.Net 大户的选择： Windows Container 在携程的应用

dataman · 2017-01-19 16:42:03 +08:00 · 4793 次点击

数人云上海&深圳两地“容器之 Mesos/K8S/Swarm 三国演义”的嘉宾精彩实录第四弹！小数已经被接连不断的干货搞晕了，沉浸技术的海洋好幸福~Windows container 在国内的实践还比较少，携程作为.Net 大户，率先进行了调研和实践应用，将其中的成果与大家分享。

> 罗勇，携程云平台开发经理 主要负责携程云平台建设和维护，熟悉 OpenStack， Docker，Linux/Windows Container 等技术领域

今天的主题是 Windows 容器。今年下半年携程开始了对 Windows container 的调研工作，目前已经有一些成果和实际应用案例，与大家分享一下，主要内容有：

- 携程为什么要使用 Windows container （可能国内大部分的人了解 Windows container 少一些，特别是具体的实践，分享重点会偏 Windows container 细节）
- 传统.Net 应用容器化实践
- 容器存储&网络&编排

携程为什么要做 Windows container ?

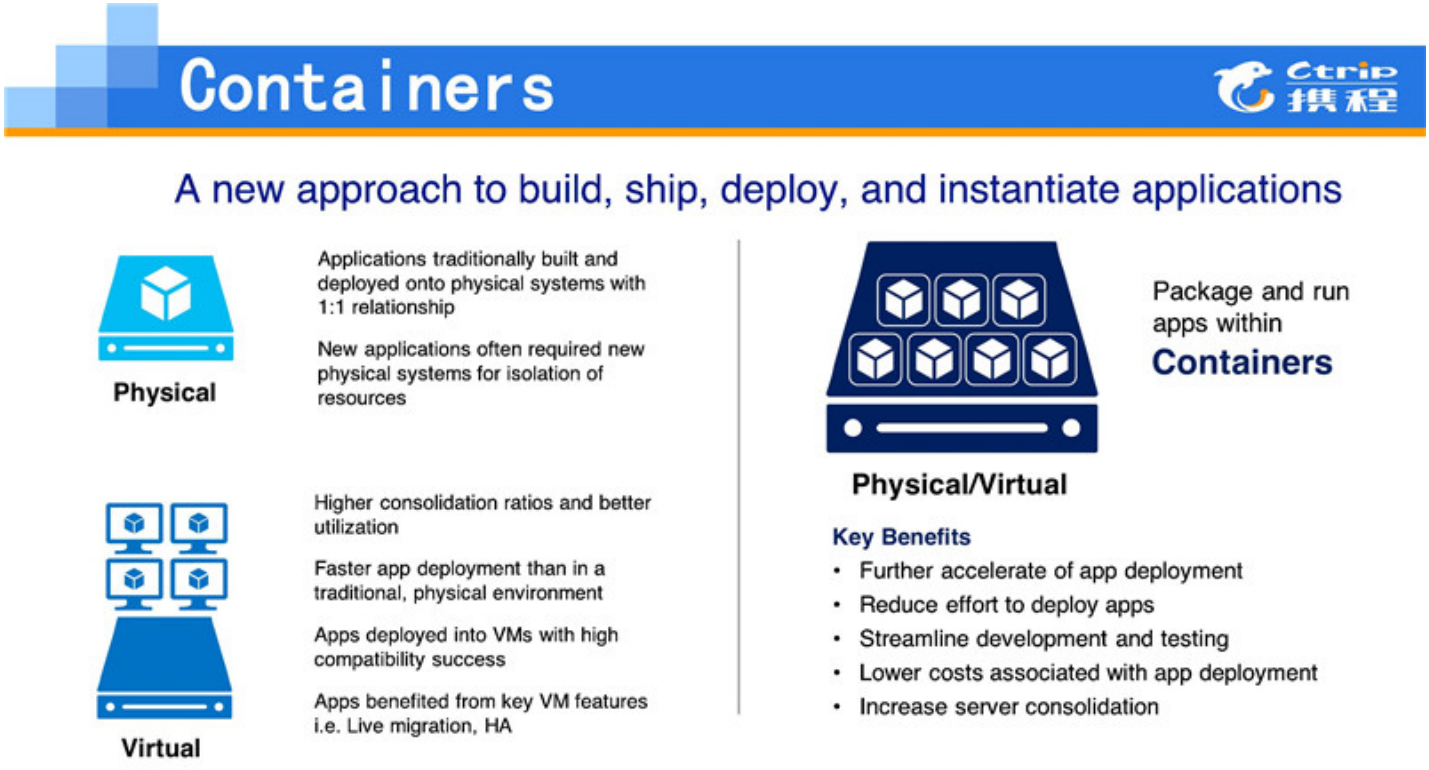
现状：

- 携程是.Net 应用大户，由技术栈决定的，早期携程整个应用架构都放在该平台上，线上跑了 3000 多个核心应用，覆盖了 20 多个 BU （业务部门），这让我们不得不关注这一庞大的系统。
- 平台要往 java 方面转，去分享 java 的红利，但是.Net 线上应用不可能都重写；
- .Net 的应用目前 90%左右的应用都跑在虚拟机上，从虚拟机自身来看，粒度太粗，对资源的使用率还不是很好。
- 持续发布，应用上线，从拿到机器环境准备好环境上线生产，虚拟机模式下周期长，扩容慢。

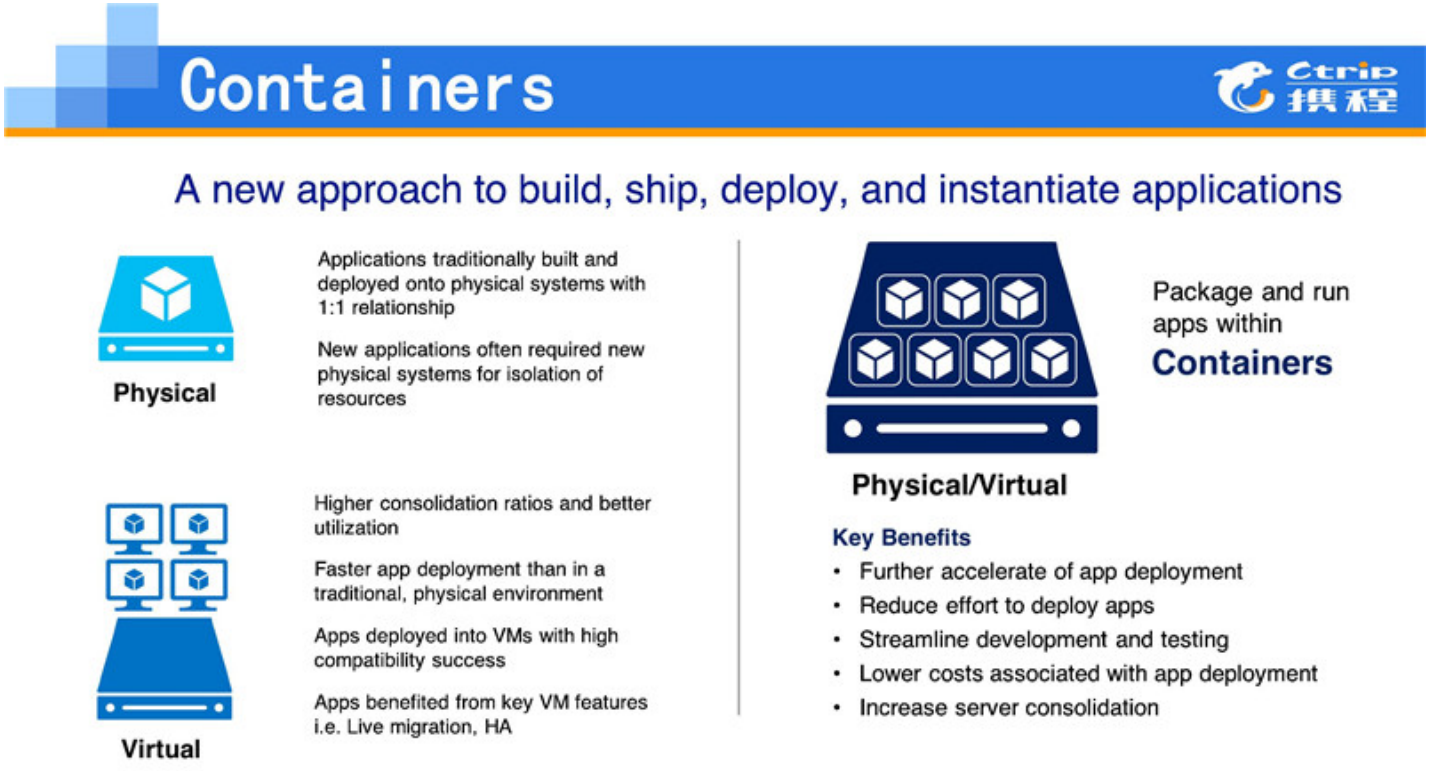
新一代发布平台的需求，希望缩短环境准备时间，做到秒级部署， Linux 平台的应用非常容易做到，但是 Windows .Net 应用在这方面支持比较难，另外，为了确保生产和测试环境高度一致，希望应用发布是单一应用、单一镜像的，最好是一个容器尽量少的包含系统进程，这样可以把资源隔离的粒度控制在小范围内，尽量榨取宿主机的资源，同时希望 Linux 容器和 Windows 容器的方案尽可能接近，比如网络、存储，不需要两套不一样的方案或是有大的有变化。

一些 Windows container 的技术细节

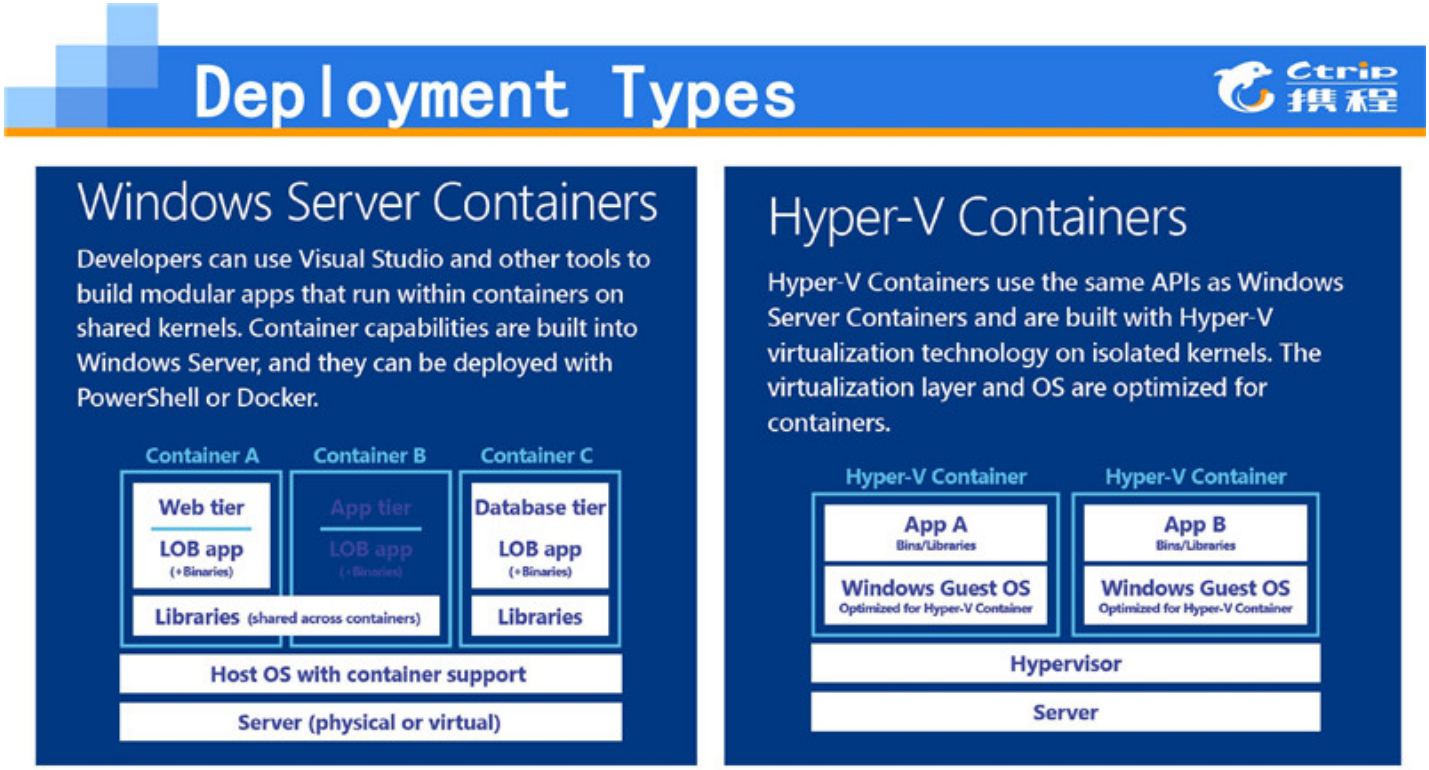
最开始的时候携程用物理机部署应用，为了保证互不冲突，用户在一个物理机上只部署一个应用。后来认为此举太浪费，就部署了多个应用，但是为管理带来了麻烦，应用之间有一定的冲突或者相互影响。之后有了虚拟机，虚拟机上可以部署更多的应用，而且隔离比较好，不过虚拟机资源隔离的粒度太粗了，于是容器走了过来，能做到把一个应用打到包里，这个包涵盖了环境配置等， run 起来可以只是一个进程，又具备一定的隔离性，同时把资源使用的粒度控制足够的细。



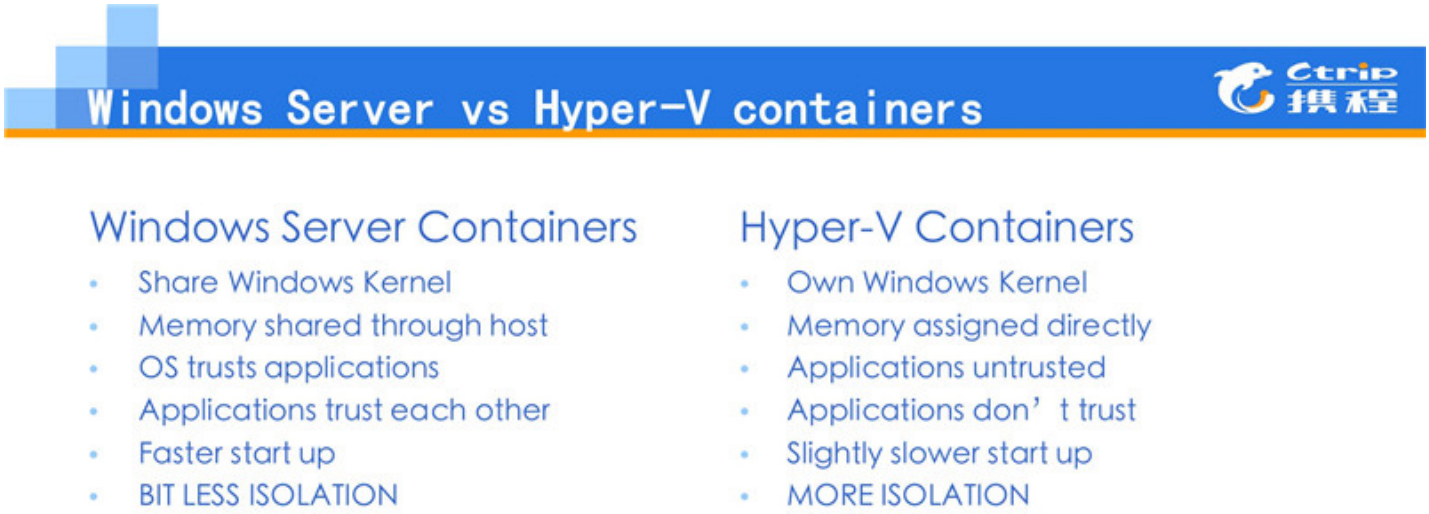
Windows container 目前支持的系统是 Windows server 2016 ， 这个版本是去年 10 月份正式发布的（携程是国内比较早的一批拿到了他们的 RTM 版本） ， 支持两类 server ， 一类是 server core ， 另一类是 nano server 。 nano server 是微软比较推荐的一类服务器系统，启动非常快，可以大幅度缩短计划内维护宕机时间，通常几秒钟就起来了，不包含硬件检测的时间，几十秒都能够起来。这块携程还没有用于生产环境，目前只测试了用 server core 作为容器宿主机的系统的情况。需要着重提一下的是，如果宿主机打了补丁或者升级，容器也要对应的做补丁或者升级。当然不一定说立马做补丁升级，但一定要比较精确的找到对应的版本做升级。



Windows container 有两种 container 类型，这两种容器都是跑到 Windows servrer 2016 的，但还有一种容器的玩法是在 Liunx 平台跑.Net core ， 这种方案我们也看过，大家很容易想到它的局限，其实只能跑到用.Net 技术开发的 Windows 的应用，一些非.Net 的应用不支持，因此这个方案被 Pass 了。直接在 Windows server 跑容器的方案更为靠谱，该方案有两种类型， Windows server 和 hyper - v container 。



有人会问， hyper-v 不就是一个虚拟机的技术吗？对，其实它有点像虚拟机，但是 hyper - v 的技术略有不同，速度会明显比虚拟机快很多，只是在申请资源或者获取资源时，比 Windows server Container 的速度稍稍慢一点点， Windows server container 可能 3 秒，它可能 4 、 5 秒。但是资源的隔离度比较好一些，类似于虚拟机，微软公有云 Azure 的容器服务也是采取这种容器类型，他们的考虑是公有云上面部署的应用不是受信任的，相互之间有可能“打架”的情况发生，他想隔离好一些。

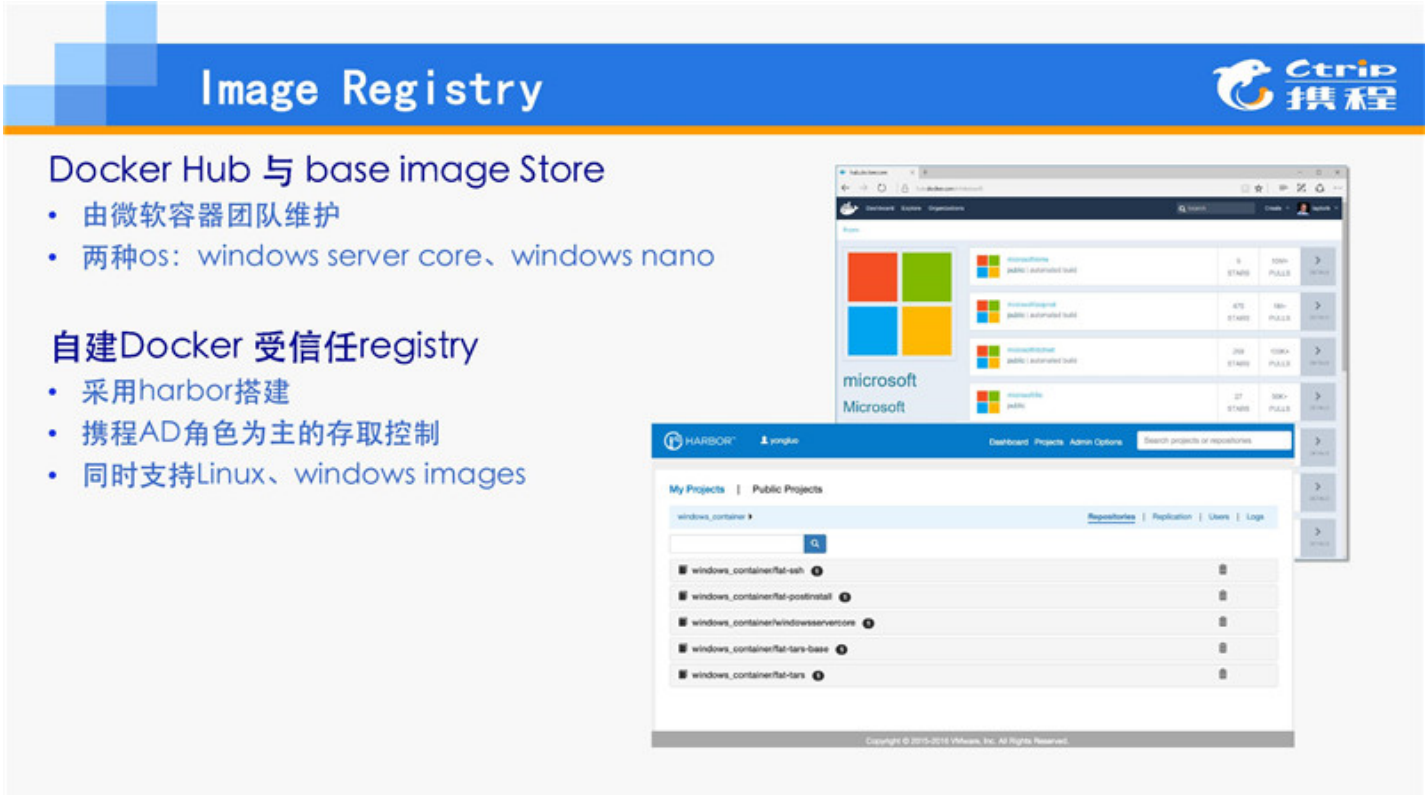


另一个区别， Windows server Container 的内核是共享的，可以在宿主机上看到每个容器里面的进程，这与 Linux 容器相似，可以直接 kill 掉。 hyper-v container 宿主机是看不见容器内进程的，像一个虚拟机一样。此外，内存资源隔离不同， Windows server Container 内存可以 share ， hyper-v container 不能 share ，hyper-v container 一旦分配就不能重新进行修改。对系统应用是信任的，这种比较适合做私有云的一些产品，因为在应用上跑的什么东西，这个应用能干什么坏事或者是相互之间有没有影响，都可以控制，但是公有云不能

这样做，应用使用率很高，会把别的容器影响到。启动速度上也会有差别，一个启动快，一个启动慢一点，当然并不是特别慢。

容器镜像，这个和 Linux 容器的镜像类似，可以分层。最下面一层是基础镜像，但是基础镜像和 Linux 有区别。Linux 镜像可以自己搞，弄一个系统把它做成镜像，但是微软没有办法自己做一个 Windows container base 镜像。或者说现在只是 Windows server 2016 的镜像，想跑一个 2012 的系统是不行的。当前系统内核只能支持 win10 ，在上面可以继续安装想要的东西，比如接着安装 Framework ，然后在最上面装应用。

镜像构建也是一样， Windows container 容器和 Docker 集成比较好，可以用 Docker 工具的一些命令进行 build ，用 Dockerfile 来 Build 一个镜像。 registry 是镜像可以直接 push 到一个平台或者是私有的 registry 上面去，通过 Docker pull 方式拉下来， Docker run 跑起来。



Windows container 的镜像，可以在 Docker 网站上可以找到关于 Windows container 的一些 base image, pull 下来大概有 8G 左右，在外网上下载可能要两天。大家可以尝试一下。也可以建私有的 registry ，携程采用的 VMware 开源的 Harbor 方案，本身没有做太多的修改，直接可以用。和携程的 AD 整合以后基本上能用了， registry 可以把 Linux 和 Windows 的镜像都放在一起，两边都能用，都能管，这部分省掉了很多的内容，不需要做额外的开发，这样 Windows 和 Linux 的平台的 image 管理方案是一致的。

传统.Net 应用迁移

迁移背景：

之前提到携程有 3000 多个.Net 应用，这些应用每天要不停的发布、测试、编译打包，是一项很大的繁琐工程，有个叫 “ build ” 的项目负责这个事情。最初这些跑在虚拟机里，资源使用率很低，白天很忙，晚上使用率很低，有一定的资源浪费，且构建环境也经常不一致。为了积攒容器应用使用的经验，我们考虑把 build 项目先容器化，也就意味着.Net 应用自己的编译在容器里面编译，看能撞出来什么样的火花。

原来写几千个应用的编译脚本，如果改了一些东西，变更维护的代价是非常大的，尽量这个方案不要用到原来以往用的工具和使用方式，不去动它，最好能够拿过来不怎么修改，就跑起来。另外，重点看一下像 vs2010 和 vs201 这样的工具能不能在容器里跑，实践证明是可以的。然后看 MSBuild 在容器里面是否兼容,支持不同的.Net Framework 版本，这些都是比较通用的软件，结果是这些功能都能够支持，另外也包括 python 、 MVC 、 GIT 等等。

迁移收益

- 首先环境，编译的环境高度一致，每个环境没有太多的区别，容器拉起来直接跑，提高了编译成功率。
- 其次资源利用率提高了，我们把虚拟机资源砍掉了一半，就只需要两台宿机机搞定整个携程 3000 多个.Net 应用的编译。
- 编译时长也缩短了，原来用一次构建平均要几分钟，现在 90 秒左右基本上能构建完成。

待解决的问题：

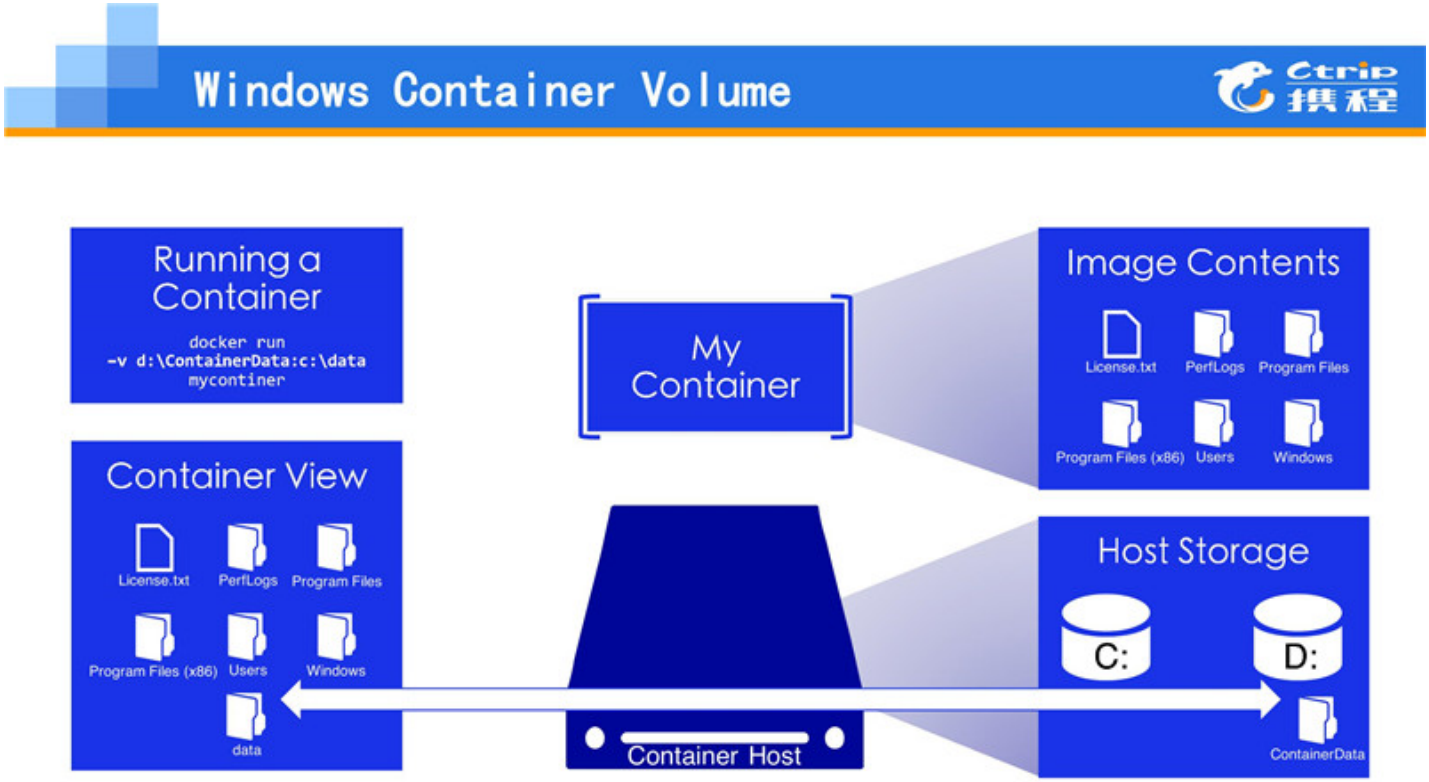
- 图形不支持，这个是某些企业想用 Windows container 的大问题，它本身图形不支持。后台程序没有太大问题，不过有一些依赖图像工具比较难支持。
- 旧应用兼容不是很好。比如遇到 MGwin 编译出来的包，一旦代码中有调用标准输出的语句程序直接就挂了，遇到这类问题，需要把源码拉下来重新编译，比较有难度。
- 不支持 RDP ，远程桌面是不能用的，那怎么做到远程访问呢？ 还好 Windows 现在支持 SSHD 安装了，只需要容器内装一个 SSHD ，然后远程 SSH 去，当然可以用 powershell 远程的登录方式，两种方案都可以用， SSH 方案更统一一些， 如果用户当前正在 Linux 平台上工作，突然想登一个 Windows 的容器怎么办？当然也可以用 linux 平台的 powershell 工具实现远程登陆容器。
- 不支持 D 盘。携程迁移过来很多老的应用是要装在 D 盘的，容器拉起来没有 D 盘，只有一个 C 盘。本身 Docker 有一个 volume 功能，可以挂一个数据的盘，问题是这样会导致在宿主机上留下一些东西，和宿主机产生耦合，如果容器删除或者迁移，宿主机上就留下了脏数据。后来我们为 D 盘做一个 link ，相当于 D 盘可以快捷的方式连到 C 盘，映射到 C 盘的某个目录，这样数据都是落地到在容器的磁盘上，如果想在别的地方拉起来这个容器，可以直接 push register ，就可以在别的地方部署且环境一样。

存储&网络&编排

接下来 Windows container 容器的存储，网络，编排方面的技术与大家分享一下。

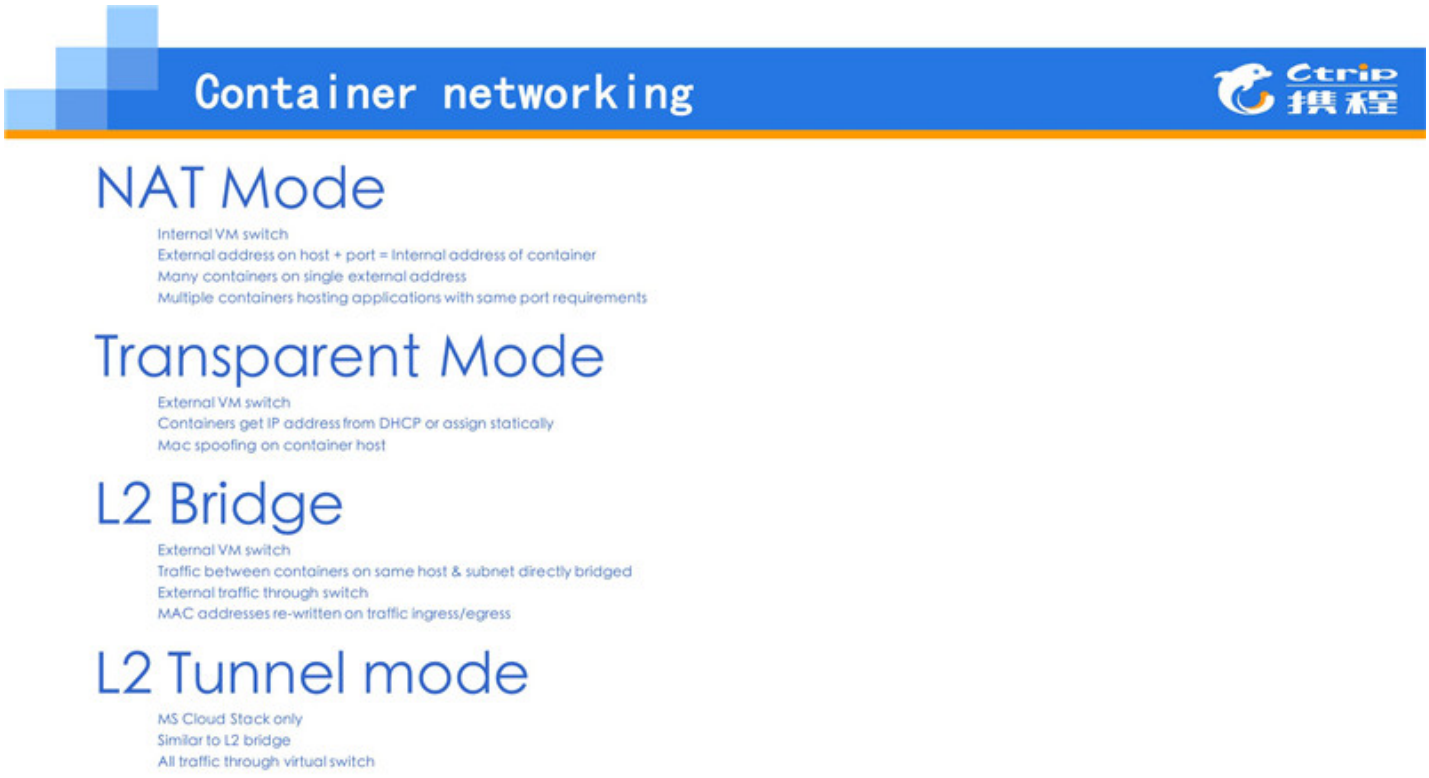
Windows container 资源的隔离方式和配置管理 API 是借用 Docker 规范，设计理念和 Linux Container 类似，也支持 CPU share 的这种方式去控制资源的分配。内存可以通过 quota 的方式去分配内存， Disk 也能够充分应用到 IO 的带宽，这一块还没有做非常多严格的测试。关于网络的支持，携程做了很多测试，整体来讲比较不错，问题较少。性能也满足需求，多个容器在一个同一个宿主机上也能尽量用到整个宿主机的带宽。

容器的存储



存储有三种：一种是镜像，镜像本身是一个存储，设计之初定义就不是一个永久的存储，当前容器存储拉掉那个存储就没有了，也不是设计安全的。另外一种存储是 volume （卷），可以挂一个数据盘到某一个容器上，在容器里扩展存储空间。同时多个容器也可以挂载宿主机上的一个同一个 volume （卷目录），这样大家可以实现 NFS 一样的效果。最后一类存储是网络存储，比如可以用 SMB 的方式挂网络盘在容器里面使用，里面如果有万兆的带宽支持还可以玩一下，如果没有万兆带宽的话就不要玩了，它只能放一些冷数据。

容器的网络



相对来讲复杂一些， Windows 支持有四种网络模型，第一种 NAT 模式大家比较熟悉，起一个本地或者是数据本地的 IP 地址，如果你想外网访问的话，把 Docker 映射出来，这种方式比较适合做一个 JOB 类型的应用在上

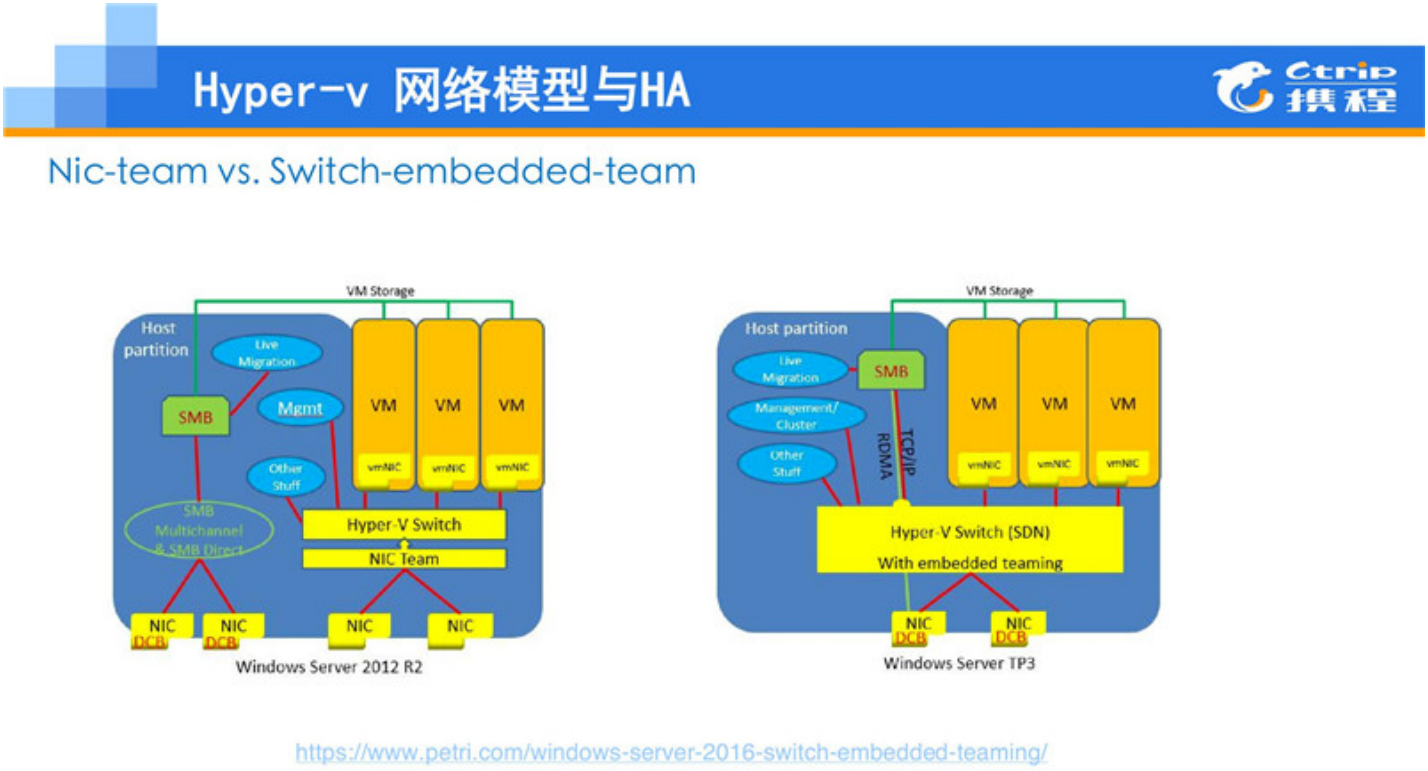
面，不需要外边可以访问它，但是容器里面可以去下载东西。之前讲的 build 项目就是用这个网络模型，非常简单，不需要考虑太多网络的模型就可以直接用。

第二种是 transparent 网络模型，这种模型是现在主要用于生产的模型，首先它是通过 mac 地址伪装实现数据透传，对网络的性能本身折损也比较少，它也支持把多个工作网卡绑到一个交换机上，然后把这个交换机给容器用。网络模型在容器宿主机以外的机器上看到 Windows 容器和一台物理机没有什么区别。

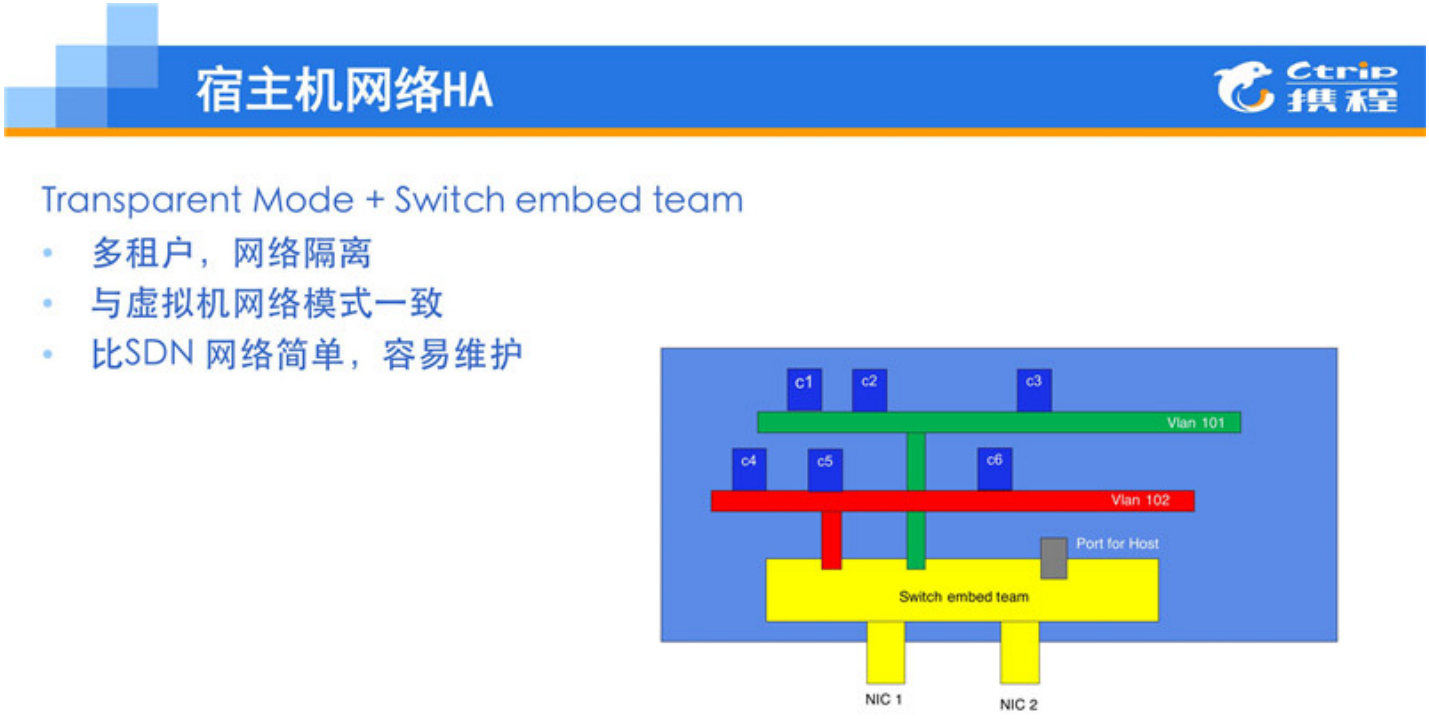
还有一种是 L2 bridge 这种采用 openstack 网络的 Flat 模式，所用的网络跟宿主机的网络是一样的，和宿主机在同一个网段，这样有很大的局限性，网络和宿主机混在一起没有办法做到多租户隔离，然后网段用光了就完了，适用于比较小的集群。

最后一种是 Tunnel mode ，没有太多研究这一种，但是微软 Azure 用的这一种网络模型。本身携程为了和虚拟机的很多的品牌网络模型一致，所以这一种没有那么快的推进。

hyper-v 的网络模型



Hyper-v 宿主机是 2012 上面 hyper-v 的网络模型，之前要求一台宿主机尽量要 4 块网卡，为什么要用 4 块网卡？两块给虚拟机用，另外两块做一些管理，比如对存储用，虚拟机迁移等，可以做宿主机的管理。另一种方式， 2016 建议的一种网络方式，这里面有一个叫做 embed team ，内嵌交换机，它的好处是把下面无论是两块还是 N 块网卡都可以绑在一个 bound ，然后把这个 bound 放在一个交换机里面，每个容器 port 全部放在交换机里面，然后容器给 port 打相应的 vlan tag ，这样容器的网就通了。



embed team 的好处是不需要要求宿主机一定要有这么多的网卡才可以用，另一个好处是对这些不同的 vlan 之间做一些流量的控制，携程的 container 的网络模型也是基于嵌入式交换机上实现的。把宿主机至少两块网卡做了 bound，放在一个 embed team 里面去，另外加一个 port 给宿主机做管理网卡。容器宿主机相比虚拟机宿主机简单，没有存储和迁移的需求，就不要以额外的划分网络了，如果需要为容器的存储单独挂一个网络的话可以加一个 Port 做这个事情。不同的网段的这种容器在上面可以再创建不同的 Docker，加不同的 port，然后容器在里面可以互通，这样的好处就是既实现了多租户、实现了网络隔离，同时和虚拟机包括 Linux 上面的网络模型是一致的。

容器编排

编排这块通常一个容器部署到多个宿主机上，同时一套应用下来有数据层、业务层、也有 web 层，这些应用要分开放，它们中间放在哪里需要有一个地方，把整个管起来，也希望这个东西能自动化，本身做编排这些，无论是 Swarm 、 K8S 、 Mesos 都是需要解决的问题。一种方案是用 Docker compose 这种方式，适合于单宿主机管理。想编排一下容器，看如何跑，这种方式用 Docker compose 就能解决。当然，微软现在对 Docker Swarm 支持好一些，实现成本比较低，基本上能管，但是性能方面没有做太多的测试，目前一些基本的调度、主机分类等等都能用。

为什么携程选择使用 Mesos ?

因为携程的 Linux 平台用的也是这套方案做的编排相关的管理，希望有一套方案能够尽量两种容器一致，于是我们的方案采用 Mesos 加上 Marathon 对它进行管理，本身它也有一些现成的工具，比如 UI 等现成的工具都可以用，这部分还在进行测试和研究中。官方下来的包不能直接跑到 Windows server 上面，要拿下来重新编译才能用。最终携程是想做到这么简单的一个容器的管理的架构，就是说希望 Mesos 在里面能够同时管 Linux 容器和 Windows 容器，对它进行统一的调度，最大限度的优化这种调度策略，提升使用率，这是最终整体的设计理念。

待解决问题

有一些急迫解决的问题，与大家交流一下。首先 Windows container 的镜像比较大，在生产环境，如果批量 pull base image ，网络的带宽会很快被打满，会对业务带宽造成影响。我们需要有一套方案来解决这个问题，如何能够比较“经济”的方式把 Based image 或者变更的 Layer 文件下发下去，是后续要解决的问题。

Windows container 的监控日志，没有现成的方案，我也有与微软团队交流过，这部分文档非常少，携程之后也会重点解决监控问题。

推行单容器、单应用的发布方式，希望后面能够把各种 FAT/UAT/Prod 环境之间打通，都可以通过 Windows 容器方式，秒级发布。

携程有 3000 多个应用，一旦容器跑起来了，宿主机的规模还是比较可观的，这种情况下，大规模容器如何管理好？这也是后面需要解决的问题。

这是我今天给大家分享的主要的内容，谢谢大家!