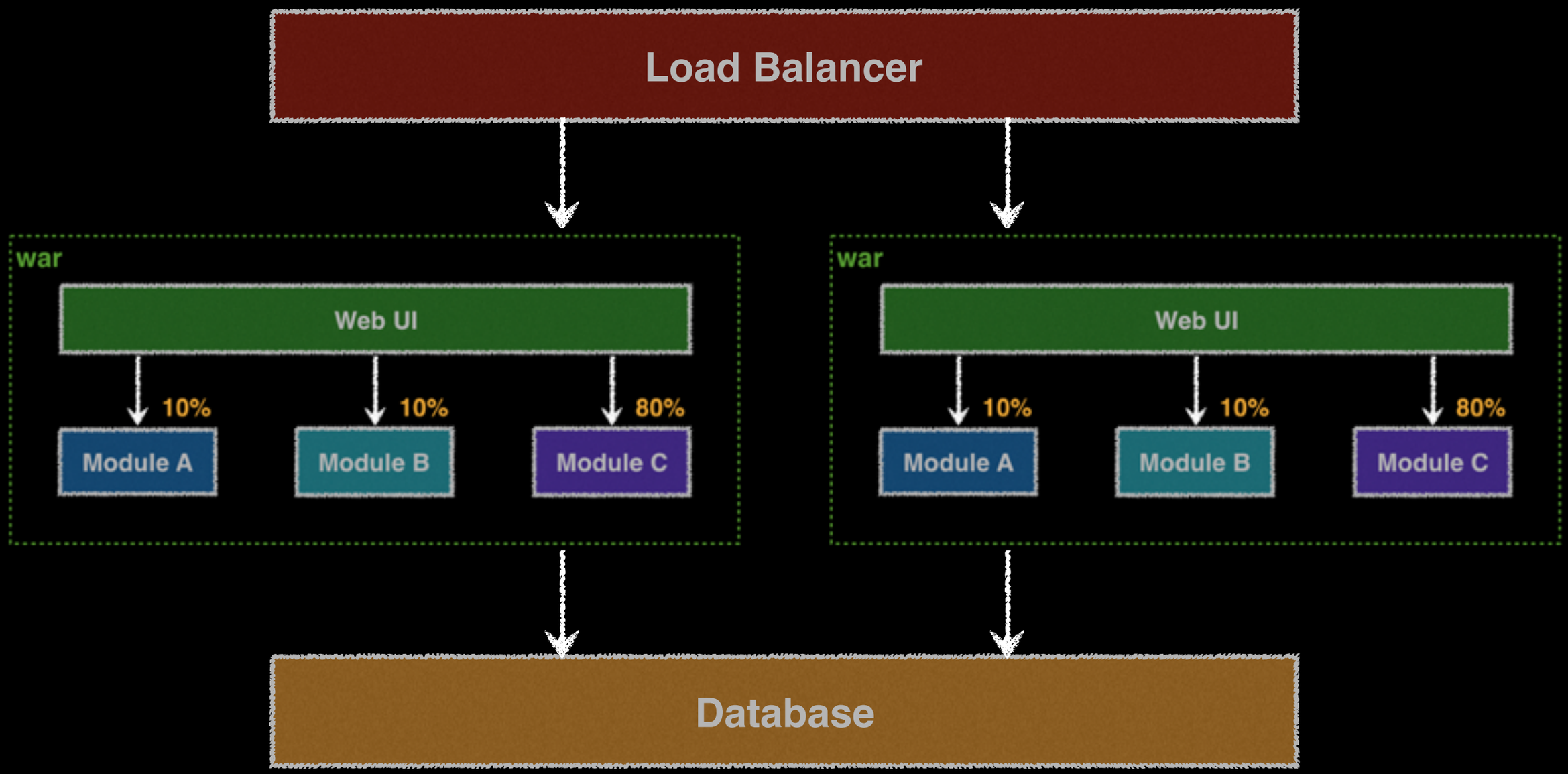
1.为什么需要微服务架构?

传统应用架构

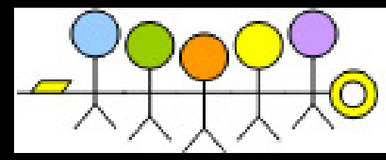


Module C 系统资源占用率过高，整个应用需要水平扩展

水平扩展



水平扩展后，Module A 与 Module B 产生了系统资源浪费



传统应用架构的问题

- 系统资源浪费

  水平扩展带来了资源浪费

- 部署效率太低

  每修改一个模块，都需要部署整个系统

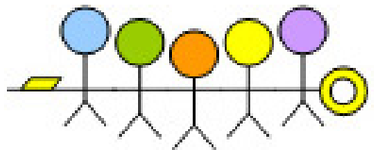
- 技术选型单一

  每个模块拥有相同的技术选型

微服务架构解决方案

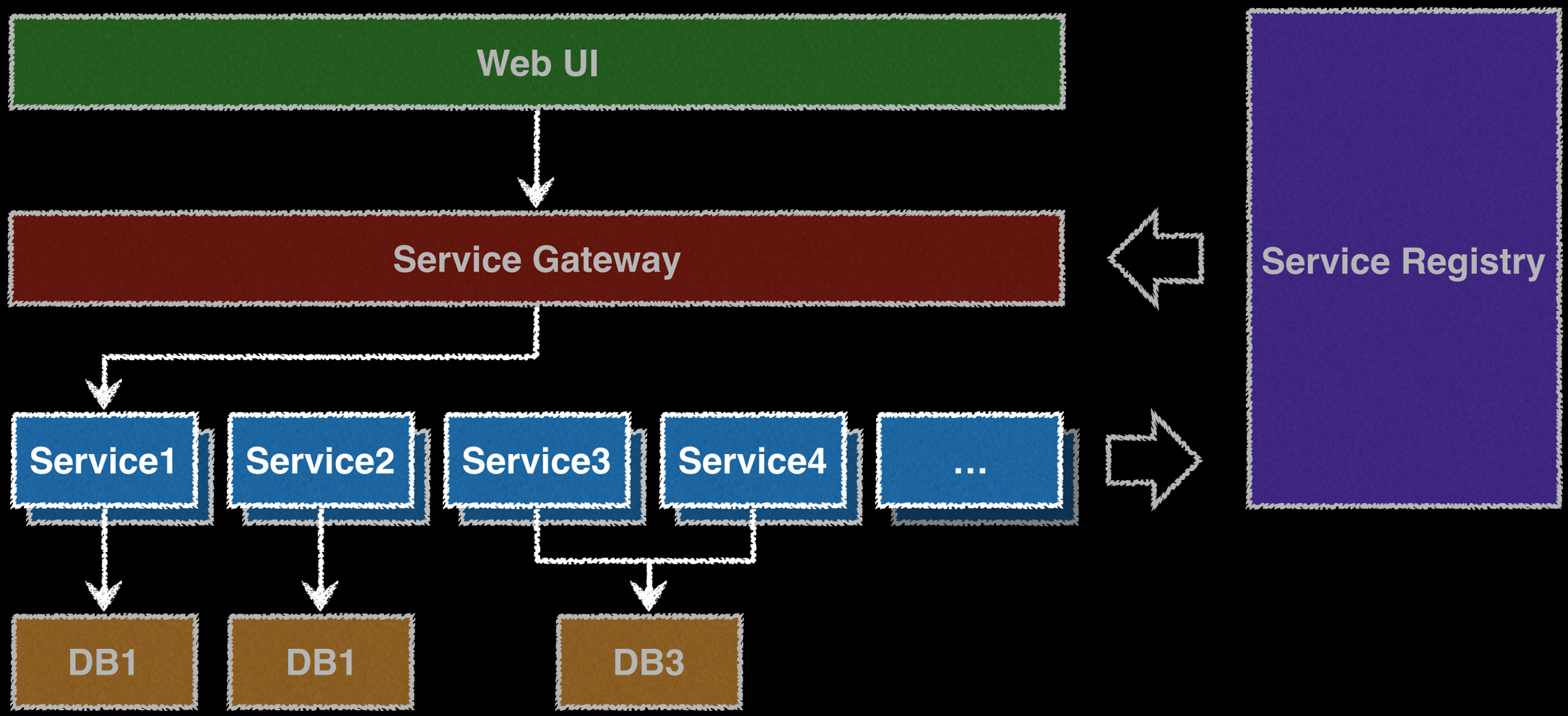
1. 根据业务模块划分服务种类
2. 每个服务可独立部署且相互隔离
3. 服务之间通过轻量级 API 进行通信
4. 服务需保证良好的高可用性

ThoughtWorks 首席科学家
Martin Fowler (福勒)

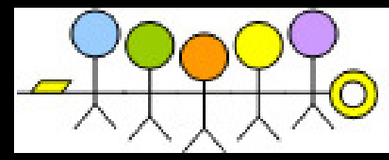


2.微服务架构是什么?

微服务架构参考图

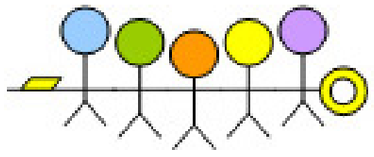


服务注册 → 服务发现 → 服务调用

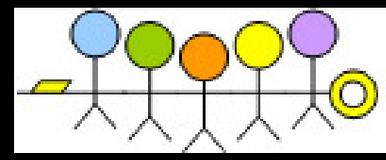


微服务架构工作流程

- 设计阶段
  - 将产品功能拆分为若干服务
  - 为每个服务设计 API 接口（REST 方式）
- 开发阶段
  - 实现 API 接口（包括单元测试）
  - 开发 UI 原型（mock 数据）
- 测试阶段
  - 前后端集成
  - 验证产品功能
- 部署阶段
  - 发布预发环境
  - 发布生产环境

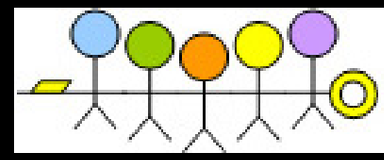


3.微服务架构有哪些特点?



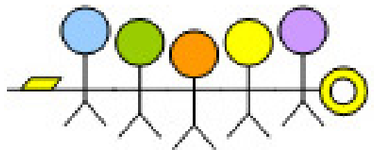
微服务架构的特点

- 粒度微小
  - 根据业务功能划分服务粒度
- 责任单一
  - 每个服务只做一件事情（单一职责原则）
- 隔离性好
  - 每个服务相互隔离且互不影响
- 管理容易
  - 自动化部署与监控预警



微服务架构的挑战

- 运维要求较高
  - 系统监控、高可用性、自动化技术
- 分布式复杂性
  - 网络延迟、系统容错、分布式事务
- 部署依赖性较强
  - 服务依赖、多版本问题
- 服务间通讯成本较高
  - 无状态性、进程间调用



4.如何搭建微服务架构?

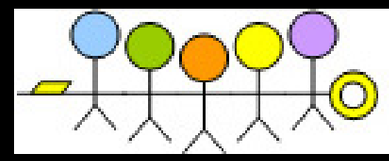
微服务架构技术选型

- 服务框架：Spring Boot

- 服务容器：Docker

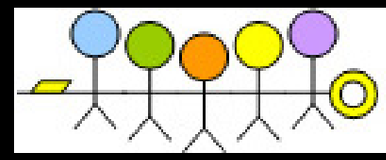
- 服务注册：ZooKeeper

- 服务网关：Node.js



微服务架构使用者

Amazon，eBay，Twitter，Netflix

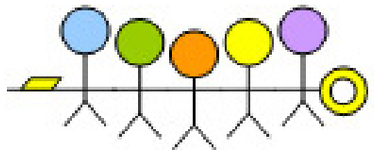


Netflix 微服务框架

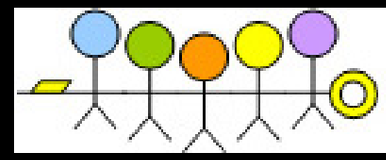
- 服务注册发现框架：Eureka
- 服务网关：Zuul
- 服务端框架：Karyon
- 客户端框架：Ribbon
- 服务容错组件：Hystrix
- 服务配置组件：Archaius
- Metrics 组件：Servo
- 日志组件：Blitz4j

Spring Cloud 解决方案

- 基于 Netflix 开源框架
- 集成 Spring Boot
- 核心功能
  - 服务配置管理：Configuration Service
  - 服务注册与发现：Discovery Service
  - API 网关：API Gateway

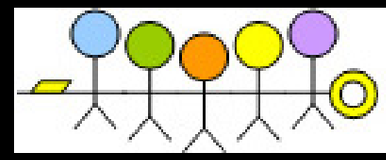


5.我开源的微服务框架



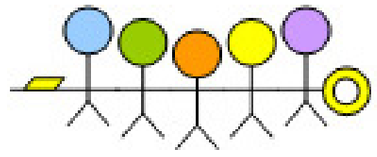
分布式 RPC 框架

- Spring
  - IOC 与 AOP
- Netty
  - 简化 NIO 编程模型
- Protostuff
  - 基于 Protobuf 序列化，基于 POJO，无需编写 .proto 文件
- ZooKeeper
  - 服务注册与服务发现，天生的集群能力



REST API 框架

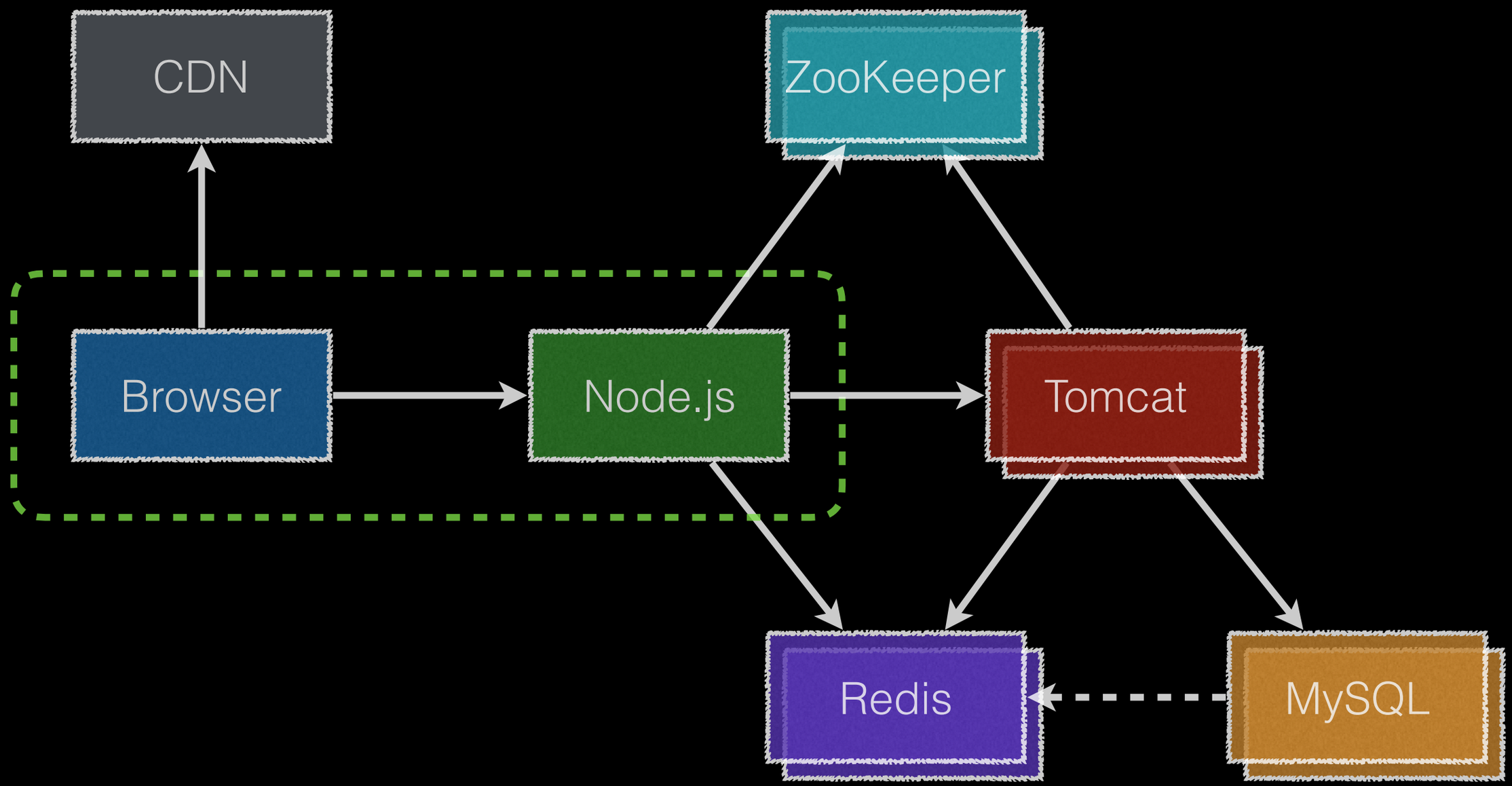
- Spring MVC
  - 基于注解方式发布 REST 服务
- MyBatis
  - 将 SQL 脚本与 Java 代码分离
- Redis / Jedis
  - 分布式 token 安全机制



6.微服务架构应用案例

特赞微服务架构

特赞系统架构



分支管理规范