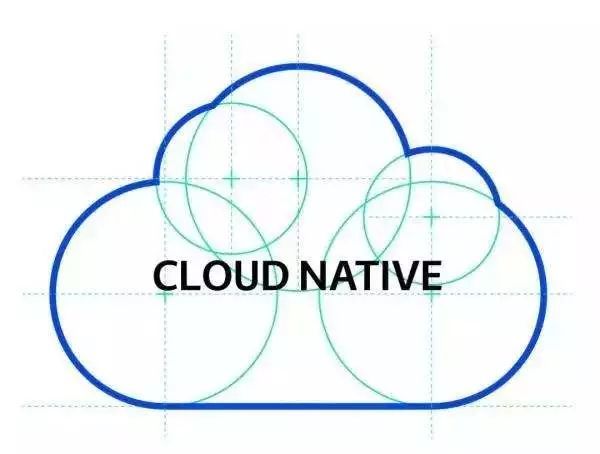
大家言必称云原生，却鲜少有人告诉你到底什么是云原生，若是找资料来看，读完大多会感觉云绕雾罩，一知半解，总之虚得很；甚至会让你一度怀疑自己的智商，不过我对于读不懂的文章，一律归因于写文章的人太蠢，当然这不一定是事实，但这样的思考方式能让我避免陷入自我怀疑的负面情绪。



云原生之所以解释不清楚，是因为云原生没有确切的定义，云原生一直在发展变化之中，解释权不归某个人或组织所有。

何谓云原生？

技术的变革，一定是思想先行，云原生是一种构建和运行应用程序的方法，是一套技术体系和方法论。云原生（CloudNative）是一个组合词，Cloud+Native。Cloud表示应用程序位于云中，而不是传统的数据中心；Native表示应用程序从设计之初即考虑到云的环境，原生为云而设计，在云上以最佳姿势运行，充分利用和发挥云平台的弹性+分布式优势。

Pivotal公司的Matt Stine于2013年首次提出云原生（CloudNative）的概念；2015年，云原生刚推广时，Matt Stine在《迁移到云原生架构》一书中定义了符合云原生架构的几个特征：12因素、微服务、自敏捷架构、基于API协作、扛脆弱性；到了2017年，Matt Stine在接受InfoQ采访时又改了口风，将云原生架构归纳为模块化、可观察、可部署、可测试、可替换、可处理6特质；而Pivotal最新官网对云原生概括为4个要点：DevOps+持续交付+微服务+容器。



2015年云原生计算基金会（CNCF）成立，CNCF掺和进来后，最初把云原生定义为包括：容器化封装+自动化管理+面向微服务；到了2018年，CNCF又更新了云原生的定义，把服务网格(Service Mesh)和声明式API给加了进来。

可见，不同的人和组织对云原生有不同的定义，相同的人和组织在不同时间点对云原生也有不同的定义，真是乱的一匹，搞得鄙人非常晕菜，我的应对很简单，选一个我最容易记住和理解的定义：DevOps+持续交付+微服务+容器。

**总而言之，符合云原生架构的应用程序应该是：采用开源堆栈（K8S+Docker）进行容器化，基于微服务架构提高灵活性和可维护性，借助敏捷方法、DevOps支持持续迭代和运维自动化，利用云平台设施实现弹性伸缩、动态调度、优化资源利用率。**

云原生构建应用简便快捷，部署应用轻松自如、运行应用按需伸缩。优点不一而足，缺点微乎其微；秒杀传统Web框架，吊打祖传IT模式，实在是保命\*\*、评优晋级不可多得的终极绝密武器。

云元素的四要素

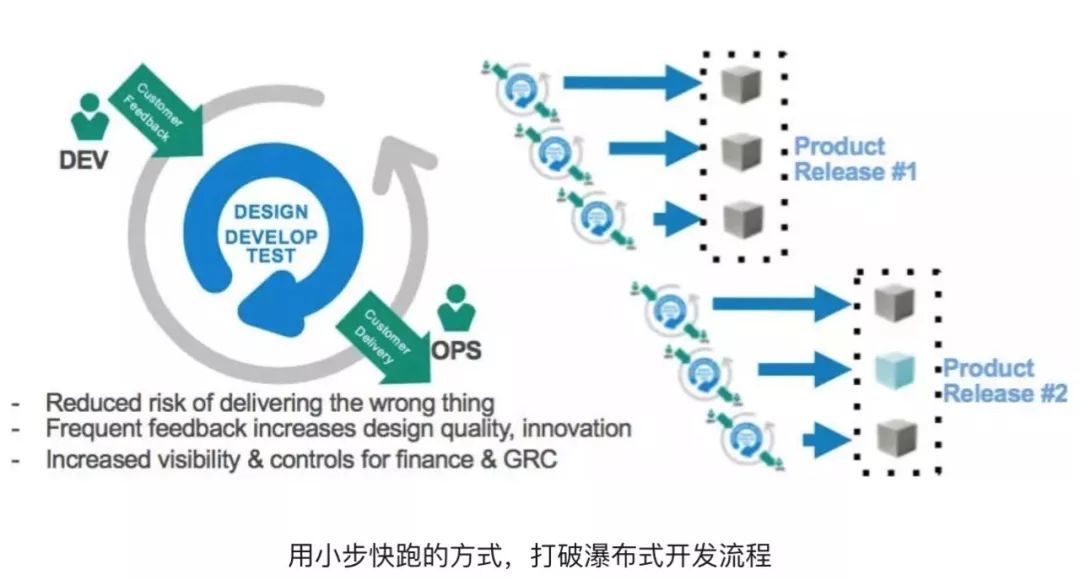
**微服务**：几乎每个云原生的定义都包含微服务，跟微服务相对的是单体应用，微服务有理论基础，那就是康威定律，指导服务怎么切分，很玄乎，凡是能称为理论定律的都简单明白不了，不然就忒没b格，大概意思是组织架构决定产品形态，不知道跟马克思的生产关系影响生产力有无关系。

微服务架构的好处就是按function切了之后，服务解耦，内聚更强，变更更易；另一个划分服务的技巧据说是依据DDD来搞。

**容器化**：Docker是应用最为广泛的容器引擎，在思科谷歌等公司的基础设施中大量使用，是基于LXC技术搞的，容器化为微服务提供实施保障，起到应用隔离作用，K8S是容器编排系统，用于容器管理，容器间的负载均衡，谷歌搞的，Docker和K8S都采用Go编写，都是好东西。

**DevOps**：这是个组合词，Dev+Ops，就是开发和运维合体，不像开发和产品，经常刀刃相见，实际上DevOps应该还包括测试，DevOps是一个敏捷思维，是一个沟通文化，也是组织形式，为云原生提供持续交付能力。

**持续交付**：持续交付是不误时开发，不停机更新，小步快跑，反传统瀑布式开发模型，这要求开发版本和稳定版本并存，其实需要很多流程和工具支撑。



## **如何云原生？**

首先，云原生借了云计算的东风，没有云计算，自然没有云原生，云计算是云原生的基础。

随着虚拟化技术的成熟和分布式框架的普及，在容器技术、可持续交付、编排系统等开源社区的推动下，以及微服务等开发理念的带动下，**应用上云已经是不可逆转的趋势**。

云计算的3层划分，即基础设施即服务(IaaS)、平台即服务(PaaS)、软件即服务(SaaS)为云原生提供了技术基础和方向指引，**真正的云化不仅仅是基础设施和平台的变化，应用也需要做出改变**，摈弃传统的土方法，在架构设计、开发方式、部署维护等各个阶段和方面都基于云的特点，重新设计，从而建设全新的云化的应用，即云原生应用。

1.本地部署的传统应用往往采用c/c++、企业级java编写，而云原生应用则需要用以网络为中心的go、node.js等新兴语言编写。

2.本地部署的传统应用可能需要停机更新，而云原生应用应该始终是最新的，需要支持频繁变更，持续交付，蓝绿部署。

3.本地部署的传统应用无法动态扩展，往往需要冗余资源以抵抗流量高峰，而云原生应用利用云的弹性自动伸缩，通过共享降本增效。

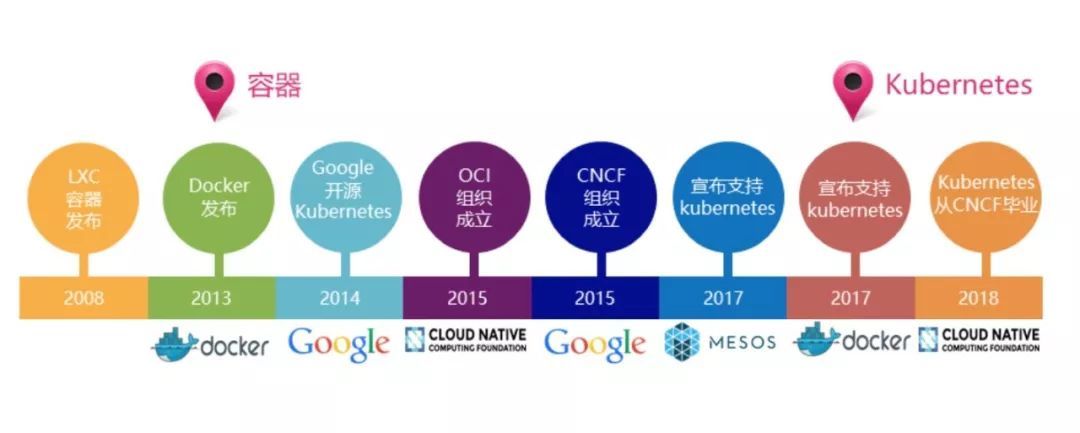
4.本地部署的传统应用对网络资源，比如ip、端口等有依赖，甚至是硬编码，而云原生应用对网络和存储都没有这种限制。

5.本地部署的传统应用通常人肉部署手工运维，而云原生应用这一切都是自动化的。

6.本地部署的传统应用通常依赖系统环境，而云原生应用不会硬连接到任何系统环境，而是依赖抽象的基础架构，从而获得良好移植性。

7.本地部署的传统应用有些是单体(巨石)应用，或者强依赖，而基于微服务架构的云原生应用，纵向划分服务，模块化更合理。

可见，要转向云原生应用需要以新的云原生方法开展工作，云原生包括很多方面：基础架构服务、虚拟化、容器化、容器编排、微服务。幸运的是，开源社区在云原生应用方面做出了大量卓有成效的工作，很多开源的框架和设施可以通过拿来主义直接用，2013年Docker推出并很快成为容器事实标准，随后围绕容器编排的混战中，2017年诞生的k8s很快脱颖而出，而这些技术极大的降低了开发云原生应用的技术门槛。



虽说云原生的推介文档有引导之嫌，但面对它列举的优点，作为杠精的我亦是无可辩驳。这么说的话，云原生也忒好了吧，应用是不是要立刻马上切换到云原生架构？我的观点是：理想很丰满，现实经常很骨感，需从应用的实际需要出发，目前的问题是否真的影响到业务发展，而推倒重来的代价能否承受得来。

## **技术的趋势和影响**

软件设计有两个关键目标：**高内聚、低耦合**，围绕这2个核心目标，又提出了单一职责、开闭原则、里氏替换、依赖导致、接口隔离、最少知识等设计原则。

软件工程师一直都在为这两个目标而努力奋斗，以求把软件编写得更加清晰、更加健壮、更加易于扩展和维护。

但后来，人们发现有更多的诉求，希望开发软件变得更简单、更快捷，程序员希望更少编写代码，非专业人员也希望能开发程序，于是，更多的更傻瓜的编程语言被发明出来，更多的编程技术和编程思想被发明出来，比如库、组件、云基础设施。

于是很多技术变成了屠龙之技，比如汇编，时代变了，建国后动物不能成精了，没有龙可以宰了，然后很多软件工程师摇身一变成了调参工程师、Call API砖家、用库包能手、拼组件达人，这是效率分工的结果，也是技术发展的使然。

纵观近二十年的科技互联网发展历程，**大的趋势是技术下沉**，特别是近些年，随着云计算的发展和普及，基础设施越来越厚实，业务开发变得越来越容易，也越来越没有技术含量，而之前困扰小团队的性能、负载、安全性、扩展性问题都不复存在，这不禁让互联网行业的油腻大叔们噤若寒蝉，仿佛分分钟就要被卷入历史洪流而万劫不复。

虽然不可否认技术的重要性在降低，但也还不至于那么悲观。遥想PC时代，当VB、Delphi、MFC出现的时候，也有类似论调，所见即所得，点点鼠标，就可以开发PC桌面程序，是不是很高端？那时候码农的担心相比现在恐怕是只多不少吧，但后来随着互联网兴起，出现了后端开发这个工种，码农很快找到了新的战场，网络、分布式、数据库、海量服务、容灾防错，于是又玩出一堆新花样。

如果说PC时代的基础设施是控件库，互联网时代的基础实施是云，那AI时代基础设施是什么？又会有什么高端玩法？

**[点击关注，第一时间了解华为云新鲜技术~](https://link.zhihu.com/?target=https://bbs.huaweicloud.com/blogs?utm_source=zhihu&utm_medium=blog-article&utm_campaign=blog-article" \t "https://zhuanlan.zhihu.com/p/_blank)**

发布于 2020-06-23

[云原生应用](https://www.zhihu.com/topic/20748165" \t "https://zhuanlan.zhihu.com/p/_blank)

[微服务](https://www.zhihu.com/topic/20552868" \t "https://zhuanlan.zhihu.com/p/_blank)

[容器（虚拟化）](https://www.zhihu.com/topic/19615961" \t "https://zhuanlan.zhihu.com/p/_blank)

​赞同 432​

​25 条评论

​分享

​喜欢​收藏​申请转载

​

赞同 432

​

分享

# **[传统容器已死，安全容器将成为云原生标配](https://zhuanlan.zhihu.com/p/99182075)**

[pockry](https://zhuanlan.zhihu.com/p/99182075)

[](https://zhuanlan.zhihu.com/p/109183544)

# **[容器、Kubernetes和云原生技术 : 企业最需要的是什么？](https://zhuanlan.zhihu.com/p/109183544)**

[fiisi...发表于进击的云计...](https://zhuanlan.zhihu.com/p/109183544)

# **[云原生的不同解释及正确含义](https://zhuanlan.zhihu.com/p/97295179)**

[云原生的解释可以说五花八门，本文从不同角度探讨云原生的内涵以及如何从不同维度准确理解它的含义。 云原生起源网上有些文章提到云原生是“Pivotal公司的Matt Stine于2013年首次提出云原生…](https://zhuanlan.zhihu.com/p/97295179)

[倚天码农](https://zhuanlan.zhihu.com/p/97295179)

作为一名100%的门外汉，坚持看完了文章。先感谢大佬科普。我自己理解，云原生应该是一种“建立在云上的多种效率提升技术的复合体"（而不是单一的技术创新），主要就是在云技术摆脱物理储存限制的基础上，进一步实现应用的专业优化（即文章里说的按功能切分）和生产效率的提升（即文章里说的弹性扩容等），从而解放人力。类似于制造业的自动化生产，只不过现在是轮到IT技术，由机器算法代替部分人力了。不知道这么理解对不对，欢迎大佬们指正

[姜伟](https://www.zhihu.com/people/jiang-wei-9-92" \t "https://zhuanlan.zhihu.com/p/_blank)01-06

DevOps感觉已经包含了持续交付的含义了。

​4​回复​踩​ 举报

[](https://www.zhihu.com/people/medsoft)

[太白](https://www.zhihu.com/people/medsoft" \t "https://zhuanlan.zhihu.com/p/_blank)07-15

云原生就是下一个内卷，不接受反驳。

​3​回复​踩​ 举报

[](https://www.zhihu.com/people/qi-qi-can-can-qi-qi-85)

[roboter.qi](https://www.zhihu.com/people/qi-qi-can-can-qi-qi-85" \t "https://zhuanlan.zhihu.com/p/_blank)07-18

理解了，就是造了个组合词，把这些所谓的容器化+devops+微服务等技术进行了抽象理解，好吧，这就是云原生，也不是什么新鲜技术了

​2​回复​踩​ 举报

[IMG_272](https://www.zhihu.com/people/dao-chang-ye)

[道常叶](https://www.zhihu.com/people/dao-chang-ye" \t "https://zhuanlan.zhihu.com/p/_blank)​05-24

简单理解，以前造房子是砖混结构，后来改成钢筋混凝土结构，设计施工一体化DEV OPS吧不用再搬砖头了，地原生

​1​回复​踩​ 举报

[](https://www.zhihu.com/people/sirs05)

[老外婆](https://www.zhihu.com/people/sirs05" \t "https://zhuanlan.zhihu.com/p/_blank)03-31

本地部署的传统应用对网络资源，比如ip、端口等有依赖，甚至是硬编码，而云原生应用对网络和存储都没有这种限制。 这说的是什么东西。。

​1​回复​踩​ 举报

[](https://www.zhihu.com/people/yi-zhi-jin-yi-wei)

[一只锦依卫](https://www.zhihu.com/people/yi-zhi-jin-yi-wei" \t "https://zhuanlan.zhihu.com/p/_blank)2020-06-23

一名云计算技术与应用的准大学生看完了这个帖子

​1​回复​踩​ 举报

[](https://www.zhihu.com/people/dong-you-ran-85)

[悠乐橙](https://www.zhihu.com/people/dong-you-ran-85" \t "https://zhuanlan.zhihu.com/p/_blank)回复[一只锦依卫](https://www.zhihu.com/people/yi-zhi-jin-yi-wei" \t "https://zhuanlan.zhihu.com/p/_blank)2020-08-20

现在学校都有这个专业了吗？

​赞​回复​踩​ 举报

[](https://www.zhihu.com/people/yi-zhi-jin-yi-wei)

[一只锦依卫](https://www.zhihu.com/people/yi-zhi-jin-yi-wei" \t "https://zhuanlan.zhihu.com/p/_blank)回复[悠乐橙](https://www.zhihu.com/people/dong-you-ran-85" \t "https://zhuanlan.zhihu.com/p/_blank)2020-08-20

不知道啊，看是新专业感觉挺好

​赞​回复​踩​ 举报

展开其他 1 条回复

[](https://www.zhihu.com/people/eaglemhy)

[eaglemhy](https://www.zhihu.com/people/eaglemhy" \t "https://zhuanlan.zhihu.com/p/_blank)08-18

最后一句“AI时代的基础设施是？” 可以关注一下这个方向啊，在这个基础设施的方向上面熟悉一个

​赞​回复​踩​ 举报

[](https://www.zhihu.com/people/ok520)

[山间之明月](https://www.zhihu.com/people/ok520" \t "https://zhuanlan.zhihu.com/p/_blank)03-13

"依赖导致"。。。是"依赖倒置"吧

​赞​回复​踩​ 举报

[](https://www.zhihu.com/people/geekcelia)

[GeekCelia](https://www.zhihu.com/people/geekcelia" \t "https://zhuanlan.zhihu.com/p/_blank)​01-21

大佬讲的很好，康威定律，文科生记住了

​赞​回复​踩​ 举报

[](https://www.zhihu.com/people/ou-mou-di)

[欧某弟](https://www.zhihu.com/people/ou-mou-di" \t "https://zhuanlan.zhihu.com/p/_blank)2020-11-28

怎么才能像你一样文笔出众。。。主要是

​赞​回复​踩​ 举报

[](https://www.zhihu.com/people/shi-guang-24-53-87)

[时光](https://www.zhihu.com/people/shi-guang-24-53-87" \t "https://zhuanlan.zhihu.com/p/_blank)2020-09-26

楼主文笔太好啦，求关注

​赞​回复​踩​ 举报

[](https://www.zhihu.com/people/maxsys)

[maxsys](https://www.zhihu.com/people/maxsys" \t "https://zhuanlan.zhihu.com/p/_blank)2020-09-14

DotNet Core + CSharp 得要接力云原生技术发力了。

​赞​回复​踩​ 举报

[](https://www.zhihu.com/people/you-you1212)

[游小唐](https://www.zhihu.com/people/you-you1212" \t "https://zhuanlan.zhihu.com/p/_blank)2020-08-16

哈哈哈，经历了k8s的我，觉得你写的很有意思，给你加鸡腿

​赞​回复​踩​ 举报

[](https://www.zhihu.com/people/linzihao-239)

[Johny Sinn](https://www.zhihu.com/people/linzihao-239" \t "https://zhuanlan.zhihu.com/p/_blank)2020-08-16

基于微服务等技术的云原生当然是Java，Spring Cloud了解一下。

​赞​回复​踩​ 举报

[](https://www.zhihu.com/people/maxsys)

[maxsys](https://www.zhihu.com/people/maxsys" \t "https://zhuanlan.zhihu.com/p/_blank)回复[Johny Sinn](https://www.zhihu.com/people/linzihao-239" \t "https://zhuanlan.zhihu.com/p/_blank)2020-09-14

C# .Net Core 的 [http://Asp.Net](http://link.zhihu.com/?target=http://Asp.Net" \t "https://zhuanlan.zhihu.com/p/_blank) Core 了解下，配合VSCode 直接远程调试，云上开发，他不更香么！？

​1​回复​踩​ 举报

[](https://www.zhihu.com/people/linzihao-239)

[Johny Sinn](https://www.zhihu.com/people/linzihao-239" \t "https://zhuanlan.zhihu.com/p/_blank)回复[maxsys](https://www.zhihu.com/people/maxsys" \t "https://zhuanlan.zhihu.com/p/_blank)2020-09-16

香啊，.Net Core 也挺不错。

​1​回复​踩​ 举报

展开其他 2 条回复

[](https://www.zhihu.com/people/desire-60-45-56)

[Desire](https://www.zhihu.com/people/desire-60-45-56" \t "https://zhuanlan.zhihu.com/p/_blank)2020-06-28

写的很详细，大佬用心了

​赞​回复​踩​ 举报

# 云原生架构

[什么是云原生架构](https://cloud.tencent.com/developer/article/1360266)

- 高可用设计（Design for Availability），依据应用业务需求，高可用分为不同级别，比如不同区域、不同机房（跨城或同城）、不同机柜、不同服务器和不同进程的高可用，云原生应用应该根据业务的可用性要求设计不同级别的架构支持。

- 可扩展设计（Design for Scale），所有应用的设计是无状态的，使得业务天生具有扩展性，在业务流量高峰和低峰时期，依赖云的特性自动弹性扩容，满足业务需求。

- 快速失败设计（Design for Failure），即包括系统间依赖的调用随时可能会失败，也包括硬件基础设施服务随时可能宕机，还有后端有状态服务的系统能力可能有瓶颈，总之在发生异常时能够快速失败，然后快速恢复，以保证业务永远在线，不能让业务半死不活地僵持着。

[360°透视：云原生架构及设计原则](https://zhuanlan.zhihu.com/p/30200943)

[什么是微服务](https://cloud.tencent.com/developer/article/1360532)

[微服务架构风格](https://cloud.tencent.com/developer/article/1360504?from=10680)

[云平台架构实践中的微服务分解原则](https://cloud.tencent.com/developer/article/1360544)

[云平台的微服务架构实践](https://cloud.tencent.com/developer/article/1360528)

[云原生应用迁云方案](https://www.aliyun.com/solution/migrating2cloudnative/index?spm=a2c6h.14922178.J\_9181122460.3.42df32c5qDw1SA)

[企业级互联网架构解决方案](https://www.aliyun.com/solution/middleware/index?spm=a2c6h.14922178.J\_9181122460.4.42df32c5qDw1SA)

[业务异地多活架构解决方案](https://www.aliyun.com/solution/middleware/msha?spm=a2c6h.14922178.J\_9181122460.2.42df32c5qDw1SA)

## 微服务架构