容器化简介

马塞洛·里卡多·德·奥利维拉

2021 年 4 月 17 日 警察

本文为经验丰富的开发人员提供了对容器化的全面了解：它的来源、工作原理，以及用于构建和部署容器化云原生应用程序的典型工具。

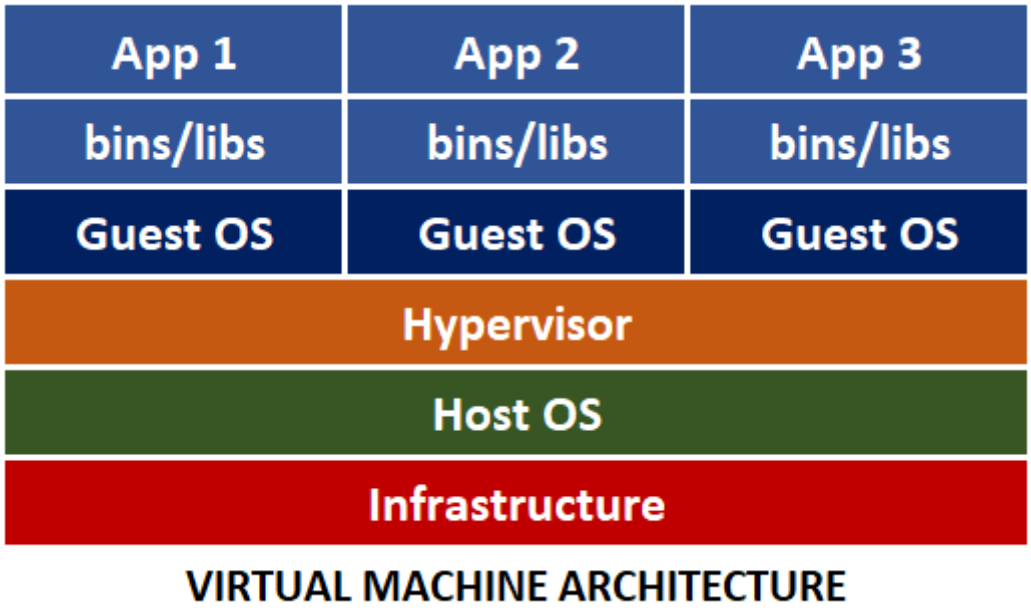
在这里，我们：看一下容器化如何开始的简史，解释什么是现代容器，讨论如何构建容器并复制您编写的应用程序，解释为什么要将容器推送到注册表以及如何直接从注册表部署到生产中，并讨论部署。

这篇文章是赞助文章。此类文章旨在向您提供我们认为对开发人员有用且有价值的产品和服务信息

很久以前，在互联网时代的早期，世界上最大的零售商投入巨资建立他们的网络运营，并在蓬勃发展的在线市场中挖掘许多机会。上线意味着他们可以全天候处理订单，但同时也带来了前所未有的运营挑战。

由于假日购物吸引了许多在线用户，而硬件最终会崩溃，因此这些零售商不得不购买比通常所需更多的服务器。购买一批新服务器非常昂贵，但在销售高峰期失去消费者是这些公司无法承受的风险。

借助 VMWare 和 Virtual Box 等产品，虚拟机改变了游戏规则，使公司能够更好地利用硬件资源。在一台主机上运行多个应用程序意味着企业不必购买这么多服务器。这就像运行“计算机中的计算机”。



VIRTUAL MACHINE ARCHITECTURE

尽管如此，虚拟化仍有其挑战。开发人员在他们的本地机器上工作，这可能与他们部署到的虚拟机不同。环境之间的这种不匹配导致了可怕的“在我的机器上工作”的借口和沮丧的开发人员，因为他们不得不担心在应用程序之外发生的故障。

虚拟机后来让位于容器化，尽管容器背后的想法并不新鲜。早在 1979 年，chroot 系统就在 Unix 系统中引入了进程隔离和文件访问隔离。2000 年，FreeBSD jails 将 Linux 计算机划分为更小的系统，2005 年，OpenVZ 使用 Linux Kernel 进行系统虚拟化和隔离。然后，在 2008 年，Linux Kernel 引入了控制组。

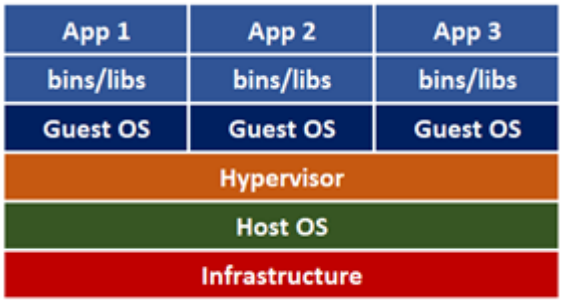
这些都是朝着我们今天使用的容器迈出的一步。最终，Docker 在竞争中脱颖而出，原因有很多，包括面向开发人员的工具、易于配置、出色的可移植性和强大的管理生态系统。

现代集装箱的兴起

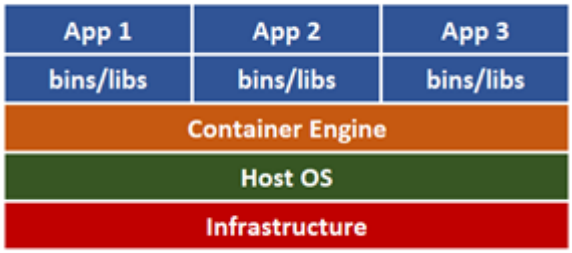
随着Docker的普及，容器化席卷了 IT 行业。容器与 VM 类似，但没有客户操作系统 (OS)，因此包更简单。

开发人员的"在我的机器上工作"的借口不再是问题，因为应用程序及其依赖项都是自包含的，并被运送到称为容器映像的同一个单元中。映像是应用程序的可部署副本，而从这些映像创建的容器实例通常在 Azure 等云平台中运行。

新架构缺少管理程序，因为不再需要它。但是，我们仍然需要管理新的容器，因此引入了容器引擎的概念。



VIRTUAL MACHINE ARCHITECTURE



CONTAINER ARCHITECTURE

容器是不可变的，这意味着您无法在其生命周期内更改容器映像：您无法应用更新、补丁或配置更改。如果您必须更新您的应用程序，您应该构建一个新镜像（它本质上是一个现有容器镜像之上的变更集）并重新部署它。不变性使容器部署变得简单和安全，并确保部署的应用程序始终按预期工作，无论在哪里。

与虚拟机相比，新容器极其轻巧便携。此外，容器启动速度要快得多。由于体积小，容器有助于最大限度地利用主机操作系统及其资源。

您可以在 Docker 容器中运行 Linux 和 Windows 程序。Docker 平台在 Linux 和 Windows 上本地运行，Docker 的工具使开发人员能够在 Linux、Windows 和 macOS 上构建和运行容器。

您不能使用容器来运行本地文件系统上的应用程序。但是，您可以使用绑定挂载和卷访问 Docker 容器外部的文件。它们很相似，但绑定挂载可以指向主机上的任何目录，并且不由 Docker 直接管理。

Docker 容器在 Linux 上访问物理主机硬件，如 GPU，但不在 Windows 上。这是因为 Linux 上的 Docker 直接在内核上运行，而 Windows 上的 Docker 工作在虚拟机中，因为 Windows 没有 Linux 内核可以直接与之通信。

容器：从编码到部署

假设您正在开发一个容器化电子商务应用程序。作为开发人员，您可以将应用程序构建到容器中。接下来，您编写一个docker 文件，其中包含在容器中为每个应用程序或服务设置和运行应用程序所需的命令。然后，您编写一个docker-compose.yml文件来设置多容器 Docker 应用程序中不同容器应用程序和服务之间的关系。

通过执行指定本地端口的 docker run 命令，然后在浏览器中打开 http://localhost:[PORT]，您可以在开发机器上本地运行和测试容器。

要构建和运行您的应用程序，您可以将其部署到 Docker 主机（VM 或物理服务器），或将其部署为组合应用程序，使用单个命令行界面 (CLI) 命令或集成开发环境 (IDE)) 像 Visual Studio。

一旦您的应用程序或服务经过测试，构建过程就会生成一个或多个容器映像。但是怎么处理它们呢？这就是Azure Container Registry 或 Docker Hub 之类的存储库的用武之地：生成容器映像后，开发人员将它们推送到公共或私有注册表。然后，他们可以将映像拉入数据中心、云中或应用程序可以运行的任何其他地方。

要了解如何实施使用 GitHub 操作将映像推送到容器注册表的持续集成和持续部署 (CI/CD)，请关注此 YouTube 播放列表：使用 GitHub 操作构建 CI/CD 工作流。

使用注册表的一些优点是：

- 您可以严格控制谁可以访问您的图像以及它们的存储位置。
- 注册表可以是您要运行的图像的唯一真实来源。
- 您可以根据需要扫描容器中的漏洞。

假设您的应用程序或服务可以在没有编排的独立容器中运行。在这种情况下，您可以使用Azure 容器实例 (ACI)等解决方案手动或从开发管道直接从容器注册表将映像部署到生产中。

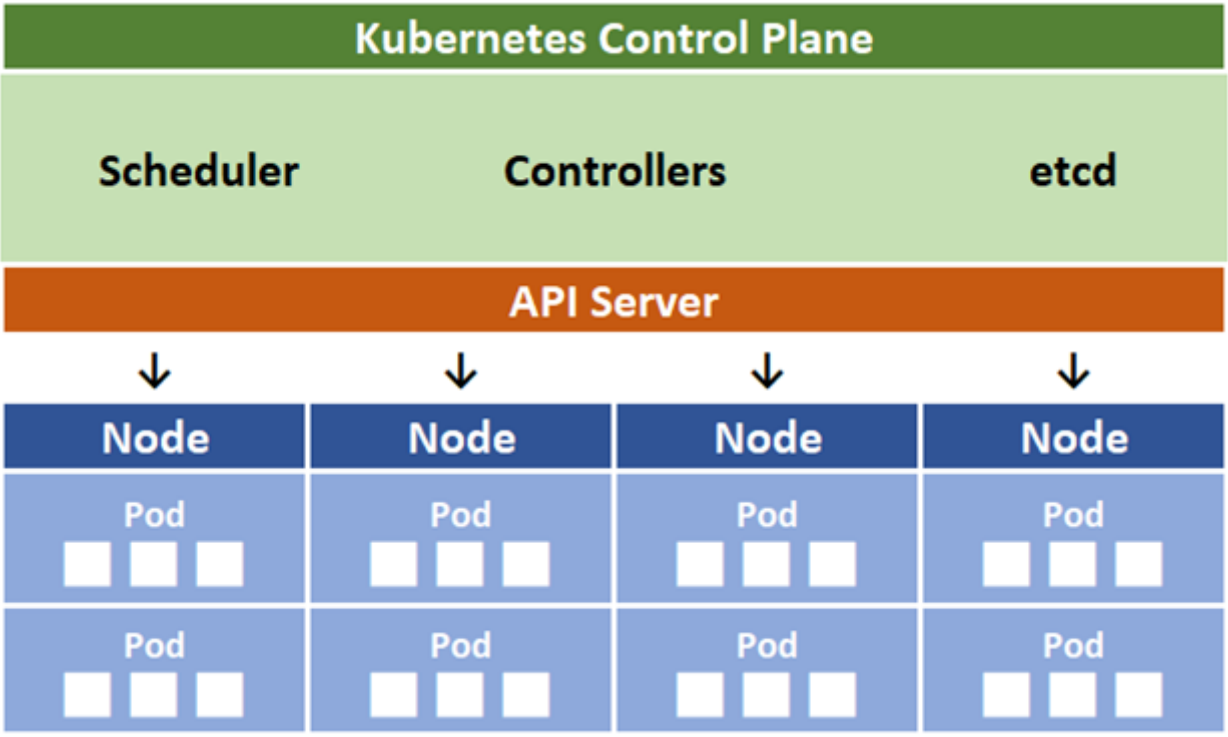
超越部署：使用 Kubernetes 进行编排

当公司开始将他们的解决方案迁移到容器时，他们意识到扩展应用程序并非易事。该Kubernetes平台很快涌现了作为金标准来运行应用程序集装箱化，从建立想要探索集装箱业务的兴趣空前的激增。

容器编排可以自动执行运行容器化工作负载和服务所需的许多任务，这需要在大型系统中付出巨大努力。Kubernetes 通过管理容器的生命周期来解决这一挑战，包括供应、部署、扩展、网络和负载平衡。

当您部署 Kubernetes 时，您将获得一个集群。Kubernetes 集群由一组称为工作器或节点的机器组成，它们在容器中运行应用程序。每个集群必须至少有一个工作节点。在工作节点内，应用程序工作负载沿着称为 Pod 的组件分布。

集群的控制平面管理工作节点和 Pod。在生产环境中，控制平面和集群通常分布在多台机器和节点上，提供高可用性和弹性。



您可以使用 minikube 或托管 Kubernetes 服务创建本地 Kubernetes 集群。Kubernetes 是一项复杂的技术，需要足够的安全知识、支持技能、频繁升级和对细节的关注。Azure Kubernetes 服务(AKS)等托管 Kubernetes 服务提供了许多方便的管理功能，可将您从这些重复性任务中解放出来，让您可以专注于您的业务。

使用 Kubernetes 的 CI/CD 管道

让我们看一下 CI/CD 管道，它将您的代码容器化，将其推送到注册表，然后从那里进入 Kubernetes 集群中的生产环境。

这一切都始于您的代码。在适当的代码审查、单元测试和其他控制之后，合并请求合并到您的 Git 存储库的主分支中。这会触发一个事件，导致构建管道将您的应用程序构建到容器映像中。生成管道然后将这些映像推送到 Azure 容器注册表 (ACR)。

可以为 Azure 容器注册表中创建的每个对象定义策略、合规性和规则。然后 CI/CD 管道从 ACR 中提取容器映像，并将它们作为 Pod 部署在 Kubernetes 集群中的工作节点内。容器镜像可以复制到多个工作节点内的多个 Pod 中，具体取决于您在部署配置文件中定义的副本数。

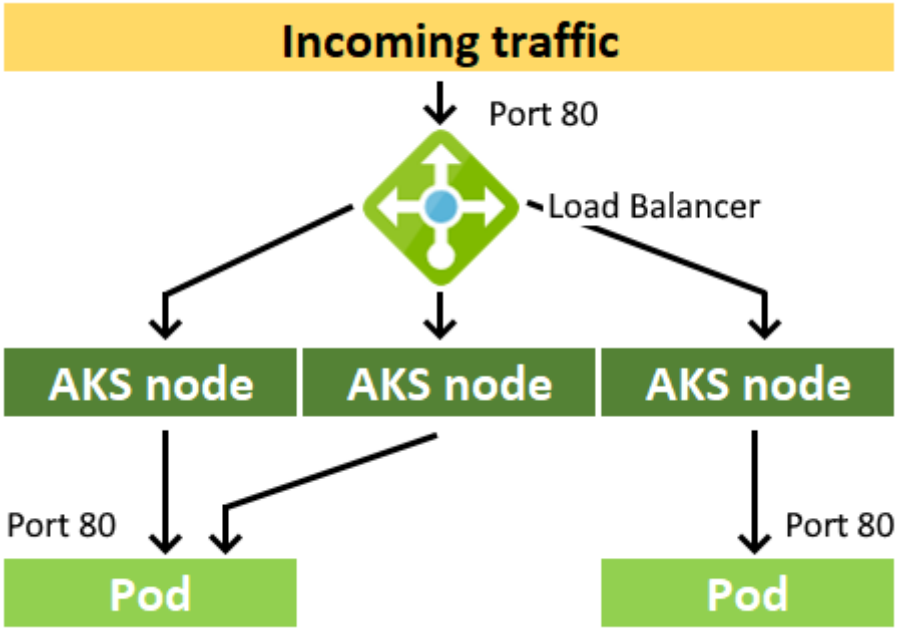
从现在开始，您的新代码已经在 Kubernetes 集群上的生产环境中运行。每当您向 Git 存储库提交新代码时，构建管道都会重复该过程。

Kubernetes 的工作原理是将一组虚拟机或物理机转换为统一的 API 接口，供开发人员与之交互的容器使用，并编排他们的应用程序，而不管底层物理机是什么。

在 Kubernetes 中，您的容器位于Pods 中，Pods是最小的部署单元。Kubernetes 的 ReplicationController 有助于确保您的应用程序跨节点的 Pod 健康。当您定义所需数量的 Pod 副本时，ReplicationController 保证该副本数量始终保持不变。因此，当 Pod 发生故障时，控制器会注意到并在集群中的其他位置启动一个 Pod。

Azure Kubernetes 服务使你能够通过自动调整运行工作负载的节点数量来维护高效、经济高效的群集。集群自动伸缩组件检测资源限制何时发生，因此节点池中的节点数量会增加以满足您的应用程序需求。同样，当需求下降时，节点数量也会减少，因此您无需为未使用的云资源付费。

除了在 Kubernetes 集群中创建额外的节点和 Pod 之外，您还必须确保这些新资源的使用程度与现有资源相同。负载平衡组件（例如 Azure 负载平衡器）从 AKS 虚拟网络内部或外部分发流，成功扩展应用程序并创建高度可用的服务。



下一步

您刚刚了解了容器为何如此重要的旋风之旅：目前，它们似乎是企业应用程序部署的未来。随着技术的成熟，越来越多的开发人员和 DevOps 团队将从容器化提高的可移植性、可扩展性和效率中受益。

想了解更多？探索以下资源：

- 学习使用 Kubernetes 构建和管理基于容器的应用程序。获取这本免费的三本电子书包，以在一个地方学习 Kubernetes 的基础知识。
- 使用 Azure 上的 Kubernetes 加速开发。通过在此 Packt 电子书中学习如何大规模部署和管理容器集群开始。
- 容器是云原生应用开发的基础。云原生应用程序基于微服务。每个应用程序都包含彼此独立运行的小型服务，利用持续交付来实现可靠性和更快的上市时间。
- 了解如何使用公共负载均衡器通过 Azure Kubernetes 服务 (AKS) 公开您的服务。

执照

分享

关于作者

马塞洛·里卡多·德·奥利维拉
讲师/培训师 Alura Cursos Online
巴西
手表该会员

Marcelo Ricardo de Oliveira 是一名高级自由软件开发人员，与他可爱的妻子 Luciana 和他的小伙伴兼继子 Kauê 住在巴西瓜鲁柳斯，是巴西电视指南电视地图的联合创始人，目前为Alura Cursos Online 工作。

他经常从事严肃的企业项目，尽管在业余时间他尝试编写有趣的代码项目文章.....

展示更多

评论和讨论

添加评论或问题 | 电子邮件提醒 | Search Comments

-- 本论坛暂无消息 --