Cloud Native 云化架构

陈皓

MegaEase Inc.

个人简介

- **18年工作经历，超大型分布式系统基础架构研发和设计**
- **擅长领域：金融、电子商务、云计算、大数据**
- **职业背景**
  - 阿里巴巴资深架构师（阿里云、天猫、淘宝）
  - 亚马逊高级研发经理（AWS、全球购、商品需求预测）
  - 汤森路透资深架构师（实时金融数据处理基础架构）
- **目前在创业**
  - 2015-2016为40+公司提供过技术咨询服务
  - 2016-至今，致力于为企业提供高并发、高可用的IT服务保障

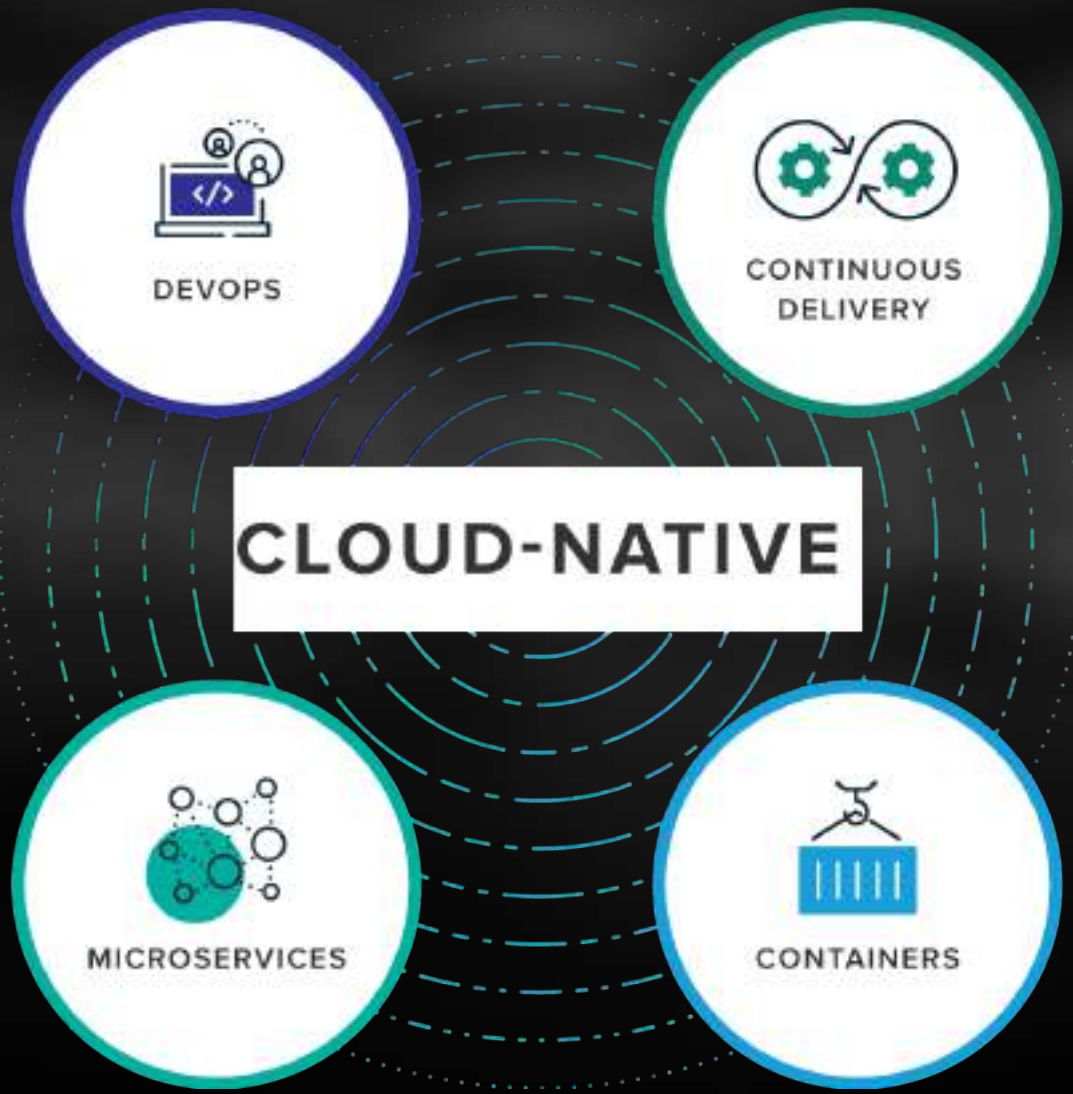
MegaEase

大纲

- Cloud Native
- 如何提高性能和稳定性
- 分布式服务化架构的关键技术
- PaaS平台的核心
- MegaEase的产品介绍

Cloud Native 是什么

MegaEase


DEVOPS
CONTINUOUS DELIVERY
CLOUD-NATIVE
MICROSERVICES
CONTAINERS

DEVOPS

BEFORE

Not my problem

Separate tools, varied incentives, opaque process

AFTER

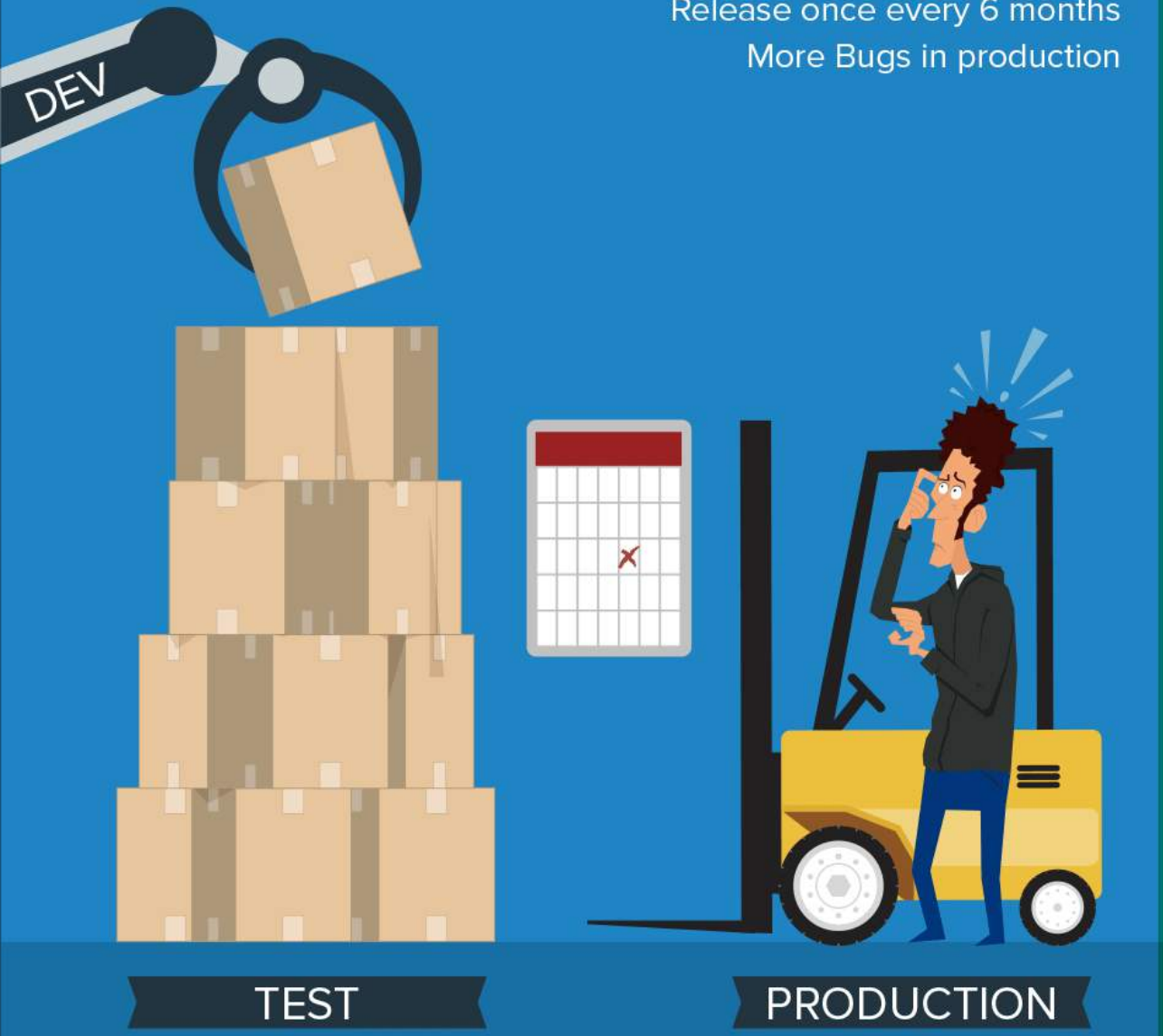
Shared responsibility

Common incentives, tools, process and culture

CONTINUOUS DELIVERY

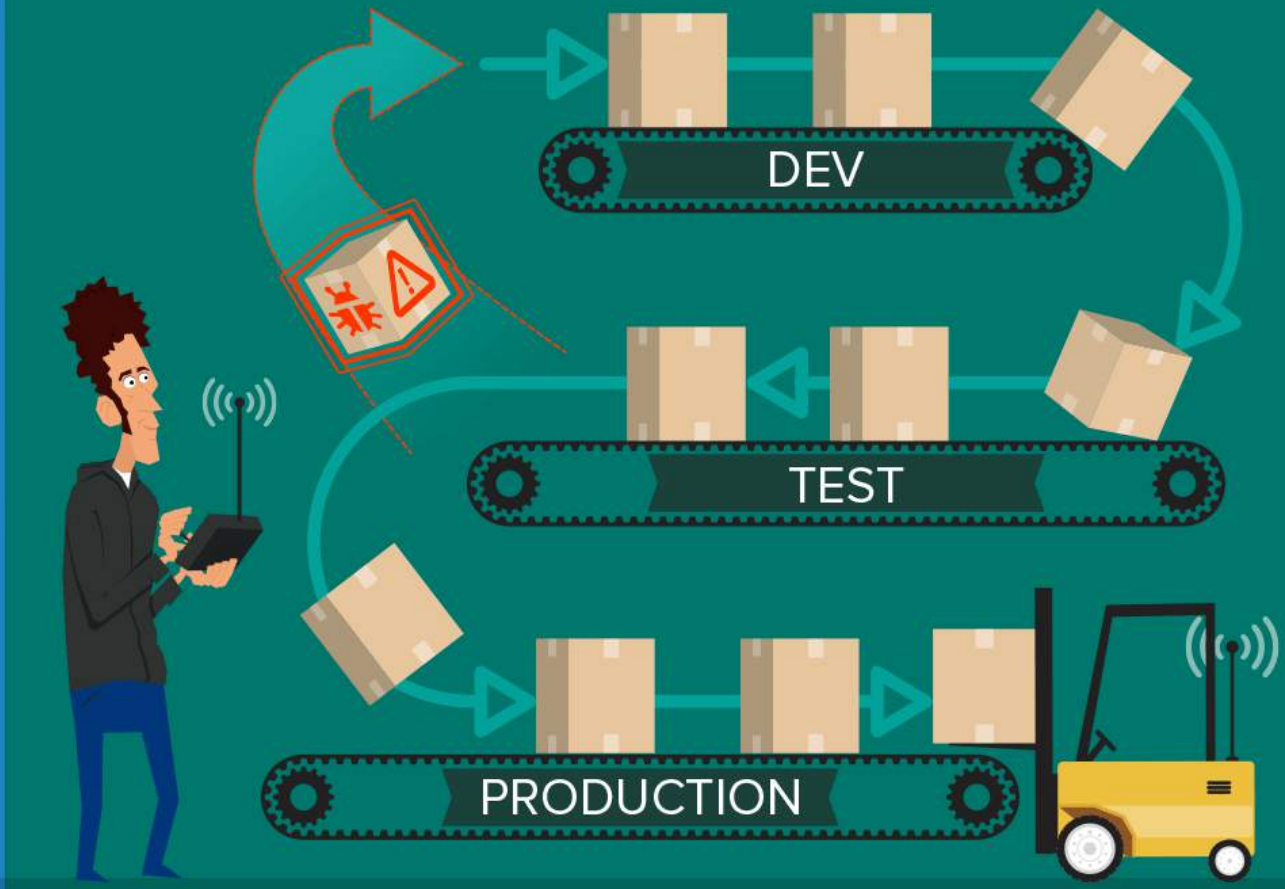
BEFORE

Release once every 6 months
More Bugs in production



AFTER

Release once a week
Higher Quality of Code



MICROSERVICES

BEFORE

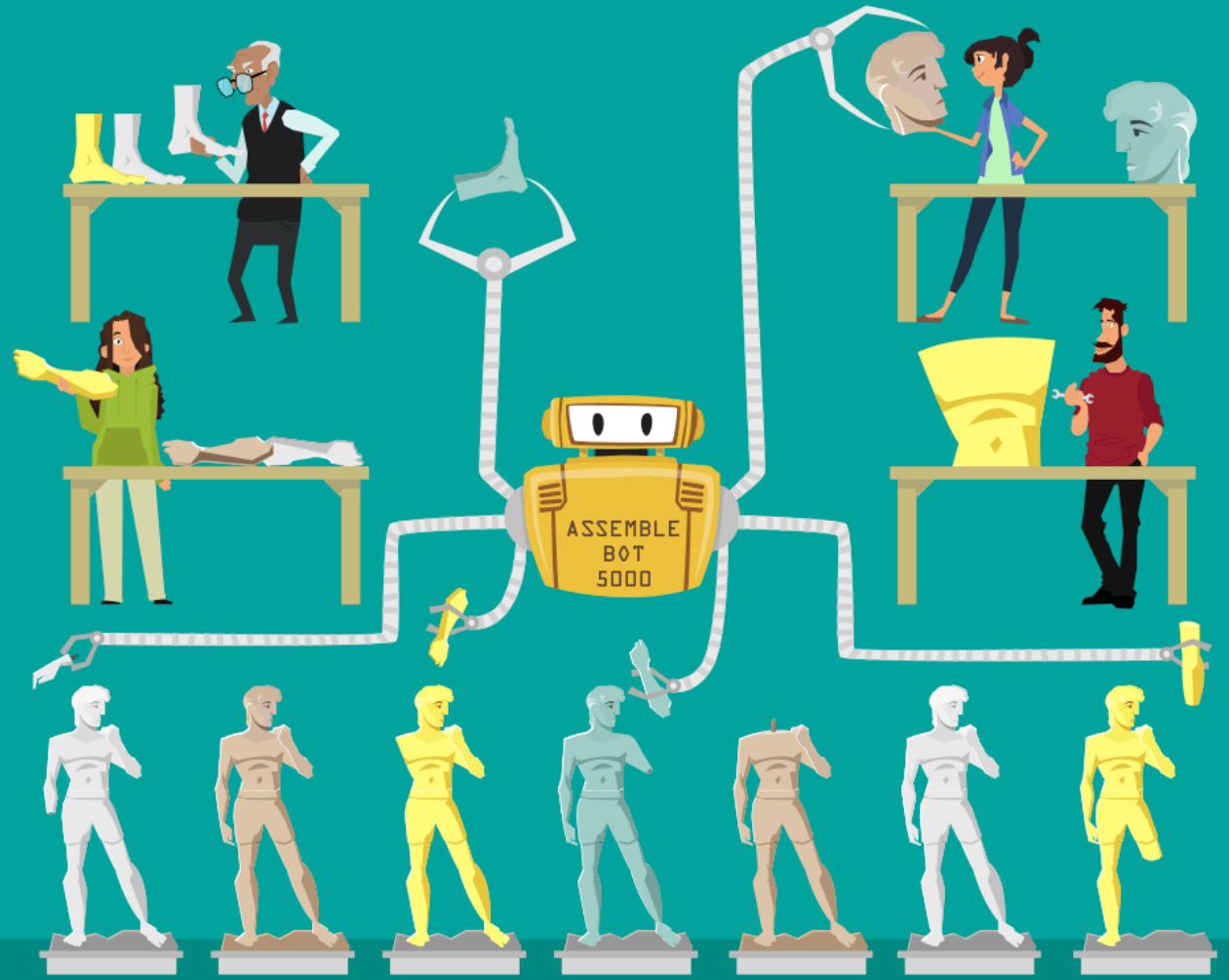
Tightly coupled components

Slow deploy cycles waiting on integrated tests and teams

AFTER

Loosely coupled components

Automated deploy without waiting on individual components

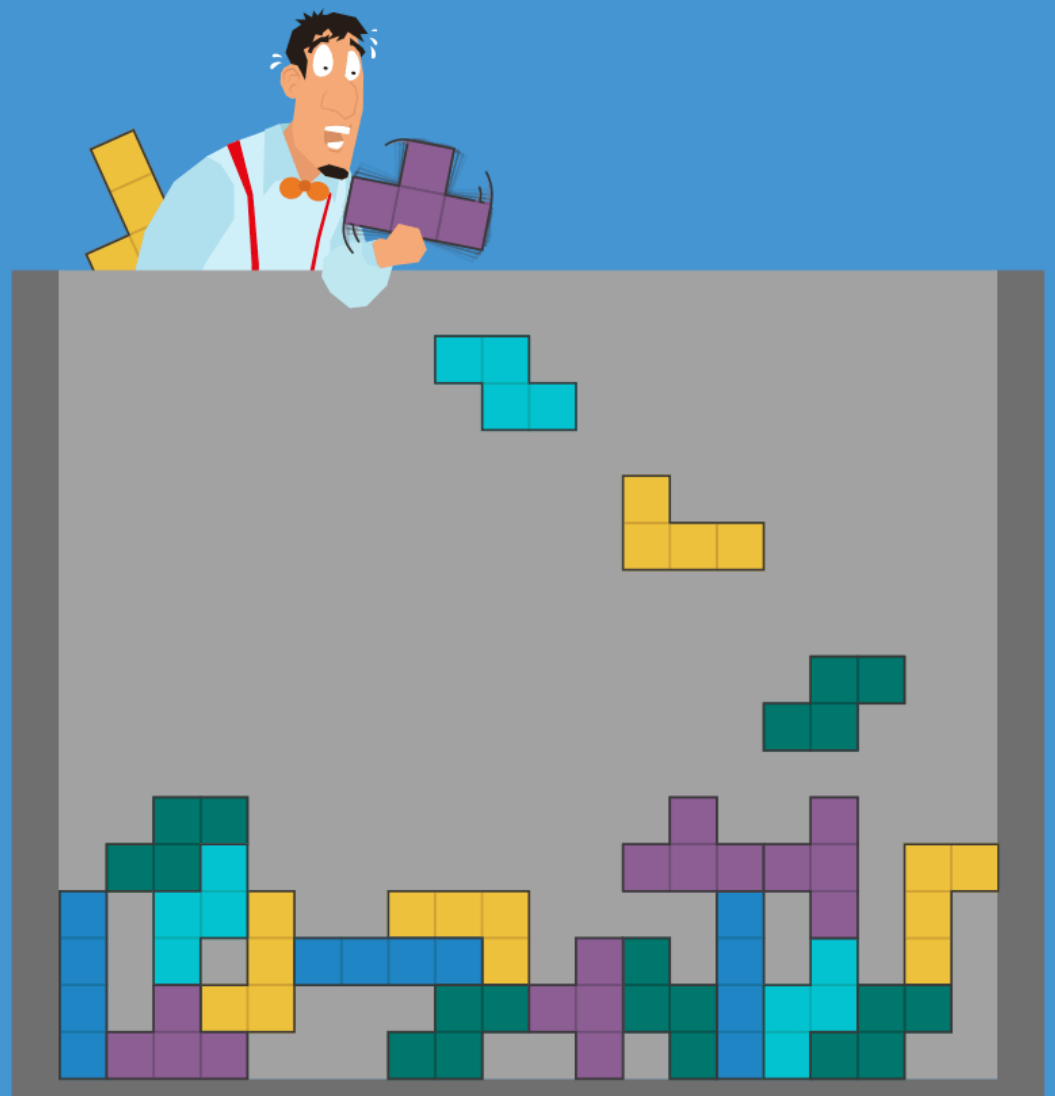


CONTAINERS

BEFORE

Make sure software matches with hardware

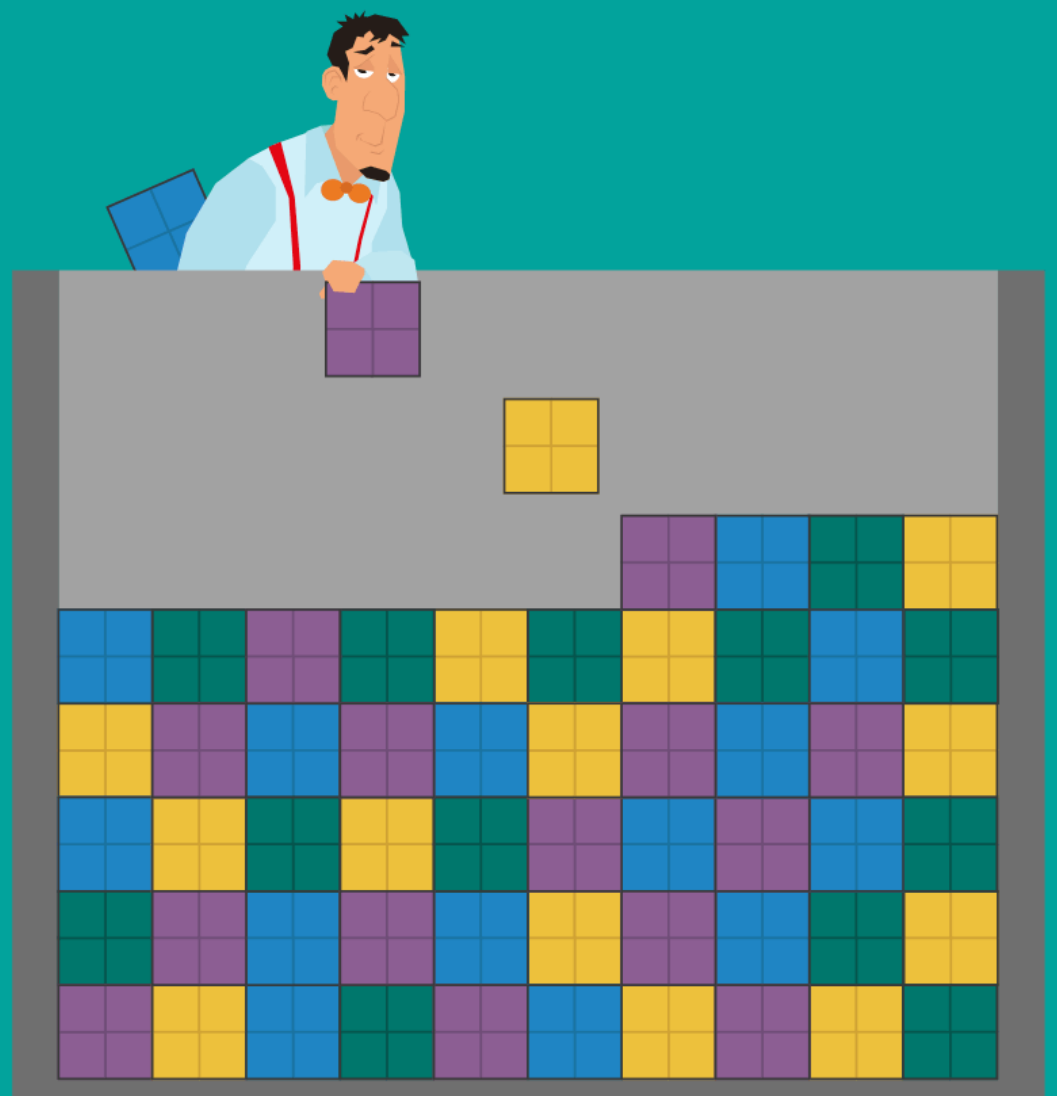
Build snowflakes for each environment



AFTER

Build once, run anywhere

A shared construct between software and hardware



需要面对的技术有哪些

分布式系统的问题

	传统单体架构	分布式服务化架构
新功能开发	新功能开发需要时间	容易开发和实现
部署	不经常且容易部署	经常发布，部署复杂
隔离性	故障影响范围大	故障影响范围小
系统性能	响应时间快，吞吐量小	响应时间慢，吞量大
系统运维	运维简单	运维复杂
新人上手	学习曲线大（应用逻辑）	学习曲线大（架构逻辑）
技术	技术单一且封闭	技术多样且开放
测试	简单	复杂
系统扩展性	扩展性很差	扩展性很好
系统管理	重点在于开发成本	重点在于服务治理和调度

Cloud Native 的相关技术

DevOps

容器化调度

微服务

服务编排

服务治理

异地多活

应用监控

高性能高并发

高可用

自动化运维

其实就是解决这几件事

快速交付

高扩展性

高可用性

提高性能

加缓存

缓存系统

缓存分区
缓存更新
缓存命中

负载均衡

网关系统

负载均衡
服务路由
服务发现

异步调用

异步系统

消息队列
消息持久
异步事务

数据镜像

数据镜像

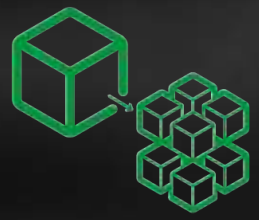
数据同步
读写分离
数据一致性

数据分区

数据分区

分区策略
数据访问层
数据一致性

提高稳定性



服务拆分

服务冗余

限流降级

高可用架构

高可用运维

服务治理

服务调用
服务依赖
服务隔离

服务调度

弹性伸缩
故障迁移
服务发现

限流降级

异步队列
降级控制
服务熔断

高可用架构

多租户系统
灾备多活
高可用服务

运维系统

全栈监控
DevOps
自动化运维

MegaEase

我们要做多少事?

- **高性能处理**
  - 缓存、弹性伸缩、异步处理、数据复制……
- **关键业务保护**
  - 高可用、故障隔离、业务降级……
- **流量控制**
  - 负载均衡、服务路由、熔断、降级……
- **整体架构监控**
  - 三层系统监控（应用层、中间件层、基础层）
- **DevOps**
  - 环境构建、持续集成、持续部署
- **架构管理**
  - 架构版本、生命周期管理，服务管理……
- **自动化运维**
  - 自动伸缩、故障迁移、配置管理，状态管理……
- **基础资源调度管理**
  - 计算、存储、网络资源调度和管理

然而还没完……

这些东西都不是功能性需求

- 不是功能性需求，容易被忽略
- 属于基础设施，不容易被理解

技术含量高，各种技术坑

- 解决了一个问题，新增多个问题
- 需要投入的时间和人力成本极高

好的技术人员越来越难招

- 能做好这些软件的人并不多
- 好的技术人员，对工作极其挑剔

MegaEase

如何面对如此纷乱的技术

MegaEase

怎么面对呢?

- **张开一个一个的网眼?**
  - 如果你是一个一个的去做，你就是在使蛮力
  - 一个一个的做，你会发现连不起来。
- **张开一个大网，需要找到“纲”**
  - 什么才是这个渔网的“纲”？
  - 如何才能做到“纲举目张”？