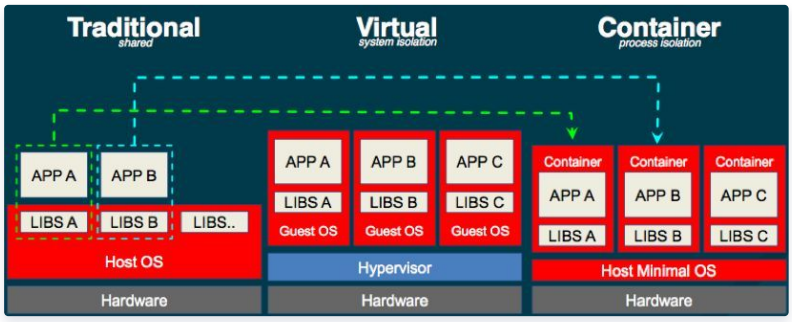
容器在公有云上的落地姿势

2019-06-28 阅读 258

1.容器天生隔离能力不足

1.1 容器是一种进程隔离技术，并非虚拟化技术



容器（container），并不是一种虚拟化（virtualization）技术，而是一种进程隔离（isolation）技术，从内核空间、资源和安全等方面对进程做隔离。

Linux 容器采用 Linux 控制组（cgroups）和命名空间（namespace），其中，cgroups 定义了一个进程能使用什么（CPU、内存、网络等资源），namespace 定义了一个进程能看到什么（uid，gid，pid，mount，filesystem等）。一方面，并非所有系统资源都可以通过这些机制来控制（比如时间和Keyring，https://blog.jessfraz.com/post/two-objects-not-namespaced-linux-kernel/）。另一方面，在Linux 容器中运行的应用程序与常规（非容器化）应用程序以相同的方式访问系统资源；直接对主机内核进行系统调用。内核以特权模式运行，允许它与必要的硬件交互并将结果返回阴应用程序。因此，即使使用了很多限制，内核仍然面向恶意程序暴露出了过多的攻击面。

除了cgroups 和 namespace，Linux 容器还会使用到象 seccomp 这样的技术。seccomp是内核防火墙，限制一个进程对内核系统调用（systemcall）的访问限制，能够在应用程序和内核之间提供更好的隔离，但是它们要求用户创建预定义的系统调用白名单。在实际中，很难事先罗列出应用程序所需的所有系统调用。如果你需要调用的系统调用存在漏洞，那么这类过滤器也很难发挥作用。

因此，容器被认为不具备和虚拟机以及沙盒（sanbox）一样的隔离能力。关于容器、虚拟机和沙盒之间的区别，Jessie 的这篇博文（Setting the Record Straight: containers vs. Zones vs. Jails vs. VMs ）给出了很好的解释。

1.2 Kubernetes 的多租户隔离

Jessie Frazelle（他的博客地址为 https://blog.jessfraz.com，强烈推荐）将多租户隔离模式分为两大类：

- 弱隔离（Soft multi-tenancy）：同一个组织中的多个用户使用同一个集群。这种隔离模式中，因为用户处于同一个组织中，因此互相之间默认是信任关系，但是也存在可能的情况，比如有恶意的员工。这种隔离模式的主要目的就是为了防止这种恶意事件。
- 强隔离（Hard multi-tenancy）:来自不同组织的多个用户使用同一个集群。这种隔离模式中，默认就假定所有用户都是潜在恶意的，因此这种模式的主要目的是阻止租户之间的互相访问。

从上面的定义可以看出，基本上，私有云的隔离模式是弱隔离模式，而公有云的隔离模式是强隔离模式。

因为容器天生隔离不足，如果只是采用传统 Linux 容器的话，公有云往往采用每个用户单独创建 Kubernetes 集群的方式来实现强隔离：

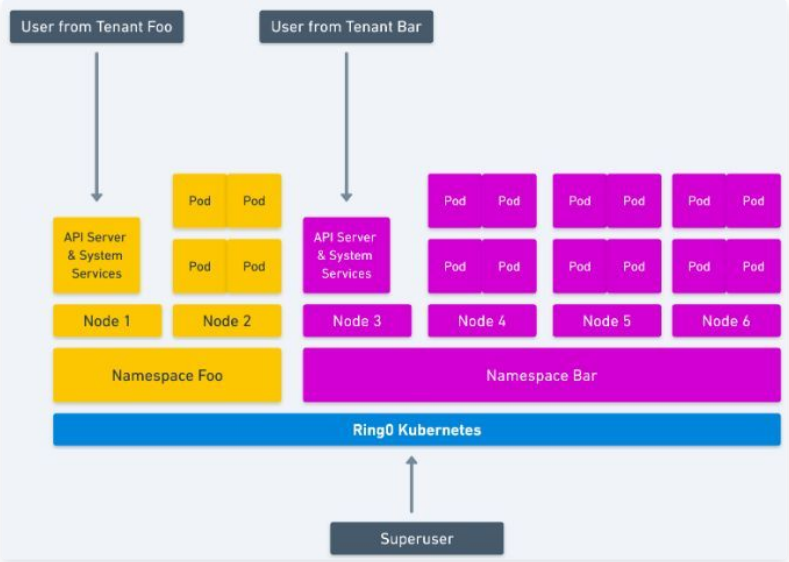
作者介绍

SammyLiu

关注 专栏

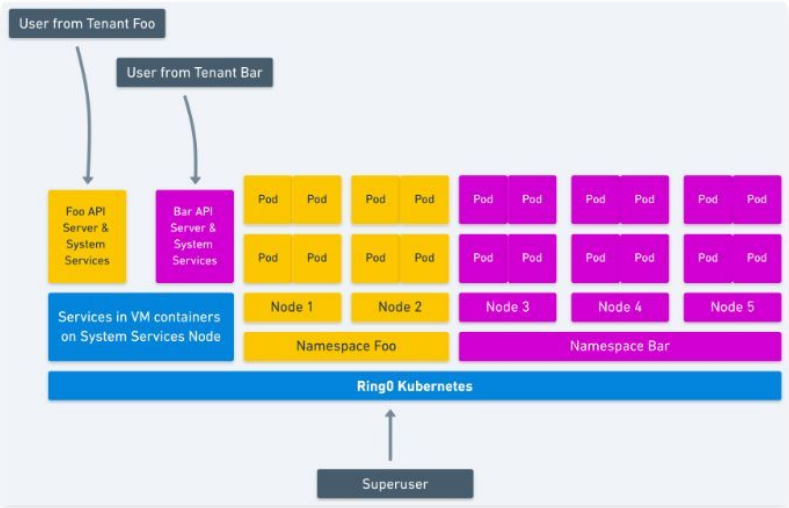
文章 55 | 阅读量 14.7K | 获赞 130 | 作者排名 3126

目录



Jessie Frazelle 的这个图是假设 K8S 能够在不同的宿主机上创建和管理不同的K8S 集群（那时候 K8S 真的成为集群操作系统了）。实际上，当前这种角色往往由公有云自己的云管平台实现，然后在若干台虚拟机或物理机上为每个用户搭建完整的Kubernetes集群，每个集群利用传统的Linux 容器来运行客户的应用。因为传统Linux容器的隔离性不足，每个用户的容器必须允许在独占的环境中。

但是，如果把运行环境从 Linux 传统容器换成微虚机（比如 kata container）的话，因为微虚机本身具有的强隔离能力，则可以在一个宿主机上创建不同用户的这种运行环境，此时这些环境在集群中是混部的。

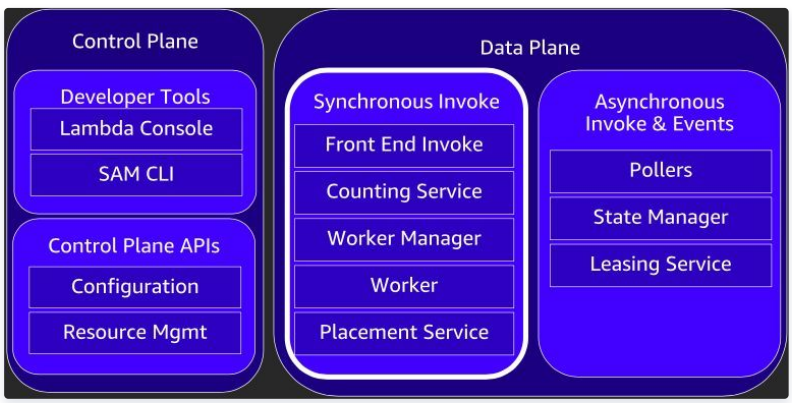


2.容器在AWS 上的落地方式（以Lambda为例）

AWS 上多个服务都利用到容器，比如 Lambda 利用了传统Linux 容器，而 ECS 和 EKS 则利用了 Docker 容器。以 Lambda 为例，我们来看看过去和现在容器在AWS上的落地方式。

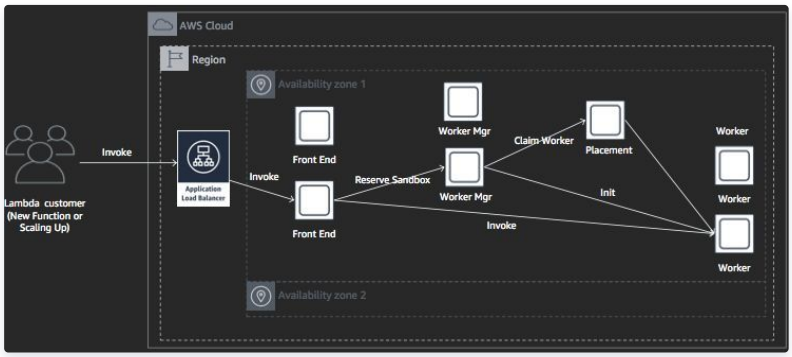
2.1 过去容器在Lambda 中的落地方式 - 用户函数运行在独占的EC2虚拟机中的Linux容器中

下图是 Lambda 的技术架构：



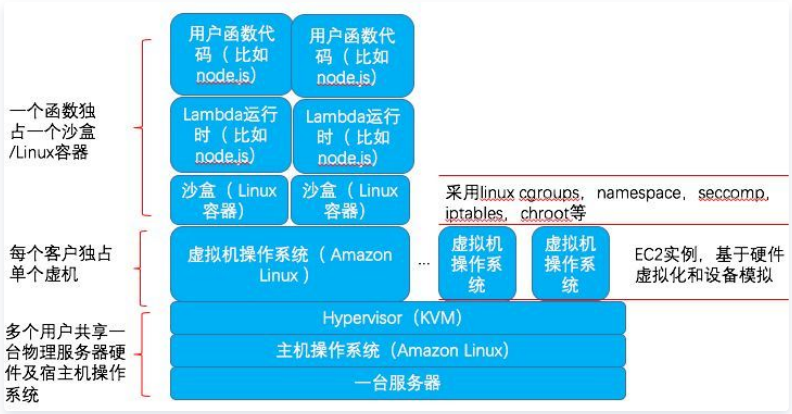
从名字上基本上就可以看出来每个组件是干什么的。其中， 一个 Worker 就是实际运行用户函数的一个安全环境。之前，一个 worker 是一个 EC2 实例，其操作系统为 Amazon Linux。

下图是 Worker 被创建以及函数被调度到 worker 中的基本流程：



其中，Worker manage 负责 worker 的创建和管理，Placement 服务负责将用户的函数调度到某个或某些worker 上运行。

此时Lambda的强隔离模式的实现方式如下图所示：



这个图还是简单明了的，具体就不解释了。

引用 AWS Lambda 团队工程师所说的，基于CPU的硬件虚拟化技术是AWS上用户之间隔离的最低要求。因此，和 AWS 上很多利用容器的服务一样，Lambda 也利用了 EC2 虚机来实现用户之间的强隔离。

但是，其局限也是显而易见的，比如：

- 资源浪费：用户的一个简单的测试函数也会占用一个虚机）
- 管理复杂：需要管理复杂的资源和安全模式
- 启动速度不够快：因为EC2虚机的创建时间原因

2.2 现在容器在Lambda 中的落地方式 - 用户函数运行在Firecracker微虚拟机中

亚马逊在2018 年 re:invent 大会上宣布了一个新的开源项目 Firecracker，并已经用在 Lambda 和 Fargate 服务之中了。 Firecracker 是一种采用基于 Linux 内核的虚拟机 (KVM) 技术的开源虚拟机监控程序(VMM)。 Firecracker 负责创建和管理微虚机（microVM）。 Firecracker 微虚机提高了效率和利用率，内存开销极低，使得在一台物理服务器上可以创建数千个微虚机。后文下面再介绍。

使用Firecracker后的 Lambda 隔离模型：

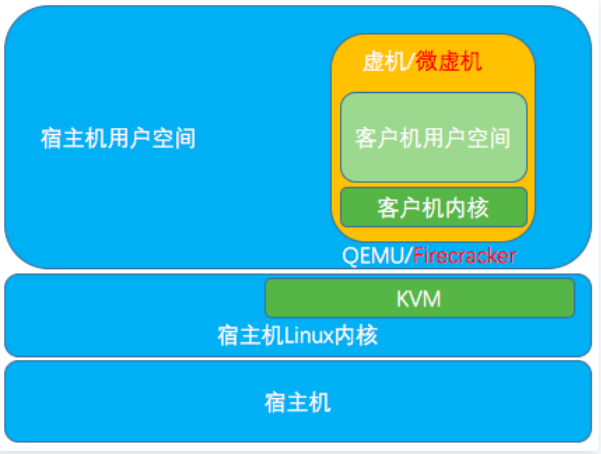
其好处也是不言自明的，比如：

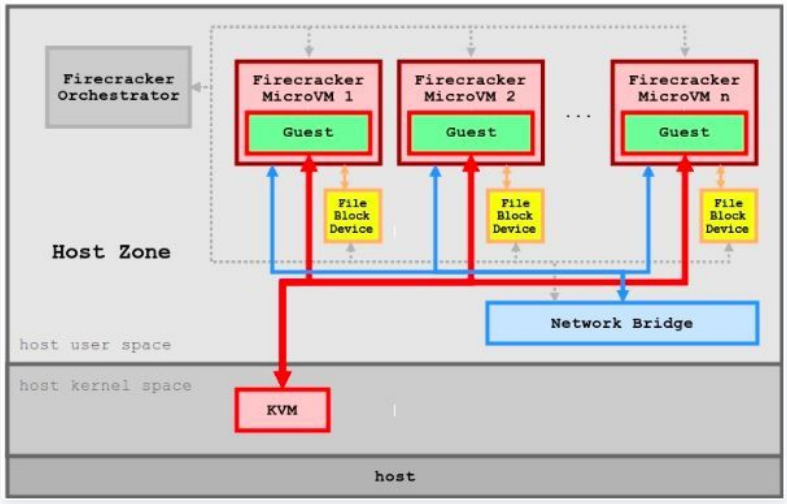
- 利用CPU硬件虚拟化实现了用户之间的强隔离。
- 提高物理硬件资源利用率。
- 缩短函数运行的启动时间。
- 简化了安全模型。
- 简化了 Lambda 编程模型。

3. Firecracker 是什么

Firecracker 的中文意思是『鞭炮』。顾名猜意，不知道AWS是不是认为在公有云上运行容器就像放鞭炮一样，看起来绚丽多彩，但是弄不好就会引起火灾。

简单地可以将 Firecracker 看做被大大简化了的 QEMU。它和 QEMU 一样，利用 KVM，负责创建和管理虚机。因为这种虚机面向 Serverless 这种场景，适合于运行暂时性（transient and short-lived）进程，因此被称为微虚机，即microVM。





因为公有云对微虚机的要求是具有象常规虚拟机一样的隔离能力，同时还有象Linux 容器那样的轻量特性（硬件开销小，启动快）。因此，Firecracker 的设计思路是：

- 内置安全性：提供了支持多租户工作负载并且不会被客户错误禁用的计算安全性屏障。客户工作负载被认为既神圣（不可侵犯）又邪恶（应当拒之门外）。
- 轻量虚拟化：重视瞬时性或无状态的工作负载，而非长时间运行或持续性的工作负载。Firecracker 的硬件资源开销是明确且又保障的。
- 功能极简主义：不会构建非AWS任务所明确要求的功能。每个功能仅实施一项。
- 计算超分：Firecracker 向来宾开放的所有硬件计算资源都可以安全地超分。

Firecracker 坚持精简主义的设计原则，它仅包含运行安全、轻量的虚拟机所需的组件。在设计过程的各个环节，都依据安全性、速度和效率要求来优化 Firecracker。例如，仅启动相对较新的 Linux 内核，并且仅启动使用特定配置选项集编译的内核（内核编译配置选项超过 1000 种）。此外，不支持任何类型的图形卡或加速器，不支持硬件透传，不支持（大多数）老旧设备。只支持四种设备虚拟化（virtio-net, virtio-block, serial console, 和只有一个按钮的键盘控制器）。

这么做的结果也是非常明显的，比如：

- 每个微虚机的内存开销小于 5MiB。
- 一台物理机上能启动的微虚机数目的限制只是硬件限制，数目可以是数千台。
- 在AWS 一次启动4000个微虚机的演示中，最长的微虚机耗时只有219毫秒，最短的只需要125毫秒。

项目的开源地址是 https://firecracker-microvm.github.io/。更多信息，可查阅更多文章，甚至阅读源码。

4. 展望未来

AWS 宣布开源 Firecracker在业界引起了很大的关注。加上之前已有的 Kata container（由 Intel，Hyper.sh 和 OpenStack主导）和 gVisor（由Google开源），微虚机越来越引起人们的重视。基于个人理解，对未来做一点不负责任的预测：

- 公有云利用微虚机来落地容器会成为通用做法。以阿里云的秉性，相信他们很快会跟进，推出和AWS类似的技术实现。很可能也会开源一个项目。

- AWS 会将 Firecracker 作为其统一的容器落地模式。

- 微虚机生态（Kata container、gVisor 和 Firecracker）会发生很多有趣的变化。当前，每个项目都有各自的面向场景。从非技术层面看，因为AWS 对开源的态度应该是『以我为主』，因此 Firecracker 还是会继续由AWS主导，以被AWS上的服务使用为主；gVisor 因为来自Google，Google 也有公有云，加上 Kubernetes 也源自Google，不知道它是否会演进为事实上的微虚机标准；而 Kata container 也许将来会以面向私有云场景为主（设想一下OpenStack支持 Kata 微虚机，而 K8S 支持支持 gVisor 微虚机，两者之间的PK是不是就成了编排能力的PK？）。

参考链接：

- https://docs.google.com/document/d/1PjlsBmZw6Jb3XZeVyZ0781m6PV7-nSUvQrwObkvz7jg/edit
- https://blog.jessfraz.com/post/containers-zones-jails-vms/
- https://www.youtube.com/watch?v=QdzV04T_kec
- https://www.slideshare.net/AmazonWebServices/a-serverless-journey-aws-lambda-under-the-hood-srv409r1-aws-reinvent-2018
- https://aws.amazon.com/blogs/opensource/firecracker-open-source-secure-fast-microvm-serverless/

容器 Linux https 网络安全 举报

点赞 3 分享

0 条评论 我来说两句

登录 后参与评论

相关文章

云中树莓派（2）：将传感器数据上传到 AWS Io…

DHT22 是一款温度与湿度传感器，它有3个针脚，左边的第一个引脚（#1）为3-5V电源，第二个引脚（#2）连接到数据输入引…

SammyLiu

[译] 在Kubernetes生产环境中运行Istio

本文翻译自 https://www.tigera.io/blog/running-istio-on-kubernetes-in-production-part-...

SammyLiu

理解Neutron (7): Neutron LBaaS v1

特别说明：本文于2015年基于OpenStack M版本发表于本人博客，现转发到公众号。因为时间关系，本文部分内容可能已过…

SammyLiu

深入理解Spring源码（一）-IOC容器的定位，载…

前言：Spring源码继承，嵌套层次非常多，读起来非常容易晕，小伙伴们在看文章的时候一定要跟着文章的思路自己去源码里…

Meet相识

IBM苏中：怎样利用深度学习、增强学习等方法…

伴随着认知计算时代的到来，如何将我们计算机的信息处理能力与人类的认知能力相结合，从而提高我们的信息处理效率，是…

数据派THU

SLAM初探（四）

OpenCV基础 这里我就不做过多的描述性问题，现在OpenCV在许多有关计算机视觉方面得到许多的应用。 OpenCV获取视频的方法及其图像转化问题 获取视频及...

Pulsar-V

大咖 | 卡耐基梅隆教授Tom Mitchell：人工智能在中国前景光明，有2…

大数据文摘

未能加载文件或程序集“Newtonsoft.Json, Versio…

_一级菜鸟

对LinqtoExcel的扩展 【数据有限性，逻辑有效性】

接着上文的内容继续讲，上文中我提到了对Excel操作帮助类库LinqToExcel类库的优缺点和使用方法。我也讲到了自己在使用中碰到的问题，我也开发了一个简单的...

小狐狸

自定义View（一）

自定义View 需求场景：当系统默认的view不能满足您的优（qi）美（pa）界面 要求时候，自定义view则进入您的视野，来满足…

用户1263308

更多文章 ›