博云
BoCloud

应用上容器

指南

CONTENT

目录

1 哪些应用适合上容器? --- P1

01 · 从架构层面看
02 · 从需求层面看
03 · 常见上容器的应用系统

2 容器化对应用要求 --- P2

01 · 应用组件全容器化
02 · 应用组件半容器化

3 容器化应用的主要工作 --- P6

01 · 应用容器化步骤
02 · 应用容器化耗时

4 应用上容器典型流程 --- P7

01 · 总体流程图
02 · 规划阶段
03 · 设计阶段
04 · 实施阶段
05 · 验证阶段
06 · 维护阶段

5 应用上容器方案 --- P14

01 · 上云实施策略定义
02 · 上云流程的裁剪
03 · 上云详细实施计划

6 应用上容器后的变化 --- P15

01 · 部署流程的变化
02 · 资源与应用绑定的习惯变化
03 · 问题解决方式的变化
04 · 应用扩展的变化
05 · 升级回滚方式的变化

7 应用上容器的注意事项 --- P17

01 · 无状态化改造问题
02 · 系统间的依赖问题
03 · 容器化与非容器化的并存问题
04 · 外部存储的问题
05 · 多进程的问题
06 · 安全性的问题
07 · 镜像制作的问题
08 · 工作流程的改变问题

哪些应用适合上容器?

适合迁移到容器云平台上的系统一般来讲有如下特征：

01.从架构层面看

- 应用本身是微服务架构或者希望变更为微服务架构
- 应用本身无状态或者易于实现状态数据和应用的分离

02.从需求层面看

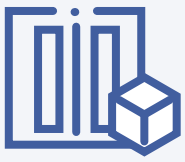
- 应用访问有突发性的特点，需要弹性支持
- 应用希望能实现快速迭代上线和快速部署

03.常见上容器的应用系统

- web 类系统，WAS/WebLogic/Tomcat/JBoss 类应用
- App 类系统，具有互联网特性的应用
- 大数据系统，Hadoop/Spark 等。
- 各类中间件，比如负载均衡、消息队列、服务注册等

容器化对应用的要求

应用容器化的方式分为两类：应用组件全容器化和应用组件半容器化。



01.应用组件全容器化

指组成应用系统的组件中，包括数据库，中间件，应用服务器等全部通过容器进行部署上线。

应用状态保持

应用分为两种一种是**有状态的应用**，一种是**无状态的应用**。

对于无状态的应用，不需要保证应用当前状态。开发过程中除了需要修改程序中的配置文件以外，其他不需要做任何改动。实施过程中，首先需要根据应用的依赖情况编写 dockerfile 文件，其次通过 dockerfile 模板生成镜像。最后通过 docker 或者 kubernetes 进行部署。

对于有状态的应用，例如会话保持（session）或者状态共享的问题。由于容器本身暂时不支持有状态的应用（单个应用实例支持），当涉及到应用实例扩容，缩容的问题，如何保证 session 共享是应用本身需要解决的问题。如果通过 LVS，Nginx 等负载方式，可以通过配置 Nginx 中 ip_hash 的方式实现 session 保持，但是无法实现真正的负载均衡。因此遇到这类问题，建议在第三方缓存（redis 或者 memche）中保存状态信息，用来实现状态共享。

除了应用服务器以外，任何一个应用组件容器化以后，搭建集群时都会遇到类似的问题，也可以通过上述方式解决。也可以在同一个容器网络内配置容器间链接实现。（例如搭建 Mysql 集群）。如果通过 kubernetes 编排工具，可以通过 PV（persist value）实现状态信息共享和同步。

应用配置

全容器化后，除了状态保持和共享以外，遇到另外一个问题是应用配置改造。在应用开发过程中，会存在 jdbc.properties 等类型的配置文件用来存储应用和其他组件交互时的必要信息。应用在启动过程中会自动加载配置文件中的参数信息。当应用容器化后，在应用启动前由于无法通过之前的方式，先修改配置，再启动。因此需要在应用容器启动之前把对应的参数信息以一种方式传递到容器内部，使得应用启动过程中能够加载。所以容器化时，应用配置也是面临的一个问题。

其实无论是哪种编排工具，都可以通过“环境变量”传递参数的方式把 value 值传递给容器，应用启动过程中可以正常启动。具体改造方式以 jdbc.properties 文件为例。未改造之前的 jdbc.properties 文件如下：

```
jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://192.168.1.216:3306/bocloud_plus?characterEncoding=utf8

jdbc.username=root

jdbc.password=onceas
```

改造完成以后，文件内容如下：

```
jdbc.driver=$DRIVER

jdbc.url=$RUL

jdbc.username=$USERNAME

jdbc.password=$PASSWORD
```

在容器启动过程中通过“-e USERNAME=root”方式就可以加载内容。

应用通信

根据应用架构的发展，现在的应用已经由单体结构，转变为 SOA 或者 RPC 类型的架构，甚至未来的微服务架构。因此原来架构聚合，功能聚合的单体应用会拆分为架构分散，功能单一的结构，每个结构之前的部署和通信会是以后面临的一个问题。所以应用容器化同样要能够解决应用系统内组件和组件之前的通信问题。

应用容器化的通信问题涉及两个方面：同一个宿主机不同组件通信和不同宿主机，不同组件相互通信。

同一个宿主机，首先给容器生成别名，在容器启动时通过—link 的参数建立关联关系即可；不同宿主机，首先要通过网络配置，保证容器能够相互通信，简单的网络方式，可以是 Overlay，也可以结合 OVS 等第三方网络实现。其次通过—link 实现应用间通信。

应用日志

应用日志正常输出到文件，容器化以后在启动过程中，通过 volume 或者 -v 参数映射到宿主机的目录下就可以。日志本身不需要改造。

版本发布

在用户的使用中，需要同时对应用容器化（镜像）和物理化部署（即软件包，由于运维部分的顾虑，暂为过渡方案）两种场景进行支持，就需要在应用开发和持续集成部分（镜像仓库和软件服务器）做对应接口规范处理。

版本部署

在应用部署时可能同时存在容器和物理部署两种情况，容器部署启动方式较为快捷方便，但物理部署中就需要对其环境变量预置、端口映射处理、日志重定向等问题进行相应的解决。

02.应用组件半容器化

组成应用系统的组件中，除了容器化以外，另外存在物理部署安装的组件。例如：应用服务器容器化 + 数据库物理部署。

应用半容器化首要要把容器化的组件和非容器化组件分开。根据我们以往的落地经验，建议应用服务器容器化，数据库不做容器化处理。应用半容器化需要保证网络互通和配置正确。

网络互通

保证能够通过容器内部网段访问到物理网络所在的网段，该部分属于实施过程中需要验证和注意的内容，开发阶段可以不考虑。

配置正确

这是应用全容器化和半容器化都必须要解决的问题，通过系统环境变量传值给容器，容器启动过程中自动加载。

容器化应用的主要工作

01.应用容器化步骤

应用容器化主要分为以下五个步骤：

1、获取应用代码

2、创建 Dockerfile

3、容器化当前应用/构建具体的镜像

4、推送镜像到仓库

5、运行应用程序

一旦应用容器化完成，就能够以镜像的形式交付并以容器的方式运行了。

02.应用容器化耗时

通常一套应用的容器化时间在3~5天左右。

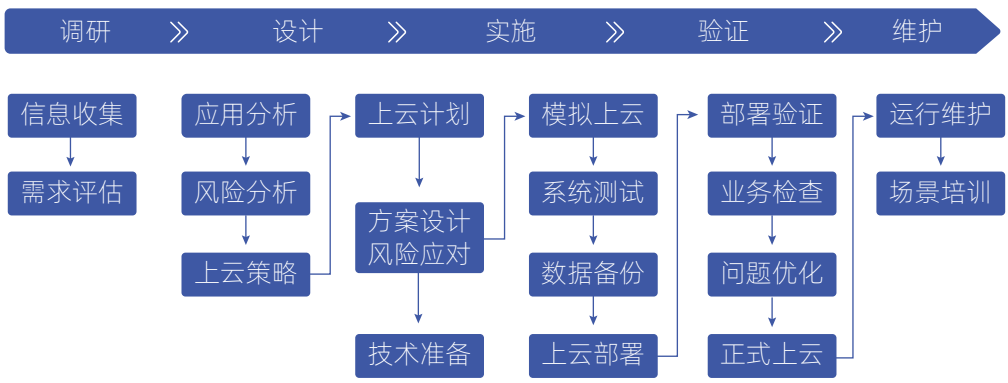
应用容器化所需时间，主要看两个指标：服务数、复杂度

- 服务数：需要容器化的应用服务数量，数量比较好理解，数量越多耗时越长。
- 复杂度：应用服务容器化的难易程度，容器化复杂度越高，所需要时间越长。

应用上容器的典型流程

注：以下内容中“应用上云”是指“应用上容器云”

01.总体流程图



02.规划阶段

信息收集

“企业上云”需要进行严谨细致的调研工作，需要收集硬件及网络环境信息、现有及将来可能增加的业务各类需求、系统配置信息、应用系统信息、数据风险等。

需求评估

从业务需求的角度来看，分析各业务的目前现状、存在的问题、是否可以云化、业务未来的发展需求，从而制定对各个业务系统迁移的目标。

从系统的角度分析各系统的目前现状，包括了主机、存储、网络及安全，分析系统存在的问题，根据评估结果进行规划。

从企业自身信息化水平的角度来看，对于有信息化基础并拥有信息系统硬件环境、维护开发队伍的企业，可根据企业发展规划，逐步进行新、老系统迁移；对于无信息化基础的企业，以企业迫切需解决问题为导向，加快相关应用上线。

应用分析

应用分析是成功上云，降低业务停滞时间的关键。根据业务的负载、特性、复杂性、关联性分析确定并量化业务上云风险可能对业务造成的影响及损失，以确定业务上云的优先分批范围及上云策略。

风险分析

根据收集到的相关信息对目前系统进行业务上云的风险分析，分析各种潜在危险并针对可能发生的危险事件，制定相应措施。

上云策略

“企业应用上云”策略可遵循：统筹规划、分步实施、由易而难、由简单到复杂的原则，制定上云顺序，例如：

（1）优先：独立应用的系统，如邮件系统、合同系统；

（2）其次：应用堆叠的应用系统，如办公 OA；

（3）最后：存在业务依赖的系统，如 CRM、ERP、MES 系统；

03.设计阶段

上云计划

企业现有的信息系统分为业务高度依赖型、业务依赖型和非业务依赖型三类：

（1）7*24 小时业务高度依赖的生产系统，迁移只能在线时间，迁移策略为：“在线迁移”；

（2）非 7*24 小时业务依赖的生产系统，迁移时可以接受一定的离线时间，迁移策略为：“离线迁移”；

（3）非业务依赖性的生产系统迁移可接受较长的离线时间，迁移策略是“分批次迁移”。

根据以上原则，综合考虑各应用系统及相关设备的调研分析情况，制定出详细的上云计划。

方案设计

方案包括：上云实施方案、应用上云方案、数据同步方案、上云回退方案等。

技术准备

技术准备包括上述方案中技术部分的相关验证工作，以及准备可能涉及的软件工具、脚本等。

04.实施阶段

模拟上云

正式上云前模拟一个批次的业务迁移（非正式迁移，业务不割接），来验证业务迁移的及时有效和正确率；针对模拟过程发现各类问题进行修正，并改进业务迁移的流程和工作手册，以满足业务的实际需要。

系统测试

模拟上云完成后对模拟上云的业务进行一次系统测试，以确定业务迁移到云环境中后能够满足业务需求。

（1）性能测试，包括：上云后系统的应用性能测试；上云后系统的网络性能测试；上云后系统软件版本性能测试等。

（2）压力测试，包括：对上云后系统进行压力测试，并取得关键性能指标达到设计目标；从分支机构发起执行版本验证测试及必要的压力测试等。

（3）业务功能，包括：上云后系统与老系统的连接测试；对上云后系统运行批处理测试；完成数据同步后，执行批处理测试；完成数据同步后，从分支机构发起执行高风险业务功能的测试等；

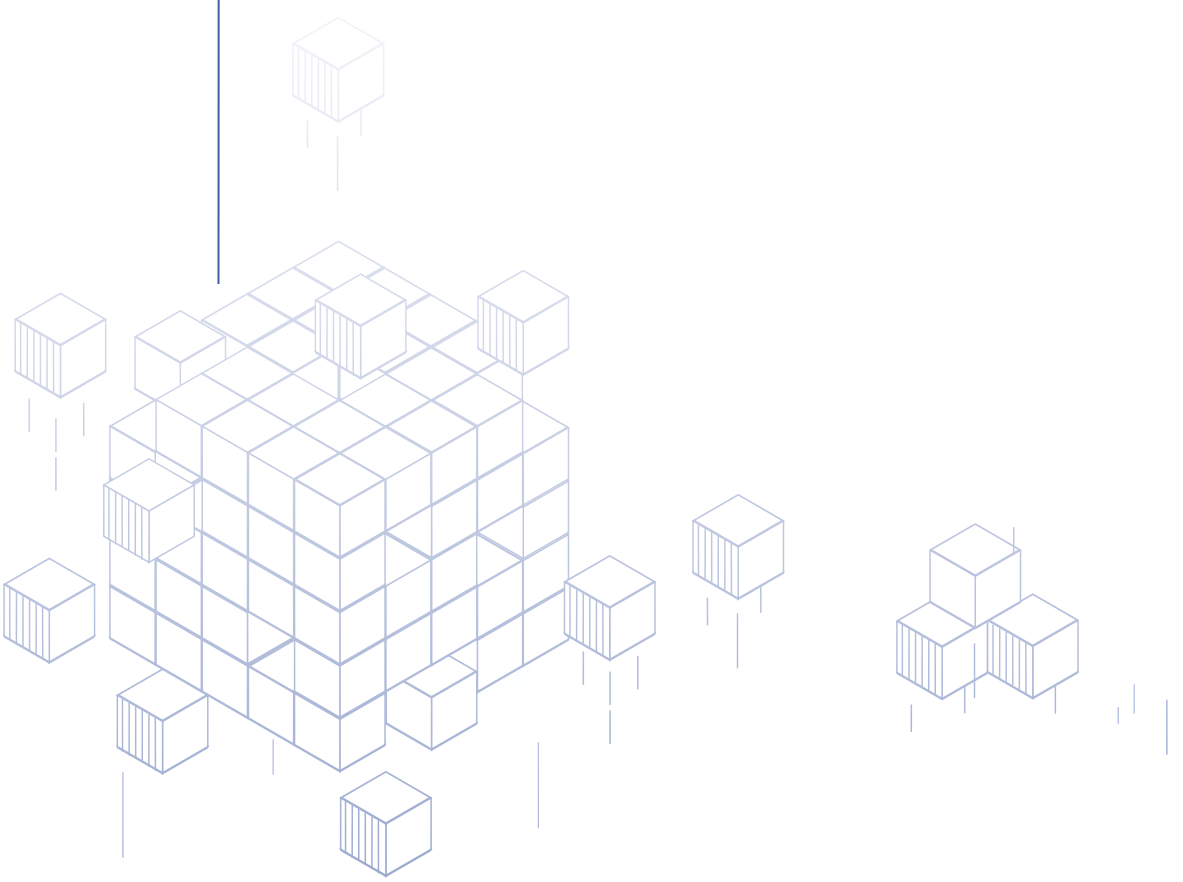
（4）系统连接性测试，包括：上云后对外围系统，进行全面的连接测试；发现问题，提出整改目标；上云后系统与网络的连接测试等。

数据备份

正式上云前，为确保业务数据的完整性、降低上云风险，需要将业务系统及数据进行备份。为提高抗风险能力，建议采用多种备份方式、多份备份数据的方式对业务系统及数据进行备份。

上云部署

根据确定的业务上云方案，实施业务上云工作；根据业务上云方案测试迁移效果，并对业务上云后的系统参数和性能进行调整，使之满足业务系统的需要，并投入实际使用。



05.验证阶段

部署验证

根据确定的业务上云方案，正式实施业务上云；根据业务上云方案测试业务上云效果，并对业务上云后的系统参数和性能进行调整，使之满足业务系统的需要，并投入实际使用。

业务检查

业务正式上云后，进行一定时间的试运行，检测业务上云后是否对业务造成影响，对出现的问题进行及时的解决。

问题优化

根据云上系统监控数据和业务系统发展规划，优化业务系统架构，消除性能瓶颈和风险，保障客户业务平稳运行。

06.维护阶段

运行维护

为上云企业提供全面专业的运维 / 运营服务，进行资源开通、辅助上云、平台监控、故障排查、容量管理、升级重保、健康检查、性能报告等服务科目，可提供驻场服务、巡场服务和远程服务三种服务类型。

场景培训

针对企业客户使用场景，帮助企业熟悉和掌握云上业务操作和云服务技术，培养企业使用习惯，解答企业用户使用过程中遇到的各类问题。

应用上容器方案

01.上云实施策略定义

实施策略如下：

· 先试点，后推广。上云优先选择业务量小、非核心、涉及周边系统接口少的应用。经过试点验证，再逐步推广。

· 先 IaaS，后 PaaS。优先选择适合 IaaS 的应用上云。原因在于，按照当前项目的项目技术方案，PaaS 部署与 IaaS 之上，并且 PaaS 可能或多或少涉及到应用改造。

· 先开发测试，后生产。在开发测试环境经过完整的新建 / 迁移验证，再实施生产环境。

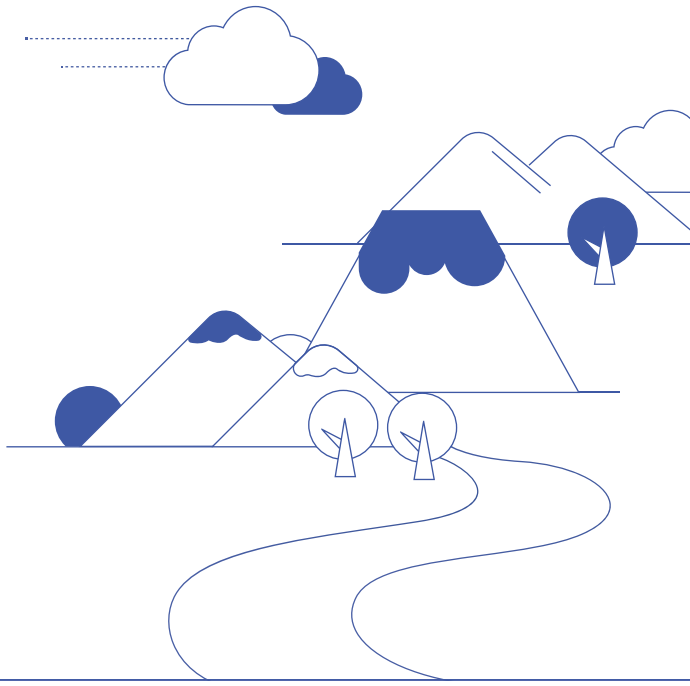
02.上云流程的裁剪

现场调研，结合需求方具体情况，与需求方共同制定有效、便捷、可行的流程方案。

03.上云详细的实施计划

与需求方共同商定。

应用上容器后的变化



01.部署流程的变化

传统的运维模式是先申请并确认资源，然后再部署应用。而在容器类应用部署过程中，资源分配是由系统自动完成的，只要整个容器集群仍有剩余资源，就不需要申请资源。或者说只需要申请部署即可，因为部署和资源的获得是一个步骤自动完成的。

02.资源与应用绑定的习惯变化

传统的运维管理中，资源与应用是绑定的，企业可以清楚地规划每台机器跑哪些应用，包括每个应用占用多少资源，包括应用的 IP 地址都是提前规划好的。而应用容器化之后，资源与应用已经解绑，无法严格进行资源和应用的绑定规划，如 IP 地址与应用的绑定，物理机与应用的绑定等，企业传统的运维方式和习惯需要进行相应的调整。

03.问题解决方式的变化

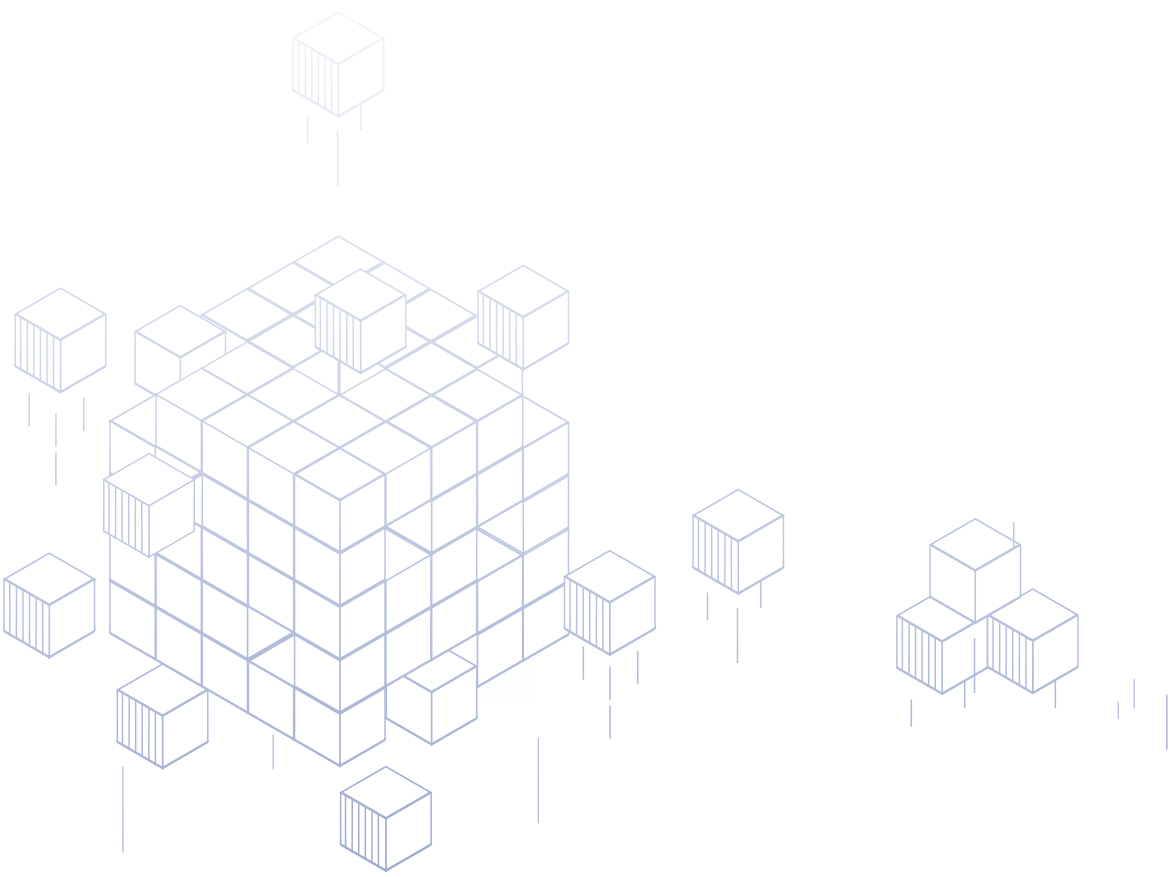
基于容器的特性，在应用出现问题的时候，最合适的方式就是重新启动一个新的容器（很多时候由系统自动启动）。先解决问题，然后再去找应用出现问题的原因并解决。另外，传统的登陆到系统内部（比如 ssh ）查看相关信息的习惯也面临挑战，当然我们也提供 ssh 到容器内部的方案。

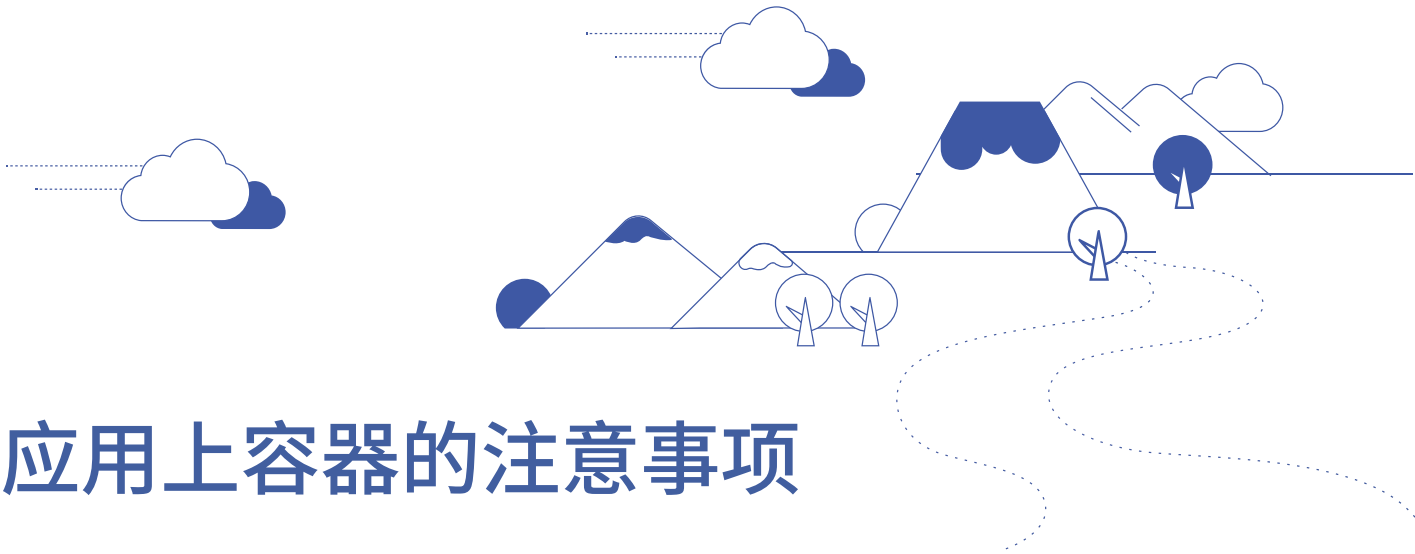
04.应用扩展的变化

过去应用扩展需要经历资源分配、安装、测试、上线的过程，而使用容器之后，一方面可以自动根据负载进行应用扩展，另一方面也可手动一键方式完成应用的扩展。

05.升级回滚方式的变化

过去的升级主要基于脚本和稳定来完成，周期长，风险大。在使用容器后，基于持续集成和自动部署的能力，可实现应用的一键自动升级，而应用的回滚也只需要一键即可完成。





应用上容器的注意事项

在应用迁移过程中，会存在以下几点问题和困难。如果企业无法解决以下问题，则不建议此应用上容器。

01.无状态化改造问题

通过无状态化改造，解耦业务逻辑与数据。

- 根据业务架构拆分应用，将业务逻辑处理部分作为无状态部分，将状态保存在有状态的中间件中
- 无状态的部分可以很容易进行横向扩展，接受业务分发的能力得到扩展
- 有状态的部分依赖中间件本身的状态保持机制完成，如消息队列、缓存、数据库、对象存储等

02.系统间的依赖问题

对于一个大的系统中，一般情况下都存在各个不同的系统间的依赖问题，系统容器化之后，由于 Docker 的网络特点，系统的对外访问信息只有在容器启动之后才能生成，这就对应用的部署和弹性过程产生了影响，需要考虑并解决系统间的解耦问题，尽量做到镜像与环境无关。如，将服务 IP 改成域名。

03.容器化与非容器化的并存问题

在一个大的系统中，经常会出现有些组件被容器化，另外一些组件没有容器化的问题，需要解决容器化组件和非容器化组件的协调问题。

例如，注册中心需要容器集群内外的组件提供服务，一种方案可以考虑使用 underlay 网络直通的方式实现服务注册；另外一种方式需要修改容器内服务的注册信息改成 ingress IP 或者 noderport 的方式。

04.外部存储的问题

在传统的应用开发模式下，如果应用需要使用持久化存储，我们可以把应用的目录挂载到远程存储服务器。

在应用容器化后，应用的持久化存储可以借助 kubernetes 的 pv 实现，只需要针对应用的持久化路径进行 PV 资源的配置，应用在使用 PV 后就可以实现持久化存储。应用代码不需要做特定的修改。

05.多进程的问题

容器本来就是用来运行单个应用的（比如 http daemon，应用服务器，数据库等等），如果一定要在一个容器里跑多个应用进程，那么考虑使用博云裸金属容器能力，以虚拟机的方式去管理容器，实现一个容器内运行多个应用进程。

06.安全性的问题

公网上的镜像普遍存在病毒或漏洞，因此需要一个安全可靠的基础镜像，为后续制作应用镜像提供基本防护。

另外，由于容器镜像运行时间普遍不长，传统的边界安全的漏扫软件难以及时发现病毒或漏洞；同时，通过容器逃逸渗透到虚拟机或物理机中危害同样巨大。因此，在容器上生产时，建议配套使用专业的容器安全产品进行防护。

07.镜像制作的问题

在具备基础镜像后，应用镜像由人工制作费时费力，建议使用流水线或 DevOps 平台自动化将代码构建成应用镜像，降低迁移对开发人员的影响。

08.工作流程的改变问题

在引入持续集成和自动部署功能后，开发人员的版本提交流程会发生一定变化，建议通过相关培训让开发人员快速使用变化后的流程。

江苏博云科技股份有限公司

地址：苏州工业园区星湖街328号创意产业园7-9F

联系电话：400-991-5335

邮箱：support@beyondcent.com

网址：www.bocloud.com.cn

博云官方服务号