畅谈云原生
（上）

蚂蚁金服中间件
服务与容器团队

前言

抛砖引玉 笨鸟先飞

背景：2018年，我们团队（中间件服务与容器团队）在进行云原生产品的实践，摸石头过河中

内容：和大家一起聊一聊对云原生的理解，围绕几个需要深度思考的问题（所以本次的段落标题都是问句☺），介绍我们团队对这些问题的思考和正在探索的思路。

目标：抛砖引玉，就云原生这个话题开一个头

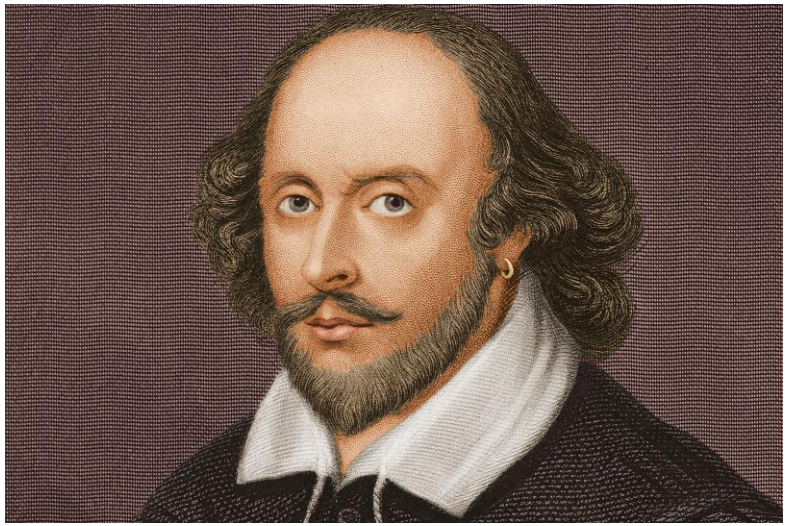
希望：后面有更多的同学继续分享云原生话题，给出更多精彩内容

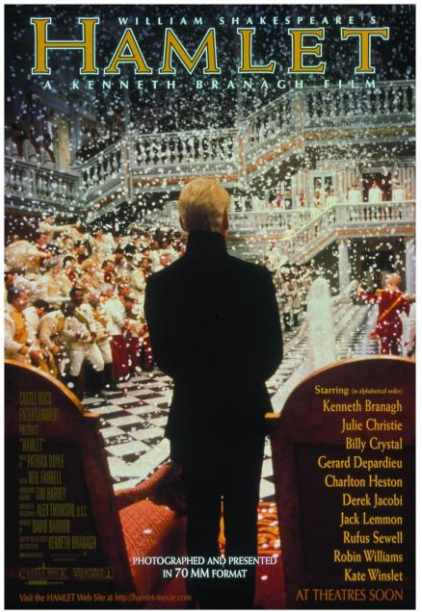
内容

1. 如何理解云原生?
2. 云原生应用应该是什么样子?
3. 云原生下的中间件该如何发展?
4. 云和应用该如何衔接?
5. 如何让产品更符合云原生?
6. 花絮：有哪些有趣的角色转变?

1 如何理解云原生?

吾日三省吾身：如何理解云原生?





Shakespeare： There are a thousand Hamlets in a thousand people's eyes.

快速回顾

云原生定义的变化

来自Pivotal的定义

Pivotal 是Cloud Native/云原生应用的提出者，并推出了Pivotal Cloud Foundry和Spring系列开发框架，是云原生的先驱者和探路者。

Pivotal

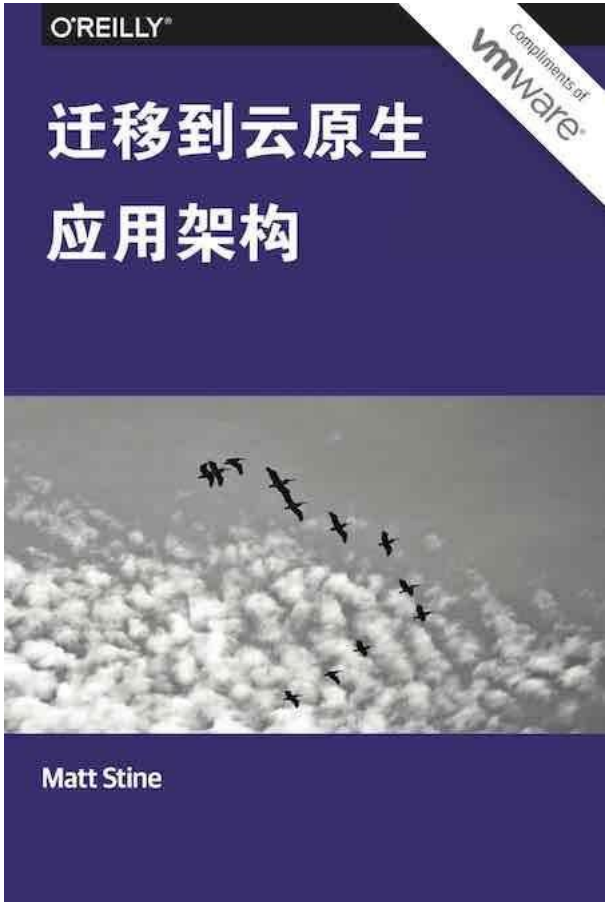
CLOUD FOUNDRY

spring

Pivotal的最初定义

2015年，来自Pivotal公司的Matt Stine编写了一本名为 迁移到云原生应用架构 的电子书，提出云原生应用架构应该具备的几个主要特征：

- 符合12因素应用(Twelve-Factor Applications)
- 面向微服务架构(Microservices)
- 自服务敏捷架构(Self-Service Agile Infrastructure)
- 基于API的协作(API-Based Collaboration)
- 抗脆弱性(Antifragility)



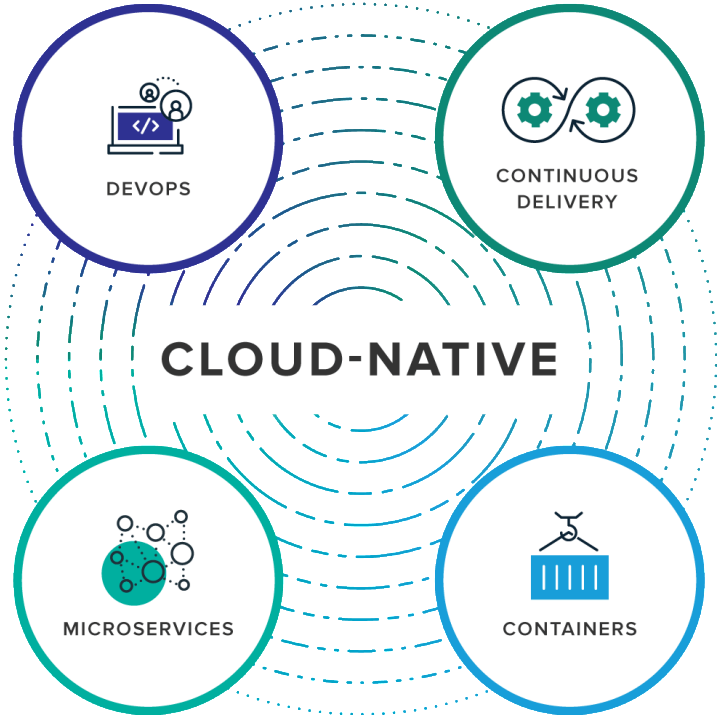
Pivotal更新后的定义

在2017年10月，也是Matt Stine，在接受InfoQ采访时，则对云原生的定义做了小幅调整，将Cloud Native Architectures定义为具有以下六个特质：

- 模块化(Modularity)：（通过微服务）
- 可观测性(Observability)
- 可部署性(Deployability)
- 可测试性(Testability)
- 可处理性(Disposability)
- 可替换性(Replaceability)

Pivotal官网最新定义

在Pivotal最新的官方网站 https://pivotal.io/cloud-native 上，对cloud native的介绍是关注四个要点：



CNCF的定义

蚂蚁金服 ANT FINANCIAL

2015年CNCF建立，开始围绕云原生的概念打造云原生生态体系，起初CNCF对云原生的定义包含以下三个方面：

- 应用容器化(software stack to be Containerized)
- 面向微服务架构(Microservices oriented)
- 应用支持容器的编排调度(Dynamically Orchestrated)

到2018年，随着社区对云原生理念的广泛认可和云原生生态的不断扩大，
还有CNCF项目和会员的大量增加，起初的定义已经不再适用

CNCF的2018年更新后的定义：v1.0

蚂蚁金服 ANT FINANCIAL

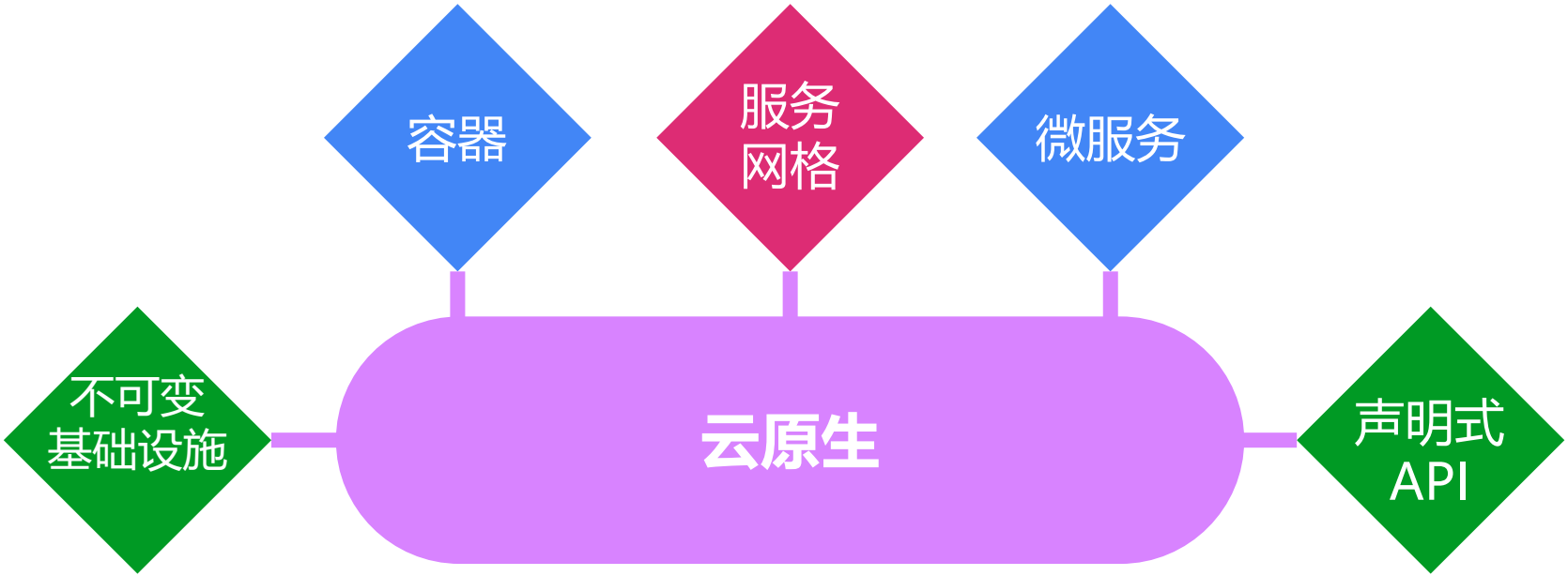
云原生技术有利于各组织在公有云、私有云和混合云等新型动态环境中，构建和运行可弹性扩展的应用。云原生的代表技术包括容器、服务网格、微服务、不可变基础设施和声明式API。

这些技术能够构建容错性好、易于管理和便于观察的松耦合系统。结合可靠的自动化手段，云原生技术使工程师能够轻松地对系统作出频繁和可预测的重大变更。

云原生计算基金会（CNCF）致力于培育和维护一个厂商中立的开源生态系统，来推广云原生技术。我们通过将最前沿的模式民主化，让这些创新为大众所用。

https://github.com/cncf/toc/blob/master/DEFINITION.md

云原生代表技术



云原生定义回顾小结：

蚂蚁金服 ANT FINANCIAL

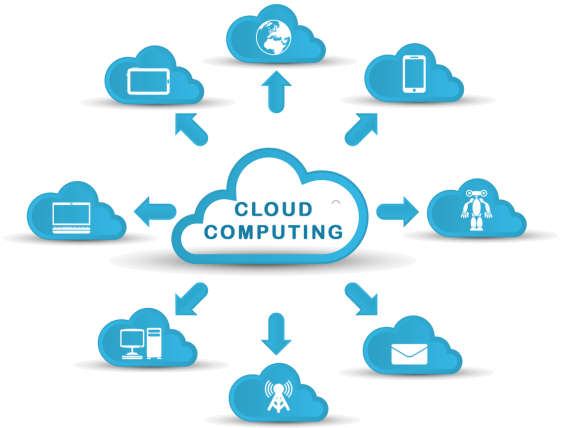
- ✓ 云原生的定义一直在变
  - 不同组织有不同的定义：Pivotal & CNCF
  - 同一个组织在不同时间点有不同的定义
  - 同一个人在不同时间点也有不同的定义
- ✓ 云原生的定义未来还会变
  - CNCF最新的定义：版本v1.0

回到话题：我们该如何理解云原生？

Cloud Native
云原生

1

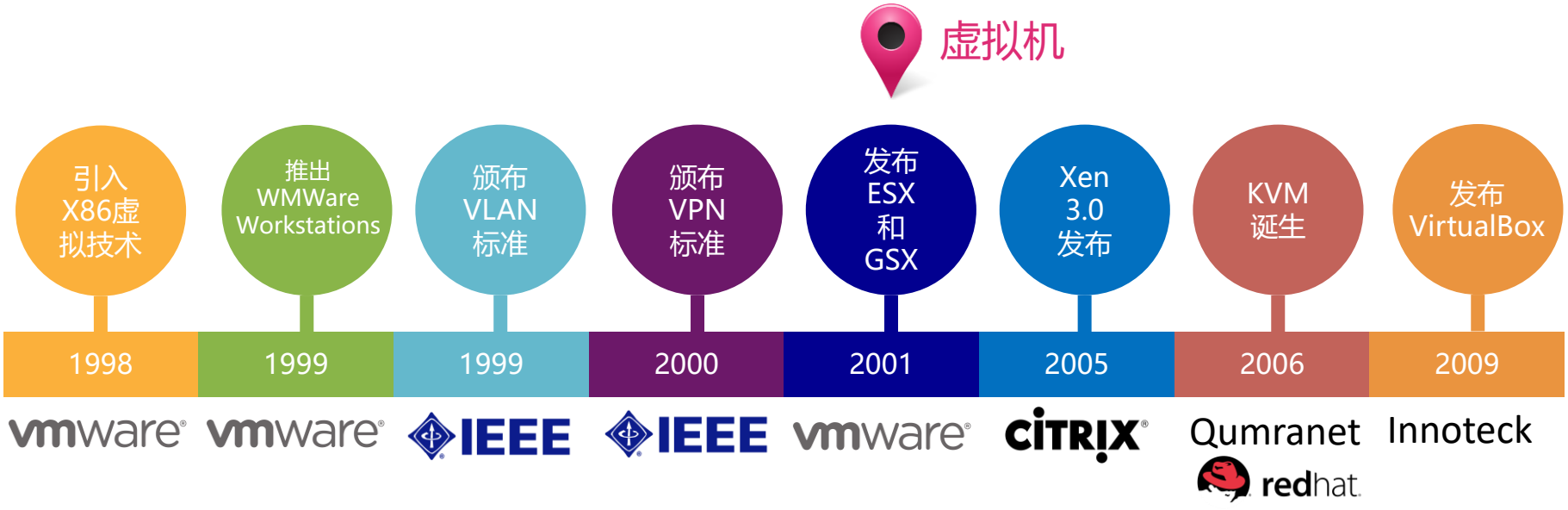
什么是Cloud?



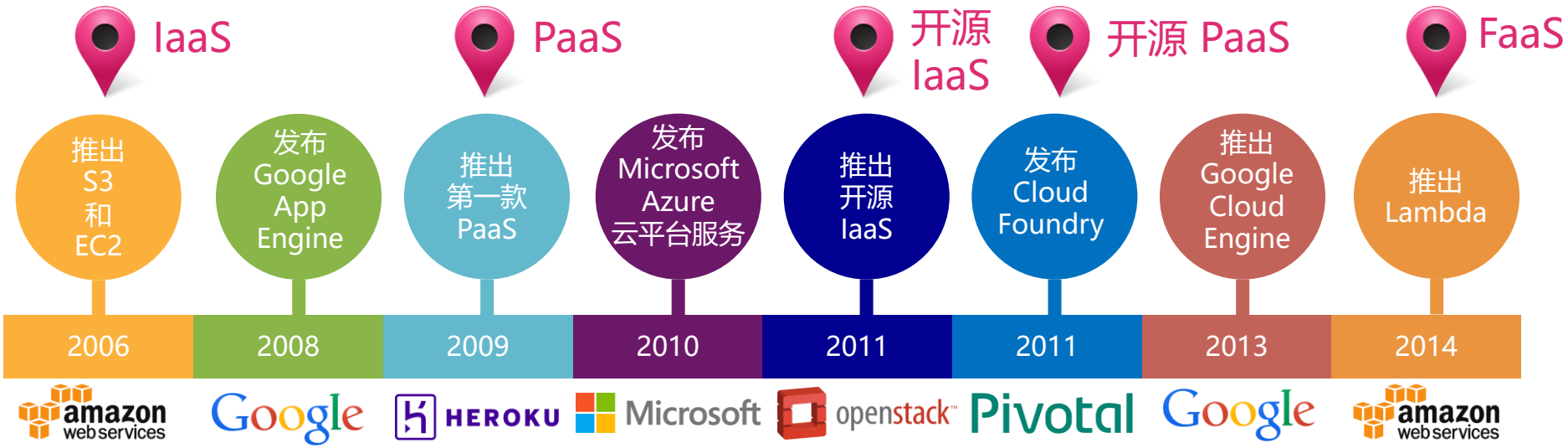
快速回顾

云计算的演进历史

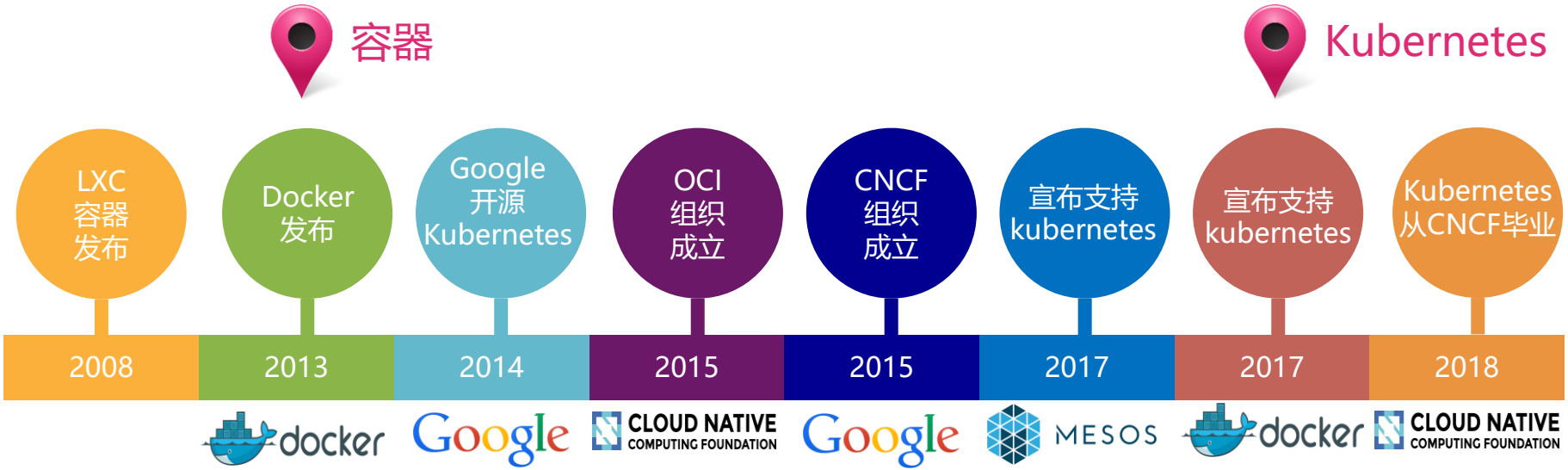
虚拟化技术的成熟


虚拟机
引入 X86虚拟技术
推出 WMWare Workstations
颁布 VLAN 标准
颁布 VPN 标准
发布 ESX 和 GSX
Xen 3.0 发布
KVM 诞生
发布 VirtualBox
1998
1999
1999
2000
2001
2005
2006
2009
vmware
vmware
IEEE
IEEE
vmware
CITRIX
Qumranet
redhat
Innoteck

基于虚拟机的云计算

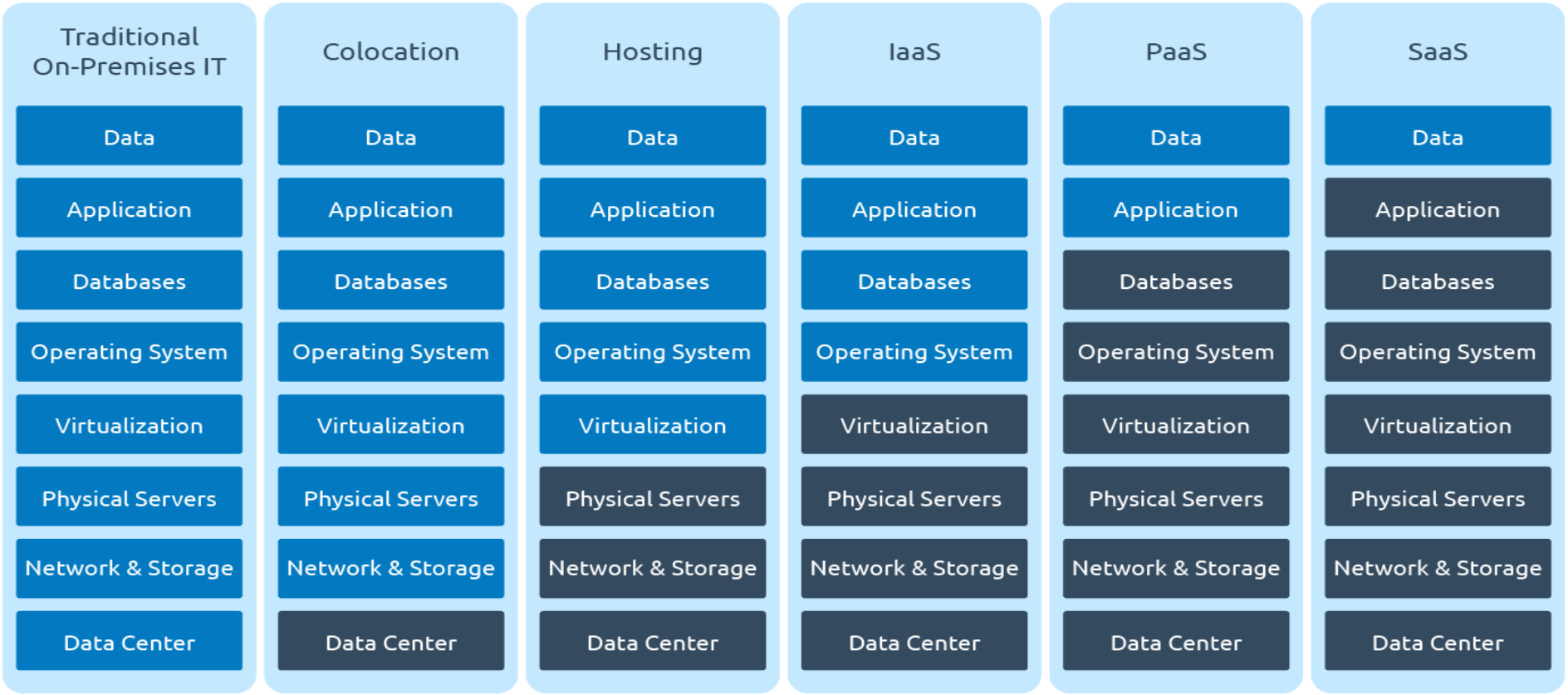

IaaS
PaaS
开源 IaaS
开源 PaaS
FaaS
推出 S3 和 EC2
发布 Google App Engine
推出 第一款 PaaS
发布 Microsoft Azure 云平台服务
推出 开源 IaaS
发布 Cloud Foundry
推出 Google Cloud Engine
推出 Lambda
2006
2008
2009
2010
2011
2011
2013
2014
amazon web services
Google
HEROKU
Microsoft
openstack
Pivotal
Google
amazon web services

容器的兴起和编排大战


容器
Kubernetes
LXC
容器
发布
Docker
发布
Google
开源
Kubernetes
OCI
组织
成立
CNCF
组织
成立
宣布支持
kubernetes
宣布支持
kubernetes
Kubernetes
从CNCF毕业
2008
2013
2014
2015
2015
2017
2017
2018
docker
Google
CLOUD NATIVE
COMPUTING FOUNDATION
Google
MESOS
docker
CLOUD NATIVE
COMPUTING FOUNDATION

云的形态变化

蚂蚁金服 ANT FINANCIAL



云计算历史回顾小结：

蚂蚁金服 ANT FINANCIAL

- ✓ 云计算二十年来变化巨大
  - 从物理机到虚拟机到容器
  - IaaS, PaaS, SaaS, Caas, FaaS 等多种形态出现
  - 公有云、私有云、混合云
  - Kubernetes出现并成为事实标准
  - 在低可用率的硬件上搭建高可用率的服务

云上的应用要如何适应？

回到话题：我们该如何理解云原生？

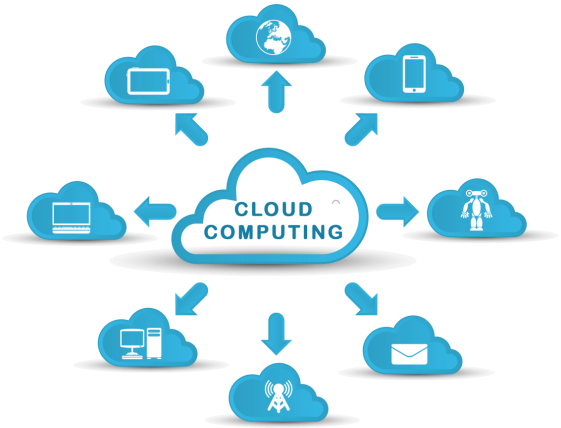
Cloud Native
云原生

1

2

什么是Cloud?

什么是Native?



字典中的Native

- belonging to a person by birth or to a thing by nature; inherent:
  - native ability; native grace.
- belonging by birth to a people regarded as indigenous to a certain place, especially a preliterate people:
  - Native guides accompanied the expedition through the rain forest.
- born in a particular place or country:
  - a native New Yorker.

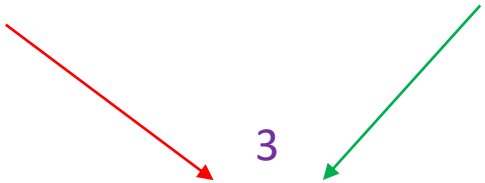
关键字：Born

与生俱来，生而知之；无牵无挂，无拘无束

继续：该如何理解云原生？

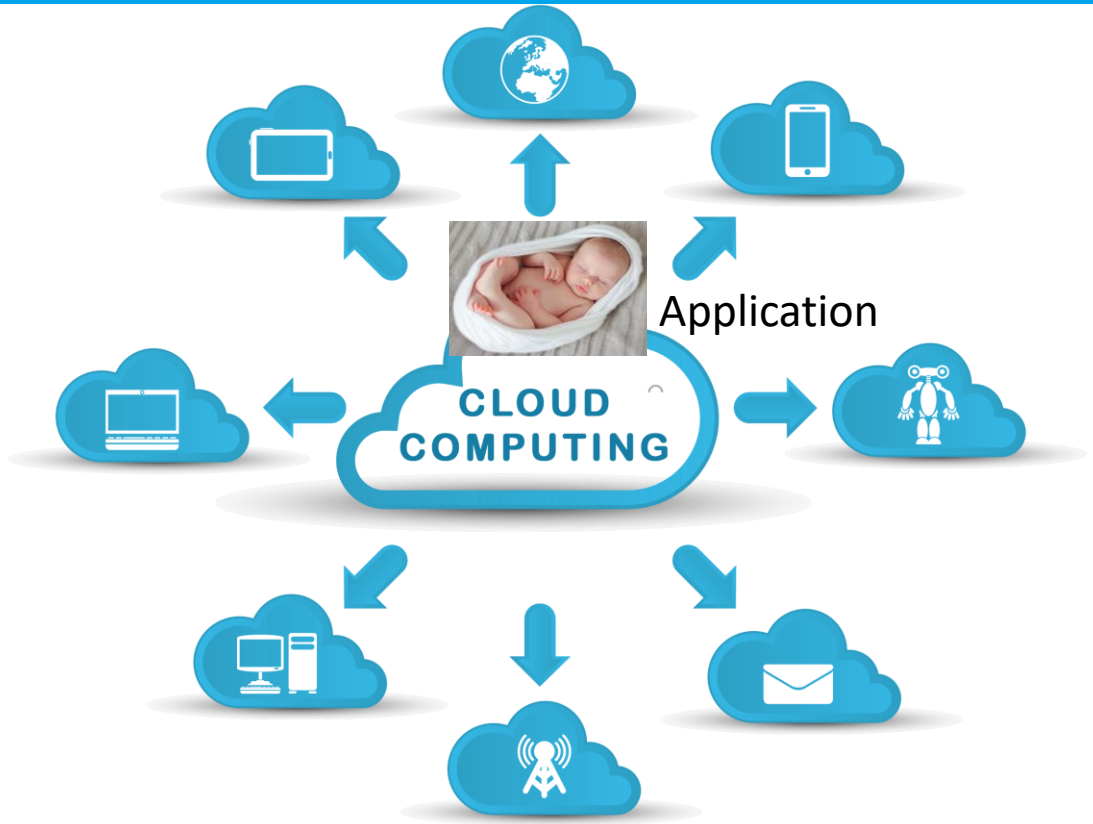
Cloud Native
云原生

什么是Cloud?　　什么是Native?



什么是Cloud Native ?

抛砖引玉：云原生 ->原生为云设计



云原生：应用原生被设计为在云上以最佳方式运行，充分发挥云的优势。

2 云原生应用应该是什么样子?

云原生之前

应用主要满足两大块需求：

- 业务需求
- 非业务需求

基础设施
虚拟机
操作系统
硬件

向上提供基本运行资源：

- CPU
- 内存
- 网络
- 存储

传统非云原生应用

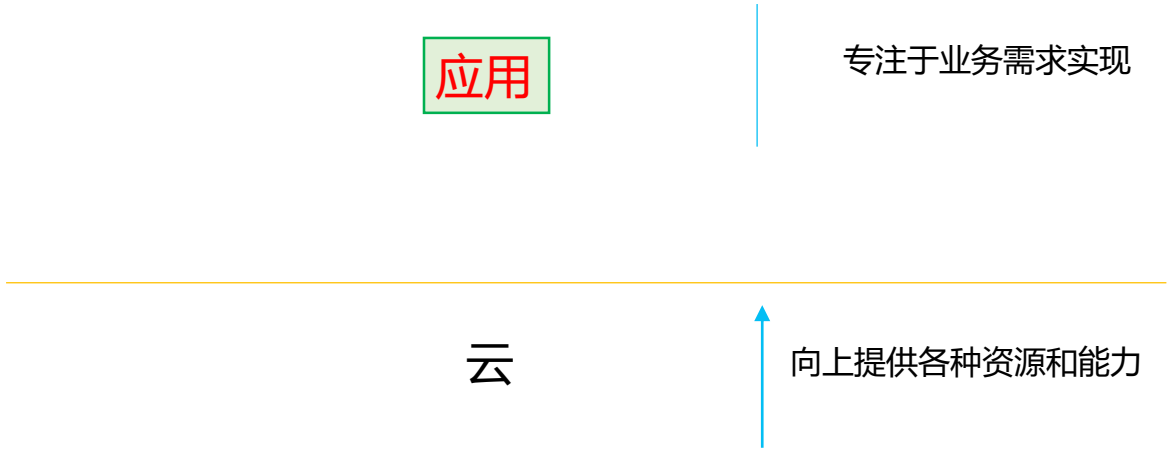
业务需求的实现

非业务需求的实现

背景：美国今年冬天特别冷，然后出现了这种极度夸张的超大号搞笑衣服。

Gigi Hadid

云原生：云的支持应该让应用更多关注业务



我们觉得：理想的云原生应用应该是这个样子 ☺

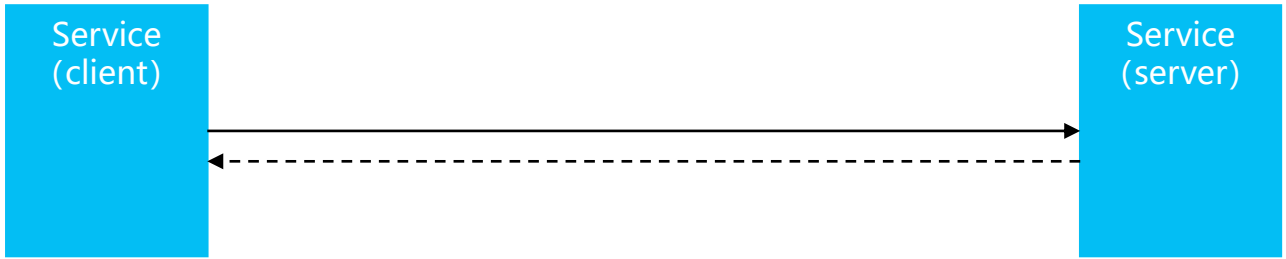
Gigi Hadid

问题：“衣服”那里去了？

蚂蚁金服 ANT FINANCIAL

- ✓ 云
- ✓ 基础设施
- ✓ 下沉到基础设施的中间件
  - 后面会详细解释什么是下沉

以服务间通讯为例

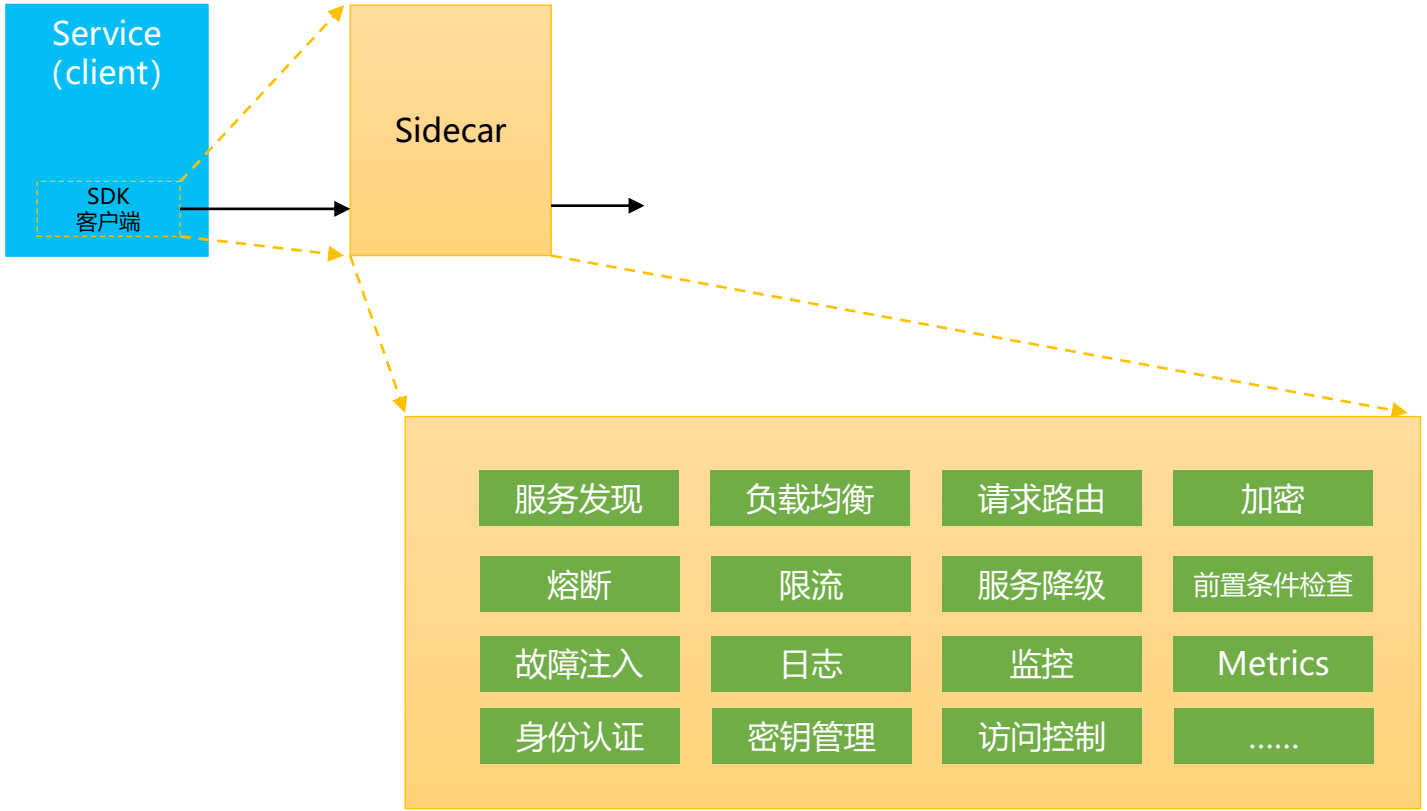


- 服务注册与服务发现
- 编解码（序列化）
- 负载均衡
- 服务路由
- 熔断，限流，重试
- 可观测性：Logging，Trace, Metrics
- 安全
- ……

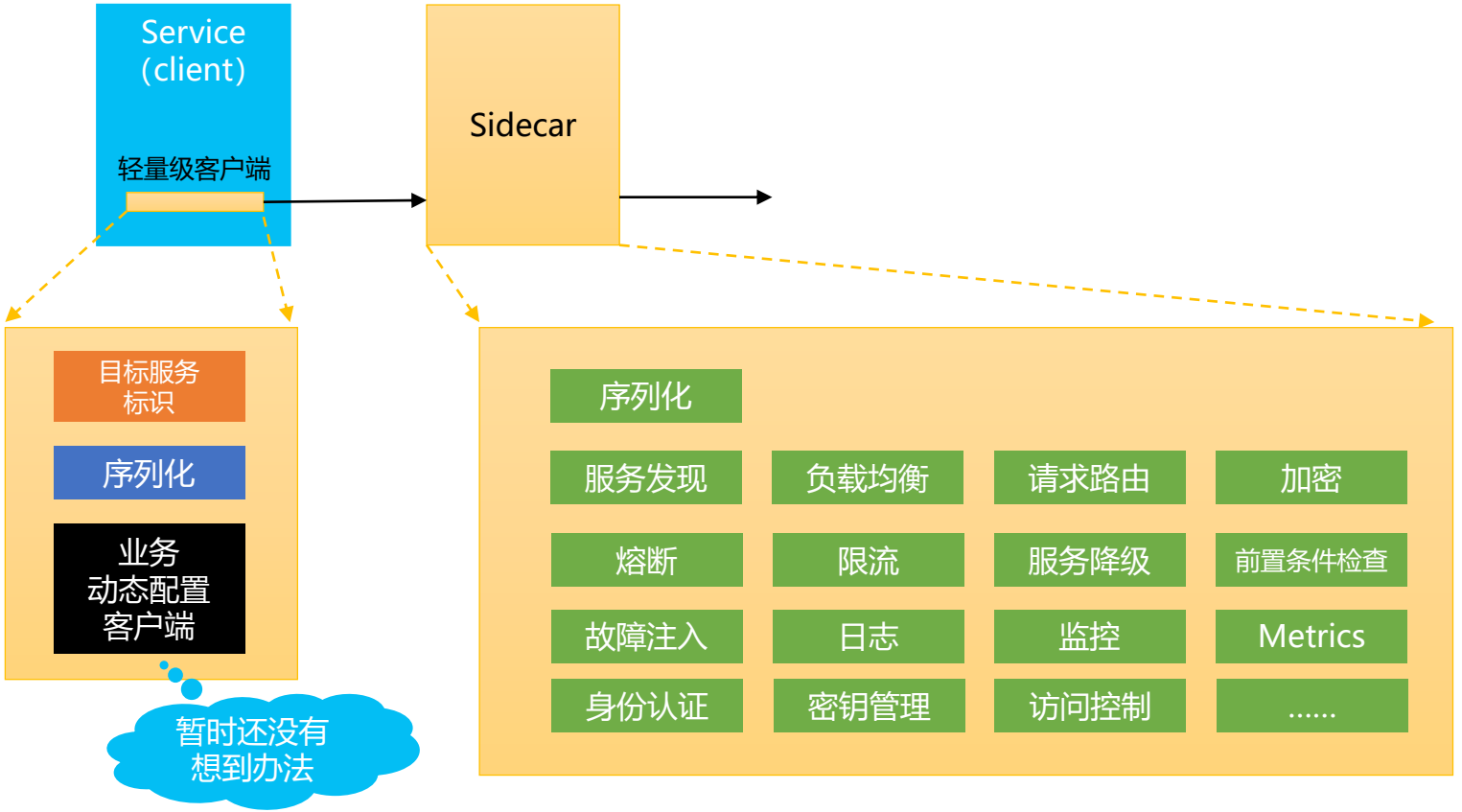
SDK的思路


Service
(client)
SDK
客户端
序列化
服务发现
负载均衡
请求路由
加密
熔断
限流
服务降级
前置条件检查
故障注入
日志
监控
Metrics
身份认证
密钥管理
访问控制
.....

Service Mesh的思路：将SDK客户端的功能剥离



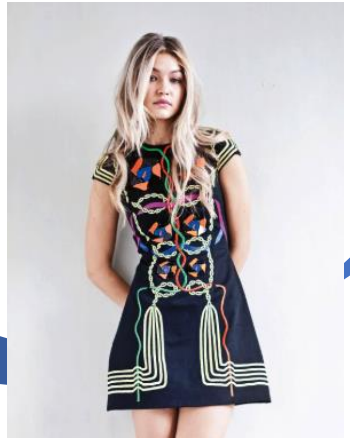
Service Mesh的思路：让应用轻量化



我们的想法：云原生应用应该往这个方向努力☺

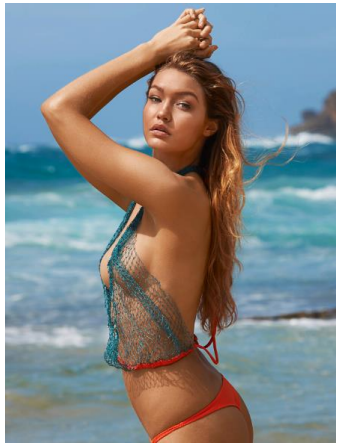
梦想

非原生



务实

轻量化



理想

举一个比较理想的案例：实现密文通讯

3

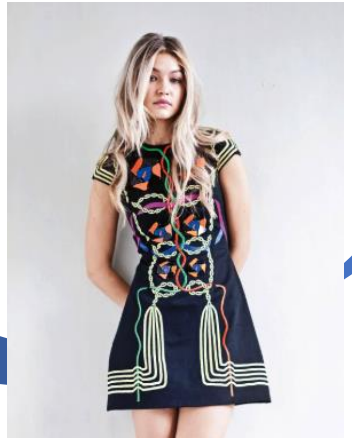
云原生下的中间件该如何发展？

大家可自行替换

隐含前提：应用最终对外提供的功能不变

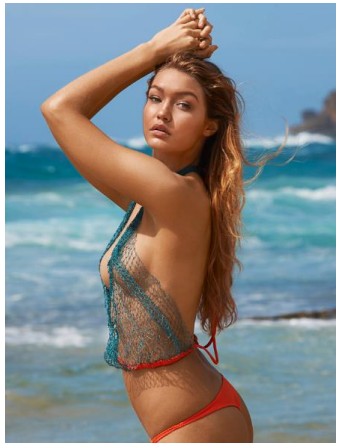
梦想

非原生



务实

轻量化

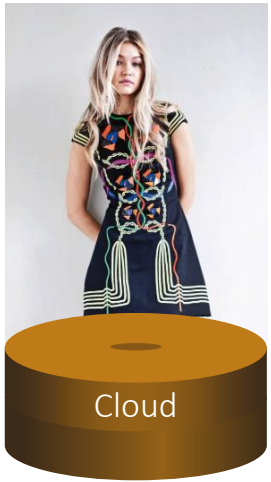


理想

应用形态依赖于云提供的能力

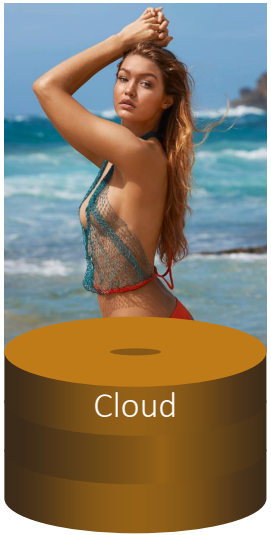
非原生

云只能提供**非常有限**的能力，应用需要自行实现



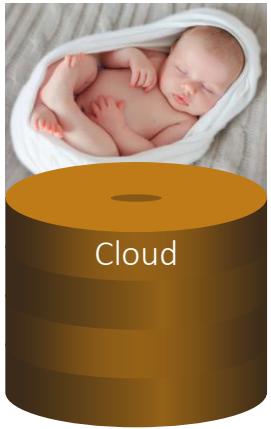
务实版
云原生

云提供**大部分**能力，部分云未能提供的能力依然需要应用自行实现



理想版
云原生

云提供**绝大部分**能力，只是在某些特定环节无法完全剥离和解耦

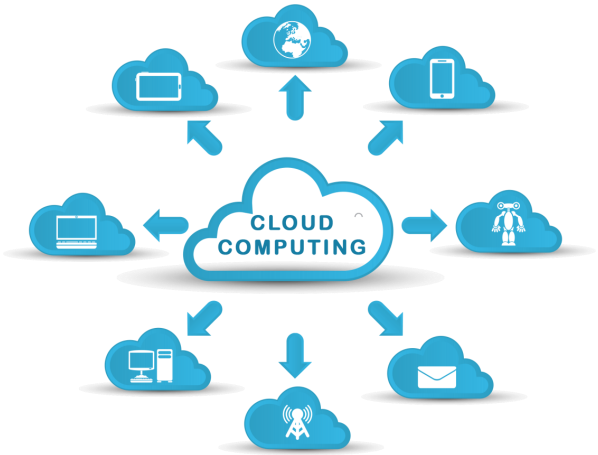


梦想版
云原生

云提供**所有**能力，所有环节都完全解耦

云的能力

中间件



现实：还远没有实现

理想：由云来提供应用需要的所有能力

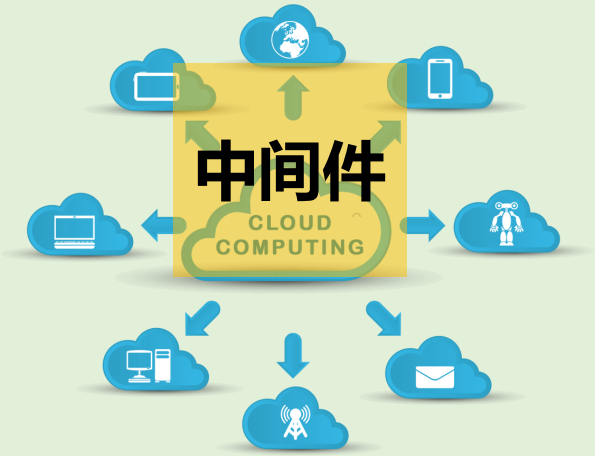
中间件的做法

应用

后端服务
(客户端和调用代码)

开发框架

类库



在云原生前：通常以类库和框架的形式

云原生：下沉到基础设施，成为云的一部分

解释：什么是“下沉”？

问题：“衣服”那里去了？

- ✓ 云
- ✓ 基础设施
- ✓ 下沉到基础设施的中间件

  - 是下沉，不是消失
  - 产品依然存在，功能继续提供
  - 但是，不和应用直接耦合（轻量化）
  - 也不被应用直接感知（原生）
  - 在运行时为应用动态赋能
    - 具体实现后面展开

云原生下中间件：功能继续，赋能方式云原生化

蚂蚁金服 ANT FINANCIAL

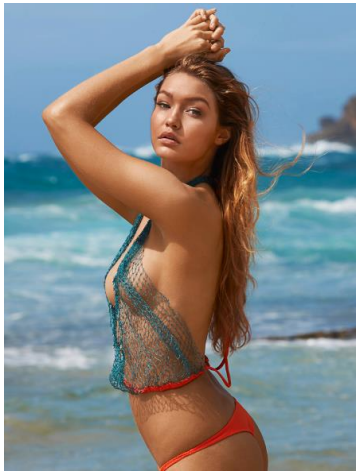
提供御寒衣服

云原生之前：

应用需要实现非常多的能力
（可以通过类库框架简化）

云原生化

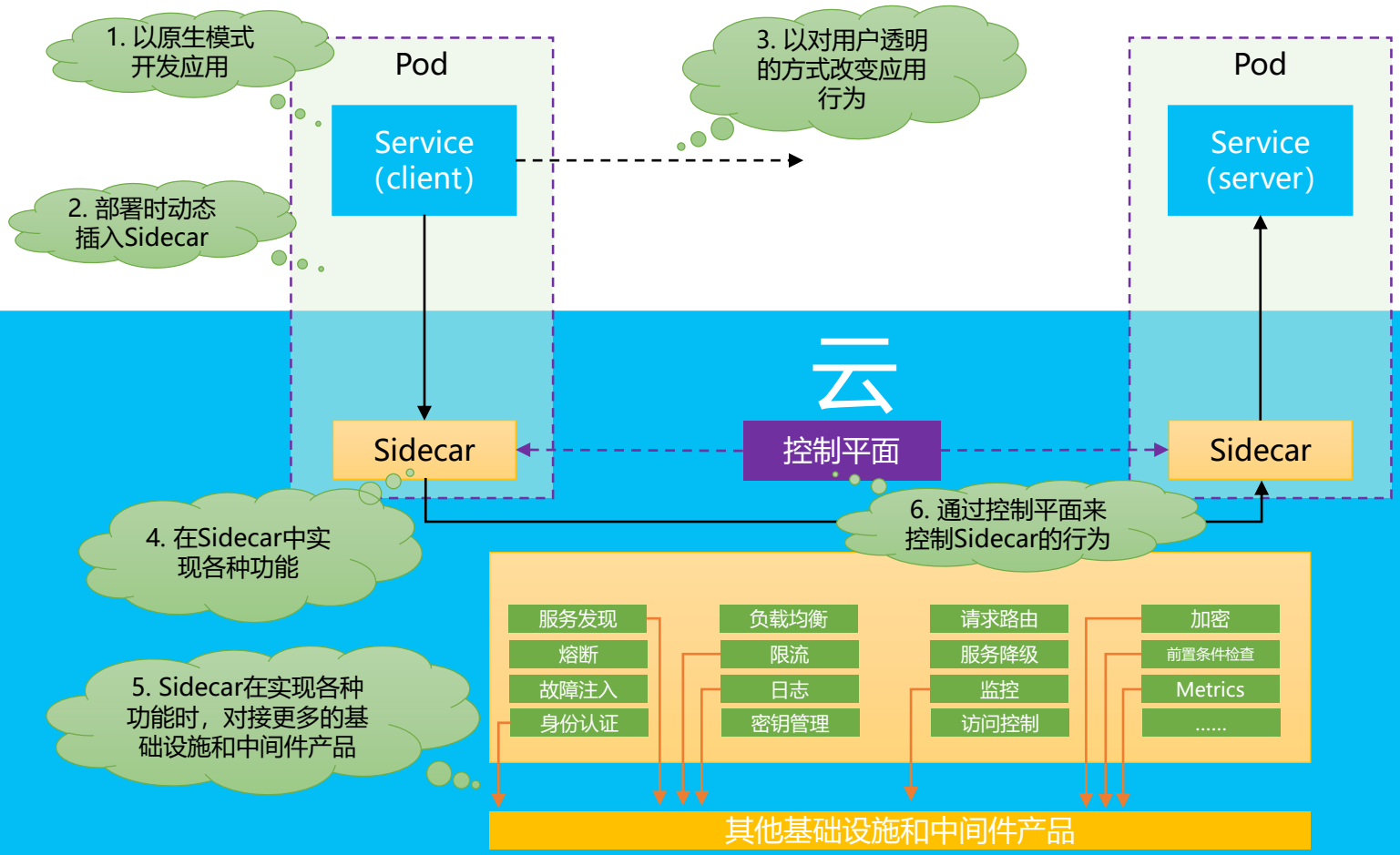
提供温暖的阳光



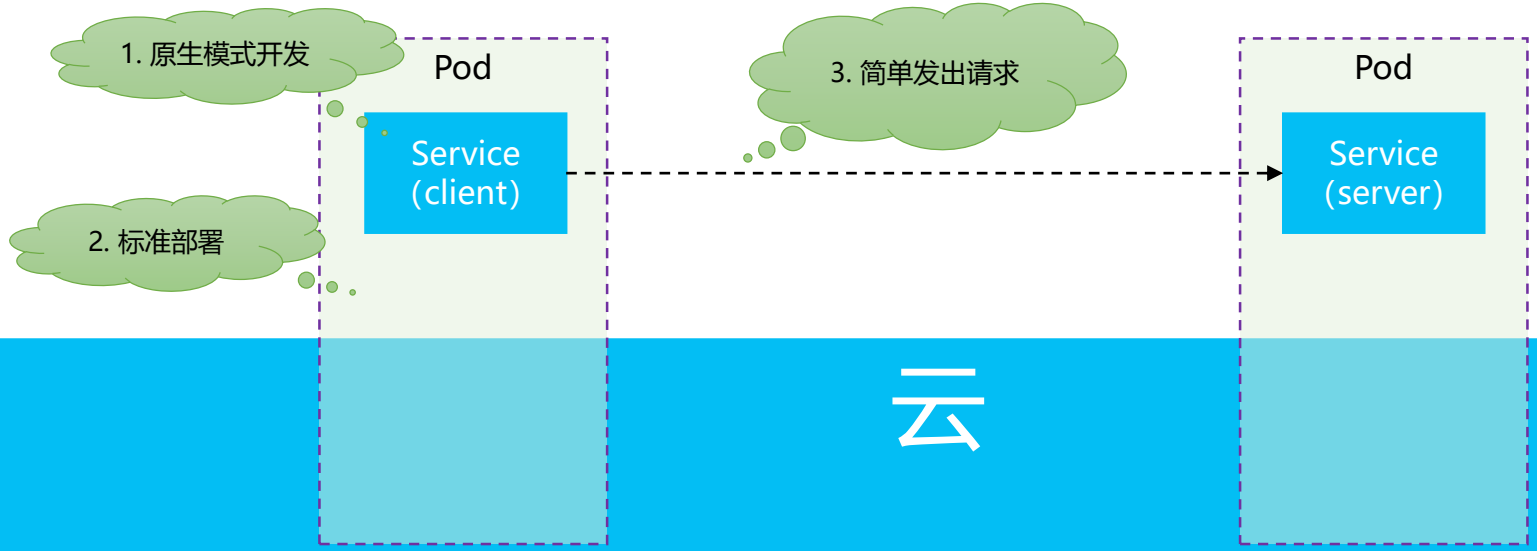
云原生：

加强和改善应用运行环境
（云）
帮助应用实现轻量化

Service Mesh 模式：工作原理（白盒）

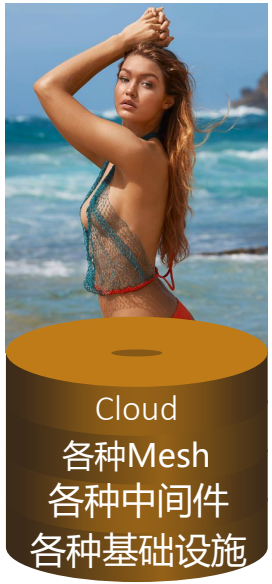


Service Mesh 模式：应用视角（黑盒）



Mesh模式可以应用于更多的场景：不仅仅是服务间通讯

非原生



云原生

- Service Mesh
- Database Mesh
- Message Mesh
- ……

Mesh模式之外的探索

蚂蚁金服 ANT FINANCIAL

- ✓ 中间件下沉到基础设施的方式
  - 不只有Mesh模式一种
  - 也不是每个中间件都需要改造为Mesh模式
    - 有些是可以通过和mesh集成来间接提供能力的
- ✓ 更多的模式有待进一步探索和实践
  - DNS（探索中）
- ✓ 基本工作原理
  - 将功能实现从应用中剥离出来
    - 轻量化
  - 在运行时为应用 动态赋能


期待更多分享
介绍更多模式
更多发展思路