GOTC

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

OPEN SOURCE , OPEN WORLD

「CNCF云原生」专场

本期议题：云原生落地实践

景显强 2021年08月01日

议题

GOTC

目录

- 为什么要云原生
- 云原生的知识体系
- 云原生平台的构建原则
- 云原生应用落地的最佳实践

为什么要云原生

GOTC

云原生的概念

云原生

定义

云原生一种将软件应用程序构建为微服务并在容器化和动态编排的平台上运行它们的方法，以利用云计算模型的优势。

特点

云原生关心的是如何创建和部署应用程序，而不是把应用放在哪里。（公有云、私有云、混合云）

云原生的代表技术包括容器、服务网格、微服务、不可变基础设施和声明式API等

云原生是采用微服务开发模式，利用容器等技术，在云平台上运行的方法

为什么要云原生

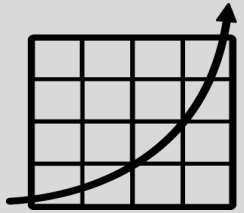
数字化转型的基础要素

云原生能更好的支撑企业数字化转型

为什么要云原生

GOTC

为什么采用云原生

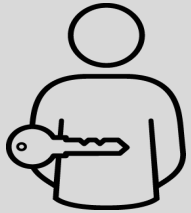


企业数字化转型的必然趋势

- 数字化时代的产物
- IT系统转型的最佳选择
- 降本增效

企业市场竞争压力的驱使

- 快速抢占业务市场
- 互联网大潮下推动



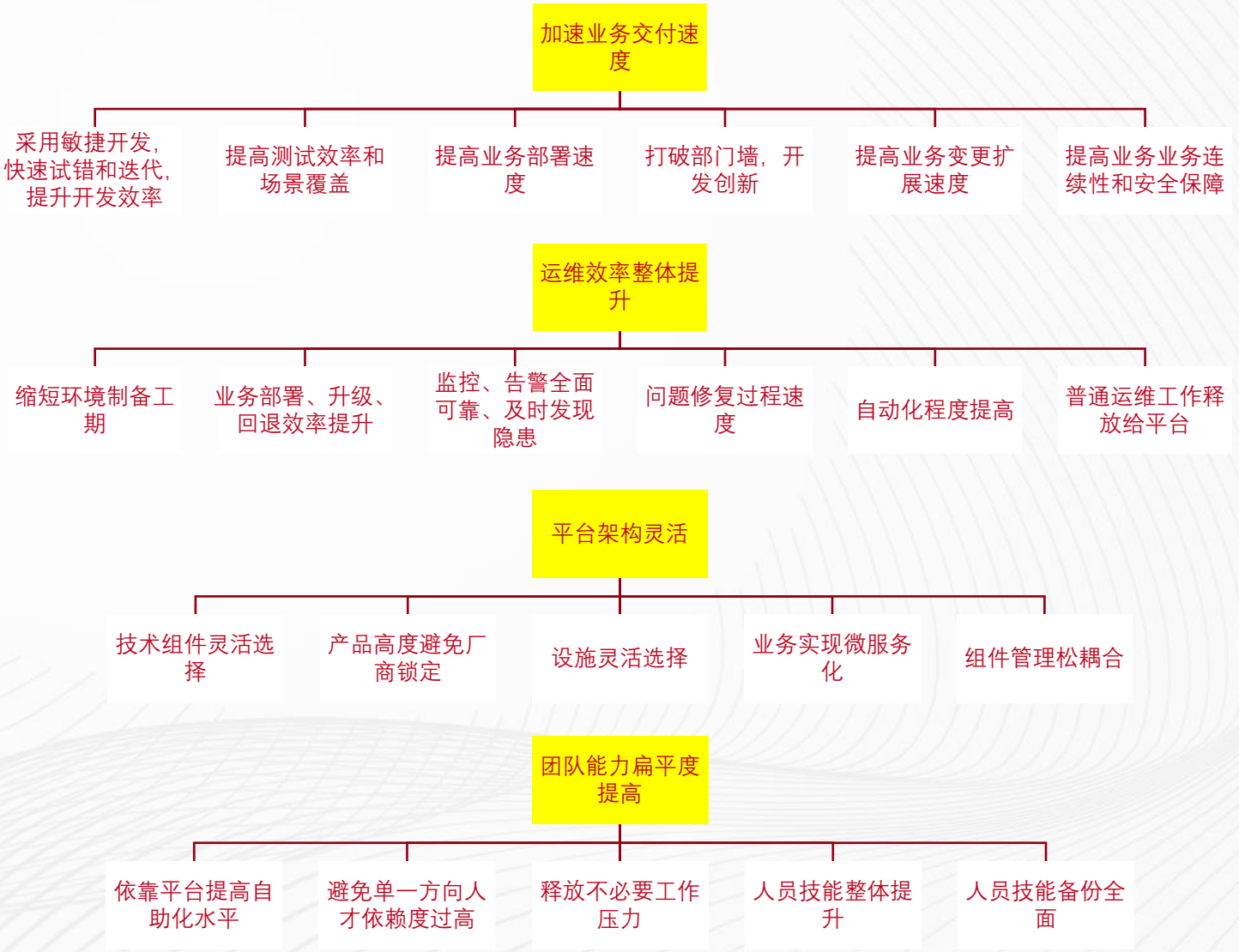
安全高效运行业务的动力驱使

- 快速上线、变更业务
- 不间断提供业务
- 高效扩展业务
- 业务在混合云下灵活迁移

为什么要云原生

GOTC

云原生带来的优势



议题

GOTC

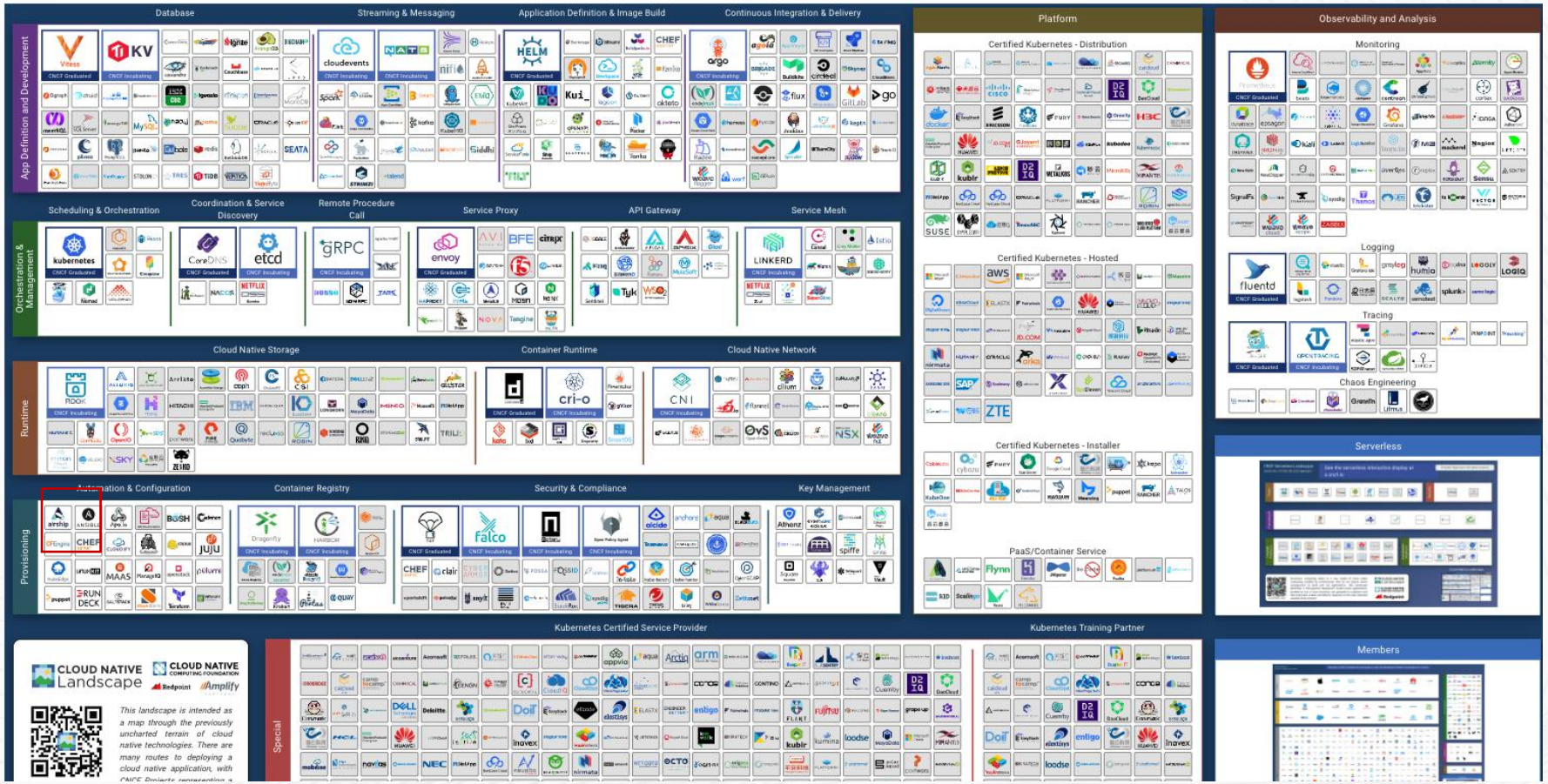
目录

- 为什么要云原生
- 云原生的知识体系
- 云原生平台的构建原则
- 云原生应用落地的最佳实践

云原生的知识体系

GOTC

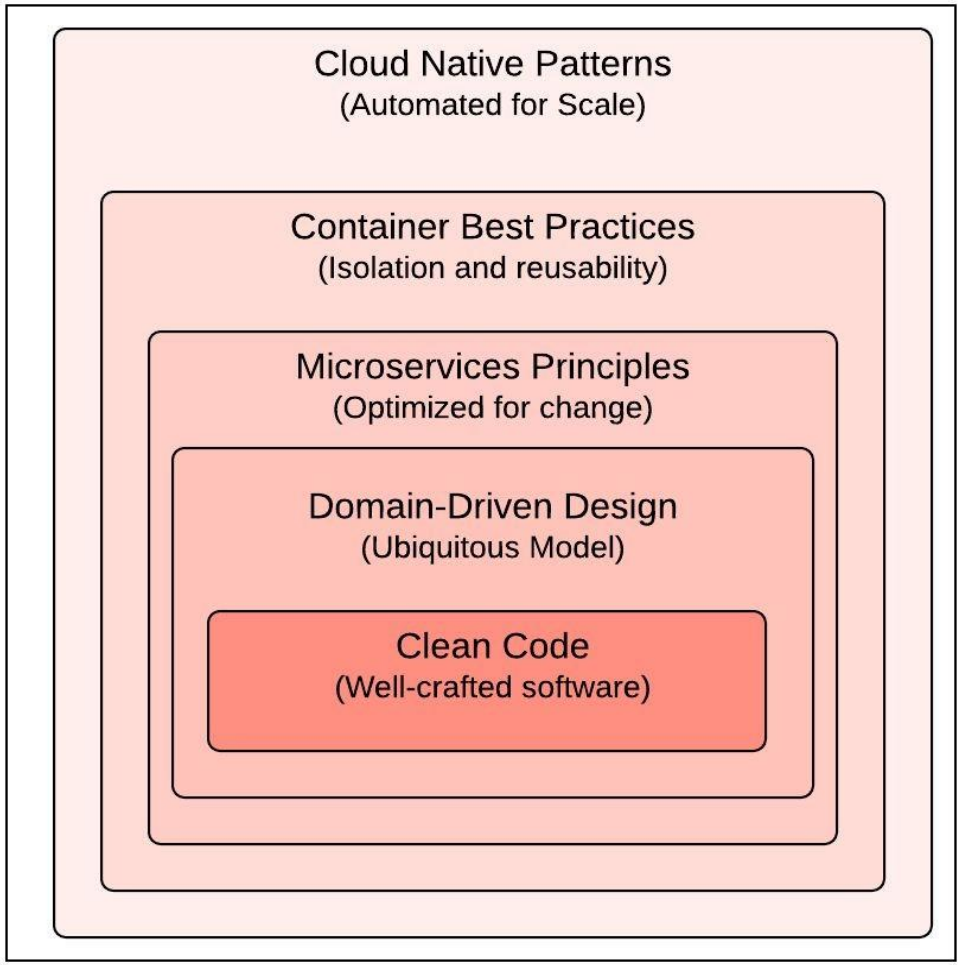
CNCF的云原生技术栈



云原生的知识体系

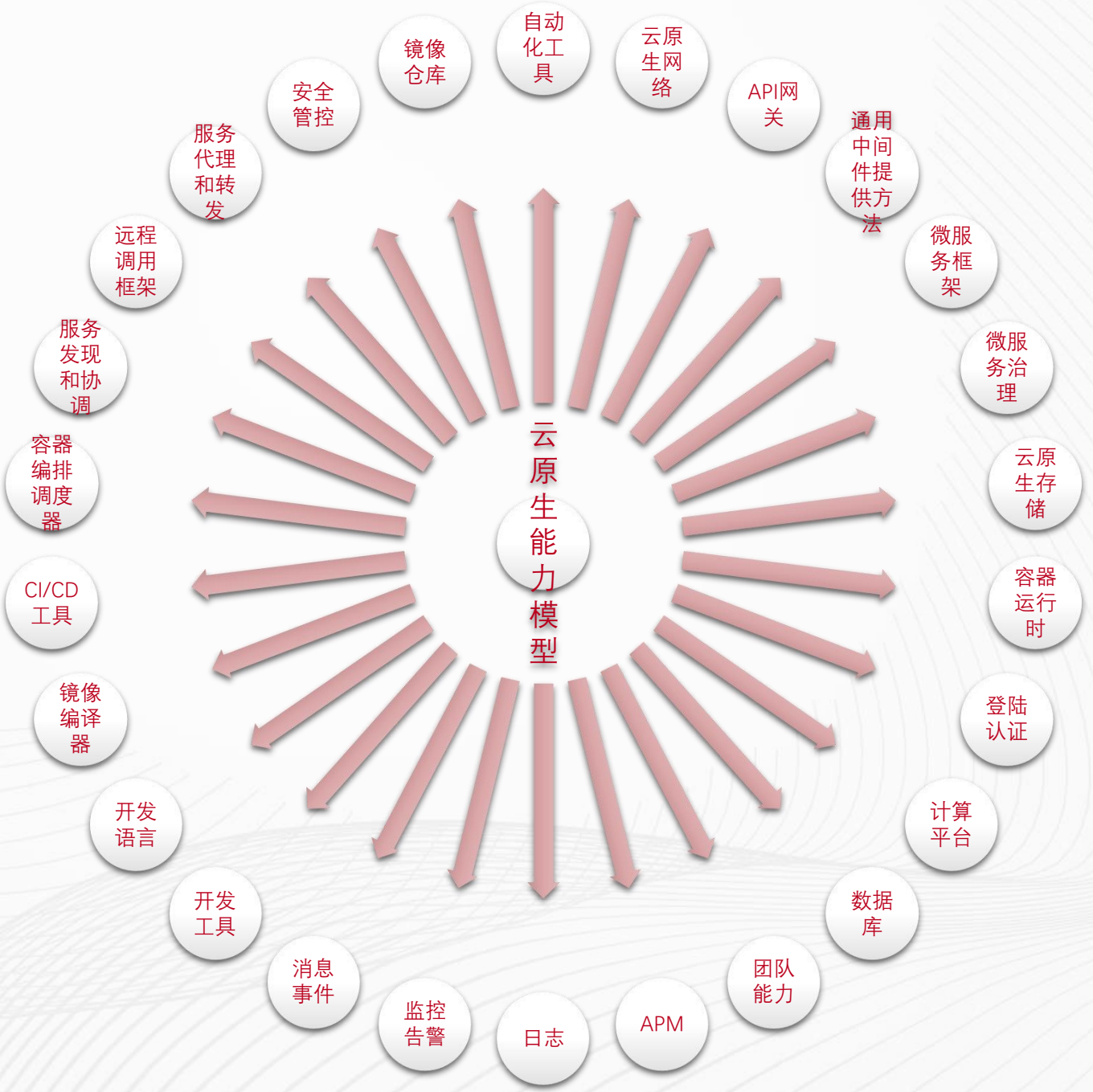
云原生体系模式

- 体系的透视图很复杂
- 不同层次复杂程度各有不同
- 每个层次需要各自不同的技能



云原生的知识体系

云原生的能力模型26条



议题

GOTC

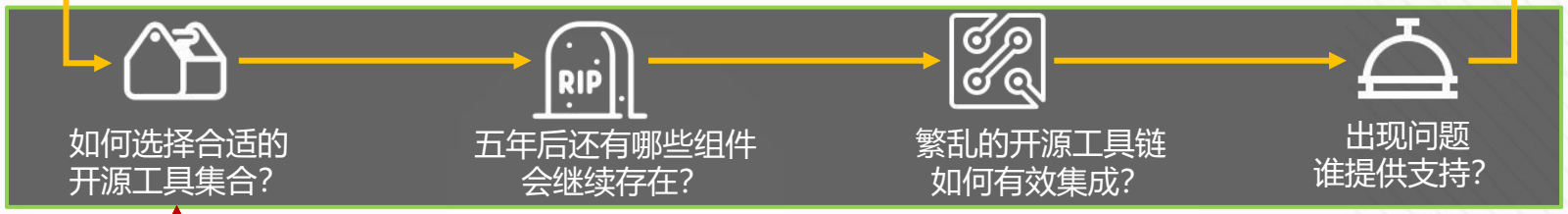
目录

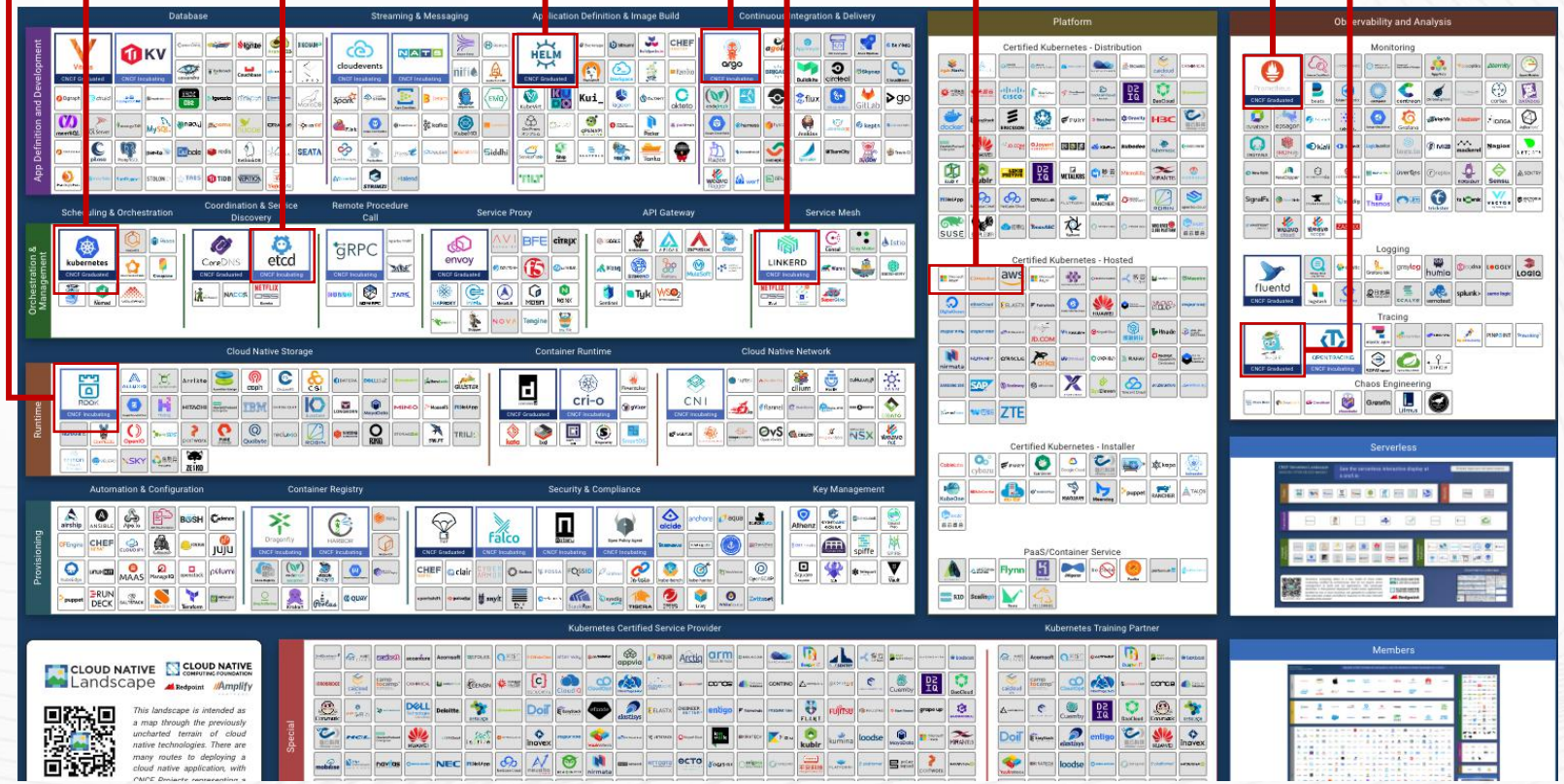
- 为什么要云原生
- 云原生的知识体系
- 云原生平台的构建原则
- 云原生应用落地的最佳实践

云原生平台的构建原则

开源技术的选型

持续演进变化（持续DIY将消耗大量资源）





云原生平台的构建原则

GOTC

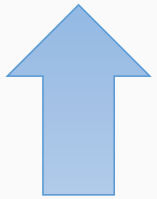
构建云原生平台考虑的非技术因素

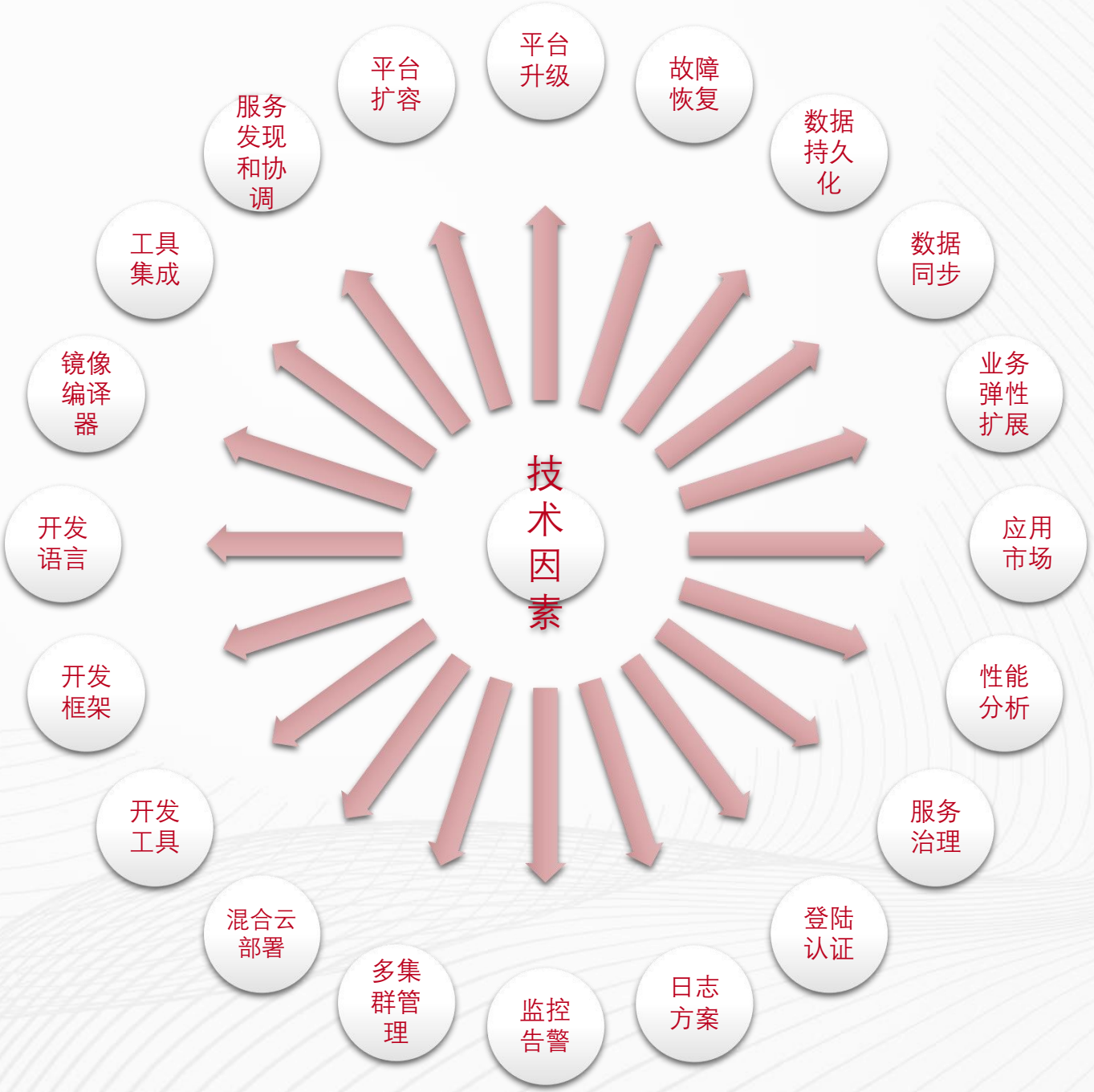
- 现在已经不是用不用开源软件的问题，而是怎么用好开源软件的问题了。
- 如果在使用了开源软件以后，没有持续的管理流程和平台，会导致没有办法修复安全漏洞，导致数据丢失。
- 开源软件很好，但是要好好地用，需要开源治理。

--- 某银行云原生负责人

云原生平台的构建原则

构建云原生平台考虑的技术因素





云原生平台的构建原则

GOTC

基于CNCF自主研发
的正确打开方式

参照平台如何选择？

- 先进性+演进性
- 开放性+可借鉴性
- 安全性+稳定性

100%自主研发云原生 Portal（自主知识产权）

业界领先的100%开源企业级多集群管理工具

先进知识转化

基于社区版自主研发的多集群管理工具（自主知识产权）

业界领先的100%开源企业级云原生平台

- ✓ 自主研发参照
- ✓ 自主研发平台备份
- ✓ 重要系统承载（第三方兜底）
- ✓ 双循环交叉点+外循环基础

先进知识转化

应用无缝漂移

基于社区版自主研发的云原生平台（自主知识产权）

- ✓ 100%实现自主可控
- ✓ 基于最优实现超越
- ✓ 承载大部分生产系统
- ✓ 内循环坚实基础

100%自主研发云原生套装

赋能服务如何选择？

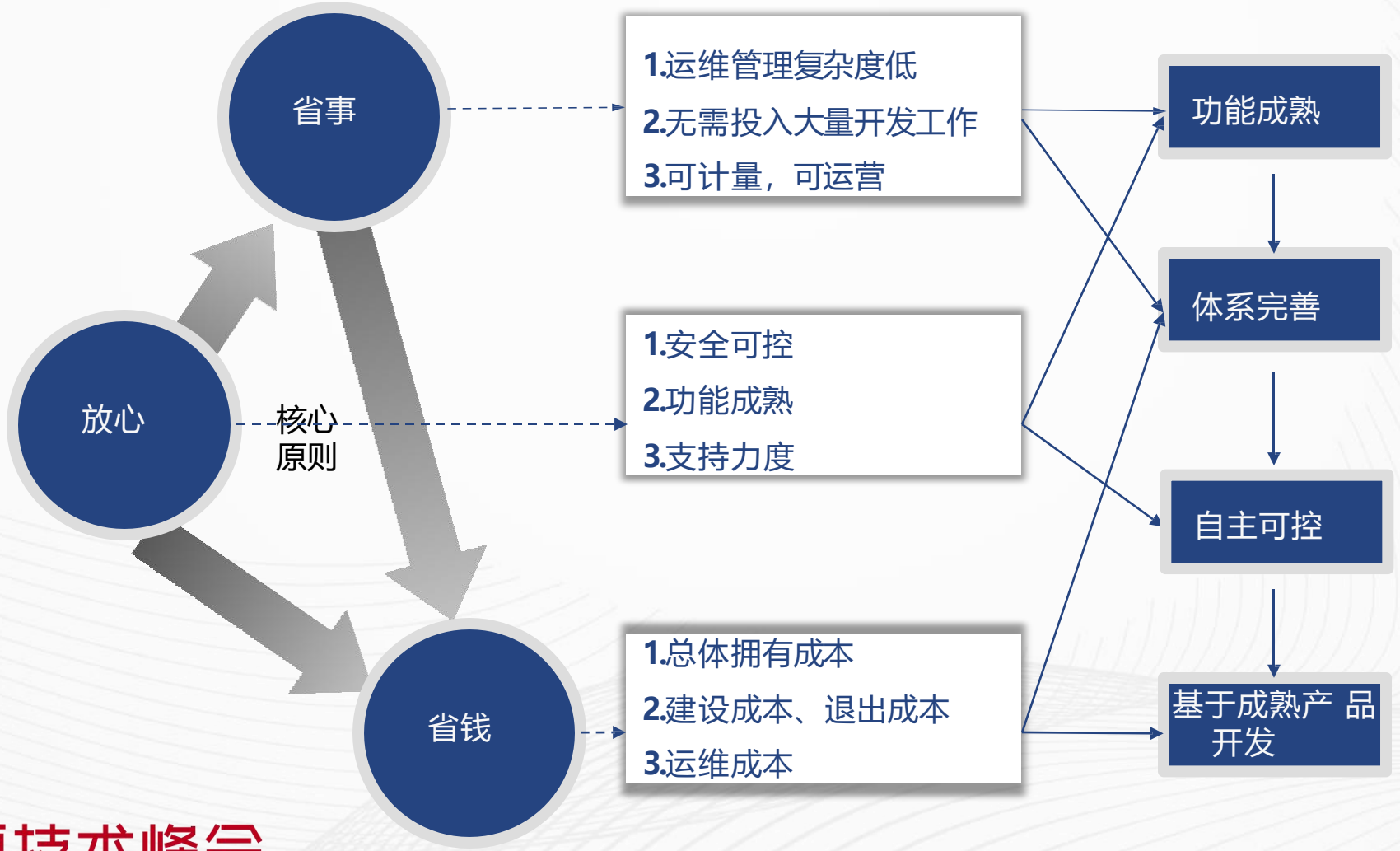
- 先进性+权威性
- 手把手教
- 不需要一直投入

开放的云原生赋能为自主研发保驾护航

云原生平台的构建原则

GOTC

构建原则



议题

GOTC

目录

- 为什么要云原生
- 云原生的知识体系
- 云原生平台的构建原则
- 云原生应用落地的最佳实践

云原生应用落地的最佳实践

GOTC

红帽的云原生方案技术选择

云原生应用落地的最佳实践

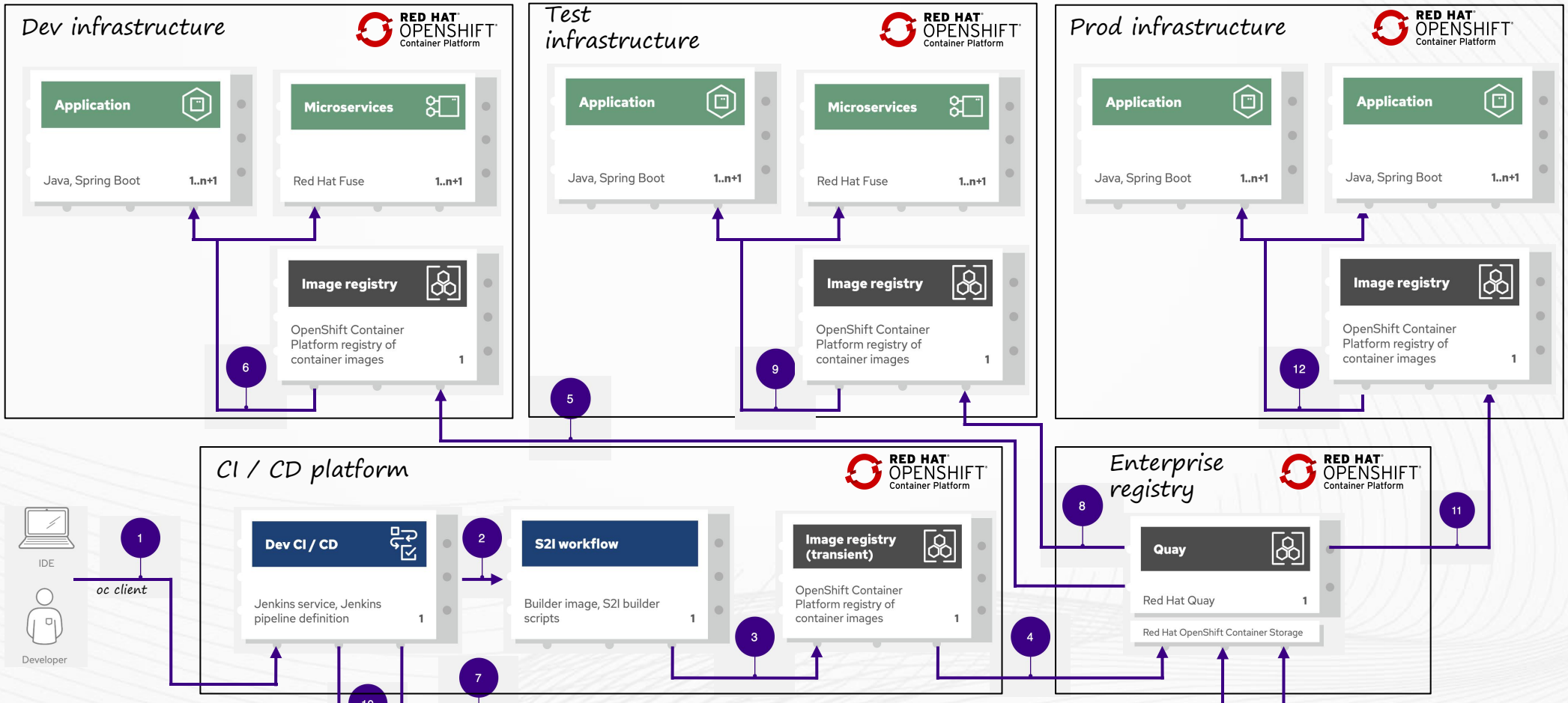
GOTC

基于红帽的云原生方案

云原生应用落地的最佳实践

GOTC

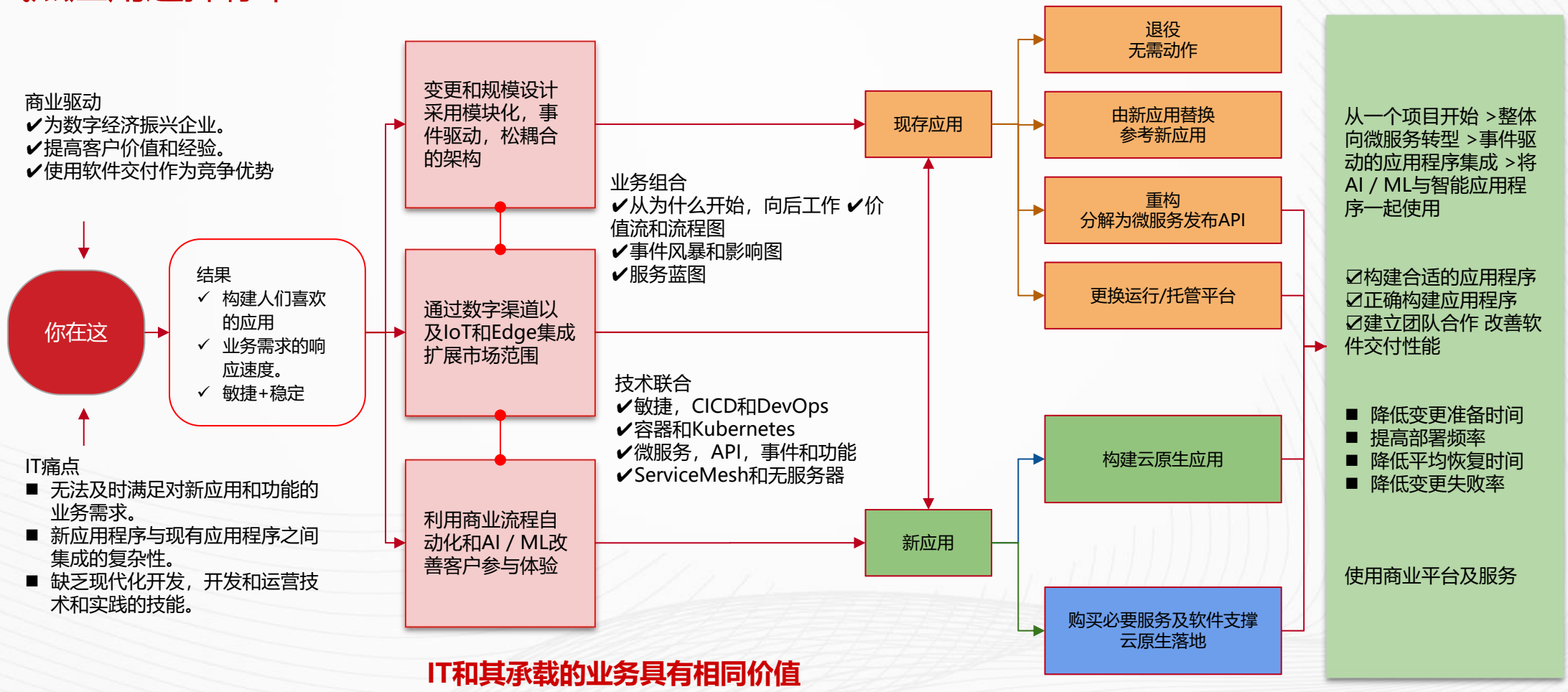
红帽的云原生落地平台部署逻辑架构



云原生应用落地的最佳实践

GOTC

云原生试点应用选择标准



IT和其承载的业务具有相同价值

云原生应用落地的最佳实践

GOTC

容器化应用的最佳实践

- **镜像要尽可能的小** - 通过清理临时文件，并避免安装不必要的软件包来构建小尺寸镜像。 这减少了容器的尺寸，构建时间和复制容器镜像的网络传输时间。
- **支持任意用户ID** - 避免使用sudo命令或要求特定用户名运行你的容器。
- **标记重要的端口** - 虽然可以在运行时指定端口号，然而使用EXPOSE命令在运行的时候指定，则可以让镜像的使用者更轻松。

云原生应用落地的最佳实践

GOTC

容器化应用的最佳实践

- **为持久数据使用卷** - 在容器摧毁之后还需要保存的容器数据的，必须将数据写入一个数据卷。
- **设置镜像元数据** - 以标签和注释形式存在的镜像元数据可以使您的容器镜像更加实用，从而为使用您的容器的开发人员提供了更好的体验。
- **使主机和镜像同步** - 一些容器应用需要容器在某些属性（如时间和机器ID）上与主机同步。

云原生应用落地的最佳实践

GOTC

容器化应用的反模式

有时候一些同学会尝试从他们对其他技术的经验中引入实践。

- 不要将整个操作系统托管在一个容器中
- 不要浪费额外的资源
- 在容器内运行SSH服务器通常不是一个好主意
- 在创建镜像之前，使用.dockerignore 删除日志，源代码等
- 解压缩所有已下载的工件后，将其删除
- 不要为开发，测试，类生产，生产环境使用不同的镜像，甚至不同的标签
- 不要将关键数据保留在容器中
- 不要将安全凭证存储在Dockerfile中

云原生应用落地的最佳实践

GOTC

容器化应用的反模式

- 不要将 registry 用作其他类型数据的通用存储库
- 不要在容器内托管多个服务，例如：不要将服务和 MySQL 托管在同一个容器中
- 创建具有稳定依赖性版本的Dockerfile，最新版的并不一定是最好的
- 将SSH守护程序放入容器中可能导致容器基础结构发生未记录的，无法追踪的更改。
- 不要依赖容器的 IP 地址来启动容器通信，创建服务
- 不要使用 –P 发布所有端口，请使用 –p 发布特定端口。否则您将面临安全风险
- 不要以 root 用户身份运行容器
- 不要在容器之间创建依赖关系，例如：应用程序和数据库
- 不要在 Dockerfile 中使用等待脚本让容器以特定顺序启动

GOTC

THANKS

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE