aws

最佳实践指南

Amazon Elastic Container Service

Amazon Elastic Container Service: 最佳实践指南

Table of Contents

Introduction

Amazon Elastic Container Service (Amazon ECS) 是一项高度可扩展的快速容器管理服务，它可轻松运行、停止和管理群集上的容器。本指南涵盖了许多最重要的操作最佳实践，同时还解释了支持基于 Amazon ECS 的应用程序工作原理的核心主题。目标是提供一种具体、可操作的方法来操作基于 Amazon ECS 的应用程序并对其进行故障排除。

本指南将定期修订，以纳入新的 Amazon ECS 最佳实践。如果您对本指南中的任何内容有任何疑问或意见，请在 GitHub 存储库中提出一个问题。有关更多信息，请参阅 。亚马逊云服务器最佳实践指南（位于 GitHub 上）。

- 最佳实践 — 网络
- 最佳实践-自动扩展和容量管理
- 最佳做法-持久性存储
- 最佳实践 — 安全

最佳实践 — 网络

现代应用程序通常由多个相互通信的分布式组件构建。例如，移动或 Web 应用程序可能与 API 终端节点通信，而 API 可能由通过互联网进行通信的多个微服务提供支持。

本指南介绍了构建网络的最佳实践，使应用程序的组件能够以可扩展的方式安全地相互通信。

主题

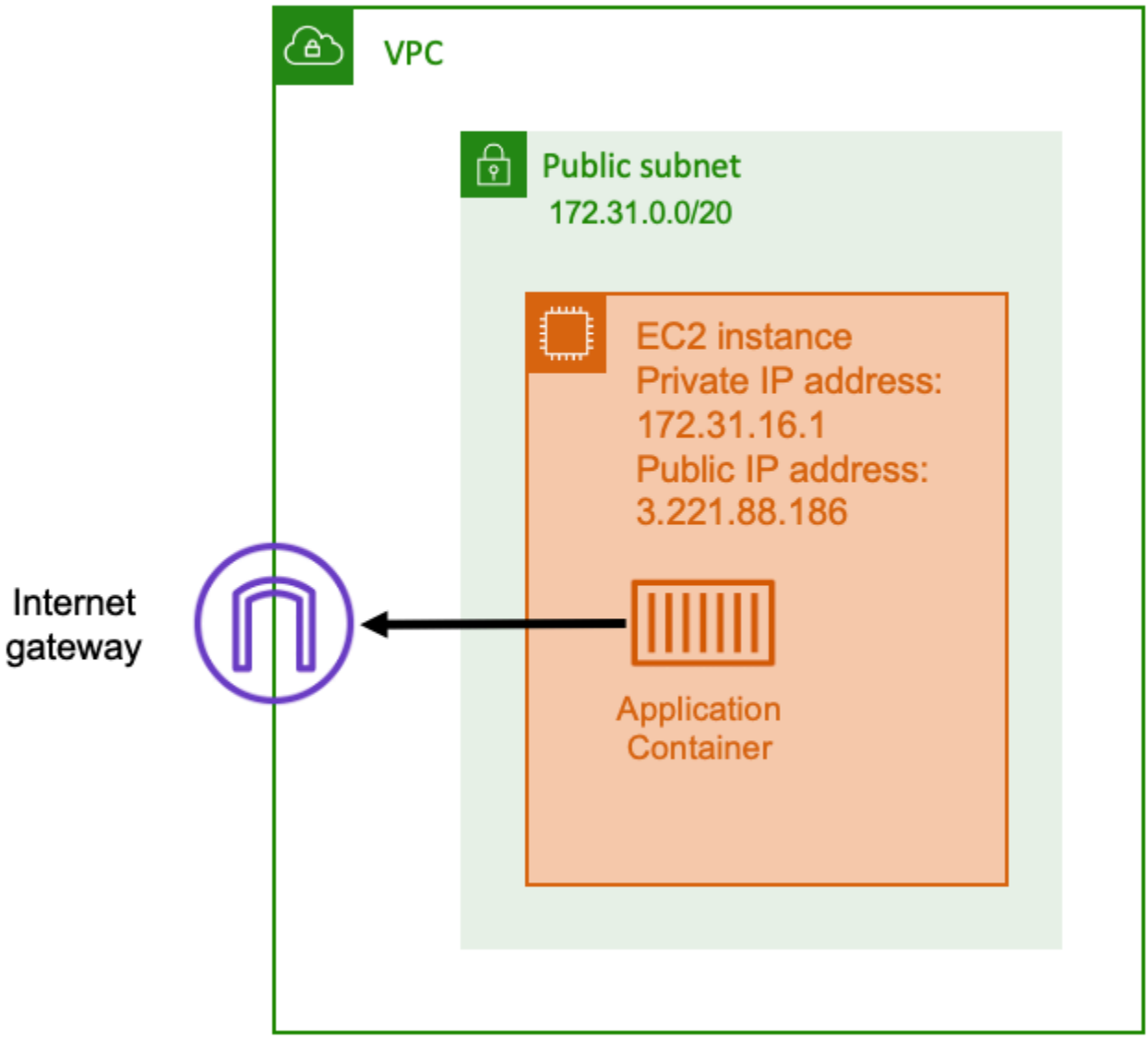
连接到互联网

大多数容器化应用程序至少有一些需要出站访问 Internet 的组件。例如，移动应用程序的后端需要出站访问推送通知。

Amazon Virtual Private Cloud 有两种主要方法来促进您的 VPC 与互联网之间的通信。

主题

使用公有子网和互联网网关



通过使用带有到互联网网关的路由的公有子网，您的容器化应用程序可以在公有子网上运行。运行您的容器的主机被分配了公有 IP 地址。此公有 IP 地址可从互联网路由。有关更多信息，请参阅 。Internet 网关中的Amazon VPC 用户指南。

此网络体系结构便于运行您的应用程序的主机与 Internet 上的其他主机之间的直接通信。通信是双向的。这意味着您不仅可以与 Internet 上的任何其他主机建立出站连接，而且 Internet 上的其他主机也可能尝试连接到您的主机。因此，您应密切注意您的安全组和防火墙规则。这是为了确保互联网上的其他主机无法打开您不希望打开的任何连接。

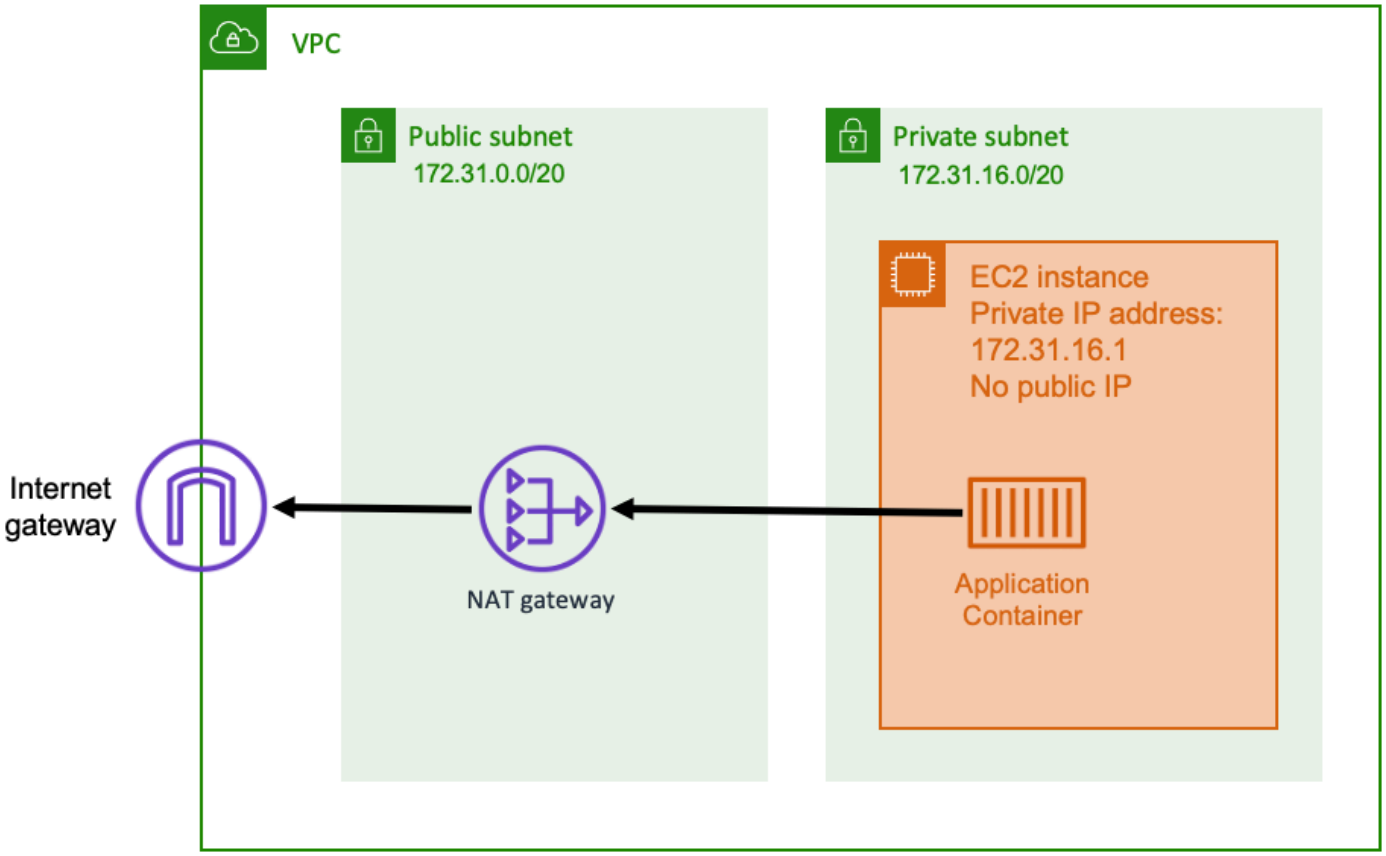
例如，如果您的应用程序在 Amazon EC2 上运行，请确保 SSH 访问的端口 22 未打开。否则，您的实例可能会收到来自互联网上混乱机器人的持续 SSH 连接尝试。这些机器人通过公有 IP 地址拖网。找到开放的 SSH 端口后，他们会尝试暴力强制密码来尝试访问您的实例。正因为如此，许多组织限制了公有子网的使用，并希望在私有子网中拥有大部分资源（如果不是全部）。

使用公有子网进行网络联网适用于需要大量带宽或最小延迟的公共应用程序。适用的使用案例包括视频流和游戏服务。

当您在 Amazon EC2 上使用 Amazon ECS 时，以及在AWS Fargate。

- 使用 Amazon EC2 — 您可以在公有子网上启动 EC2 实例。Amazon ECS 使用这些 EC2 实例作为集群容量，并且在实例上运行的任何容器都可以使用主机的底层公有 IP 地址进行出站网络。这适用于`host`和`bridge`网络模式。但是，`awsvpc`网络模式不提供具有公有 IP 地址的任务 ENI。因此，他们无法直接使用互联网网关。
- 使用 Fargate — 创建 Amazon ECS 服务时，请为服务的网络配置指定公有子网，并确保分配公有 IP 地址选项处于启用状态。每个 Fargate 任务都在公有子网中进行联网，并且具有自己的公有 IP 地址，用于与互联网直接通信。

使用私有子网和 NAT 网关



通过使用私有子网和 NAT 网关，您可以在私有子网中的主机上运行容器化应用程序。因此，此主机具有可在 VPC 内路由的私有 IP 地址，但不能从互联网路由。这意味着 VPC 内的其他主机可以使用其私有 IP 地址连接到主机，但互联网上的其他主机无法与主机进行任何入站通信。

使用私有子网，您可以使用网络地址转换 (NAT) 网关允许私有子网内的主机连接到 Internet。互联网上的主机会收到一个入站连接，该连接似乎来自公有子网内 NAT 网关的公有 IP 地址。NAT 网关负责作为互联网和私有 VPC 之间的桥梁。出于安全原因，此配置通常是首选的，因为这意味着您的 VPC 受到保护，防止互联网上的攻击者直接访问。有关更多信息，请参阅 。NAT 网关中的Amazon VPC 用户指南。

此专用网络方法适用于希望保护容器免受直接外部访问的场景。适用的方案包括支付处理系统或存储用户数据和密码的容器。您在账户中创建和使用 NAT 网关会产生费用。NAT 网关小时使用费率和数据处理费率也适用于此。出于冗余目的，您应在每个可用区中有一个 NAT 网关。这样，单个可用区域的可用性不会损害您的出站连接。因此，如果您的工作负载较小，则使用专用子网和 NAT 网关可能更具成本效益。

无论是在 Amazon EC2 上使用 Amazon ECS，还是在AWS Fargate。

- 使用 Amazon EC2 — 您可以在私有子网上启动 EC2 实例。在这些 EC2 主机上运行的容器使用底层主机网络，出站请求通过 NAT 网关。
- 使用 Fargate — 创建 Amazon ECS 服务时，请为服务的网络配置指定私有子网，并且不要启用分配公有 IP 地址选项。每个 Fargate 任务都托管于私有子网中。其出站流量通过您与该私有子网关联的任何 NAT 网关进行路由。

从 Internet 接收传入连接

如果您运行公共服务，则必须接受来自互联网的入站流量。例如，您的公共网站必须接受来自浏览器的入站 HTTP 请求。在这种情况下，Internet 上的其他主机也必须启动到应用程序主机的入站连接。

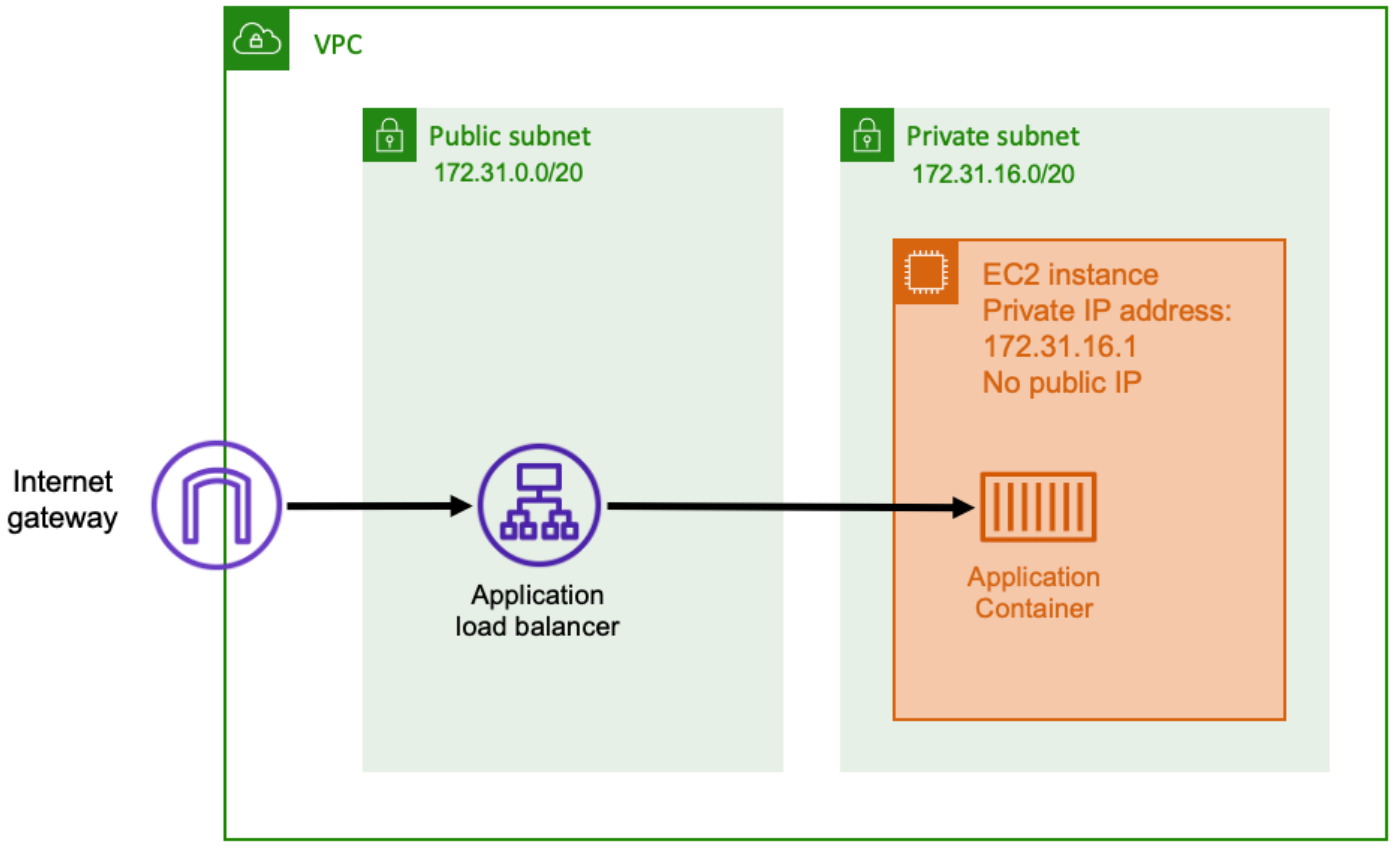
解决此问题的一种方法是在具有公有 IP 地址的公有子网中的主机上启动容器。但是，我们不推荐大型应用程序使用。为此，更好的方法是拥有一个位于 Internet 和应用程序之间的可扩展输入层。对于这种方法，您可以使用任意AWS服务作为输入列出。

主题

- 应用程序负载均衡器
- 网络负载均衡器
- Amazon API Gateway HTTP HTTP API

应用程序负载均衡器

应用程序负载均衡器在应用程序层运行。这是开放系统互连 (OSI) 模型的第七层。这使得应用程序负载均衡器适用于公共 HTTP 服务。如果您有网站或 HTTP REST API，则 Application Load Balancer 是适合此工作负载的负载均衡器。有关更多信息，请参阅 。什么是应用程序负载均衡器？中的适用于应用程序负载均衡器的用户指南。



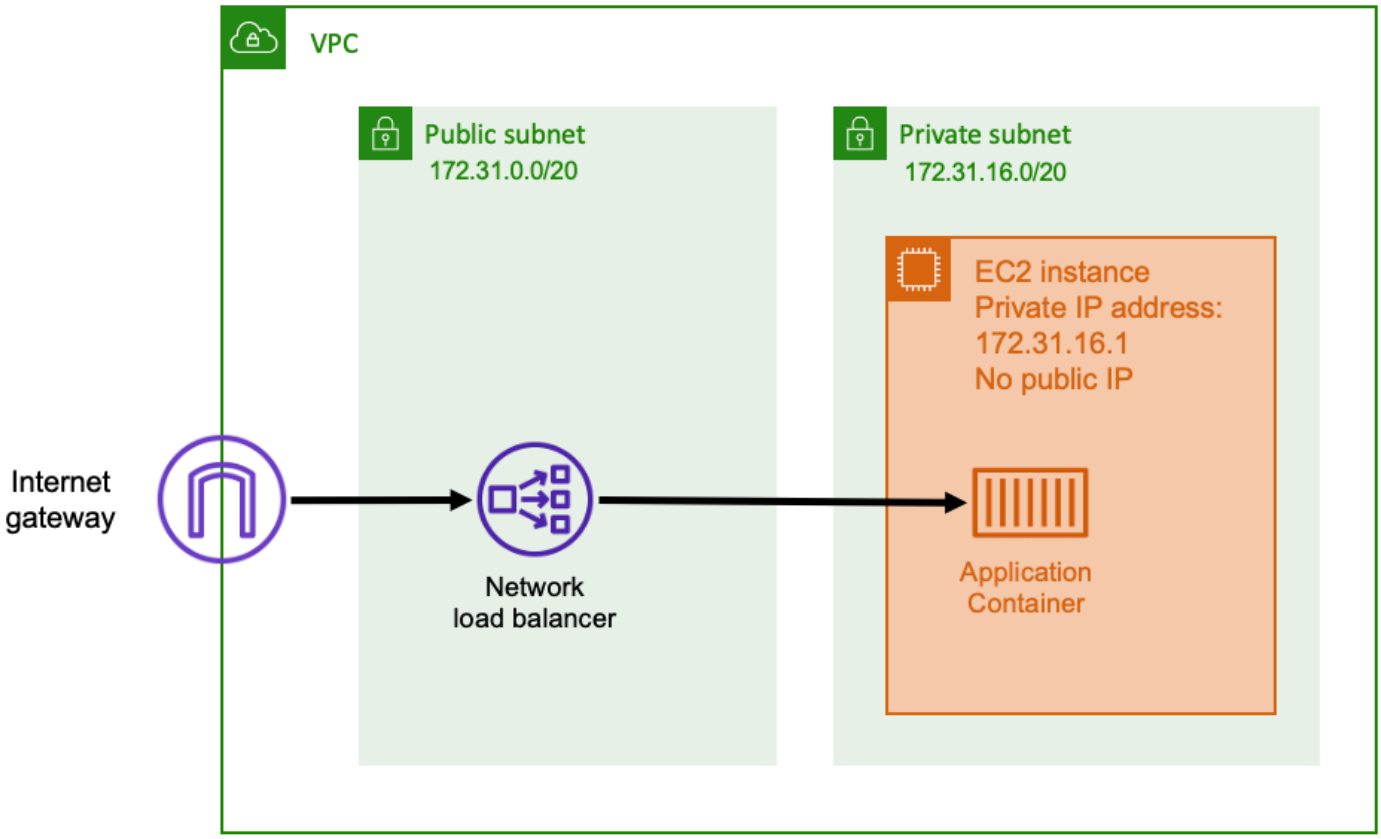
使用此体系结构，您可以在公有子网中创建 Application Load Balancer，以便其具有公有 IP 地址并可以接收来自 Internet 的入站连接。当应用 Application Load Balancer 收到入站连接或更具体地说是 HTTP 请求时，它会使用其私有 IP 地址打开与应用程序的连接。然后，它通过内部连接转发请求。

Application Load Balancer 具有以下优势。

- SSL/TLS 终止 — Application Load Balancer 可以维持安全的 HTTPS 通信和证书，以便与客户端进行通信。它可以选择在负载均衡器级别终止 SSL 连接，以便您不必在自己的应用程序中处理证书。
- 高级路由 — 应用程序负载均衡器可以具有多个 DNS 主机名。它还具有高级路由功能，可根据指标（如主机名或请求路径）将传入的 HTTP 请求发送到不同的目的地。这意味着您可以使用单个应用 Application Load Balancer 作为许多不同内部服务的输入，甚至可以在 REST API 的不同路径上使用微服务。
- GRPC 支持和网络套接字 — 应用程序负载均衡器可以处理的不仅仅是 HTTP。它还可以负载平衡 GRPC 和基于网络套接字的服务，并支持 HTTP/2。
- 安全性 — 应用程序负载均衡器有助于保护您的应用程序免受恶意流量 它包括 HTTP 同步缓解措施等功能，并与 AWS Web 应用程序防火墙 (AWS WAF)。AWS WAF可以进一步过滤掉可能包含攻击模式（如 SQL 注入或跨站点脚本）的恶意流量。

网络负载均衡器

网络负载均衡在开放系统互连 (OSI) 模型的第四层运行。它适用于非 HTTP 协议或需要端到端加密但不具有与 Application Load Balancer 相同的 HTTP 特定功能的场景。因此，Network Load Balancer 最适合不使用 HTTP 的应用程序。有关更多信息，请参阅 。什么是 Network Load Balancer？中的适用于网络负载均衡器的用户指南。



当 Network Load Balancer 用作输入时，它的功能类似于应用 Application Load Balancer。这是因为它是在一个公有子网中创建的，并且具有可在 Internet 上访问的公有 IP 地址。然后，Network Load Balancer 打开与运行容器的主机的私有 IP 地址的连接，并将数据包从公共端发送到私有端。

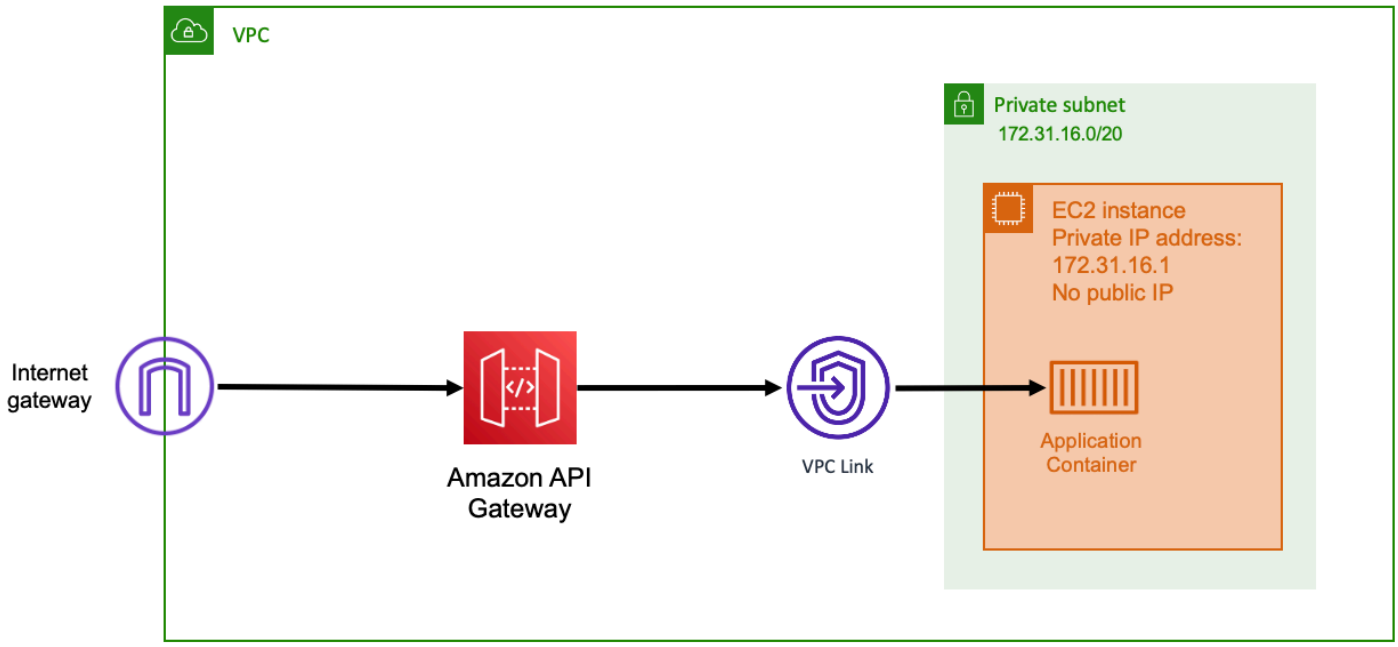
由于 Network Load Balancer 在网络堆栈的较低级别上运行，因此它不具有与 Application Load Balancer 相同的功能集。但是，它确实具有以下重要功能。

- 端到端加密 — 因为 Network Load Balancer 在 OSI 模型的第四层运行，所以它不读取数据包的内容。这使得它适合需要端到端加密的负载均衡通信。
- TLS 加密 — 除了端到端加密之外，Network Load Balancer 还可以终止 TLS 连接。这样，您的后端应用程序就不必实现自己的 TLS。

- UDP 支持 — 由于 Network Load Balancer 在 OSI 模型的第四层运行，因此它适用于非 HTTP 工作负载和 TCP 以外的协议。

Amazon API Gateway HTTP HTTP API

Amazon API Gateway HTTP API 是一种较少入口的服务器，适用于请求量突然突发或请求量少的 HTTP 应用程序。有关更多信息，请参阅 。什么是 Amazon API Gateway？中的API Gateway 开发人员指南。



Application Load Balancer 和 Network Load Balancer 的定价模型包括小时价格，以保持负载均衡器可随时接受传入连接。相比之下，API Gateway 会对每个请求单独收费。这样做的结果是，如果没有提出请求，则不收取任何费用。在高流量负载下，应用程序负载均衡器或 Network Load Balancer 可以以比 API Gateway 更便宜的每个请求价格处理更多的请求。但是，如果您的请求总体数量较少或流量较低，则使用 API Gateway 的累积价格应比支付小时费用来维护未充分利用的负载均衡器更具成本效益。

API Gateway 使用 VPC 链接功能，该链接允许AWS托管服务使用私有 IP 地址连接到 VPC 私有子网内的主机。它可以检测这些私有 IP 地址，方法是查看AWS Cloud Map由 Amazon ECS 服务发现管理的服务发现记录。

API Gateway 支持以下功能。

- SSL/TLS 终端节点

- 将不同的 HTTP 路径路由到不同的后端微服务

除了上述功能外，API Gateway 还支持使用自定义 Lambda 授权器，您可以使用这些授权器来保护 API 免受未经授权的使用。有关更多信息，请参阅 。字段注意：使用亚马逊云服务器和 Amazon API Gateway 的无服务器基于容器的 API。

选择网络模式

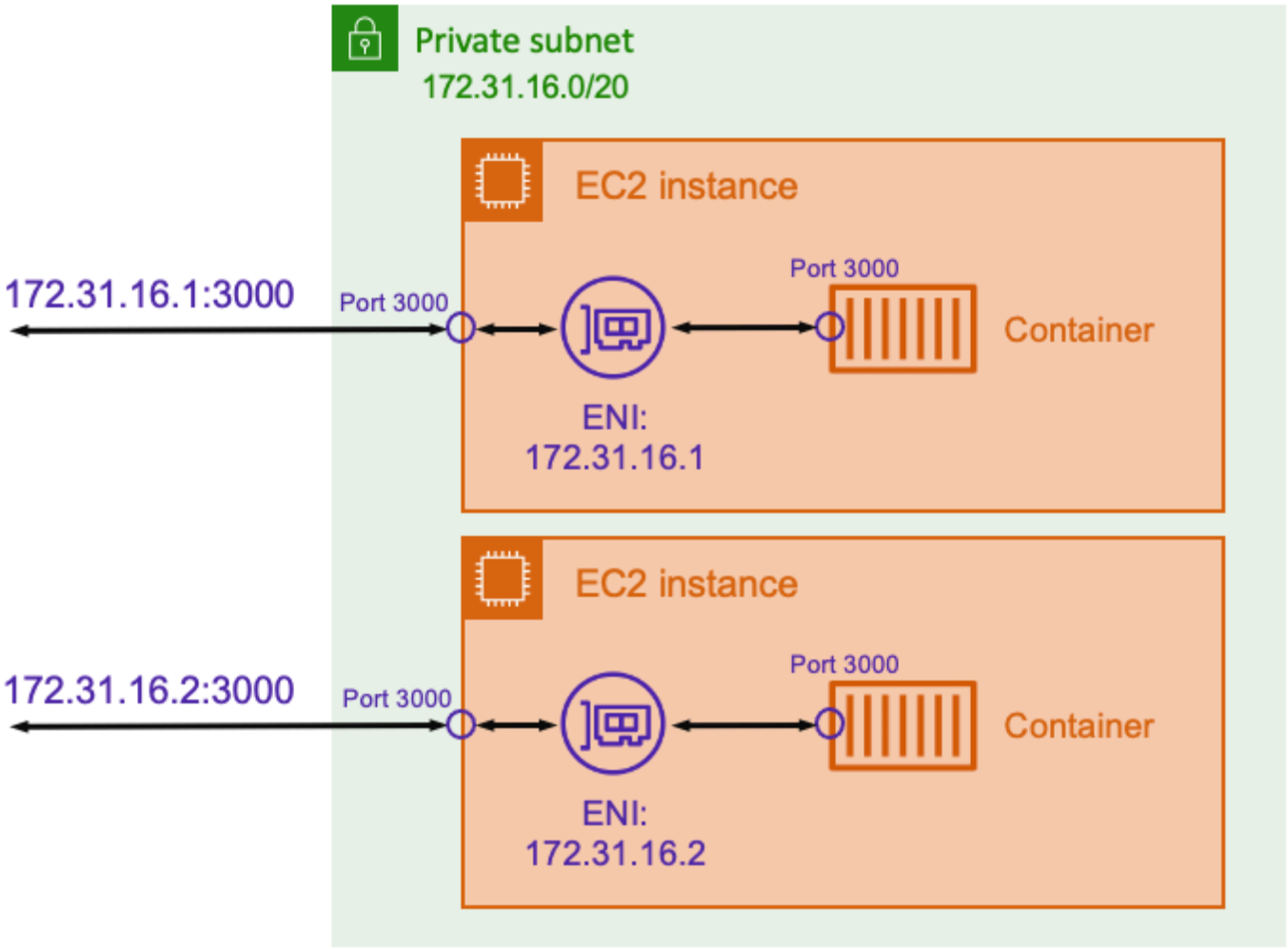
前面提到的用于构建入站和出站网络连接的方法可以适用于AWS，即使它们不在容器内。运行容器时AWS，您需要考虑另一级别的网络。使用容器的主要优点之一是，您可以将多个容器打包到单个主机上。执行此操作时，您需要选择如何在同一主机上运行的容器网络。以下是可供选择的选项。

主题

- 主机模式
- 桥接模式
- AWSVPC 模式

主机模式

这些区域有：host网络模式是 Amazon 云服务器支持的最基本的网络模式。使用主机模式，容器的网络连接直接绑定到运行容器的底层主机。



假定您正在运行一个 Node.js 容器与侦听端口的快速应用程序3000类似于前面的示意图中所示的。当host网络模式时，容器会使用底层主机 Amazon EC2 实例的 IP 地址在端口 3000 上接收流量。我们不建议使用此模式。

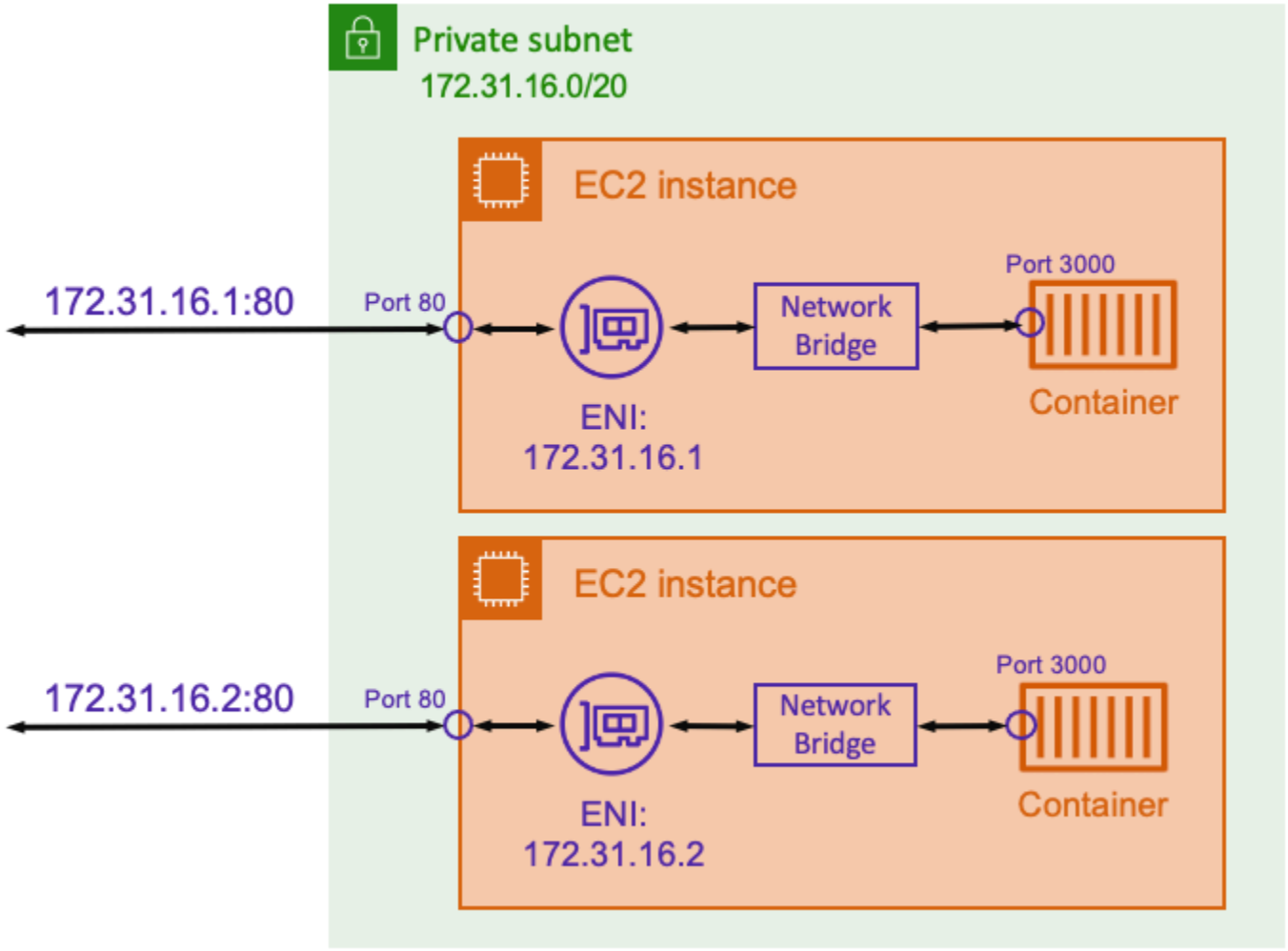
使用此网络模式有很大的缺点。您不能在每台主机上运行多个任务实例化。这是因为只有第一个任务可以绑定到 Amazon EC2 实例上的所需端口。当它使用host网络模式。例如，如果应用程序需要侦听特定端口号，则无法直接重新映射端口号。相反，您必须通过更改应用程序配置来管理任何端口冲突。

在使用host网络模式。此模式允许容器模拟主机，并允许容器连接到主机上的专用环回网络服务。

这些区域有：host网络模式仅支持 Amazon EC2 实例上托管的 Amazon ECS 任务。在 Fargate 上使用亚马逊云服务器时不支持此功能。

桥接模式

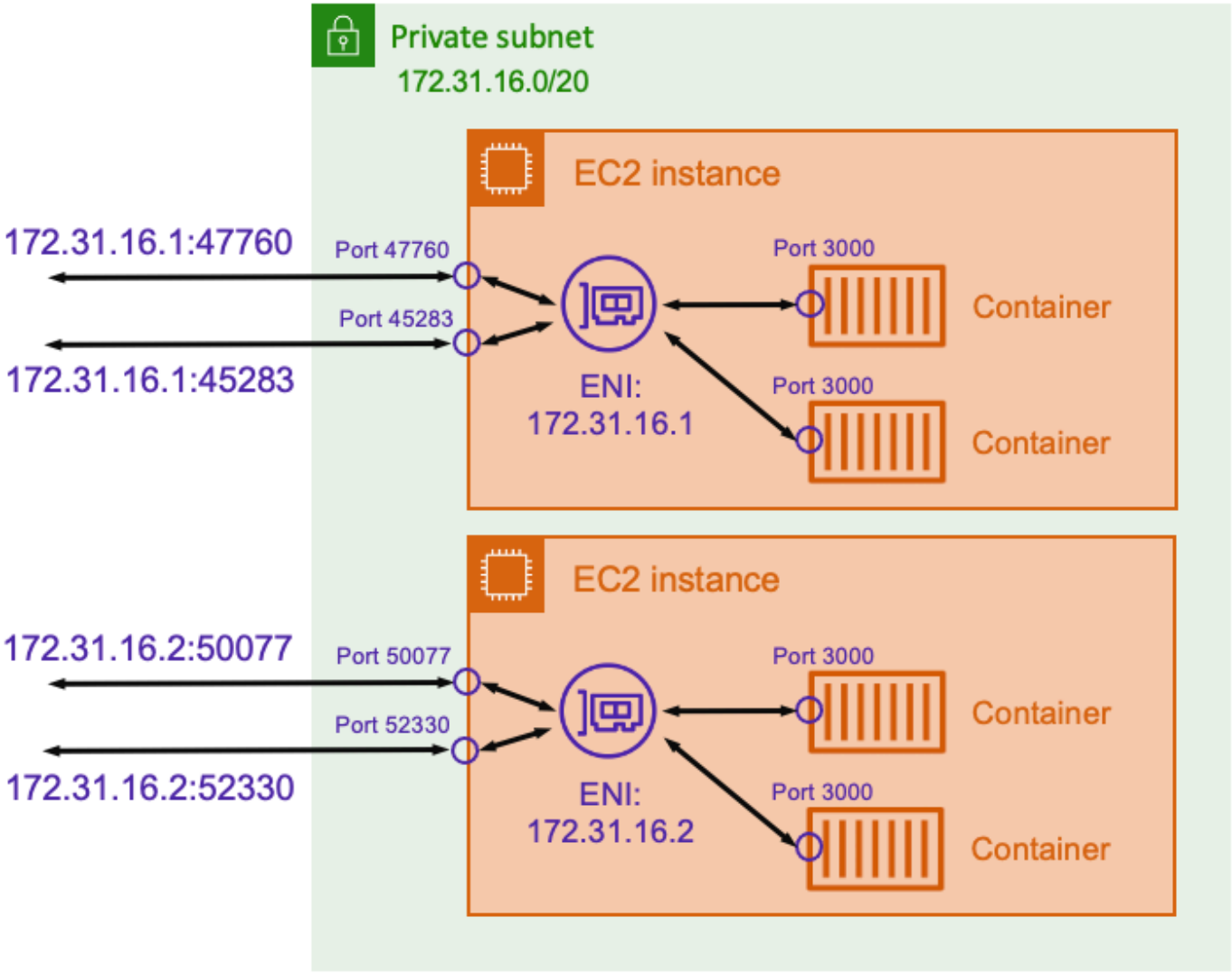
与bridge模式时，您正在使用虚拟网络桥接在主机和容器网络之间创建一个层。通过这种方式，您可以创建将主机端口重新映射到容器端口的端口映射。映射可以是静态的，也可以是动态的。



通过静态端口映射，您可以显式定义要映射到容器端口的主机端口。使用上述示例，端口80正在将主机上映射到端口3000（在容器上）。要与容器化应用程序进行通信，请将流量发送到端口80添加到 Amazon EC2 实例的 IP 地址。从容器化应用程序的角度来看，端口上的入站流量3000。

如果您只想更改流量端口，则适合使用静态端口映射。但是，这仍然具有与使用host网络模式。您不能在每台主机上运行多个任务实例化。这是因为静态端口映射仅允许将单个容器映射到端口 80。

要解决此问题，请考虑使用bridge具有动态端口映射的网络模式，如下图所示。



通过不在端口映射中指定主机端口，您可以让 Docker 从临时端口范围中选择一个未使用的随机端口，并将其指定为容器的公共主机端口。例如，Node.js 应用程序侦听端口`3000`可能会被分配一个随机的高数端口，例如`47760`（在 Amazon EC2 主机上）。执行此操作意味着您可以在主机上运行该容器的多个副本。此外，每个容器都可以在主机上分配自己的端口。容器的每个副本都接收端口上的流量`3000`。但是，向这些容器发送流量的客户端使用随机分配的主机端口。

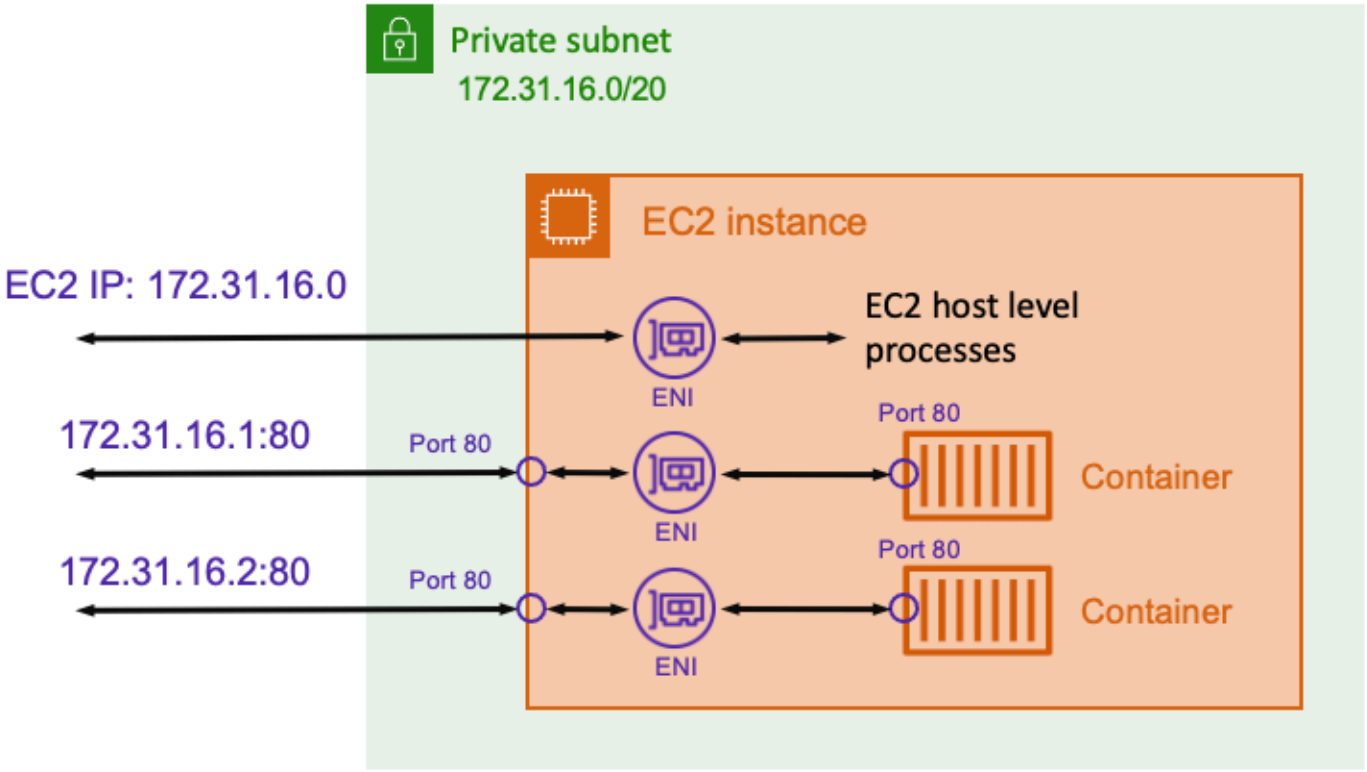
Amazon ECS 可帮助您跟踪随机分配的每项任务的端口。它通过自动更新负载均衡器目标组和AWS Cloud Map服务发现以获得任务 IP 地址和端口列表。这样，就可以更轻松地使用使用`bridge`模式与动态端口。

但是，使用bridge网络模式的原因是，很难将服务锁定到服务通信。由于服务可能会分配给任何未使用的随机端口，因此有必要在主机之间打开广泛的端口范围。但是，创建特定规则并不容易，以便特定服务只能与其他特定服务进行通信。这些服务没有用于安全组网络规则的特定端口。

这些区域有：bridge网络模式仅支持 Amazon EC2 实例上托管的 Amazon ECS 任务。在 Fargate 上使用亚马逊云服务器时不支持此功能。

AWSVPC 模式

随着awsvpc网络模式下，Amazon ECS 会为每个任务创建并管理弹性网络接口 (ENI)，每个任务在 VPC 内接收自己的私有 IP 地址。此 ENI 是独立于底层主机 ENI。如果 Amazon EC2 实例正在运行多个任务，则每个任务的 ENI 也是单独的。



在前面的示例中，Amazon EC2 实例被分配给 ENI。ENI 表示 EC2 实例的 IP 地址，用于主机级网络通信。每个任务还有一个相应的 ENI 和一个私有 IP 地址。由于每个 ENI 都是独立的，因此每个容器都可以绑定到端口80在任务 ENI. 因此，您不需要跟踪端口号。相反，您可以将流量发送到端口80在任务 ENI 的 IP 地址。

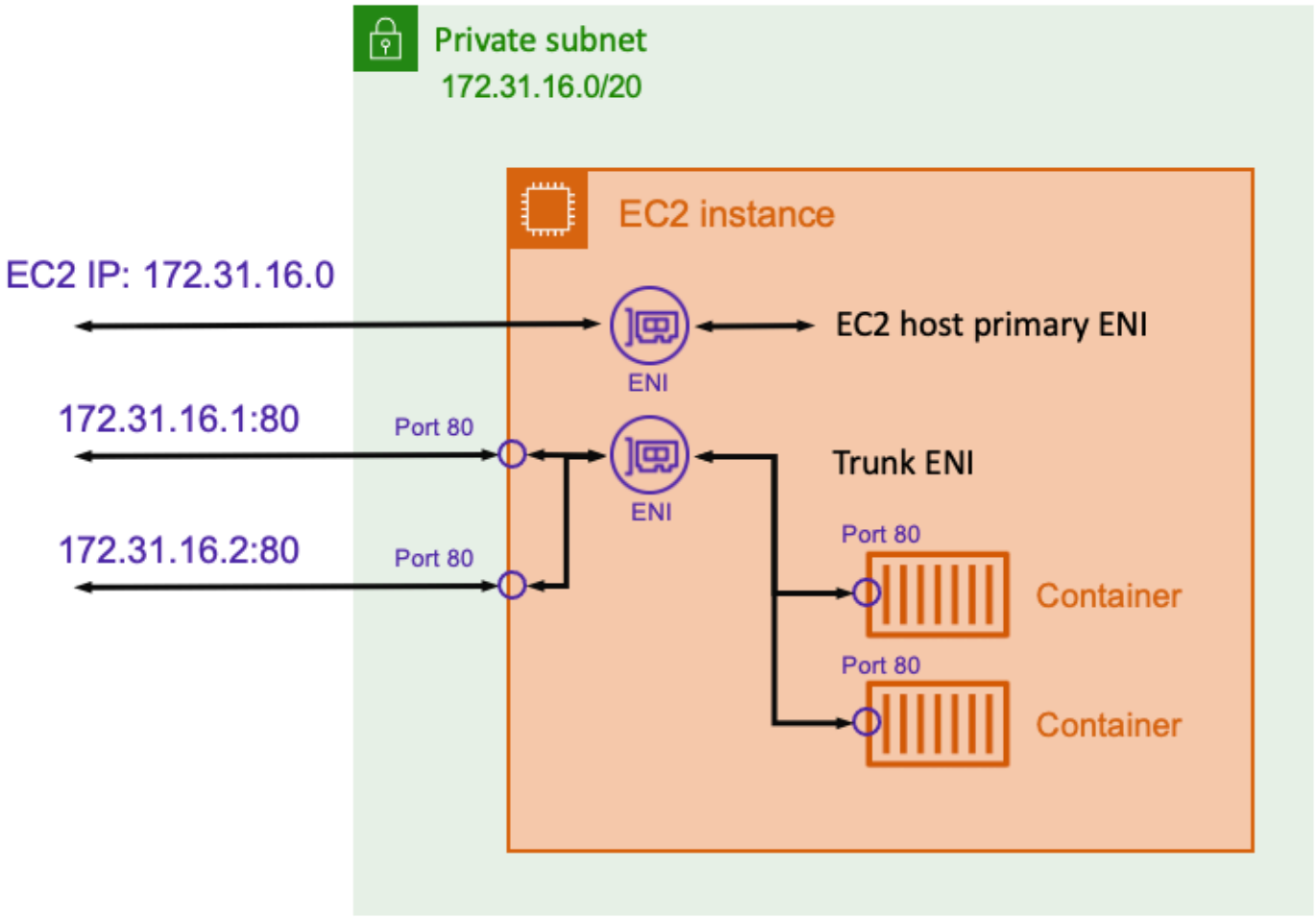
使用awsvpc网络模式是每个任务都有一个单独的安全组来允许或拒绝通信。这意味着您可以更加灵活地控制任务和服务之间的通信，以更精细的级别。您还可以将任务配置为拒绝来自位于同一主机上的另一个任务的传入流量。

这些区域有：awsvpc网络模式支持在 Amazon EC2 和 Fargate 上托管的亚马逊云服务器任务。请注意，在使用 Fargate 时，awsvpc网络模式。

使用awsvpc网络模式，您应该注意一些挑战。

使用 ENI 中继提高任务密度

使用awsvpc网络模式中，Amazon EC2 实例对可以附加到它们的 ENI 数量有限制。这会限制您可以在每个实例上放置多少任务。Amazon 弹性云服务器提供 ENI 中继功能，增加可用 ENI 的数量，从而实现更高的任务密度。



使用 ENI 中继时，默认使用两个 ENI 附件。第一个是实例的主 ENI，用于任何主机级进程。第二个是亚马逊云服务器创建的中继 ENI。仅在特定 Amazon EC2 实例类型上支持此功能。

考虑此示例。如果没有 ENI 中继，则`c5.large`实例只能托 vCPUs 两个任务。但是，使用 ENI 中继，`c5.large`实例最多可以承载十个任务。每个任务都有不同的 IP 地址和安全组。有关可用实例类型及其密度的更多信息，请参阅受支持的 Amazon EC2 实例类型中的Amazon Elastic Container Service 开发人指南。

ENI 中继在延迟或带宽方面对运行时性能没有影响。但是，它会增加任务启动时间。如果使用 ENI 中继，您应该确保依赖于任务启动时间的自动缩放规则和其他工作负载仍然可以正常工作。

有关更多信息，请参阅 。弹性网络接口中继中的Amazon Elastic Container Service 开发人指南。

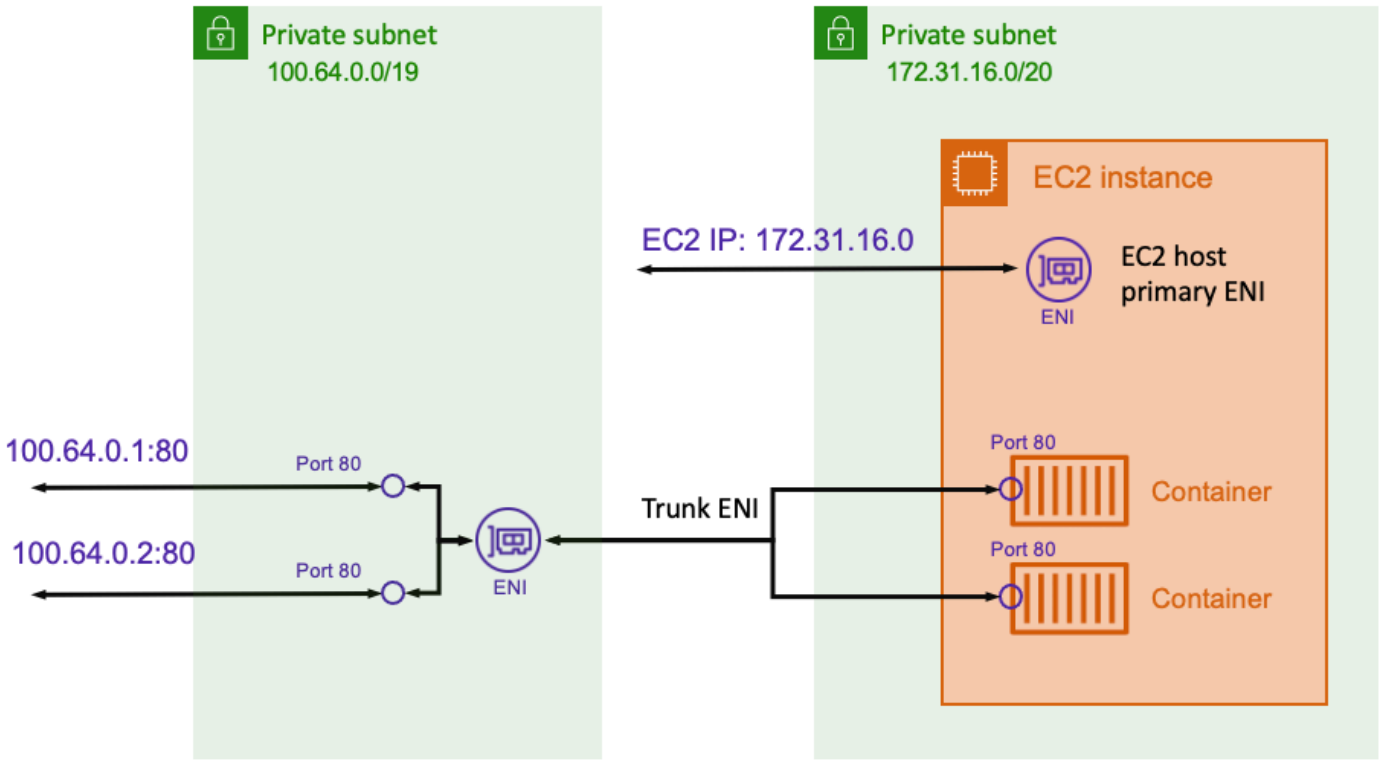
防止 IP 地址耗尽

通过为每个任务分配单独的 IP 地址，您可以简化整体基础架构并维护提供高级别安全性的安全组。但是，此配置可能导致 IP 耗尽。

您的默认 VPCAWS帐户具有预置备子网，这些子网具有`/20`CIDR 范围。这意味着每个子网都有 4,091 个可用的 IP 地址。请注意，`/20`范围是用于 AWS 特定使用的。考虑此示例。您可以在三个可用区中将您的应用程序分发到三个子网以提供高可用性。在这种情况下，您可以跨三个子网使用大约 12,000 个 IP 地址。

使用 ENI 中继，您启动的每个 Amazon EC2 实例都需要两个 IP 地址。一个 IP 地址用于主 ENI，另一个 IP 地址用于中继 ENI。实例上的每个 Amazon ECS 任务都需要一个 IP 地址。如果要启动极大的工作负载，可用 IP 地址可能会耗尽。这可能会导致 Amazon EC2 启动失败或任务启动失败。出现这些错误的原因是，如果没有可用的 IP 地址，ENI 无法在 VPC 内添加 IP 地址。

使用`awsvpc`网络模式，您应该评估您的 IP 地址要求，并确保您的子网 CIDR 范围满足您的需求。如果您已经开始使用具有小子网的 VPC，并开始用尽地址空间，则可以添加辅助子网。



通过使用 ENI 中继，Amazon VPC CNI 可以配置为在与主机不同的 IP 地址空间中使用 ENI。通过执行此操作，您可以为您的 Amazon EC2 主机和您的任务提供不同的 IP 地址范围，这些地址不会重叠。在示例图中，EC2 主机 IP 地址位于`172.31.16.0/20`IP 范围。但是，在主机上运行的任务会在`100.64.0.0/19`范围。通过使用两个独立的 IP 范围，您无需担心任务占用过多的 IP 地址，也无需为实例留下足够的 IP 地址。

使用 IPv6 双堆栈模式

这些区域有：`awsvpc`网络模式与配置为 IPv6 双堆栈模式的 VPC 兼容。使用双堆栈模式的 VPC 可通过 IPv4、或 IPv6 进行通信。VPC 中的每个子网都可以具有 IPv4 CIDR 范围和 IPv6 CIDR 范围。有关更多信息，请参阅 。VPC 中的 IP 寻址中的Amazon VPC 用户指南。

您不能为 VPC 和子网禁用 IPv4 支持以解决 IPv4 耗尽问题。但是，借助 IPv6 支持，您可以使用一些新功能，特别是仅出口互联网关。仅出口互联网网关允许任务使用其公开路由 IPv6 地址发起到互联网的出站连接。但仅出口 Internet 网关不允许从 Internet 连接。有关更多信息，请参阅 。仅出口互联网网关中的Amazon VPC 用户指南。

连接到AWS从您的 VPC 内部提供的服务

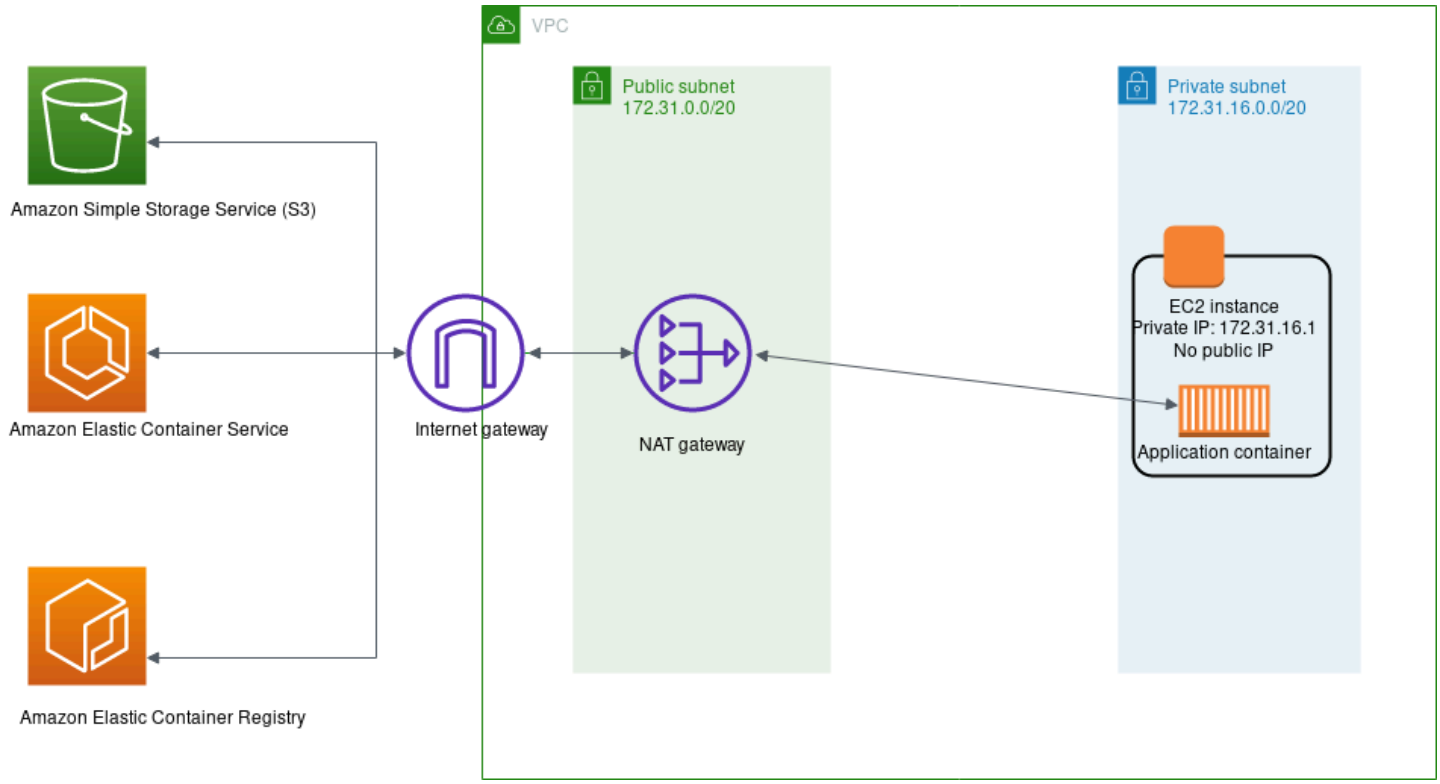
要使 Amazon 云服务器正常运行，每台主机上运行的 ECS 容器代理必须与 Amazon 云服务器控制平面通信。如果您要将容器映像存储在 Amazon ECR 中，则 Amazon EC2 主机必须与 Amazon ECR 服务终端节点和存储图像图层的 Amazon S3 进行通信。如果您使用其他AWS服务（如保留存储在 DynamoDB 中的数据），请仔细检查这些服务是否也具有必要的网络支持。

主题

- NAT 网关
- AWS PrivateLink

NAT 网关

使用 NAT 网关是确保您的 Amazon ECS 任务可以访问其他AWS服务。有关此方法的更多信息，请参阅使用私有子网和 NAT 网关。



以下是使用此方法的缺点：

- 您无法限制 NAT 网关可以与哪些目标进行通信。您也无法限制您的后端轮胎可以与哪些目的地进行通信，而不会中断 VPC 的所有出站通信。

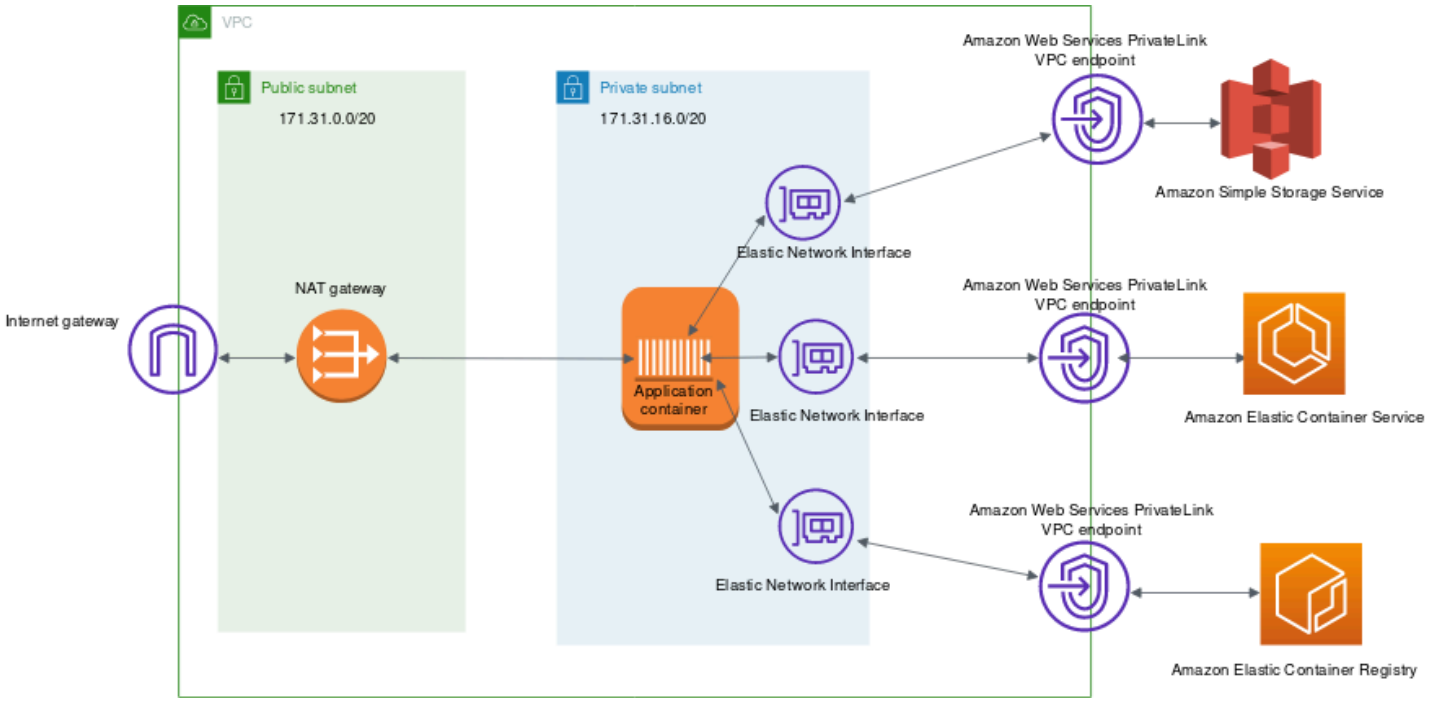
- NAT 网关会对通过的每 GB 数据进行收费。如果您使用 NAT 网关从 Amazon S3 下载大文件，或者对 DynamoDB 执行大量数据库查询，则需要为每 GB 带宽收费。此外，NAT 网关支持 5 Gbps 带宽并可自动扩展到 45 Gbps。如果通过单个 NAT 网关进行路由，则需要非常高带宽连接的应用程序可能会遇到网络限制。解决方法是，您可以跨多个子网划分工作负载，并为每个子网指定其自己的 NAT 网关。

AWS PrivateLink

AWS PrivateLink提供 VPC 之间的私有连接，AWS服务和本地网络，而不会将流量暴露给公共互联网。

用于实现此目的的技术之一是 VPC 终端节点。通过 VPC 终端节点可在您的 VPC 与受支持的AWS服务和 VPC 终端节点服务。VPC 和其他服务之间的流量不会脱离 Amazon 网络。VPC 终端节点不需要 Internet 关联、虚拟私有网关、NAT 设备、VPN 连接或AWS Direct Connect连接。VPC 中的 Amazon EC2 实例无需公有 IP 地址便可与服务中的资源进行通信。

下图演示了如何与AWS服务在您使用 VPC 终端节点而不是互联网网关时起作用。AWS PrivateLink在子网内部配置弹性网络接口 (ENI)，VPC 路由规则用于通过 ENI 将任何通信发送到服务主机名，直接发送到目的地AWS服务。此流量不再需要使用 NAT 网关或互联网网关。



以下是与亚马逊云服务器服务一起使用的一些常见 VPC 终端节点。

- S3 网关 VPC 终端节点

- DynamoDB VPC 终端节点
- Amazon ECS VPC 终端节点
- Amazon ECR VPC 终端节点

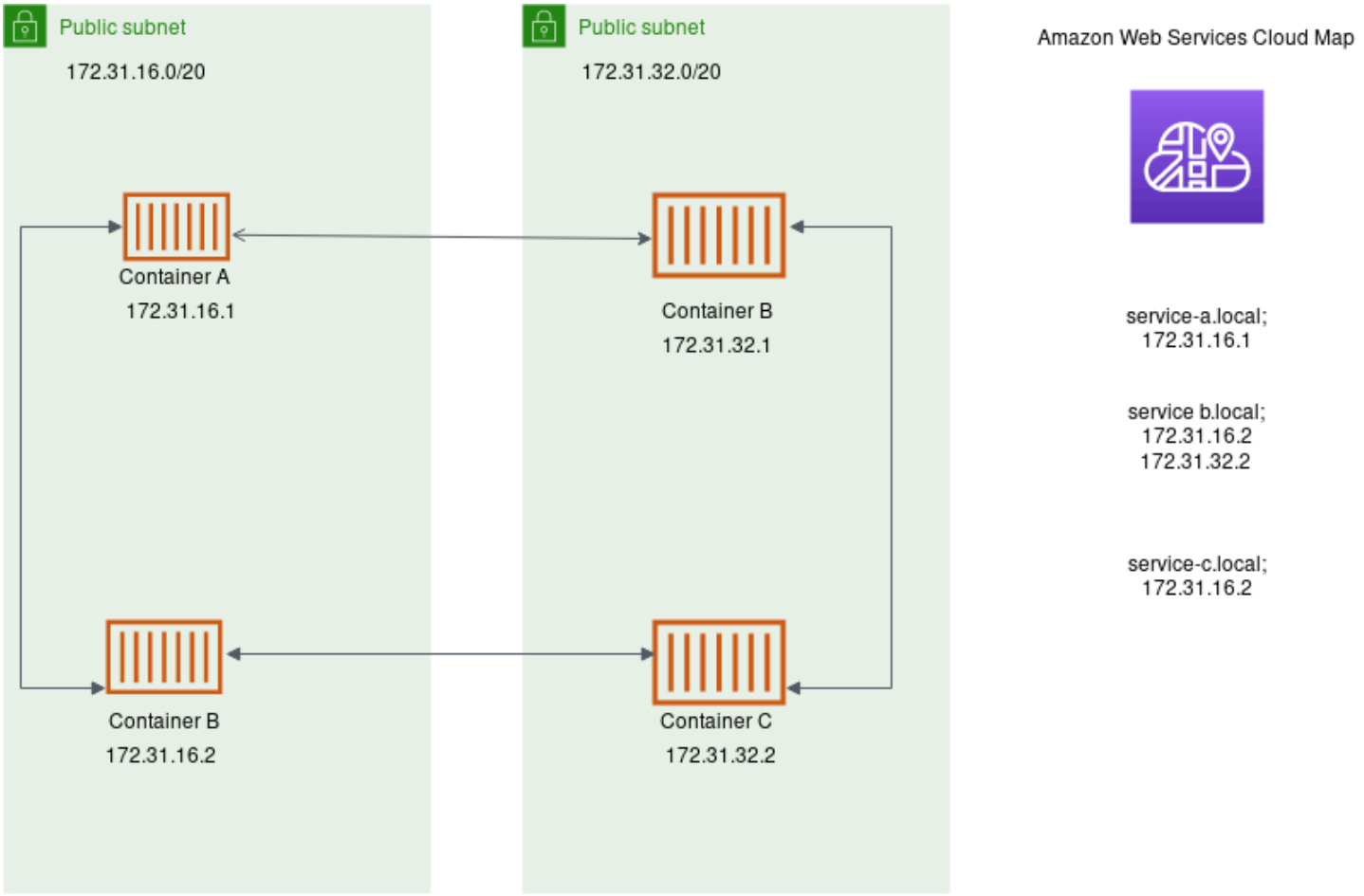
多个其他AWS服务支持 VPC 终端节点。如果您大量使用任何AWS服务，您应该查找该服务的特定文档以及如何为该流量创建 VPC 终端节点。

VPC 中的亚马逊云服务器服务之间的联网

使用 VPC 中的 Amazon ECS 容器，您可以将整体应用程序分解为可在安全环境中独立部署和扩展的单独部分。但是，要确保 VPC 内外的所有这些部分都可以相互通信，这可能是一项挑战。有几种促进沟通的方法，所有方法都有不同的优点和缺点。

使用服务发现

使用服务到服务通信的方法之一是使用服务发现进行直接通信。在这种方法中，您可以使用AWS Cloud Map服务发现与亚马逊云服务器的集成。通过服务发现，Amazon 云服务器将启动的任务列表同步到AWS Cloud Map，它维护一个 DNS 主机名，该主机名可解析为该特定服务中一个或多个任务的内部 IP 地址。Amazon VPC 中的其他服务可以使用此 DNS 主机名使用其内部 IP 地址将流量直接发送到另一个容器。有关更多信息，请参阅 。服务发现中的Amazon Elastic Container Service 开发人指南。



在前面的示意图中，有三种服务。serviceA有一个容器，并与serviceB，其中有两个容器。serviceB还必须与serviceC，它有一个容器。所有这三个服务中的每个容器都可以使用AWS Cloud Map从需要通信的下游服务中查找容器的内部 IP 地址。

这种服务到服务通信方法提供了较低的延迟。乍一看，这也很简单，因为容器之间没有额外的组件。流量从一个集装箱直接传输到另一个集装箱。

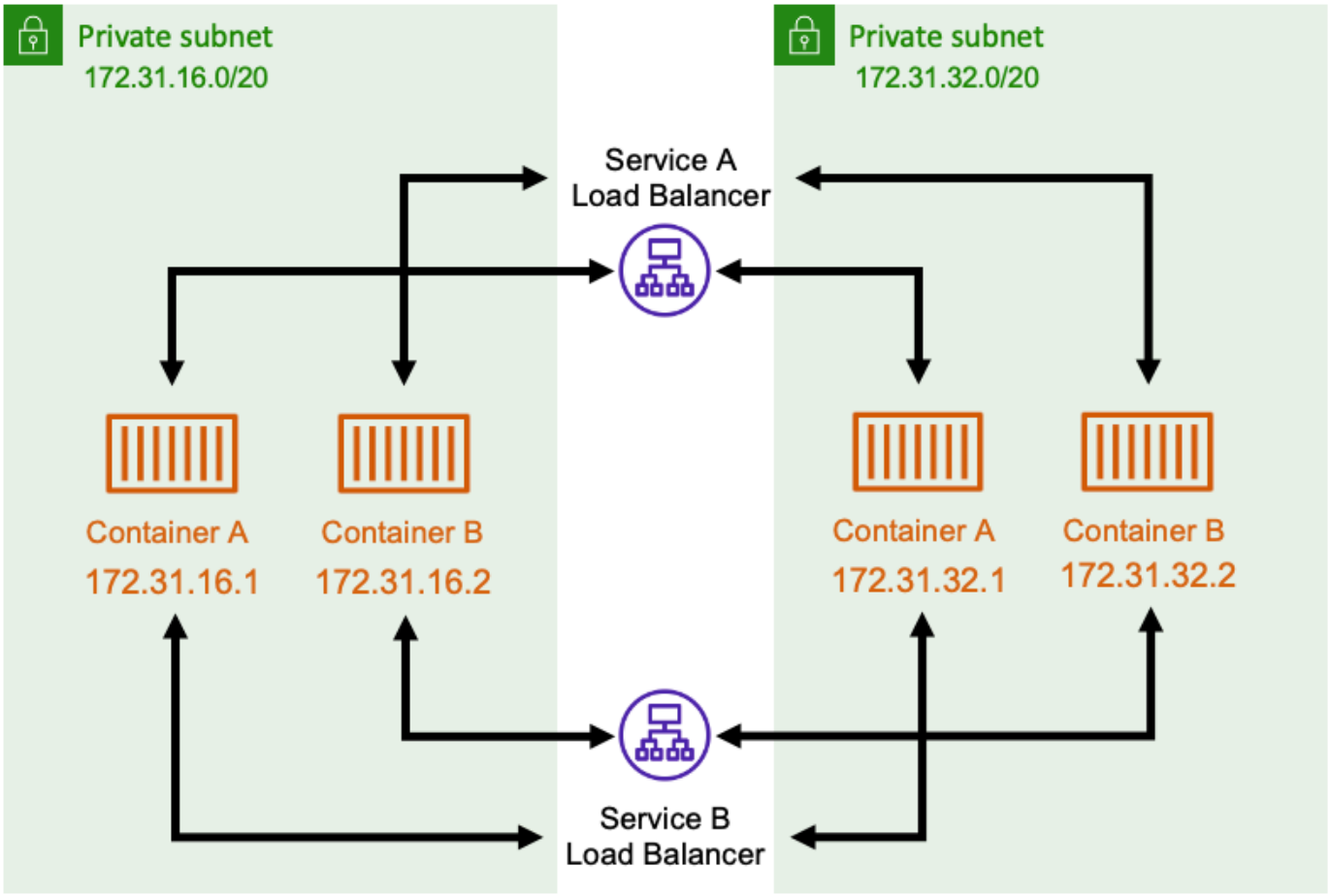
这种方法适用于使用awsvpc网络模式，其中每个任务都有自己的唯一 IP 地址。大多数软件只支持使用 DNSA记录，这些记录直接解析为 IP 地址。使用awsvpc网络模式下，每个任务的 IP 地址都是A记录。但是，如果您使用的是bridge网络模式时，多个容器可能共享相同的 IP 地址。此外，动态端口映射会导致容器随机分配该单个 IP 地址上的端口号。在这一点上，A记录不再足以用于服务发现。您还必须使用SRV记录。此类记录可以跟踪 IP 地址和端口号，但要求您适当配置应用程序。您使用的某些预构建应用程序可能不支持SRV记录。

另一个优势awsvpc网络模式的原因是，每个服务都有一个唯一的安全组。您可以将此安全组配置为仅允许来自需要与该服务通信的特定上游服务的传入连接。

使用服务发现直接服务到服务通信的主要缺点是，您必须实现额外的逻辑才能重试并处理连接故障。DNS 记录具有生存时间 (TTL) 期间，用于控制缓存时长。DNS 记录更新和缓存过期需要一些时间，以便您的应用程序能够获取最新版本的 DNS 记录。因此，您的应用程序可能最终解析 DNS 记录，以指向另一个不再存在的容器。您的应用程序需要处理重试，并具有忽略坏后端的逻辑。

使用内部负载均衡器

服务到服务通信的另一种方法是使用内部负载均衡器。内部负载均衡器完全存在于您的 VPC 内部，并且只能对 VPC 内部的服务进行访问。



负载均衡器通过在每个子网中部署冗余资源来维护高可用性。当一个容器`serviceA`需要与来自`serviceB`，它会打开与负载均衡器的连接。然后，负载均衡器从`service B`。负载均衡器可作为管理每个服务之间的所有连接的集中位置。

如果一个容器来自`serviceB`停止，则负载均衡器可以从池中删除该容器。负载均衡器还会对池中的每个下游目标进行运行状况检查，并且可以自动从池中删除坏目标，直到它们再次恢复正常状态。应用程序不再需要知道有多少下游容器。他们只是打开与负载均衡器的连接。

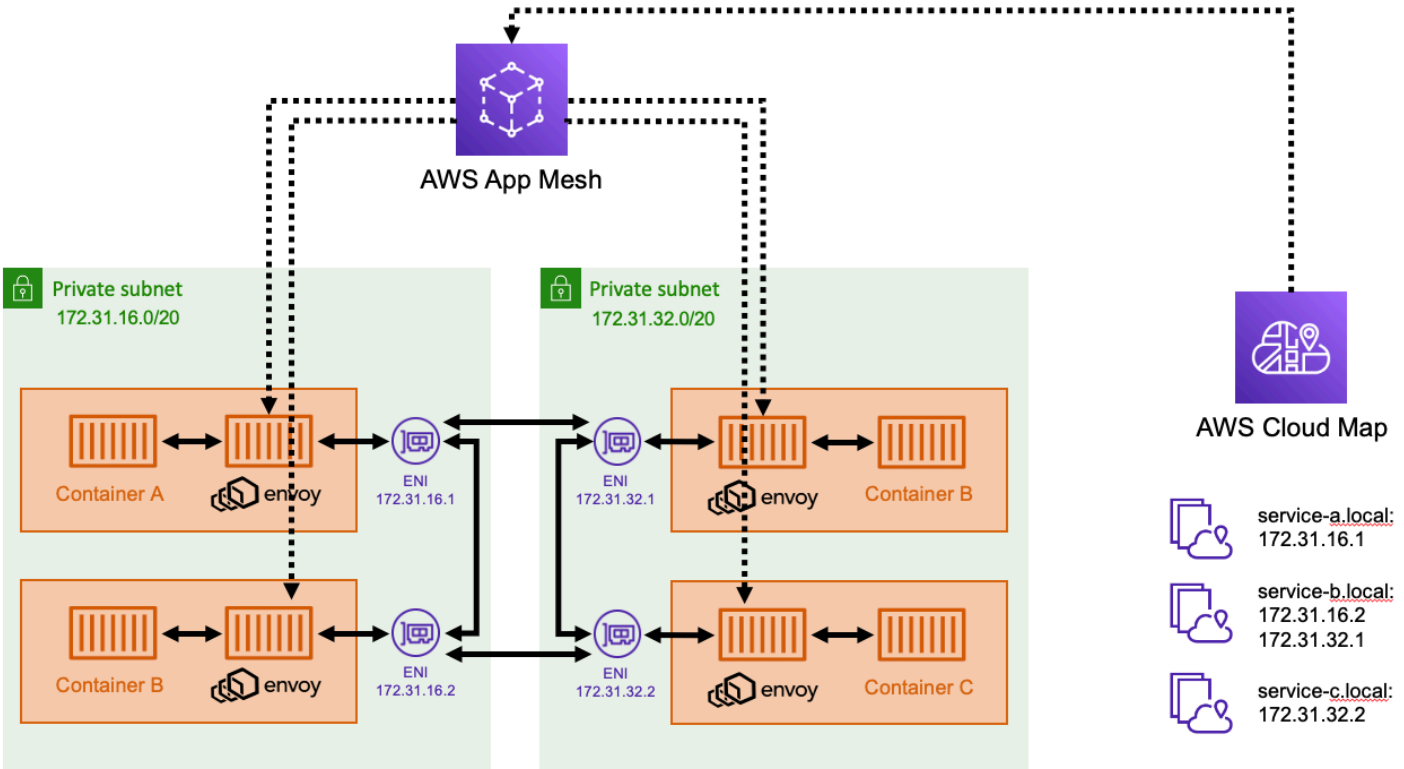
此方法适用于所有网络模式。负载均衡器可以在使用awsvpc网络模式，以及 IP 地址和端口的更高级组合使用bridge网络模式。它在所有 IP 地址和端口组合之间均匀分配流量，即使多个容器实际上托管在同一 Amazon EC2 实例上，仅在不同的端口上也是如此。

这种做法的一个缺点是成本。要实现高可用性，负载均衡器需要在每个可用区域中拥有资源。这会增加额外的成本，因为为负载均衡器付费的开销以及通过负载均衡器的流量。

但是，您可以通过让多个服务共享一个负载均衡器来降低开销成本。这特别适用于使用应用 Application Load Balancer 的 REST 服务。您可以创建基于路径的路由规则，将流量路由到不同服务。例如，/api/user/*可能会路由到属于user服务，而/api/order/*可能会路由到关联的order服务。使用此方法，您只需为一个 Application Load Balancer 付费，并为您的 API 提供一个一致的 URL。但是，您可以将流量拆分到后端上的各种微服务。

使用服务网格

AWS App Mesh是一种服务网格，可帮助您管理大量服务，并更好地控制服务之间的流量路由方式。App Mesh 充当基本服务发现和负载平衡之间的中介。借助 App Mesh，应用程序不会直接相互交互，但它们也不使用集中式负载均衡器。相反，您的任务的每个副本都伴随着一个特使代理边车。有关更多信息，请参阅 AWS App Mesh 用户指南 中的什么是 AWS App Mesh。



在前面的示意图中，每个任务都有一个 Enway 代理边车。此边车负责代理任务的所有入站和出站流量。App Mesh 控制平面使用AWS Cloud Map获取可用服务列表和特定任务的 IP 地址。然后，App Mesh 将配置提供给特使代理边车。此配置包括可连接到的可用容器的列表。特使代理边车还对每个目标进行运行状况检查，以确保它们可用。

此方法提供了服务发现的功能，并且易于管理的负载均衡器。应用程序在其代码中没有实现尽可能多的负载平衡逻辑，因为使用代理边车来处理负载平衡。可以将使用代理配置为检测故障并重试失败的请求。此外，还可以将其配置为使用 MTL 对传输中的流量进行加密，并确保您的应用程序与已验证的目的地进行通信。

Ever 代理与负载均衡器之间有一些区别。简而言之，使用特使代理，您负责部署和管理自己的特使代理边车。特使代理边车使用您分配给 Amazon 云服务器任务的一些 CPU 和内存。这增加了任务资源消耗的一些开销，并增加了在需要时维护和更新代理所需的额外操作工作量。

App Mesh 和特使代理允许在任务之间实现极低的延迟。这是因为特使代理运行与每个任务并置。在一个使用代理和另一个使用代理之间，只有一个实例进行网络跳转。这意味着与使用负载平衡器相比，网络开销也更少。使用负载平衡器时，有两个网络跳转。第一个是从上游任务到负载均衡器，第二个是从负载均衡器到下游任务。

跨网络服务AWS帐户和 VPC

如果您是一个拥有多个团队和部门的组织的一部分，则可能会将服务单独部署到AWS帐户或与多个单独关联的 VPCAWS帐户。无论您以何种方式部署服务，我们都建议您补充网络组件，以帮助在 VPC 之间路由流量。为此，几个AWS服务可用于补充现有网络组件。

- AWS Transit Gateway — 您应该首先考虑此网络服务。此服务可作为在亚马逊 VPC 之间路由连接的中心枢纽，AWS帐户和本地网络。有关更多信息，请参阅 。什么是中转网关？中的亚马逊 VPC 中转网关指南。
- Amazon VPC 和 VPN 支持 — 您可以使用此服务创建站点到站点 VPN 连接，以将本地网络连接到您的 VPC。有关更多信息，请参阅 。是什么AWS Site-to-Site VPN?中的AWS Site-to-Site VPN用户指南。
- Amazon VPC — 您可以使用 Amazon VPC 对等功能来帮助您在同一账户中或跨账户连接多个 VPC。有关更多信息，请参阅 。什么是 VPC 对等？中的Amazon VPC 对等指南。
- 共享 VPC — 您可以跨多个 VPC 和 VPC 子网AWS帐户。有关更多信息，请参阅 。使用共享 VPC中的Amazon VPC 用户指南。

优化和问题排查

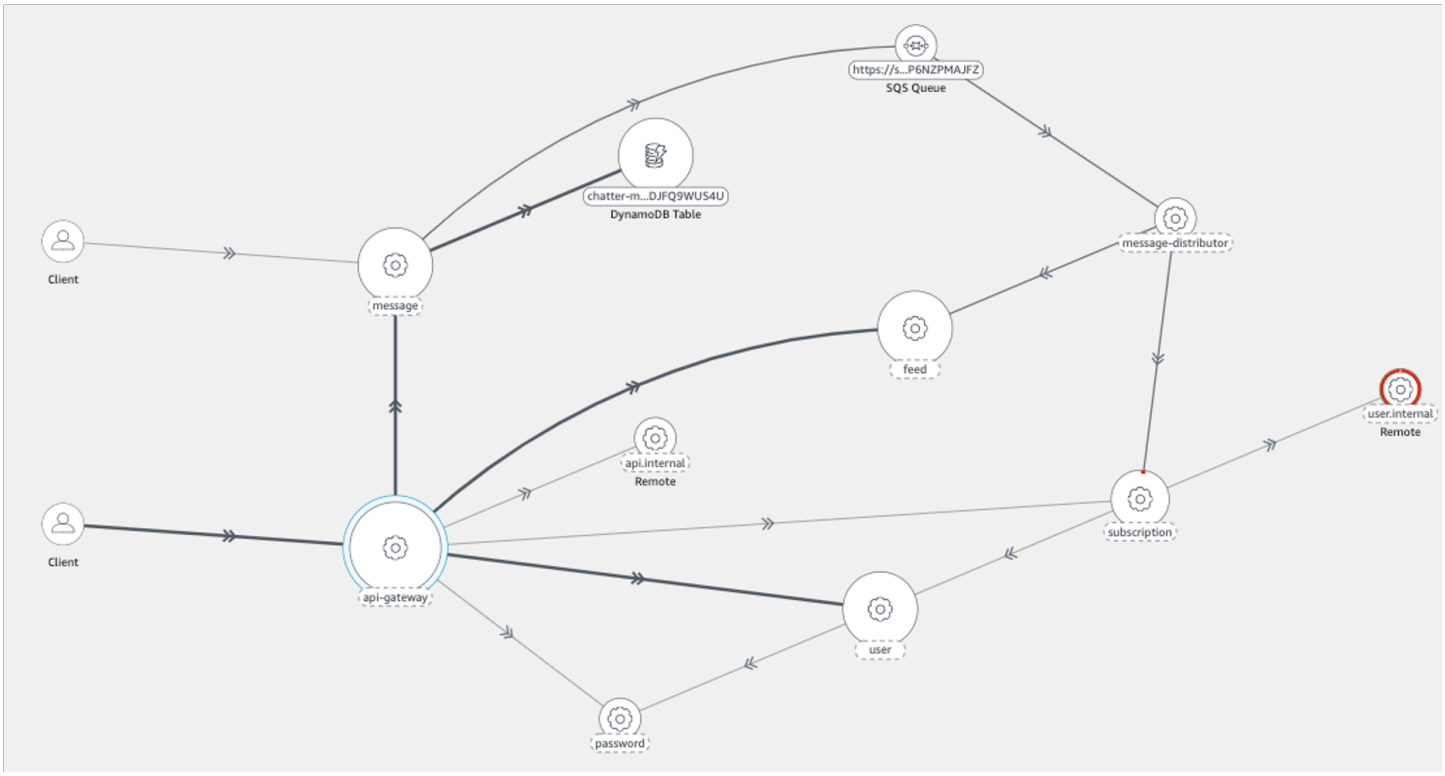
以下服务和功能可帮助您深入了解您的网络和服务配置。您可以使用此信息排查网络问题并更好地优化服务。

CloudWatch 容器洞察

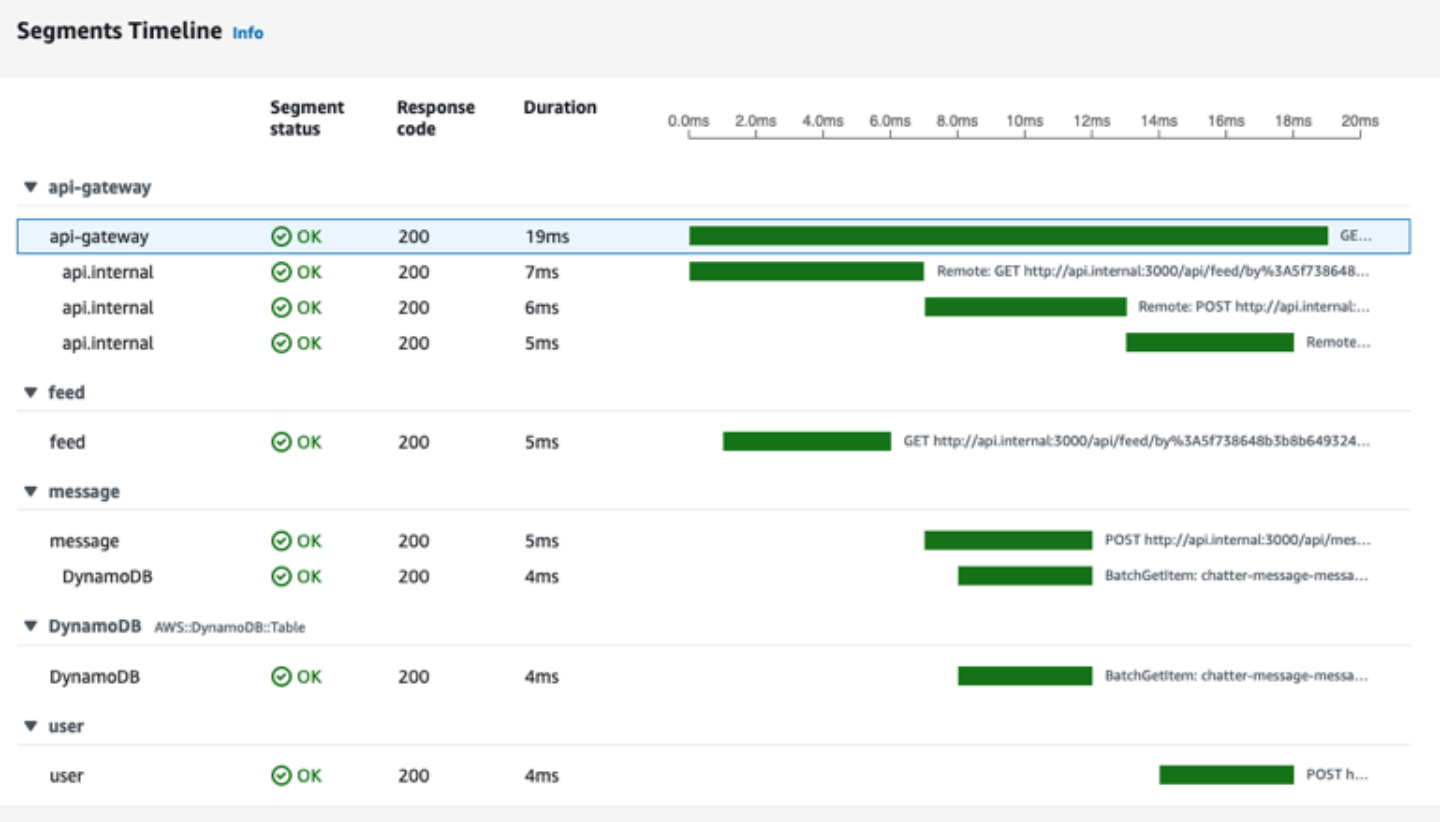
CloudWatch 容器见解从容器化应用程序和微服务中收集、聚合和汇总指标和日志。指标包括资源的使用率，如 CPU、内存、磁盘和网络。它们可以在 CloudWatch 自动仪表板中使用。有关更多信息，请参阅 。在亚马逊云服务器上设置容器见解中的Amazon CloudWatch 用户指南。

AWS X-Ray

AWS X-Ray是一种跟踪服务，可用于收集有关应用程序发出的网络请求的信息。您可以使用 SDK 来测量您的应用程序，并捕获服务之间以及服务和服务之间的流量的时间和响应代码AWS服务终端节点。有关更多信息，请参阅 。是什么AWS X-Ray中的AWS X-Ray开发人员指南。



您还可以浏览AWS X-Ray您的服务如何互相网络的图形。或者，使用它们来探索有关每个服务到服务链接的性能的聚合统计信息。最后，您可以深入了解任何特定事务，了解代表网络调用的区段如何与该特定事务相关联。



您可以使用这些功能来确定是否存在网络瓶颈，或者网络中的特定服务是否未按预期运行。

VPC 流日志

您可以使用 Amazon VPC 流日志来分析网络性能和调试连接问题。启用 VPC 流日志后，您可以捕获 VPC 中所有连接的日志。其中包括与 Elastic Load Balancing、Amazon RDS、NAT 网关和其他密钥相关联的网络接口的连接AWS服务。有关更多信息，请参阅 。VPC 流日志中的Amazon VPC 用户指南。

网络调节提示

您可以对一些设置进行微调，以改善您的网络。

无文件 Ulimit

如果您希望应用程序具有较高的流量并处理许多并发连接，则应考虑允许的文件数量的系统配额。当打开大量网络套接字时，每个套接字都必须用一个文件描述符表示。如果您的文件描述符配额过低，则会限制您的网络套接字。这会导致连接失败或错误。您可以更新 Amazon ECS 任务定义中文件数的特定容器配额。如果您在 Amazon EC2 上运行（而不是AWS Fargate），那么您可能还需要在底层 Amazon EC2 实例上调整这些配额。

Syssctl 网

另一个可调的设置类别是`sysctl`NET 设置。您应该参考所选 Linux 发行版的具体设置。许多这些设置都会调整读取和写入缓冲区的大小。在某些情况下，当运行包含大量容器的大规模 Amazon EC2 实例时，这会有所帮助。

最佳实践-自动扩展和容量管理

Amazon ECS 用于运行各种规模的容器化应用程序工作负载。这包括极端的测试环境和在全球范围内运行的大型生产环境。

使用亚马逊云服务器，像所有AWS服务，您可以按实际用量付费。经过适当的架构设计后，您可以通过让您的应用程序在需要时仅占用它所需的资源来节省成本。本最佳实践指南介绍了如何以符合服务级别目标的方式运行 Amazon ECS 工作负载，同时仍以经济高效的方式运行。

主题

确定任务大小

在 Amazon ECS 上部署容器时，最重要的选择之一是容器和任务大小。容器和任务大小对于扩展和容量规划都是必不可少的。在 Amazon ECS 中，有两个用于容量的资源指标：CPU 和内存。CPU 的测量单位为完整 vCPU 的 1/1024（其中 1024 个单位等于整个虚拟 CPU）。内存以兆字节为单位。在任务定义中，您可以声明资源保留和限制。

当您声明预留时，您声明了任务所需的最小资源量。您的任务至少会收到请求的资源量。您的应用程序可能能够使用比您声明的预留更多的 CPU 或内存。但是，这受您同时声明的任何限制。使用超过预留金额称为突发。在亚马逊云服务器中，保证预订。例如，如果您使用 Amazon EC2 实例提供容量，Amazon ECS 不会在无法完成预留的实例上放置任务。

限制是容器或任务可以使用的 CPU 单位或内存的最大数量。任何尝试使用更多的 CPU 比这个模拟都会导致限制。任何尝试使用更多内存都会导致容器停止。

选择这些值可能是一个难题。这是因为最适合您的应用程序的值在很大程度上取决于应用程序的资源需求。负载测试应用程序是成功规划资源需求和更好地了解应用程序要求的关键。

无状态申请

对于水平扩展的无状态应用程序（如负载均衡器后面的应用程序），我们建议您首先确定应用程序在处理请求时占用的内存量。为此，您可以使用传统工具，例如`ps`或者`top`或监控解决方案（如 CloudWatch 容器见解）。

确定 CPU 预留时，请考虑如何扩展应用程序以满足业务需求。您可以使用较小的 CPU 预留（例如 256 个 CPU 单位（或 1/4 vCPU）），以最大限度地降低成本的细粒度方式向外扩展。但是，它们的扩展速度可能不够快，无法满足重大的需求峰值。您可以使用较大的 CPU 预留来更快地扩展和向外扩展，从而更快地匹配需求峰值。但是，较大的 CPU 预留成本就越高。

其他应用程序

对于不能横向扩展的应用程序（如单例工作人员或数据库服务器），可用容量和成本是您最重要的考虑因素。您应根据负载测试表明您需要提供流量以满足服务级别目标的情况来选择内存量和 CPU 量。Amazon ECS 确保将应用程序放置在具有足够容量的主机上。

配置服务 Auto Scaling

亚马逊云服务器服务是托管的任务集合。每个服务都有一个关联的任务定义、所需的任务计数和一个可选的放置策略。Amazon ECS 服务自动扩展是通过 Application Auto Scaling 服务实现的。Application Auto Scaling 使用 CloudWatch 指标作为扩展指标的源。它还使用 CloudWatch 警报来设置何时扩展或向外扩展服务的阈值。您可以通过设置度量目标（称为目标跟踪扩展，或者通过指定阈值（称为步进扩展。配置 Application Auto Scaling 后，它会持续计算服务的相应所需任务计数。它还会通过向外扩展或扩展任务计数在所需任务数发生变化时通知 Amazon ECS。

要有效地使用服务自动扩展，您必须选择适当的扩展衡量指标。我们将在以下部分讨论如何选择指标。

确定应用程序的特征

正确扩展应用程序需要了解应用程序应在何处扩展以及何时扩展应用程序。实质上，如果预测需求超出容量，应该扩展应用程序。相反，可以扩展应用程序，以便在资源超出需求时节省成本。

确定利用率度量

为了有效扩展，确定指示利用率或饱和度的度量至关重要。此衡量指标必须具有以下属性才能对扩展有用。

- 该指标必须与需求相关。当资源保持稳定，但需求发生变化时，度量值也必须更改。当需求增加或减少时，指标应增加或减少。

- 指标值必须按容量成比例扩展。当需求保持不变时，添加更多资源必须导致指标值的成比例更改。因此，任务数增加一倍会导致指标减少 50%。

确定利用率指标的最佳方法是在预生产环境（如临时环境）中进行负载测试。广泛提供商用和开源负载测试解决方案。这些解决方案通常可以产生合成负载或模拟实际用户流量。

要启动负载测试过程，您应首先为应用程序的利用率指标构建仪表板。这些指标包括 CPU 利用率、内存利用率、I/O 操作、I/O 队列深度和网络吞吐量。您可以使用诸如 CloudWatch 容器见解之类的服务收集这些指标。或者，通过使用亚马逊 Prometheus 托管服务和亚马逊 Managed Service for Grafana 来完成此操作。在此过程中，请确保收集和绘制应用程序响应时间或工作完成率的指标。

进行负载测试时，从较小的请求或作业插入速率开始。保持此速度稳定几分钟，以便您的应用程序进行热身。然后，慢慢增加速率并保持稳定几分钟。重复此循环，每次增加速率，直到应用程序的响应或完成时间太慢，无法满足服务级别目标 (SL)。

在负载测试时，检查每个利用率指标。随负载而增加的指标是作为最佳利用率指标的最佳候选指标。

接下来，确定达到饱和度的资源。同时，还要检查利用率度量，以查看哪一个在高级别首先展平。或者，检查哪一个达到峰值，然后首先崩溃您的应用程序。例如，如果在添加负载时 CPU 利用率从 0% 增加到 70-80%，那么在添加更多负载后保持在该 lebel 处，那么可以安全地说 CPU 是饱和的。根据 CPU 体系结构，它可能永远不会达到 100%。例如，假设内存利用率随着您添加负载而增加，然后您的应用程序在达到任务或 Amazon EC2 实例内存限制时突然崩溃。在这种情况下，很可能是内存已被完全消耗的情况。您的应用程序可能会占用多个资源。因此，请选择表示首先耗尽的资源的度量。

最后，在任务或 Amazon EC2 实例数增加一倍后，再次尝试加载测试。假设关键指标的增加或减少速率是之前的一半。如果是这种情况，则指标与容量成正比。对于自动扩展，这是一个很好的利用率指标。

现在考虑这种假设情况。假设您加载测试一个应用程序，并发现 CPU 利用率最终达到 80%，每秒 100 个请求。当添加更多负载时，它不会再提高 CPU 利用率。但是，它确实会使您的应用程序响应更慢。然后，您再次运行负载测试，使任务数增加一倍，但保持速率在其以前的峰值值。如果您发现平均 CPU 利用率降至 40% 左右，则平均 CPU 利用率是扩展指标的理想选择。另一方面，如果增加任务数后 CPU 利用率仍保持在 80%，则平均 CPU 利用率不是一个很好的扩展指标。在这种情况下，需要进行更多的研究，以找到合适的衡量标准。

常见应用程序模型和扩展属性

所有类型的软件都在AWS。许多工作负载都是自行开发的，而其他工作负载则基于流行的开源软件。无论它们来自何处，我们都观察到了一些常见的服务设计模式。如何有效地缩放在很大程度上取决于模式。

高效的 CPU 绑定服务器

高效的 CPU 绑定服务器除了 CPU 和网络吞吐量之外，几乎没有使用任何其他资源。每个请求都可以由应用程序单独处理。请求不依赖于其他服务，如数据库。该应用程序可以处理成千上万的并发请求，并且可以有效地利用多个 CPU 来完成此操作。每个请求都由内存开销较低的专用线程处理，或者在每个服务请求的 CPU 上运行异步事件循环。应用程序的每个副本都同样能够处理请求。CPU 之前可能耗尽的唯一资源是网络带宽。在 CPU 绑定服务中，即使在峰值吞吐量下，内存利用率也是可用资源的一小部分。

这种类型的应用程序适用于基于 CPU 的自动缩放。该应用程序在扩展方面享有最大的灵活性。它可以通过向其提供更大的 Amazon EC2 实例或 Fargate vCPUs 来进行垂直扩展。而且，它也可以通过添加更多副本来水平缩放。添加更多副本或将实例大小增加一倍，可将相对于容量的平均 CPU 利用率降低一半。

如果您将 Amazon EC2 容量用于此应用程序，请考虑将其放置在计算优化的实例上，例如`c5`或者`c6g`系列。

高效的内存绑定服务器

高效的内存绑定服务器为每个请求分配大量内存。在最大并发时（但不一定是吞吐量），内存会在 CPU 资源耗尽之前耗尽。请求结束时释放与请求关联的内存。只要有可用的内存，就可以接受额外的请求。

这种类型的应用程序适用于基于内存的自动缩放。该应用程序在扩展方面享有最大的灵活性。它可以通过向其提供更大的 Amazon EC2 或 Fargate 内存资源来垂直扩展。而且，它也可以通过添加更多副本来水平缩放。添加更多副本或将实例大小增加一倍，可以将相对于容量的平均内存利用率降低一半。

如果您将 Amazon EC2 容量用于此应用程序，请考虑将其放置在内存优化的实例上，例如`r5`或者`r6g`系列。

某些内存绑定的应用程序不会在请求结束时释放与请求相关联的内存，因此减少并发不会导致使用的内存减少。为此，我们不建议您使用基于内存的扩展。

基于工作者的服务器

基于工作程序的服务器为每个单独的工作线程逐个处理一个请求。工作线程可以是轻量级线程，例如 POSIX 线程。它们也可以是重量较重的线程，例如 UNIX 进程。无论它们是哪个线程，应用程序总是可以支持的最大并发性。通常，并发限制按比例设置可用的内存资源。如果达到并发限制，则会将其他请求放入积压队列。如果积压队列溢出，则会立即拒绝其他传入请求。适合此模式的常见应用程序包括 Apache Web 服务器和 Gunicorn。

请求并发通常是扩展此应用程序的最佳衡量指标。由于每个副本都有并发限制，因此在达到平均限制之前向外扩展非常重要。

获取请求并发指标的最佳方法是让您的应用程序向 CloudWatch 报告这些指标。应用程序的每个副本都可以高频率地将并发请求数作为自定义指标发布。我们建议将频率设置为至少每分钟一次。收集多个报告后，您可以使用平均并发作为扩展衡量指标。此度量的计算方法是：获取总并发性并将其除以副本数。例如，如果总并发性为 1000，副本数为 10，则平均并发性为 100。

如果您的应用程序位于应用程序负载均衡器后面，则还可以使用`ActiveConnectionCount`指标作为扩展指标中的一个因子。这些区域有：`ActiveConnectionCount`度量必须除以复制副本数才能获得平均值。平均值必须用于缩放，而不是原始计数值。

为了使这种设计发挥最佳效果，响应延迟的标准差在低请求速率下应该很小。我们建议在低需求期间，大多数请求在短时间内得到答复，而且没有太多的请求比平均时间长得多。平均响应时间应接近第 95 个百分位数响应时间。否则，可能会导致队列溢出。这会导致错误。我们建议您在必要时提供其他副本，以减少溢出风险。

等待服务器

等待服务器会对每个请求执行一些处理，但它高度依赖于一个或多个下游服务才能运行。容器应用程序通常大量使用下游服务，如数据库和其他 API 服务。这些服务可能需要一些时间才能响应，特别是在高容量或高并发情况下。这是因为这些应用程序往往使用的 CPU 资源很少，而且它们在可用内存方面的最大并发性。

等待服务适用于内存绑定的服务器模式或基于工作者的服务器模式，具体取决于应用程序的设计方式。如果应用程序的并发性仅受内存限制，则应将平均内存利用率用作扩展衡量指标。如果应用程序的并发性基于工作线程限制，则平均并发应用作扩展衡量指标。

基于 Java 的服务器

如果基于 Java 的服务器受 CPU 约束，并且按 CPU 资源比例扩展，那么它可能适用于高效的 CPU 绑定服务器模式。如果是这种情况，则平均 CPU 利用率可能适合作为扩展指标。但是，许多 Java 应用程序不受 CPU 约束，从而使它们难以扩展。

为了获得最佳性能，我们建议您为 Java 虚拟机 (JVM) 堆分配尽可能多的内存。最新版本的 JVM（包括 Java 8 更新 191 或更高版本）会自动设置尽可能大的堆大小以适应容器。这意味着，在 Java 中，内存利用率很少与应用程序利用率成正比。随着请求速率和并发性的增加，内存利用率保持不变。因此，我们不建议根据内存利用率扩展基于 Java 的服务器。相反，我们通常建议扩展 CPU 利用率。

在某些情况下，基于 Java 的服务器在耗尽 CPU 之前会遇到堆耗尽。如果您的应用程序在高并发时容易出现堆耗尽，那么平均连接是最佳的扩展指标。如果您的应用程序在高吞吐量时容易出现堆耗尽，则平均请求速率是最佳扩展指标。

使用其他垃圾收集运行时的服务器

许多服务器应用程序都基于执行垃圾回收（如 .NET 和 Ruby）的运行时。这些服务器应用程序可能适用于前面描述的模式之一。但是，与 Java 一样，我们不建议根据内存扩展这些应用程序，因为它们观察到的平均内存利用率通常与吞吐量或并发无关。

对于这些应用程序，如果应用程序受 CPU 限制，我们建议您扩展 CPU 利用率。否则，我们建议您根据负载测试结果扩展平均吞吐量或平均并发性。

Job 处理器

许多工作负载都涉及异步作业处理。它们包括不实时接收请求的应用程序，而是订阅工作队列以接收作业。对于这些类型的应用程序，正确的扩展衡量指标几乎始终是队列深度。队列增长表示待处理工作超出了处理能力，而空队列表示存在的容量多于工作要完成的容量。

AWS消息传送服务（如 Amazon SQS 和 Amazon Kinesis Data Streams）提供可用于扩展的 CloudWatch 指标。对于 Amazon SQS，`ApproximateNumberOfMessagesVisible`是最佳衡量指标。对于 Kinesis Data Streams，请考虑使用`MillisBehindLatest`指标，由 Kinesis 客户端库 (KCL) 发布。在使用此指标进行扩展之前，应该在所有使用者之间对其进行平均处理。

容量和可用性

应用程序可用性对于提供无错误的体验和最大限度地减少应用程序延迟至关重要。可用性取决于是否具有可访问的资源，并具有足够的容量来满足需求。AWS提供了多种机制来管理可用性。对于 Amazon 云服务器上托管的应用程序，这些应用程序包括自动扩展和可用区 (AZ)。Autocaling 根据您定义的指标管理任务或实例的数量，而可用区则允许您将应用程序托管在隔离但地理位置相近的位置。

与任务规模一样，容量和可用性提出了您必须考虑的某些权衡。理想情况下，容量将完全符合需求。总是有足够的容量来满足请求和处理作业，以满足服务级别目标 (SL)，包括低延迟和低错误率。容量永远不会过高，导致成本过高；也不会太低，从而导致高延迟和错误率。

自动缩放是一个潜在的过程。首先，必须将实时指标交付到 CloudWatch。然后，需要对它们进行聚合以进行分析，这可能需要几分钟时间，具体取决于指标的粒度。CloudWatch 将指标与警报阈值进行比较，以确定资源短缺或过剩。为了防止不稳定，请将警报配置为要求在警报熄灭前超过设定的阈值几分钟。配置新任务和终止不再需要的任务也需要时间。

由于所述系统中存在这些潜在的延迟，因此通过过度配置保持一定的空间非常重要。这样做可以帮助适应短期的需求突发。这也有助于您的应用程序在不达到饱和度的情况下处理其他请求。作为一种良好做法，您可以将利用率的 60-80% 设置为扩展目标。这有助于您的应用程序更好地处理突发的额外需求，同时还在配置额外容量。

我们建议您过度配置的另一个原因是，您可以快速响应可用区故障。AWS建议从多个可用区域为生产工作负载提供服务。这是因为，如果发生可用区故障，在剩余可用区中运行的任务仍然可以满足需求。如果您的应用程序在两个可用区中运行，则需要将正常任务计数翻一番。这样，您就可以在任何潜在故障期间立即提供容量。如果您的应用程序在三个可用区中运行，我们建议您运行正常任务计数的 1.5 倍。也就是说，为普通服务所需的每两个运行三个任务。

最大限度扩展速度

自动缩放是一个被动过程，需要时间才能生效。但是，有一些方法可以帮助最大限度地缩短横向扩展所需的时间。

最小化图像大小。 较大的图像需要更长时间才能从图像存储库下载和解压缩。因此，保持较小的图像大小可以减少容器启动所需的时间。要减小图像大小，您可以遵循以下具体建议：

- 如果您可以构建静态二进制文件或使用 Golang，请构建您的映像FROM从头开始，并仅在生成的图像中包含您的二进制应用程序。
- 使用来自上游发行版供应商（如 Amazon Linux 或 Ubuntu）的最小化基础映像。
- 不要在最终映像中包含任何构建工件。使用多阶段构建可以帮助解决这个问题。
- 紧凑型RUN在可能的情况下阶段. EACHRUN阶段会创建一个新的影像图层，从而导致额外的往返行程来下载图层。单个RUN阶段，具有多个命令&&具有的图层少于具有多个RUN阶段。
- 如果要在最终映像中包含数据（例如 ML 推理数据），则只包含启动和开始提供流量所需的数据。如果您在不影响服务的情况下从 Amazon S3 或其他存储中按需获取数据，请将数据存储在这些位置。

保持您的图像接近。 网络延迟越高，下载映像所需的时间就越长。将您的图像托管在同一个AWS您的工作负载所在的区域。Amazon ECR 是一个高性能映像存储库，适用于亚马逊云服务器所在的每个区域。避免通过互联网或 VPN 链接下载容器映像。在同一地区托管映像可提高整体可靠性。它可以降低不同地区的网络连接问题和可用性问题的风险。或者，您也可以实施 Amazon ECR 跨区域复制来帮助解决这一问题。

降低负载均衡器运行状况检查阈值。 负载均衡器在向应用程序发送流量之前执行运行状况检查。目标组的默认运行状况检查配置可能需要 90 秒或更长时间。在此期间，它会检查运行状况并接收请求。降低运行状况检查间隔和阈值计数可以使您的应用程序更快地接受流量并减少其他任务的负载。

考虑冷启动性能。 某些应用程序使用运行时（如 Java）执行即时 (JIT) 编译。编译过程至少在启动时可以显示应用程序性能。解决方法是使用不会造成冷启动性能损失的语言重写工作负载中的延迟关键部分。

使用步进扩展策略，而不使用目标跟踪扩展策略。 您有多个 Application Auto Scaling 选项用于 Amazon ECS 任务。目标跟踪是使用最简单的模式。有了它，您只需设置一个指标的目标值，例如 CPU 平均利用率。然后，自动缩放器会自动管理获得该值所需的任务数。但是，我们建议您改用步进缩放，以便您能够更快地响应需求的变化。使用步骤扩展，您可以为扩展指标定义特定阈值，以及超过阈值时要添加或删除的任务数。而且，更重要的是，您可以通过最大限度地减少阈值警报违反的时间来对需求变化作出快速反应。有关更多信息，请参阅 。Auto Scaling 服务中的Amazon Elastic Container Service 开发指南。

如果您使用 Amazon EC2 实例提供集群容量，请考虑以下建议：

使用更大的 Amazon EC2 实例和更快的 Amazon EBS 卷。 您可以通过使用更大的 Amazon EC2 实例和更快的 Amazon EBS 卷来提高映像下载和准备速度。在给定的 Amazon EC2 实例系列中，网络和 Amazon EBS 最大吞吐量随实例大小的增加而增加（例如，从`m5.xlarge`到`m5.2xlarge`)。此外，您还可以自定义 Amazon EBS 卷，以提高其吞吐量和 IOPS。例如，如果您使用的是`gp2`卷，请使用可提供更多基准吞吐量的较大卷。如果您使用的是`gp3`卷，请在创建卷时指定吞吐量和 IOPS。

对 Amazon EC2 实例上运行的任务使用桥接网络模式。 使用的任务`bridge`网络模 Amazon EC2 动速度比使用`awsvpc`网络模式。何时`awsvpc`使用网络模式，Amazon ECS 会在启动任务之前将 elastic network interface (ENI) 附加到实例。这会导致额外的延迟。虽然使用桥接网络有几个折衷。这些任务没有自己的安全组，并且对负载平衡有一些影响。有关更多信息，请参阅 。负载均衡器目标组中的Elastic Load Balancing 用户指南。

应对需求冲击

某些应用程序经历了突然的大量需求冲击。发生这种情况的原因有多种：新闻事件、大销售、媒体事件或其他事件，这些事件会引起病毒化并导致流量在很短的时间内迅速显著增加。如果计划外，则可能会导致需求快速超过可用资源。

处理需求冲击的最佳方式是预测需求冲击并作出相应的规划。由于自动缩放可能需要时间，因此我们建议您在需求冲击开始之前横向扩展应用程序。为了获得最佳效果，我们建议您制定一个业务计划，该计划涉及使用共享日历的团队之间的紧密协作。计划活动的团队应提前与负责申请的团队紧密合作。这使团队有足够的时间制定明确的计划计划。他们可以安排容量，以便在事件发生之前向外扩展，并在事件发生后扩展。有关更多信息，请参阅 。计划的扩展中的Application Auto Scaling 用户指南。

如果您有企业 Support 计划，请务必与您的技术客户经理 (TAM) 合作。您的 TAM 可以验证您的服务配额，并确保在事件开始之前提高任何必要的配额。这样，您就不会意外达到任何服务配额。他们还可以通过预热服务（如负载均衡器）为您提供帮助，以确保您的活动顺利进行。

处理计划外的需求冲击是一个更为困难的问题。计划外冲击如果幅度足够大，可能会迅速导致需求超出容量。它还可以超过自动缩放反应的能力。为应对计划外冲击做好准备的最佳方式是过度提供资源。您必须有足够的资源来随时处理最大的预期流量需求。

在预期意外需求冲击的情况下保持最大容量可能是成本高昂的。要减轻成本影响，请查找预测即将发生大量需求冲击的主要指标或事件。如果度量或事件可靠地提供了重要的提前通知，请在事件发生时或度量超过您设置的特定阈值时立即开始向外扩展过程。

如果您的应用程序容易受到突然意外需求冲击，请考虑在应用程序中添加一种高性能模式，该模式会牺牲非关键功能，但为客户保留关键功能。例如，假定您的应用程序可以从生成昂贵的自定义响应切换到提供静态响应页。在这种情况下，您可以在根本不扩展应用程序的情况下显著提高吞吐量。

最后，您可以考虑拆分整体服务，以更好地应对需求冲击。如果您的应用程序是运行成本高昂且扩展速度较慢的单片服务，则您可能能够提取或重写性能关键部分，并将它们作为单独的服务运行。然后，这些新服务可以独立于不太关键的组件扩展。灵活地将性能关键型功能与应用程序的其他部分分开扩展，既可减少添加容量所需的时间，又有助于节省成本。

集群容量

在本主题前面，我们讨论了如何使用扩展指标向外扩展您的副本帐户。您的任务还需要在资源上运行，包括 CPU 和内存资源。这再次涉及到能力问题。在 Amazon ECS 中，容量通过两个主要提供商提供：AWS Fargate和 Amazon EC2。

您可以通过多种方式为 Amazon ECS 集群提供容量。例如，您可以使用 Amazon ECS 容器代理启动 Amazon EC2 实例，并在启动时将其注册到集群。但是，此方法可能具有挑战性，因为您需要自行管理扩展。因此，我们建议您使用 Amazon ECS 容量提供商。他们为您管理资源扩展。容量提供商有三种：Amazon EC2、Fargate 和 Fargate 现货。

Fargate 和 Fargate 现货容量提供商为您处理法门任务的生命周期。Fargate 提供按需容量，而 Fargate 竞价提供竞价容量。启动任务后，弹性云服务器会为您预配一个 Fargate 资源。此 Fargate 资源随附的内存和 CPU 单元直接对应于您在任务定义中声明的任务级限制。每个任务都接收自己的 Fargate 资源，使任务和计算资源之间的关系具有 1:1 的关系。

在 Fargate 点上运行的任务可能会中断。中断是在两分钟的警告之后发生的。这些情况发生在需求量大的时期。Fargate 竞价最适合容忍中断的工作负载，例如批处理作业、开发或临时环境。它们也适用于任何其他不需要高可用性和低延迟的场景。

您可以与 Fargate 按需任务一起运行远门点击任务。通过将它们结合使用，您可以以较低的成本获得预配置 “突发” 容量。

云服务器还可以为您的任务管理 Amazon EC2 实例容量。每个 Amazon EC2 容量提供商都与您指定的 Amazon EC2 Auto Scaling 组相关联。当您使用 Amazon EC2 容量提供程序时，ECS 集群 Auto Scaling 将保持 Amazon EC2 自动扩展组的大小，以确保可以放置所有计划任务。

集群容量最佳实践

为您的服务增加空间，而不是容量提供商。 Amazon EC2 容量提供商提供了目标容量值。如果您将该值设置为低于 100%，则 ECS 预配置的 Amazon EC2 实例数量超过了满足您的任务所需的数量。让多个 Amazon EC2 实例准备好接受任务可能会很有帮助。但是，当您使用 Amazon Virtual Private Cloud 时，启动新任务需要额外的时间才能下载映像并连接网络接口。这种增加的延迟可能会损害您的利润。

因此，我们建议您执行以下操作。通过修改服务的目标跟踪扩展衡量指标或步长扩展阈值来增加服务中的副本数量，而不是减少容量提供程序的目标容量。有关相关扩展策略的更多信息，请参阅。目标跟踪扩展策略或者步进扩展策略中的Amazon Elastic Container Service 开发指南。Amazon EC2 容量提供商通过向 Auto Scaling 组添加更多实例来配置额外任务所需的容量。这有助于确保计算资源和应用程序资源在您需要时都可用。例如，弹性云服务中的任务数增加一倍，以满足即时的 100% 突发需求，从而提供帮助。

选择 Fargate 任务大小

如果您在AWS Fargate，您必须在任务定义中声明任务 CPU 和内存限制。弹性云服务器使用这些限制来确定要在其上运行任务的 Fargate 实例类型。您确定的限制必须大于或等于您声明的任何预留。在大多数情况下，您可以将它们设置为任务定义中声明的每个容器的预留总和。然后，还将数字四舍五入到最近的 Fargate 实例大小。有关可用大小的更多信息，请参阅。任务 CPU 和内存中的Amazon Elastic Container Service 开发指南。

选择 Amazon EC2 实例类型

如果您使用 Amazon EC2 为您的 ECS 集群提供容量，您可以从大量实例类型中进行选择。所有 Amazon EC2 实例类型和系列都与弹性云服务器兼容。

要确定您可以使用哪些实例类型，请首先消除不符合应用程序特定要求的实例类型或实例系列。例如，如果您的应用程序需要 GPU，则可以排除没有 GPU 的任何实例类型。但是，您也应该考虑其他要求。例如，考虑 CPU 体系结构、网络吞吐量以及是否需要实例存储。接下来，检查每种实例类型提供的 CPU 和内存量。作为一般规则，CPU 和内存必须足够大，以容纳要运行的任务的至少一个副本。

您可以从与应用程序兼容的实例类型中进行选择。使用较大的实例，您可以同时启动更多任务。而且，对于较小的实例，您可以以更精细的方式进行横向扩展，以节省成本。您无需选择适合集群中所有应用程序的单个 Amazon EC2 实例类型。相反，您可以创建多个 Auto Scaling 组。每个组可具有不同的实例类型。然后，您可以为这些组中的每个组创建 Amazon EC2 容量提供程序。最后，在服务和任务的容量提供程序策略中，您可以选择最适合其需求的容量提供程序。

使用 Amazon EC2 Spot 和远门 _SPOT

与按需实例相比，竞价型容量可以显著节约成本。竞价容量是过剩容量，其价格大大低于按需容量或预留容量。竞价容量适用于批处理和机器学习工作负载以及开发和暂存环境。更一般而言，它适用于容忍临时停机的任何工作负载。

了解以下后果，因为竞价容量可能并非始终可用。

- 首先，在需求极高的时期，竞价容量可能不可用。这可能会导致 Fargate 竞价任务和 Amazon EC2 竞价型实例启动延迟。在这些事件中，ECS 服务会重试启动任务，Amazon EC2 Auto Scaling 组也会重试启动实例，直到所需容量变为可用。Fargate 和 Amazon EC2 不会用按需容量取代竞价容量。
- 其次，当容量总体需求增加时，竞价型实例和任务可能会终止，但只有两分钟的警告。发送警告后，在实例完全终止之前，如有必要，任务应开始有序关闭。这有助于最大限度地减少错误的可能性。有关正常关机的更多信息，请参阅。使用弹性云服务器实现优雅关机。

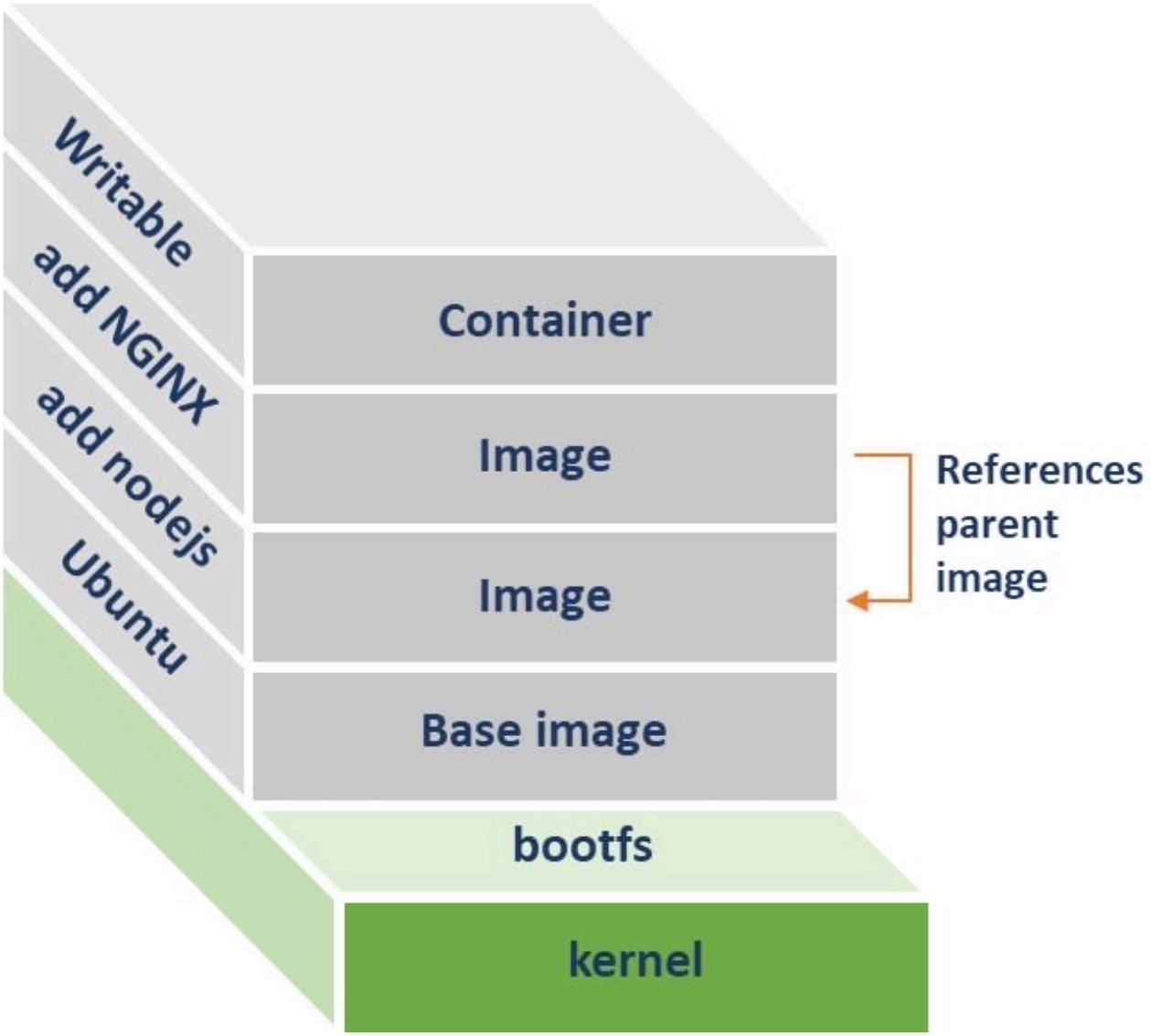
要帮助最大限度地减少竞价容量短缺，请考虑以下建议：

- 使用多个区域和可用区。 竞价容量因地区和可用区而异。您可以通过在多个区域和可用区域运行工作负载来提高竞价可用性。如果可能，请在运行任务和实例的区域中的所有可用区域中指定子网。
- 使用多个 Amazon EC2 实例类型。 当您将混合实例策略与 Amazon EC2 Auto Scaling 结合使用时，会在您的 Auto Scaling 组中启动多种实例类型。这可以确保在需要时满足竞价容量的请求。为了最大限度地提高可靠性并最大限度地降低复杂性，请在混合实例策略中使用具有大致相同 CPU 和内存量的实例类型。这些实例可以来自不同层代，也可以是相同基本实例类型的变体。请注意，它们可能附带了您可能不需要的其他功能。此类列表的一个示例可以包括 m4 大型、m5 大型、m5 大型、大型、m5 分级大型、M5 分级大型、M5 分级大型和大型。有关更多信息，请参阅 。Auto Scaling 组，具有多个实例类型和购买选项中的Amazon EC2 Auto Scaling 用户指南。
- 使用容量优化的竞价分配策略。 借助 Amazon EC2 竞价，您可以在容量和成本优化的分配策略之间进行选择。如果您在启动新实例时选择容量优化策略，Amazon EC2 竞价将选择在选定可用区中具有最高可用性的实例类型。这有助于减少实例在启动后立即终止的可能性。

最佳做法-持久性存储

您可以使用 Amazon ECS 大规模运行有状态的容器化应用程序，方法是使用AWS存储服务（例如亚马逊 EFS、亚马逊 EBS 或适用于 Windows 文件服务器的 Amazon FSX），可为本身的短暂容器提供数据持久性。术语数据持久性意味着数据本身超出了创建它的过程。中的数据持久性AWS是通过耦合计算和存储服务来实现的。与 Amazon EC2 类似，您还可以使用 Amazon ECS 将容器化应用程序的生命周期与它们使用和生成的数据分离。使用AWS存储服务，即使在任务终止后，Amazon ECS 任务也可以保留数据。

默认情况下，容器不会保留它们生成的数据。当容器终止时，它写入到其可写层的数据将随容器一起销毁。这使得容器适合不需要在本地存储数据的无状态应用程序。需要数据持久性的容器化应用程序需要一个存储后端，该存储后端在应用程序的容器终止时不会被销毁。



容器图像是基于一系列图层构建的。每个图层代表 Docker 文件中的一条指令，该指令是从中构建映像。每个图层都是只读的，但容器除外。也就是说，当您创建容器时，将在基础图层上添加一个可写图层。容器创建、删除或修改的任何文件都会写入可写层。当容器终止时，也会同时删除可写层。使用相同图像的新容器具有自己的可写层。此图层不包括任何更改。因此，容器的数据必须始终存储在容器可写层之外。

借助 Amazon ECS，您可以使用卷运行有状态的容器。亚马逊云服务器与亚马逊 EFS 本机集成，并使用与 Amazon EBS 集成的卷。对于 Windows 容器，亚马逊弹性云服务器与适用于 Windows 文件服务器的亚马逊 FSX 集成，以提供持久存储。

主题

选择适合容器的存储类型

在 Amazon 云服务器集群中运行的应用程序可以使用各种AWS存储服务和第三方产品，为有状态工作负载提供持久存储。您应该根据应用程序的体系结构和存储要求为容器化应用程序选择存储后端。有关的更多信息AWS存储服务，请参阅上的云存储AWS。

对于包含 Linux 实例或与 Fargate 一起使用的亚马逊云服务器集群，亚马逊云服务器与亚马逊 EFS 和亚马逊 EBS 集成以提供容器存储。Amazon EFS 和 Amazon EBS 之间最明显的区别在于，您可以在数千个亚马逊云服务器任务上同时挂载 Amazon EFS 文件系统。相比之下，Amazon EBS 卷不支持并发访问。鉴于此，Amazon EFS 是水平扩展的容器化应用程序的推荐存储选项。这是因为它支持并发。Amazon EFS 跨多个可用区冗余存储您的数据，并提供来自 Amazon ECS 任务的低延迟访问（无论哪个可用区域）的低延迟访问。Amazon EFS 支持同时在 Amazon EC2 和 Fargate 上运行的任务。

假设您有一个应用程序（如事务数据库），该应用程序需要亚毫秒级延迟，并且在水平缩放时不需要共享文件系统。对于此类应用程序，我们建议使用 Amazon EBS 卷进行永久性存储。目前，亚马逊 ECS 仅支持 Amazon EBS 卷用于托管在 Amazon EC2 上的任务。对 Amazon EBS 卷的 Support 不适用于 Fargate 上的任务。在将 Amazon EBS 卷与 Amazon ECS 任务结合使用之前，您必须首先将 Amazon EBS 卷附加到容器实例，然后在任务的生命周期内单独管理卷。

对于包含 Windows 实例的集群，适用于 Windows 文件服务器的 Amazon FSX 为容器提供持久存储。Amazon FSx for Windows File Server 系统支持多可用区部署。通过这些部署，您可以与跨多个可用区域运行的 Amazon ECS 任务共享文件系统。

您还可以将 Amazon EC2 实例存储用于使用绑定挂载或 Docker 卷托管在 Amazon EC2 上的 Amazon ECS 任务的数据持久性。使用绑定装载或 Docker 卷时，容器将数据存储在容器实例文件系统上。将主机文件系统用于容器存储的一个限制是，数据一次只能在单个容器实例上提供。这意味着容器只能在数据所在的主机上运行。因此，只建议在应用程序级别处理数据复制的情况下使用主机存储。

Amazon EFS 卷

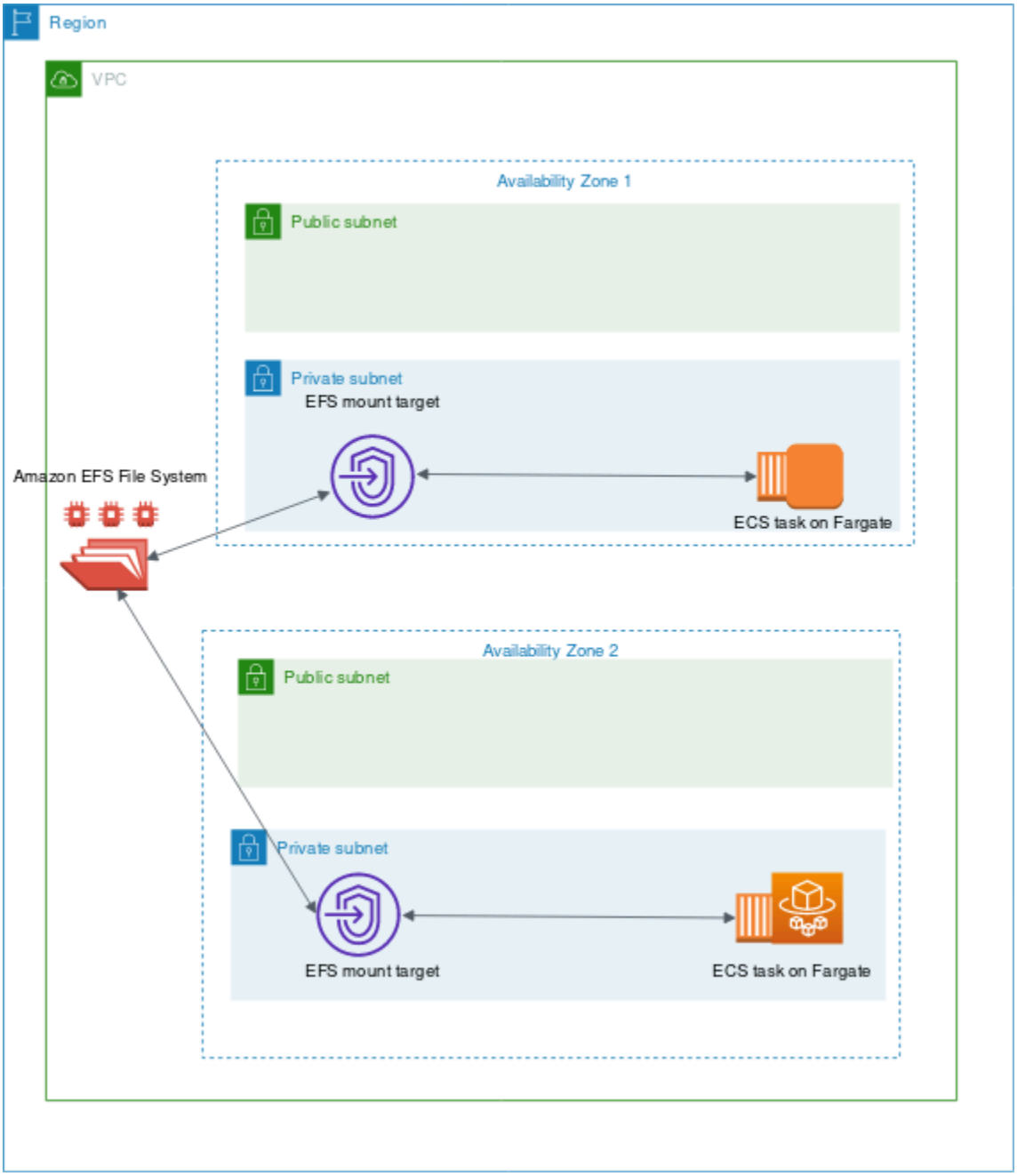
Amazon Elastic File System (Amazon EFS) 提供简单、可扩展、完全托管的弹性 NFS 文件系统。它的构建目的是能够按需扩展到 PB 级，而不会中断应用程序。它可以在添加和删除文件时进行扩展或缩小。

您可以使用 Amazon EFS 卷提供持久存储，在 Amazon ECS 中运行有状态应用程序。使用平台版本在 Amazon EC2 实例或 Fargate 上运行的 Amazon 云服务器任务`1.4.0`，以及更高版本可以挂载现有 Amazon EFS 文件系统。鉴于多个容器可以同时装载和访问 Amazon EFS 文件系统，因此，无论它们托管在何处，您的任务都可以访问相同的数据集。

要在容器中挂载 Amazon EFS 文件系统，您可以在 Amazon ECS 任务定义中引用 Amazon ECS 任务定义中的 Amazon EFS 文件系统和容器挂载点。以下是任务定义的片段，其中使用 Amazon EFS 进行容器存储。

```
...
"containerDefinitions":[
    {
        "mountPoints": [
            {
                "containerPath": "/opt/my-app",
                 "sourceVolume": "Shared-EFS-Volume"
            }
    }
  ]
...
"volumes": [
    {
      "efsVolumeConfiguration": {
        "fileSystemId": "fs-1234",
        "transitEncryption": "DISABLED",
        "rootDirectory": ""
      },
      "name": "Shared-EFS-Volume"
    }
  ]
```

Amazon EFS 在单个区域内多个可用区间冗余地存储数据。Amazon 云服务器任务通过使用 Amazon EFS 装载目标在其可用区域中安装 Amazon EFS 文件系统。只有当 Amazon EFS 文件系统在运行任务的可用区中具有装载目标时，Amazon ECS 任务才能装载 Amazon EFS 文件系统。因此，最佳做法是在您计划托管 Amazon ECS 任务的所有可用区域中创建 Amazon EFS 装载目标。



有关更多信息，请参阅 。Amazon EFS 卷中的Amazon Elastic Container Service 开发者指南。

安全和访问控制

Amazon EFS 提供访问控制功能，您可以使用这些功能来确保存储在 Amazon EFS 文件系统中的数据是安全的，并且只能从需要它的应用程序访问。您可以通过启用静态和传输中的加密来保护数据。有关更多信息，请参阅 。Amazon EFS 中的数据加密中的Amazon Elastic File System 用户指南。

除了数据加密之外，您还可以使用 Amazon EFS 来限制对文件系统的访问。可以通过三种方式在 EFS 中实现访问控制。

- 安全组— 使用 Amazon EFS 装载目标，您可以配置用于允许和拒绝网络流量的安全组。您可以将附加到 Amazon EFS 的安全组配置为允许来自附加到您的 Amazon ECS 实例的安全组的 NFS 流量（端口 2049），或者在使用awsvpc网络模式下执行亚马逊云服务器任务。
- IAM您可以使用 IAM 限制对 Amazon EFS 文件系统的访问。配置后，Amazon 弹性云服务器任务需要 IAM 角色才能安装 EFS 文件系统。有关更多信息，请参阅 。使用 IAM 控制文件系统数据访问中的Amazon Elastic File System 用户指南。

  IAM 策略还可以强制执行预定义的条件，例如要求客户端在连接到 Amazon EFS 文件系统时使用 TLS。有关更多信息，请参阅 。针对客户的亚马逊 EFS 条件密钥中的Amazon Elastic File System 用户指南。
- Amazon EFS 接入点— Amazon EFS 访问点是 Amazon EFS 文件系统中特定于应用程序的入口点。您可以使用访问点为通过访问点发出的所有文件系统请求强制执行用户身份（包括用户的 POSIX 组）。访问点还可以为文件系统强制执行不同的根目录。这样，客户端只能访问指定目录或其子目录中的数据。

考虑在 Amazon EFS 文件系统上实施所有三种访问控制，以获得最大的安全性。例如，您可以将附加到 Amazon EFS 装载点的安全组配置为仅允许来自与您的容器实例或 Amazon ECS 任务关联的安全组的入口 NFS 流量。此外，您还可以将 Amazon EFS 配置为要求 IAM 角色来访问文件系统，即使连接源自允许的安全组。最后，您可以使用 Amazon EFS 访问点强制执行 POSIX 用户权限并为应用程序指定根目录。

以下任务定义代码段演示如何使用访问点挂载 Amazon EFS 文件系统。

```
"volumes": [
    {
      "efsVolumeConfiguration": {
        "fileSystemId": "fs-1234",
        "authorizationConfig": {
          "acessPointId": "fsap-1234",
          "iam": "ENABLED"
        },
        "transitEncryption": "ENABLED",
        "rootDirectory": ""
      },
      "name": "my-filesystem"
    }
```

```
]
```

Performance

亚马逊 EFS 提供两种性能模式：通用和最大 I/O。常规用途适用于延迟敏感的应用程序，如内容管理系统和 CI/CD 工具。相比之下，Max I/O 文件系统适用于数据分析、媒体处理和机器学习等工作负载。这些工作负载需要从数百甚至数千个容器执行并行操作，并需要尽可能高的聚合吞吐量和 IOPS。有关更多信息，请参阅 。Amazon EFS 性能模式中的Amazon Elastic File System 用户指南。

某些对延迟敏感的工作负载既需要由最大 I/O 性能模式提供更高的 I/O 级别，也需要由通用性能模式提供的更低延迟。对于此类工作负载，我们建议创建多个通用性能模式文件系统。这样，只要工作负载和应用程序可以支持它即可，即可将应用程序工作负载分散到所有这些文件系统间。

Throughput

所有 Amazon EFS 文件系统都有一个关联的计量吞吐量，该吞吐量取决于预配置吞吐量或存储在 EFS 标准或 One 区域存储类中的数据量，以使用突增吞吐量。有关更多信息，请参阅 。了解计量吞吐量中的Amazon Elastic File System 用户指南。

Amazon EFS 文件系统的默认吞吐量模式为拆分模式。在突发模式下，文件系统可用的吞吐量会随着文件系统的增长而扩展或缩小。因为基于文件的工作负载通常会猛增，其余时间吞吐量较低，因此 Amazon EFS 被设计为突增，以允许在一段时间内实现高吞吐量。此外，由于许多工作负载都是读取繁重的，因此读取操作的计量与其他 NFS 操作（如写入）的比率为 1:3。

对于每 TB 的 Amazon EFS 标准存储或 Amazon EFS 单区存储，所有 Amazon EFS 文件系统都可提供 50 MB/s 的一致基准性能。所有文件系统（无论大小如何）都可以突发到 100 MB/s。具有超过 1TB 的 EFS 标准或 EFS 单区域存储的文件系统每 TB 可以突发为 100 MB/s。由于读取操作以 1:3 的比率计量，因此每个 TiB 的读取吞吐量可以驱动多达 300 MIBS/s。向文件系统添加数据时，文件系统可用的最大吞吐量会随 Amazon EFS Standard 存储类中的存储而自动进行线性扩展。如果您需要的吞吐量超过存储数据量所能达到的吞吐量，则可以将预配置吞吐量配置为工作负载所需的特定量。

文件系统吞吐量在连接到文件系统的所有 Amazon EC2 实例之间共享。例如，吞吐量突增到 100 MB/s 的 1TB 文件系统可以从单个 Amazon EC2 实例驱动 100 MB/s，每个驱动器可以驱动 10 MB/s。有关更多信息，请参阅 。Amazon EFS 绩效中的Amazon Elastic File System 用户指南。

成本优化

Amazon EFS 为您简化了扩展存储。随着您添加更多数据，Amazon EFS 文件系统会自动增长。尤其是 Amazon EFS突增吞吐量模式下，Amazon EFS 上的吞吐量会随着标准存储类别中文件系统大小的

增大而增加。要提高吞吐量，而无需为 EFS 文件系统上的预配置吞吐量支付额外费用，您可以与多个应用程序共享 Amazon EFS 文件系统。使用 Amazon EFS 接入点，您可以在共享的 Amazon EFS 文件系统中实现存储隔离。通过这样做，即使应用程序仍然共享同一个文件系统，除非您授权，否则它们无法访问数据。

随着数据的增长，Amazon EFS 可帮助您自动将不经常访问的文件移动到较低的存储类别。Amazon EFS 标准不经常访问 (IA) 存储类别降低了不经常访问的文件的存储成本。它可以在不牺牲 Amazon EFS 提供的高可用性、高持久性、弹性和 POSIX 文件系统访问。有关更多信息，请参阅 。Amazon EFS 存储类中的Amazon Elastic File System 用户指南。

考虑使用 Amazon EFS 生命周期策略通过将不经常访问的文件移动到 Amazon EFS IA 存储来自动节省资金。有关更多信息，请参阅 。Amazon EFS 生命周期管理中的Amazon Elastic File System 用户指南。

创建 Amazon EFS 文件系统时，您可以选择 Amazon EFS 跨多个可用区（标准）复制您的数据，还是将数据冗余存储在单个可用区中。与 Amazon EFS 标准存储类相比，Amazon EFS One 区域存储类别可以大幅降低存储成本。考虑对不需要多可用区恢复的工作负载使用 Amazon EFS One Zone 存储类。您可以通过将不经常访问的文件移动到 Amazon EFS One 区域不经常访问，进一步降低 Amazon EFS One 区域存储的成本。有关更多信息，请参阅 。Amazon EFS 不经常访问。

数据保护

Amazon EFS 使用标准存储类别跨文件系统的多个可用区冗余存储您的数据。如果您选择 Amazon EFS One Zone 存储类，则您的数据将冗余存储在单个可用区内。此外，Amazon EFS 被设计为可在一年内提供 99.9999999%（11 9）的持久性。

与任何环境一样，最佳做法是进行备份并构建防止意外删除的保护措施。对于 Amazon EFS 数据，最佳实践包括使用AWS Backup。默认情况下，使用 Amazon EFS One Zone 存储类的文件系统配置为在文件系统创建时自动备份文件，除非您选择禁用此功能。有关更多信息，请参阅 。Amazon EFS 的数据保护中的Amazon Elastic File System 用户指南。

使用案例

Amazon EFS 提供并行共享访问，随着文件的添加和删除，它会自动增长和收缩。因此，Amazon EFS 适用于需要具有低延迟、高吞吐量和写后读取一致性等功能的存储的任何应用程序。Amazon EFS 是水平扩展并需要共享文件系统的应用程序的理想存储后端。Amazon EFS 常见的 Amazon EFS 使用案例中，例如数据分析、媒体处理、内容管理和 Web 服务。

Amazon EFS 可能不适合的一个使用案例是需要亚毫秒级延迟的应用程序。这通常是事务性数据库系统的要求。我们建议运行存储性能测试，以确定将 Amazon EFS 用于延迟敏感应用程序的影响。如

果在使用 Amazon EFS 时应用程序性能下降，请考虑 Amazon EBS io2 块 Express，它可以在硝基实例上提供亚毫秒级、低方差的 I/O 延迟。有关更多信息，请参阅 。Amazon EBS 卷类型中的适用于 Linux 实例的 Amazon EC2 用户指南。

如果某些应用程序的底层存储意外更改，则会失败。因此，Amazon EFS 不是这些应用程序的最佳选择。相反，您可能更愿意使用不允许从多个位置并发访问的存储系统。

Docker 卷

Docker 卷是 Docker 容器运行时的一项功能，允许容器通过从主机的文件系统挂载目录来保存数据。Docker 卷驱动程序（也称为插件）用于将容器卷与外部存储系统（例如 Amazon EBS）集成。只有在 Amazon EC2 实例上托管 Amazon ECS 任务时，才支持 Docker 卷。

Amazon 弹性云服务器任务可以使用 Docker 卷保存使用 Amazon EBS 卷保存数据。这可以通过将 Amazon EBS 卷附加到 Amazon EC2 实例，然后使用 Docker 卷在任务中装载该卷来完成。Docker 卷可以在主机上的多个 Amazon 云服务器任务之间共享。

Docker 卷的限制在于，任务使用的文件系统与特定 Amazon EC2 实例相关。如果实例由于任何原因停止，并且任务被放置在另一个实例上，则数据将丢失。您可以将任务分配给实例，以确保关联的 EBS 卷始终可用于任务。

有关更多信息，请参阅 。Docker 卷中的Amazon Elastic Container Service 开发者指南。

Amazon EBS 卷生命周期

容器存储和 Amazon EBS 有两种关键的使用模式。首先，当应用程序需要保留数据并在其容器终止时防止数据丢失时。这种类型的应用程序的一个例子是像 MySQL 这样的事务数据库。当 MySQL 任务终止时，需要另一个任务替换它。在这种情况下，卷的生命周期与任务的生命周期分开。使用 EBS 保留容器数据时，最佳做法是使用任务放置约束限制将任务放置在连接 EBS 卷的单个主机上。

第二种情况是卷的生命周期独立于任务生命周期。这对于需要高性能和低延迟存储但在任务终止后不需要保留数据的应用程序尤其有用。例如，处理大量数据的 ETL 工作负载可能需要高吞吐量存储。Amazon EBS 适用于此类工作负载，因为它提供了高性能卷，可预配高达 256,000 IOPS。任务终止后，可以将替换副本安全放置在集群中的任何 Amazon EC2 主机上。只要任务能够访问能够满足其性能要求的存储后端，任务就可以执行其功能。因此，在这种情况下，不需要任务放置约束。

如果集群中的 Amazon EC2 实例具有多种类型的 Amazon EBS 卷，您可以使用任务放置约束来确保任务放置在附加了适当 Amazon EBS 卷的实例上。例如，假设集群有一些具有gp2卷，而另一些则使用io1卷。您可以将自定义属性附加到io1卷，然后使用任务放置约束来确保 I/O 密集型任务始终放置在具有io1卷。

以下AWS CLI命令用于在 Amazon ECS 容器实例上放置属性。

```
aws ecs put-attributes \
     --attributes name=EBS,value=io1,targetId=<your-container-instance-arn>
```

Amazon EBS 数据可用性

容器通常寿命短，经常创建，并随着应用程序水平扩展和向外扩展而终止。作为最佳做法，您可以在多个可用区中运行工作负载，以提高应用程序的可用性。Amazon ECS 为您提供了一种使用任务放置策略和任务放置约束来控制任务放置的方法。当工作负载使用 Amazon EBS 卷保留其数据时，其任务需要放置在与 Amazon EBS 卷相同的可用区中。我们还建议您设置放置约束，以限制任务可以放置的可用区。这可确保您的任务及其对应的卷始终位于同一可用区中。

运行独立任务时，您可以通过使用可用区属性设置放置约束来控制任务放置哪个可用区。

```
attribute:ecs.availability-zone == us-east-1a
```

在运行在多个可用区中运行的应用程序时，请考虑为每个可用区域创建不同的 Amazon ECS 服务。这可确保需要 Amazon EBS 卷的任务始终与关联卷位于同一可用区中。

我们建议在每个可用区中创建容器实例，使用启动模板，并添加自定义属性添加到实例，以区分它们与 Amazon ECS 集群中的其他容器实例。创建服务时，配置任务放置约束，以确保 Amazon ECS 将任务放置在正确的可用区和实例中。有关更多信息，请参阅 。任务放置约束示例中的Amazon Elastic Container Service 开发者指南。

Docker 卷插件

Docker 插件（如 Portworx）提供了 Docker 卷和 Amazon EBS 卷之间的抽象。当您需要卷的任务启动时，这些插件可以动态创建 Amazon EBS 卷。当容器终止时，Portworx 还可以将卷附加到新主机，并将其后续副本放置在不同的容器实例上。它还可以在 Amazon ECS 节点之间和跨可用区域复制每个容器的卷数据。有关更多信息，请参阅 。波特沃克斯。

Amazon FSx for Windows File Server 的

适用于 Windows 文件服务器的 Amazon FSX 提供了完全托管、高度可靠且可扩展的文件存储，可通过行业标准的服务器消息块 (SMB) 协议访问。它基于 Windows Server 构建，提供了广泛的管理功能，如用户配额、最终用户文件恢复和 Microsoft 活动目录 (AD) 集成。它提供了单可用区和多可用区部署选项、完全托管的备份以及静态和传输中数据的加密。

亚马逊云服务器支持在亚马逊云服务器 Windows 任务定义中使用 Windows 文件服务器的 Amazon FSX，使用称为全球映射的 SMB 功能，通过 SMBv3 协议将永久存储作为装载点。

要设置适用于 Windows 文件服务器和亚马逊云服务器集成的 Amazon FSX，Windows 容器实例必须是活动目录域服务 (AD DS) 上的域成员，该服务由AWS Directory Service for Microsoft Active Directory、本地活动目录或 Amazon EC2 上的自托管活动目录。AWS Secrets Manager用于存储敏感数据，如 Active Directory 凭据的用户名和密码，该凭据用于映射 Windows 容器实例上的共享。

要为容器使用 Amazon FSx to Windows File Server 文件卷，您必须在任务定义中指定卷和挂载点配置。以下是任务定义的片段，其中使用 Amazon FSx for Windows File Server 进行容器存储。

```
{
 "containerDefinitions": [{
  "name": "container-using-fsx",
  "image": "iis:2",
  "entryPoint": [
   "powershell",
   "-command"
  ],
  "mountPoints": [{
   "sourceVolume": "myFsxVolume",
   "containerPath": "\\mount\\fsx",
   "readOnly": false
  }]
 }],
 "volumes": [{
  "fsxWindowsFileServerVolumeConfiguration": {
   "fileSystemId": "fs-ID",
   "authorizationConfig": {
    "domain": "ADDOMAIN.local",
    "credentialsParameter": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:SecretName"
   },
   "rootDirectory": "share"
  }
 }]
}
```

有关更多信息，请参阅 。适用于 Windows File Server 卷的 Amazon FSx 卷中的Amazon Elastic Container Service 开发者指南。

安全和访问控制

Amazon FSx for Windows File Server 系统中存储的 Amazon FSx to Windows File Server 文件系统中的数据是安全的，只有从需要它的应用程序访问。

数据加密

Amazon FSx for Windows File Server 支持两种形式的文件系统加密。它们是传输中的数据加密和静态加密。在支持 SMB 协议 3.0 或更高版本的容器实例上映射的文件共享上，支持对传输中的数据进行加密。在创建 Amazon FSx 文件系统时，将自动启用静态数据加密。当您访问文件系统时，Amazon FSX 使用 SMB 加密自动加密传输中的数据，而无需修改应用程序。有关更多信息，请参阅 。Amazon FSx 中的数据加密中的适用于 Windows File Server 的 Amazon FSx 用户指南。

使用 Windows ACL 进行文件夹级别访问控制

Windows Amazon EC2 实例使用活动目录证书访问亚马逊 FSX 文件共享。它使用标准 Windows 访问控制列表 (ACL) 进行精细的文件和文件夹级别访问控制。您可以创建多个凭据，每个凭据用于映射到特定任务的共享中的特定文件夹。

在以下示例中，任务可以访问文件夹App01使用保存在 Secrets Manager 中的凭据。Amazon 资源名称 (ARN) 是1234。

```
"rootDirectory": "\\path\\to\\my\\data\App01",
"credentialsParameter": "arn-1234",
"domain": "corp.fullyqualified.com",
```

在另一个示例中，任务可以访问文件夹App02使用保存在 Secrets Manager 中的凭据。其 ARN 为 6789。

```
"rootDirectory": "\\path\\to\\my\\data\App02",
"credentialsParameter": "arn-6789",
"domain": "corp.fullyqualified.com",
```

使用案例

容器不是为了保留数据而设计的。但是，某些容器化的 .NET 应用程序可能需要本地文件夹作为永久存储来保存应用程序输出。Amazon FSx for Windows File Server 在容器中提供本地文件夹。这允许多个容器在由 SMB 共享支持的同一文件系统上读写。

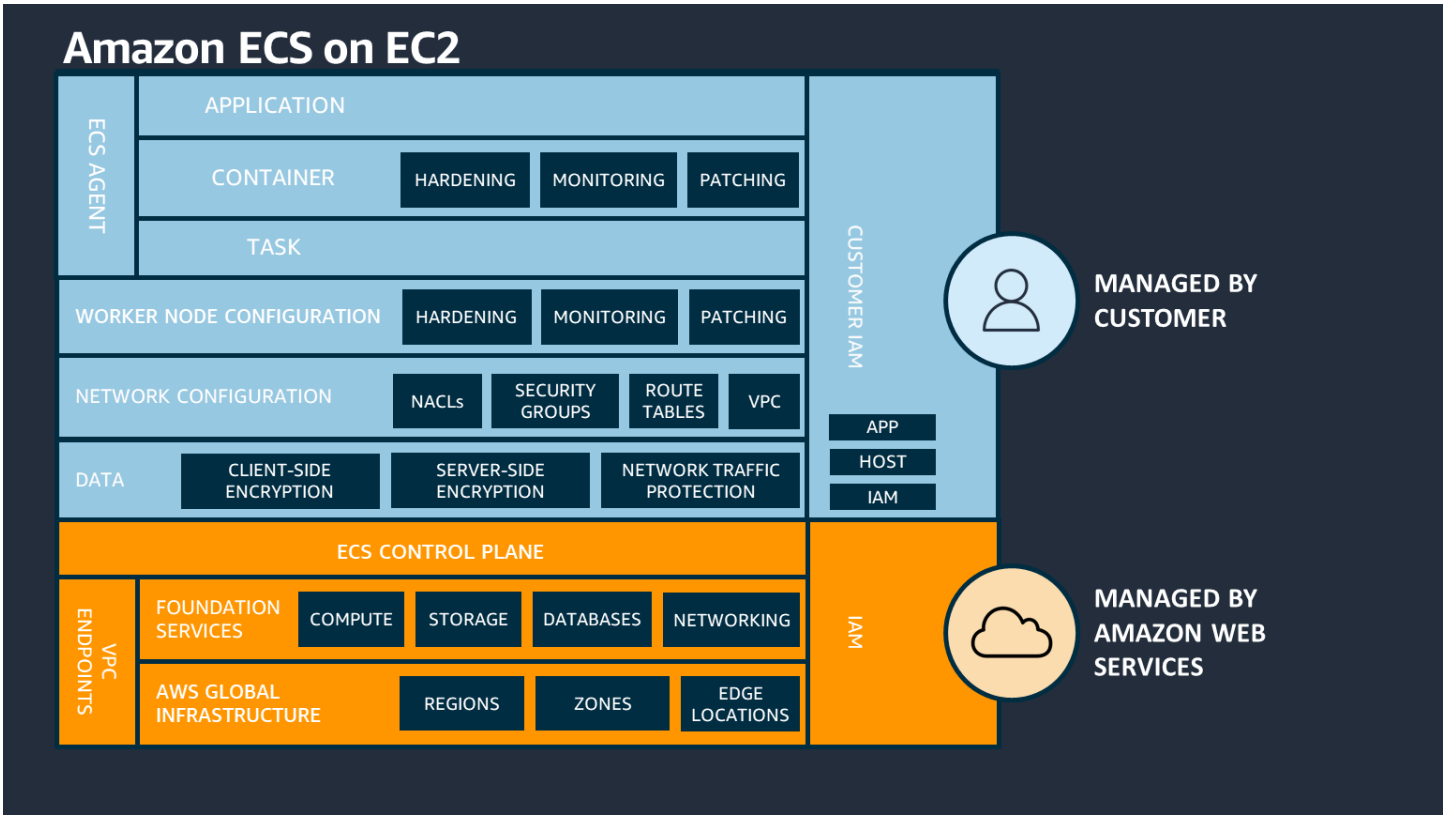
最佳实践 — 安全

本指南提供安全性和合规性建议，以保护您的信息、系统和依赖 Amazon ECS 的其他资产。它还介绍了一些风险评估和缓解策略，您可以使用这些策略更好地掌握针对 Amazon ECS 群集及其支持的工作负载构建的安全控制。本指南中的每个主题都从简要概述开始，然后是可用于保护 Amazon ECS 集群的建议和最佳实践列表。

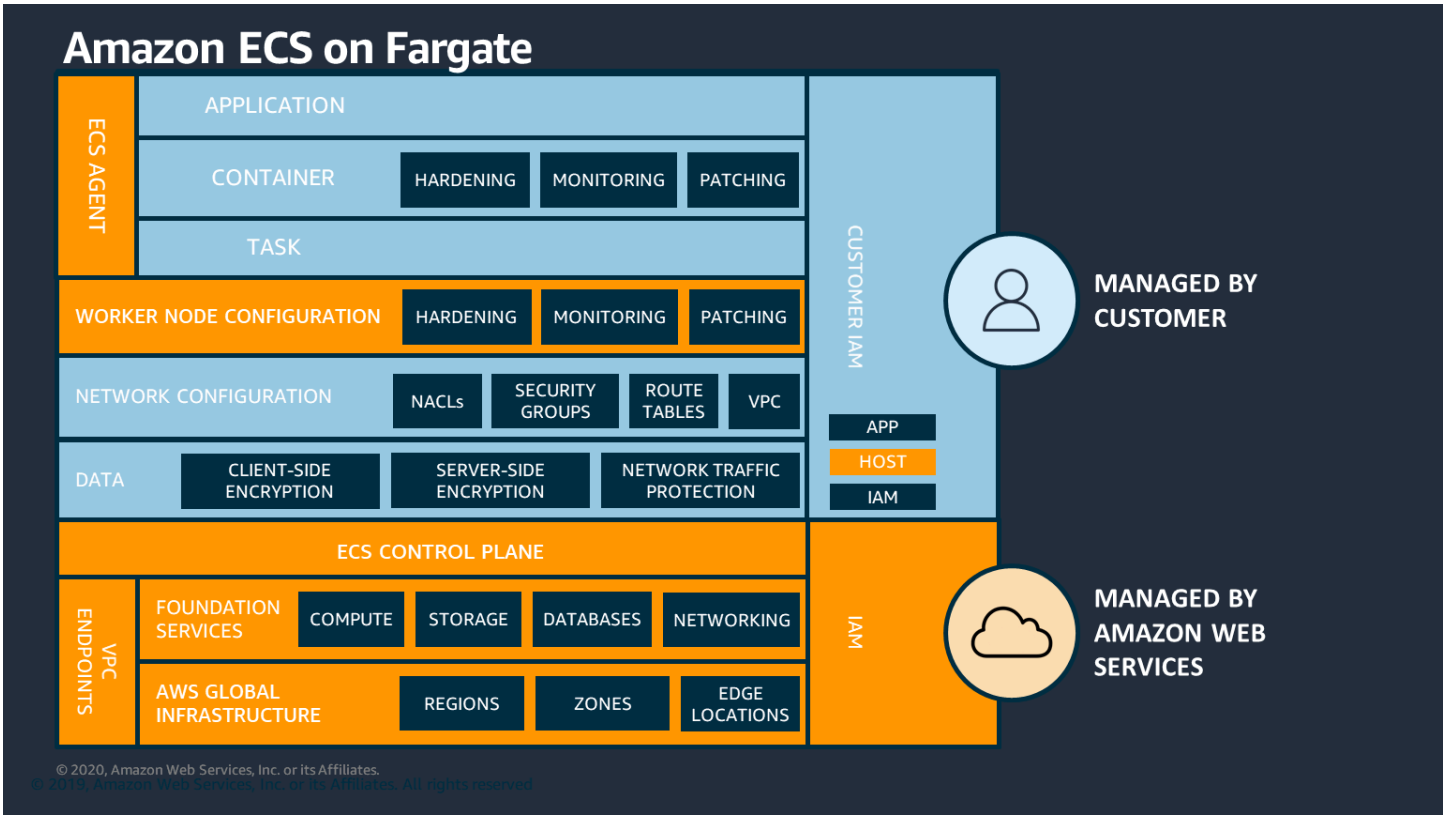
主题

责任共担模式

像 Amazon ECS 这样的托管服务的安全性和合规性是您和AWS。一般而言,AWS负责 “云” 的安全性，而您（客户）则负责 “云中的” 安全性。AWS负责管理 Amazon ECS 控制平面，包括提供安全可靠服务所需的基础设施。而且，您主要负责本指南中的主题。这包括数据、网络和运行时安全，以及日志记录和监控。



关于基础设施安全,AWS承担更多责任AWS Fargate资源与其他自我管理实例相比。随着 FargateAWS管理云中基础实例的安全性以及用于运行任务的运行时。Fargate 还可以代表您自动扩展您的基础架构。



在将服务扩展到云之前，您应该了解您负责哪些方面的安全性和合规性。

有关共担责任模型的更多信息，请参阅责任共担模式。

AWS Identity and Access Management

您可以使用AWS Identity and Access Management(IAM) 来管理和控制对AWS服务和资源，通过基于规则的策略进行身份验证和授权。更具体地说，通过此服务，您可以控制对AWS资源，使用应用于 IAM 用户、组或角色的策略。在这三个用户中，IAM 用户是可以访问您的资源的账户。而且，IAM 角色是一组可由经过身份验证的身份代入的权限，该身份与 IAM 外部的特定身份无关联。有关更多信息，请参阅 。IAM 中的策略和权限？。

管理对亚马逊云服务器的访问

您可以通过创建和应用 IAM 策略来控制对亚马逊云服务器的访问。这些策略由一组应用于特定资源的操作组成。策略的操作定义了允许或拒绝的操作列表（如 Amazon ECS API），而资源控制操作应用于哪些 Amazon ECS 对象。可以将条件添加到策略中以缩小其范围。例如，可以编写策略，以便仅允许针对具有特定标记集的任务执行操作。有关更多信息，请参阅 。Amazon ECS 如何与 IAM 协同工作中的Amazon Elastic Container Service 开发者指南。

Recommendations

建议您在设置 IAM 角色和策略时执行以下操作。

遵循最低特权访问策略

创建范围为允许用户执行其规定的作业的策略。例如，如果开发人员需要定期停止某个任务，请创建一个仅允许该特定操作的策略。以下示例仅允许用户停止属于特定`task_family`在具有特定 Amazon 资源名称 (ARN) 的集群上。在条件中引用 ARN 也是使用资源级权限的一个示例。您可以使用资源级权限指定要将操作应用到的资源。

> **Note**
>
> 在策略中引用 ARN 时，请使用新的较长 ARN 格式。有关更多信息，请参阅 。Amazon 资源名称 (ARN) 和 ID中的Amazon Elastic Container Service 开发者指南。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:StopTask"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"
        }
      },
      "Resource": [
        "arn:aws:ecs:<region>:<aws_account_id>:task-definition/<task_family>:*"
      ]
    }
  ]
}
```

让群集资源作为管理边界

范围过窄的策略可能会导致角色激增，并增加管理开销。创建作用域为群集的角色，并将群集用作主管理边界，而不是创建仅限于特定任务或服务的角色。

通过创建自动化管道，将最终用户与 Amazon ECS API 隔离开来

您可以通过创建自动打包和部署应用程序到 Amazon ECS 群集的管道来限制用户可以使用的操作。这有效地将创建、更新和删除任务的工作委托给管道。有关更多信息，请参阅 。教程：具有 CodePipeline 的亚马逊云服务器标准部署中的AWS CodePipeline用户指南。

使用策略条件来增强安全层

当您需要额外的安全层时，请将条件添加到您的策略中。如果您正在执行特权操作，或者您需要限制可针对特定资源执行的操作集，则此操作非常有用。以下示例策略在删除集群时需要多因素授权。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DeleteCluster"
      ],
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
        }
      },
    "Resource": ["*"]
    }
  ]
}
```

应用于服务的标签将传播到属于该服务一部分的所有任务。因此，您可以创建具有特定标签的 Amazon ECS 资源范围的角色。在以下策略中，IAM 委托人启动和停止标签键为Department和一个标签值Accounting。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:RunTask"
```

```
            ],
            "Resource": "arn:aws:ecs:*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Department": "Accounting"}
            }
        }
    ]
}
```

定期审核对亚马逊云服务器 API 的访问

用户可能会更改角色。更改角色后，先前授予他们的权限可能不再适用。请确保您审核谁有权访问 Amazon ECS API，以及该访问权限是否仍然保证。考虑将 IAM 与用户生命周期管理解决方案集成，该解决方案可在用户离开组织时自动撤销访问权限。有关更多信息，请参阅 。Amazon ECS 安全审核指南中的Amazon Web Services 一般参考。

将 IAM 角色与亚马逊云服务器任务结合使用

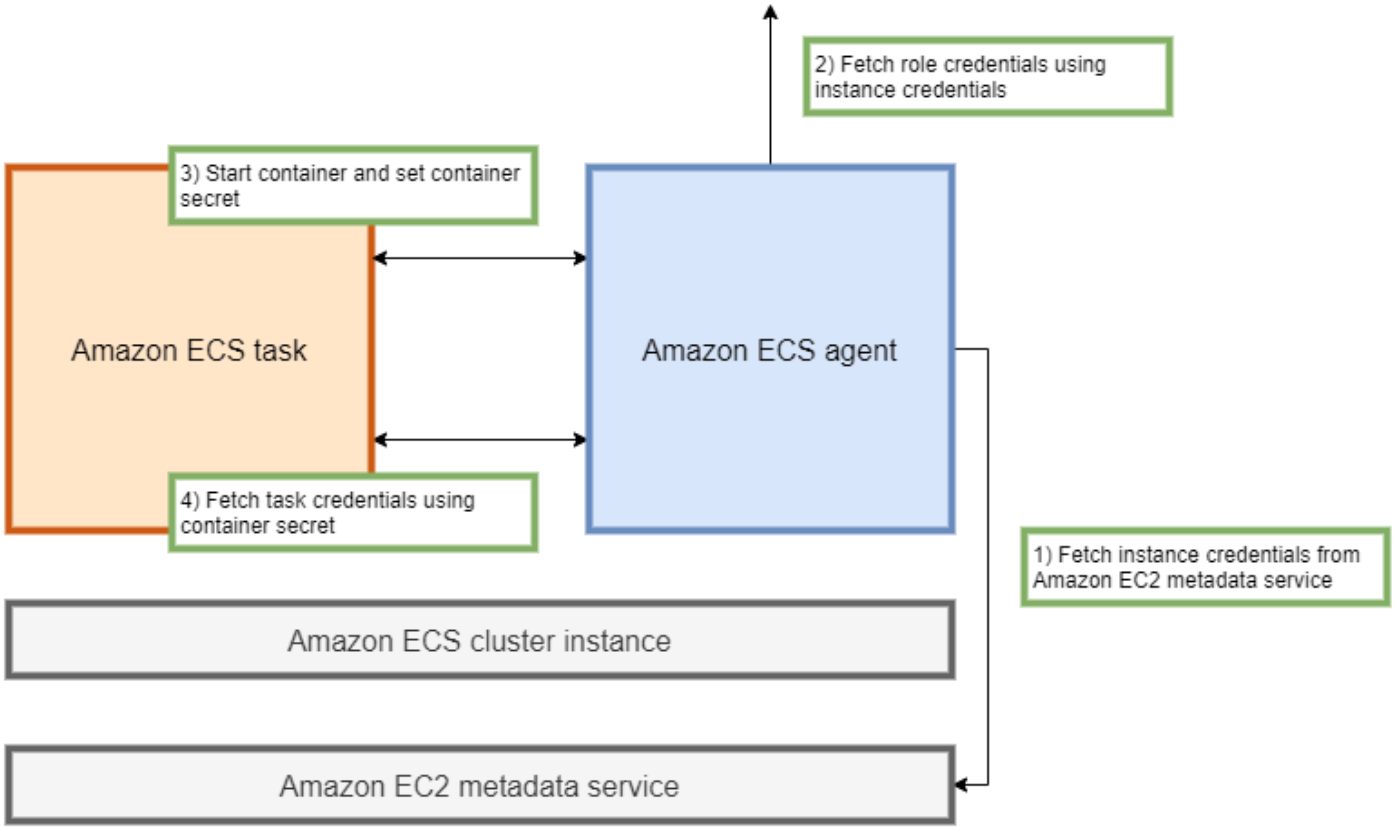
我们建议您为任务分配 IAM 角色。它的角色可以与运行它的 Amazon EC2 实例的角色区分开来。为每个任务分配一个角色符合最少权限访问的原则，并允许对操作和资源进行更精细的控制。

为任务分配 IAM 角色时，您必须使用延伸信任策略，以便您的每个任务都可以代入与 EC2 实例使用的 IAM 角色不同的 IAM 角色。这样，您的任务就不会继承 EC2 实例的角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

向任务定义添加任务角色时，Amazon ECS 容器代理会自动创建具有唯一凭据 ID 的令牌（例如12345678-90ab-cdef-1234-567890abcdef) 为任务。然后将此令牌和角色凭据添加到代理的

内部缓存中。代理填充环境变量AWS_CONTAINER_CREDENTIALS_RELATIVE_URI在容器中使用凭据 ID 的 URI（例如/v2/credentials/12345678-90ab-cdef-1234-567890abcdef）。



您可以从容器内部手动检索临时角色证书，方法是将环境变量附加到 Amazon ECS 容器代理的 IP 地址，然后运行curl命令在结果字符串上。

```
curl 192.0.2.0$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

预期的输出如下所示：

```
{
 "RoleArn": "arn:aws:iam::123456789012:role/SSMTaskRole-SSMFargateTaskIAMRole-
DASWWSF2WGD6",
 "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
 "Token": "IQoJb3JpZ2luX2VjEEM/Example==",
 "Expiration": "2021-01-16T00:51:53Z"
}
```

较新版本的AWS软件开发工具包会自动从AWS_CONTAINER_CREDENTIALS_RELATIVE_URI环境变化AWSAPI 调用。

输出包括访问密钥对，由私有访问密钥 ID 和您的应用程序用于访问的私有密密钥组成。AWS资源的费用。它还包括一个令牌AWS用于验证凭证是否有效。默认情况下，分配给使用任务角色的任务的凭据有效期为六小时。之后，Amazon ECS 容器代理会自动轮换。

任务执行角色

任务执行角色用于授予 Amazon ECS 容器代理调用特定AWS代表您执行 API 操作。例如，当您使用 AWS Fargate，Fargate 需要一个 IAM 角色，该角色允许它从亚马逊 ECR 中提取映像并将日志写入 CloudWatch Logs。当任务引用存储在AWS Secrets Manager，例如图像拉取机密。

> **Note**
>
> 如果您以经过身份验证的用户身份提取图像，则不太可能受到码头集线器的拉力限制。有关更多信息，请参阅容器实例的私有注册表身份验证。
> 通过使用亚马逊 ECR 和亚马逊 ECR 公共，您可以避免 Docker 施加的限制。如果您从 Amazon ECR 提取映像，这也有助于缩短网络提取时间，减少流量离开 VPC 时的数据传输变化。

> **Important**
>
> 当您使用 Fargate 时，您必须使用repositoryCredentials。无法设置 Amazon ECS 容器代理环境变量ECS_ENGINE_AUTH_TYPE或者ECS_ENGINE_AUTH_DATA或修改ecs.config文件，用于托管在 Fargate 上的任务。有关更多信息，请参阅 。任务的私有注册表身份验证。

Amazon EC2 容器实例角色

Amazon ECS 容器代理是一个在 Amazon ECS 集群中的每个 Amazon EC2 实例上运行的容器。它在亚马逊云服务器之外使用init命令，该命令在操作系统上可用。因此，无法通过任务角色向其授予权限。相反，必须将权限分配给代理在其上运行的 Amazon EC2 实例。示例中的操作列表AmazonEC2ContainerServiceforEC2Role策略需要授予ecsInstanceRole。如果您不执行此操作，则您的实例将无法加入集群。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:UpdateContainerInstancesState",
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

在此策略中，ecr和logsapi 操作允许在您的实例上运行的容器从 Amazon ECR 中提取映像，并将日志写入 Amazon CloudWatch。这些区域有：ecs操作允许代理注册和取消注册实例，并与 Amazon ECS 控制平面进行通信。其中，ecs:CreateCluster操作是可选的。

服务相关角色

您可以使用 Amazon ECS 的服务链接角色来授予 Amazon ECS 服务代表您调用其他服务 API 的权限。Amazon ECS 需要具有创建和删除网络接口、向目标组注册和取消注册目标的权限。它还需要创建和删除扩展策略所需的权限。这些权限通过服务相关角色授予。此角色是在您第一次使用服务时代表您创建的。

> **Note**
>
> 如果您无意中删除了服务相关角色，则可以重新创建该角色。有关说明，请参阅创建服务相关角色。

Recommendations

建议您在设置任务 IAM 角色和策略时执行以下操作。

阻止对 Amazon EC2 元数据的访问

当您在 Amazon EC2 实例上运行任务时，我们强烈建议您阻止对 Amazon EC2 元数据的访问，以防止容器继承分配给这些实例的角色。如果您的应用程序必须调用AWSAPI 操作，请使用 IAM 角色执行任务。

要防止任务运行在桥模式访问 Amazon EC2 元数据，请运行以下命令或更新实例的用户数据。有关更新实例的用户数据的更多说明，请参阅AWSSupport 文章。有关任务定义桥梁模式的更多信息，请参阅任务定义网络模式。

```
sudo yum install -y iptables-services; sudo iptables --insert FORWARD 1 --in-interface
 docker+ --destination 192.0.2.0/32 --jump DROP
```

若要在重新启动后保持此更改，请运行以下特定于您的 Amazon 系统映像 (AMI) 的命令：

- Amazon Linux 2

```
sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now
 iptables
```

- Amazon Linux

```
sudo service iptables save
```

对于使用`awsvpc`网络模式下，设置环境变量`ECS_AWSVPC_BLOCK_IMDS`到`true`中的`/etc/ecs/ecs.config`文件。

您应该设置`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST`变量`false`来防止在 ecs-agent 配置文件中运行的容器`host`网络访问 Amazon EC2 元数据。

使用**`awsvpc`**网络模式

使用网络`awsvpc`网络模式来限制不同任务之间或您的任务与 Amazon VPC 内运行的其他服务之间的流量流。这增加了一个额外的安全层。这些区域有：`awsvpc`网络模式为 Amazon EC2 上运行的任务提供了任务级网络隔离。这是AWS Fargate。它是唯一可用于将安全组分配给任务的网络模式。

使用 IAM 访问顾问优化角色

我们建议您删除任何从未使用过或一段时间未使用的操作。这可以防止发生不需要的访问。为此，请查看 IAM Access Advisor 生成的结果，然后删除从未使用过或最近未使用过的操作。您可以按照以下步骤执行此操作。

运行以下命令生成报告，显示引用策略的上次访问信息：

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

使用JobId来运行以下命令。执行此操作后，您可以查看报告的结果。

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

有关更多信息，请参阅 。IAM 访问顾问。

显示器AWS CloudTrail对于可疑活动

您可以监控AWS CloudTrail查看任何可疑活动。大多数AWSAPI 调用会记录到AWS CloudTrail作为事件。它们被分析AWS CloudTrail见解，并且您会收到任何与writeAPI 调用。这可能包括呼叫音量的峰值。这些警报包括异常活动发生的时间和导致 API 的顶级标识 ARN 等信息。

您可以识别由具有 IAM 角色的任务执行的操作AWS CloudTrail通过查看事件的userIdentity属性。在下面的示例中，arn包括承担的角色的名称，s3-write-go-bucket-role，后跟任务的名称，7e9894e088ad416eb5cab92afExample。

```
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROA36C6WWEJ2YEXAMPLE:7e9894e088ad416eb5cab92afExample",
    "arn": "arn:aws:sts::123456789012:assumed-role/s3-write-go-bucket-
role/7e9894e088ad416eb5cab92afExample",
    ...
}
```

> **Note**
>
> 当担任角色的任务在 Amazon EC2 容器实例上运行时，Amazon ECS 容器代理将请求记录到位于/var/log/ecs/audit.log.YYYY-MM-DD-HH格式的日期和时间。有关更多信息，请参阅 。任务 IAM 角色日志和记录跟踪的见解事件。

网络安全

网络安全是一个广泛的主题，包括几个子主题。其中包括传输中加密、网络分段和隔离、防火墙、流量路由和可观察性。

传输中加密

加密网络流量可防止未经授权的用户在通过网络传输数据时拦截和读取数据。借助 Amazon ECS，可以通过以下任一方式实施网络加密。

- 使用服务网格 (TLS)：

  与AWS App Mesh中，您可以在使用网格端点部署的 Express 代理之间配置 TLS 连接。两个示例是虚拟节点和虚拟网关。TLS 证书可以来自AWS Certificate Manager(ACM). 或者，它可以来自您自己的私有证书颁发机构。

  - 启用传输层安全性 (TLS)
  - 启用服务之间的流量加密AWS App Mesh使用 ACM 证书或客户提供的证书
  - TLS ACM 演练
  - TLS 文件演练
  - Envoy
- 使用硝基实例：

  默认情况下，以下 Nitro 实例类型之间的流量会自动加密：
  C5n、G4、M5dn、M5dn、M5dn、M5dn、M5dn、M5dn、M5dn、M5dn、M5dn、M5dn、M5dn、M5dn、
  当流量通过传输网关、负载均衡器或类似中介路由时，流量不会被加密。

  - 传输中加密
  - 2019 年的新会计
  - 本次演讲来自 2019 年重点：
- 对应用 Application Load Balancer 器使用服务器名称指示 (SNI)：

  应用程序负载平衡器 (ALB) 和 Network Load Balancer (NLB) 支持服务器名称指示 (SNI)。通过使用 SNI，您可以将多个安全应用程序放在一个监听器后面。为此，每个人都有自己的 TLS 证书。我们建议您使用AWS Certificate Manager(ACM)，然后将它们添加到监听程序的证书列表中。这些区域有：AWS负载均衡器使用带有 SNI 的智能证书选择算法。如果客户端提供的主机名与证书列表中的一个证书匹配，则负载均衡器将选择该证书。如果客户端提供的主机名与列表中的多个证书匹配，则负载均衡器将选择客户端可支持的证书。示例包括自签名证书或通过 ACM 生成的证书。

- SNI，带 Application Load Balancer
- SNI，带 Network Load Balancer

- 使用 TLS 证书的端到端加密：

  这涉及与任务一起部署 TLS 证书。这可以是自签名证书，也可以是来自可信证书颁发机构的证书。您可以通过引用证书的密钥来获取证书。否则，您可以选择运行向 ACM 发出证书签名请求 (CSR) 的容器，然后将生成的密钥装载到共享卷。

  - 使用具有 Amazon ECS 第 1 部分的 Network Load Balancer，从而维护您的容器的传输层安全
  - 保持传输层安全性 (TLS) 到您的容器第 2 部分：使用 AWS Private Certificate Authority

任务联网

以下建议考虑了亚马逊云服务器的工作原理。亚马逊云服务器不使用覆盖网络。相反，将任务配置为在不同的网络模式下运行。例如，配置为使用`bridge`模式从每台主机上运行的 Docker 网络获取不可路由的 IP 地址。配置为使用`awsvpc`网络模式从主机子网获取 IP 地址。配置的任务`host`网络使用主机的网络接口。`awsvpc`是首选网络模式。这是因为它是您可以用来将安全组分配给任务的唯一模式。这也是唯一可用于AWS Fargate任务。

任务的安全组

我们建议配置任务以使用`awsvpc`网络模式。将任务配置为使用此模式后，Amazon ECS 代理会自动预配置并将弹性网络接口 (ENI) 附加到任务。置备 ENI 后，任务将注册到AWS安全组。安全组充当虚拟防火墙，您可以使用它来控制入站和出站流量。

服务网格和互传输层安全 (MTL)

您可以使用服务网格，例如AWS App Mesh来控制网络流量。默认情况下，虚拟节点只能与其配置的服务后端通信，例如虚拟节点将与之通信的虚拟服务。如果虚拟节点需要与网格外部的服务进行通信，则可以使用`ALLOW_ALL`出站过滤器或在网格内为外部服务创建虚拟节点。有关更多信息，请参阅 。库贝内特人出口操作指导演练。

App Mesh 还使您能够使用相互传输层安全性 (MTL)，其中客户端和服务器都使用证书进行相互验证。然后使用 TLS 对客户端和服务器之间的后续通信进行加密。通过在网格中的服务之间要求 MTL，您可以验证流量是否来自受信任的源。有关更多信息，请参阅以下主题：

- MTL 身份验证
- MTL 秘密发现服务 (SDS) 演练

- MTL 文件演练

AWS PrivateLink

AWS PrivateLink是一种网络技术，允许您为不同的AWS服务，包括亚马逊云服务器。在没有连接到 Amazon VPC 的 Internet 网关 (IGW) 且没有通往互联网的替代路由的沙盒环境中，终端节点是必需的。使用AWS PrivateLink确保对亚马逊云服务器服务的调用保持在 Amazon VPC 内，且不会遍历互联网。有关如何创建AWS PrivateLink亚马逊云服务器和其他相关服务的终端节点，请参阅Amazon ECS 接口 Amazon VPC 终端节点。

> **Important**
>
> AWS Fargate任务不需要AWS PrivateLink终端节点。

亚马逊 ECR 和亚马逊云服务器都支持终端节点策略。这些策略允许您优化对服务 API 的访问。例如，您可以为 Amazon ECR 创建一个终端节点策略，该策略仅允许将映像推送到注册表，特别是AWS帐户。此类策略可用于防止数据通过容器映像泄露，同时仍允许用户推送到授权的 Amazon ECR 注册表。有关更多信息，请参阅 。使用 VPC 终端节点策略。

以下策略允许所有AWS以仅针对您的 Amazon ECR 存储库执行所有操作：

```
{
  "Statement": [
    {
      "Sid": "LimitECRAccess",
      "Principal": "*",
      "Action": "*",
      "Effect": "Allow",
      "Resource": "arn:aws:ecr:region:your_account_id:repository/*"
    },
  ]
}
```

您可以通过设置一个使用新PrincipalOrgID属性。这可以防止 IAM 委托人推送和拉取图像，该委托人不属于AWS Organizations。有关更多信息，请参阅 。aws:PrincipalOrgID。

我们建议将相同的策略应用于com.amazonaws.*region*.ecr.dkr和com.amazonaws.*region*.ecr.api终端节点。

Amazon ECS 容器代理设置

Amazon ECS 容器代理配置文件包含多个与网络安全相关的环境变量。`ECS_AWSVPC_BLOCK_IMDS`和`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST`用于阻止任务访问 Amazon EC2 元数据。`HTTP_PROXY`用于将代理配置为通过 HTTP 代理路由以连接到互联网。有关配置代理和 Docker 运行时以通过代理路由的说明，请参阅HTTP 代理配置。

> **Important**
>
> 这些设置在您使用AWS Fargate。

Recommendations

我们建议您在设置 Amazon VPC、负载均衡器和网络时执行以下操作。

在适用情况下使用网络加密

您应在适用的情况下使用网络加密。如果数据包含持卡人数据，某些合规性程序（例如 PCI DSS）要求您对传输中的数据进行加密。如果您的工作负载具有类似要求，请配置网络加密。

当连接到不安全的站点时，现代浏览器会警告用户。如果您的服务由面向公共的负载均衡器提供，请使用 TLS/SSL 加密从客户端浏览器到负载均衡器的流量，并在有保证的情况下重新加密到后端。

使用**awsvpc**当您需要控制任务之间或任务与其他网络资源之间的流量时，网络模式和安全组

您应使用`awsvpc`网络模式和安全组时，您需要控制任务之间以及任务与其他网络资源之间的流量。如果您的服务位于 ALB 之后，则使用安全组仅允许来自其他网络资源的入站流量与 ALB 相同的安全组。如果您的应用程序位于 NLB 后面，请将任务的安全组配置为仅允许来自 Amazon VPC CIDR 范围和分配给 NLB 的静态 IP 地址的入站流量。

安全组还应用于控制 Amazon VPC 中的任务与其他资源（如 Amazon RDS 数据库）之间的流量。

当网络流量需要严格隔离时，在单独的 Amazon VPC 中创建集群

当网络流量需要严格隔离时，您应在单独的 Amazon VPC 中创建集群。避免在工作负载不必遵守这些要求的群集上运行具有严格安全要求的工作负载。如果强制执行严格的网络隔离，请在单独的 Amazon VPC 中创建集群，并使用 Amazon VPC 终端节点选择性地将服务公开给其他 Amazon VPC。有关更多信息，请参阅 。Amazon VPC 终端节点。

配置AWS PrivateLink终端节点（如果有保证）

您应配置AWS PrivateLink终端节点（如果有保证）。如果您的安全策略阻止您将 Internet Gateway (IGW) 连接到您的 Amazon VPC，请配置AWS PrivateLink亚马逊云服务器和其他服务的终端节点，例如亚马逊 ECR、AWS Secrets Manager和 Amazon CloudWatch。

使用 Amazon VPC 流日志分析进出长时间运行的任务的流量

您应该使用 Amazon VPC 流日志来分析进出长时间运行的任务的流量。使用的任务`awsvpc`网络模式获取自己的 ENI。执行此操作，您可以使用 Amazon VPC 流日志监控进出单个任务的流量。最近对 Amazon VPC 流日志 (v3) 的更新使用包括 VPC ID、子网 ID 和实例 ID 在内的流量元数据丰富了日志。此元数据可用于帮助缩小调查范围。有关更多信息，请参阅 。Amazon VPC 流日志。

> **Note**
>
> 由于容器的临时性质，流日志可能并不总是分析不同容器或容器与其他网络资源之间的流量模式的有效方法。

密钥管理

应用程序经常使用 API 密钥和数据库凭据等密钥来访问其他系统。它们通常由用户名和密码、证书或 API 密钥组成。对这些密钥的访问应限制在使用 IAM 并在运行时注入到容器中的特定 IAM 委托人。

秘密可以无缝地注入容器AWS Secrets Manager和 Amazon EC2 Systems Manager Parameter Store。这些秘密可以在您的任务中作为以下任何一项引用。

1. 它们被引用为环境变量，它们使用`secrets`容器定义参数。
2. 它们被引用为`secretOptions`如果您的日志记录平台需要验证。有关更多信息，请参阅 。日志配置选项。
3. 它们被引用为使用`repositoryCredentials`容器定义参数，如果从中提取容器的注册表需要身份验证。从 Docker Hub 提取图像时，请使用此方法。有关更多信息，请参阅 。任务的私有注册表身份验证。

Recommendations

我们建议您在设置机密管理时执行以下操作。

使用AWS Secrets Manager或 Amazon EC2 Systems Manager Parameter Store，用于存储密钥材料

您应该将 API 密钥、数据库凭据和其他秘密材料安全地存储在AWS Secrets Manager或 Amazon EC2 Systems Manager Parameter Store 中的加密参数。这些服务是相似的，因为它们都是使用AWS KMS来加密敏感数据。AWS Secrets Manager，但是，还包括自动旋转机密、生成随机机密以及跨AWS帐户。如果您认为这些重要功能，请使用AWS Secrets Manager否则使用加密参数。

> **Note**
>
> 引用来自密钥的任务AWS Secrets Manager或 Amazon EC2 Systems Manager Parameter Store 中的任务执行角色的策略，该策略允许 Amazon ECS 访问所需密钥，如果适用，还允许AWS KMS密钥用于加密和解密该密钥。

> **Important**
>
> 任务中引用的密文不会自动轮换。如果您的密钥发生更改，则必须强制执行新部署或启动新任务以检索最新的密钥值。有关更多信息，请参阅以下主题：
>
> - AWS Secrets Manager：将数据作为环境变量注入
> - Amazon EC2 Systems Manager Parameter Store：将数据作为环境变量注入

从加密的 Amazon S3 存储桶中检索数据

因为环境变量的值可能会无意中泄漏到日志中，并在运行`docker inspect`，您应将密钥存储在加密的 Amazon S3 存储桶中，并使用任务角色限制对这些密钥的访问。执行此操作时，必须写入您的应用程序才能从 Amazon S3 存储桶读取密钥。有关说明，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

使用侧车容器将秘密装入卷

由于环境变量的数据泄漏风险较高，因此您应该运行一个侧车容器，从AWS Secrets Manager并将其写入共享卷。此容器可以在应用程序容器之前运行和退出，方法是使用Amazon ECS 容器订购。执行此操作时，应用程序容器随后挂载到写入密钥的卷。与 Amazon S3 存储桶方法一样，您的应用程序必须写入才能从共享卷读取密钥。由于卷的作用域为任务，因此在任务停止后将自动删除该卷。有关边车集装箱的示例，请参阅AWS 秘密侧侧侧注射器项目。

> **Note**
>
> 在 Amazon EC2 上，向其写入密钥的卷可以使用AWS KMS客户托管密钥。在 上AWS Fargate，则使用服务托管密钥自动加密卷存储。

其他资源

- 将密钥传递到 Amazon ECS 任务中的容器
- 分庭是在 Amazon EC2 Systems Manager Parameter Store 中存储密密的包装器

Compliance

您在使用 Amazon ECS 时的合规性责任由您数据的敏感性、您公司的合规性目标以及适用的法律法规决定。

AWS提供以下资源来帮助实现合规性：

- 安全性和合规性快速入门指南：这些部署指南讨论了架构注意事项，并提供了在AWS。
- HIPAA 安全性和合规性架构设计白皮书：本白皮书介绍了公司如何使用AWS来创建符合 HIPAA 要求的应用程序。
- AWS合规性计划范围内的服务：此列表包含AWS服务范围内的特定合规性计划。有关更多信息，请参阅 。AWS合规性计划。

支付卡行业数据安全标准 (PCI DSS)

在遵守 PCI DSS 时，您必须了解环境中持卡人数据 (CHD) 的完整流程。CHD 流程决定 PCI DSS 的适用性，定义持卡人数据环境 (CDE) 的边界和组件，以及 PCI DSS 评估的范围。准确确定 PCI DSS 范围是定义安全状态和最终成功评估的关键。客户必须有一个确定范围的程序，以确保其完整性并检测范围的变化或偏离。

容器化应用程序的临时性质在审核配置时提供了额外的复杂性。因此，客户需要保持对所有容器配置参数的了解，以确保在容器生命周期的所有阶段都能满足法规遵从性要求。

有关在亚马逊云服务器上实现 PCI DSS 合规性的更多信息，请参阅以下白皮书。

- 在 Amazon ECS 上构建 PCI DSS 合规性

- 用于 PCI DSS 范围划分和分段的架构 AWS

HIPAA（美国 Health 保险流通与责任法案）

将 Amazon ECS 与处理受保护的运行状况信息 (PHI) 的工作负载一起使用，无需进行其他配置。Amazon ECS 充当协调服务，用于协调 Amazon EC2 上容器的启动。它不会对正在协调的工作负载中的数据进行操作，也不会对其进行操作。符合 HIPAA 法规和AWS业务助理附录：如果通过 Amazon ECS 启动的容器访问，则应在运输过程中和静态 PHI 进行加密。

各种静态加密机制可用于每个AWS存储选项，例如 Amazon S3、亚马逊 EBS 和AWS KMS。您可以部署覆盖网络（如 VNS3 或 Weave Net），以确保对容器之间传输的 PHI 进行完全加密，或提供冗余的加密层。还应启用完整日志记录，并且所有容器日志都应定向到 Amazon CloudWatch。有关更多信息，请参阅 。HIPAA 安全性和合规性架构设计。

Recommendations

您应尽早与企业内的合规性计划所有者接触，并使用AWS责任共担模式确定合规性控制所有权，以便成功完成相关合规性计划。

日志记录和监控

日志记录和监控是保持 Amazon ECS 和您的AWS解决方案。AWS提供了多种工具来监控 Amazon ECS 资源并对潜在的事件做出响应：

- Amazon CloudWatch 警报
- Amazon CloudWatch Logs
- Amazon CloudWatch Events
- AWS CloudTrail 日志

您可在任务中配置容器以将日志信息发送到 Amazon CloudWatch Logs。如果您使用的是AWS Fargate启动类型，您可以从容器查看日志。如果使用 Amazon EC2 启动类型，您可在一个方便的位置查看容器中的不同日志。这也防止您的容器日志占用您容器实例上的磁盘空间。

有关 Amazon CloudWatch Logs 的更多信息，请参阅监控 Amazon EC2 实例中的日志中的Amazon CloudWatch 用户指南。有关将任务的容器日志发送到 Amazon CloudWatch Logs 的说明，请参阅使用awslogs日志驱动程序。

使用流利位进行容器日志记录

AWS提供了 Fluent Bit 映像以及用于 Amazon CloudWatch Logs 和 Amazon Kinesis Data Firehose 的插件。此图像提供了将日志路由到 Amazon CloudWatch 和 Amazon Kinesis Data Firehose 目标（包括 Amazon S3、Amazon Elasticsearch Service 和 Amazon Redshift）的功能。我们建议使用 Fluent Bit 作为日志路由器，因为其资源利用率低于 Fluentd。有关更多信息，请参阅 。流利位的 Amazon CloudWatch Logs和Amazon Kinesis Data Firehose 适用于流利位。

这些区域有：AWS对于流利位图像，请访问：

- 亚马逊 ECR 公共展览馆上的亚马逊 ECR
- Amazon ECR 存储库（在大多数高可用性区域）
- Docker Hub

以下显示 Docker CLI 使用的语法。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

例如，您可以将最新的AWS使用此 Docker CLI 命令进行流利位映像：

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:latest
```

有关 Fluent Bit 和相关功能的更多信息，请参阅以下博客文章：

- 亚马逊 EKS 的流利位 AWS Fargate
- 使用流量位的集中式容器记录
- 构建可扩展的日志解决方案聚合器AWS Fargate、Fluentd 和 Amazon Kinesis Data Firehose

自定义日志路由-适用于亚马逊弹性云服务器的 FireLens

通过适用于 Amazon 弹性云服务器的 FireLens，您可以使用任务定义参数将日志路由到AWS服务或AWS用于日志存储和分析的合作伙伴网络 (APN) 目标。FireLens 与 Fluentd 和 Fluent Bit 配合使用。我们提供AWS用于流利位图像。或者，您可以使用自己的 Fluentd 或 Fluentd 或 Fluentd Bit 图像。

在为亚马逊云服务器使用 FireLens 时，您应考虑以下条件和注意事项：

- 支持亚马逊弹性云服务器的 FireLens 用于托管在AWS Fargate和 Amazon EC2。

- Amazon ECS 的 FireLens 在AWS CloudFormation模板。有关更多信息，请参阅 。AWS::ECS::TaskDefinition FirelensConfiguration中的AWS CloudFormation用户指南。
- 对于使用bridge网络模式下，具有 FireLens 配置的容器必须在依赖它的任一应用程序容器启动之前启动。要控制容器启动的顺序，请在任务定义中使用依赖条件。有关更多信息，请参阅 。容器依赖项。

AWS Fargate 安全性

建议您在使用时考虑以下最佳实践。AWS Fargate。

使用AWS KMS对临时存储进行加密

你应该让你的临时存储通过AWS KMS。对于托管在AWS Fargate使用平台版本1.4.0或更高版本，每个任务都会收到 20 GB 的短暂存储。存储容量不可调整。对于这些在 2020 年 5 月 28 日或之后启动的任务，将使用 AES-256 加密算法通过托管加密密钥来加密短暂存储空间。AWS Fargate。

示例：在上启动 Amazon ECS 任务AWS Fargate平台版本 1.4.0，带有短暂存储加密

以下命令将启动 Amazon ECS 任务。AWS Fargate平台版本 1.4。由于此任务是作为 Amazon ECS 群集的一部分启动的，因此它使用自动加密的 20 GB 临时存储。

```
aws ecs run-task --cluster clustername \
  --task-definition taskdefinition:version \
  --count 1
  --launch-type "FARGATE" \
  --platform-version 1.4.0 \
  --network-configuration
 "awsvpcConfiguration={subnets=[subnetid],securityGroups=[securitygroupid]}" \
  --region region
```

用于内核系统调用跟踪的 SYS_PTRACE 功能

添加或从容器中删除的 Linux 功能的默认配置由 Docker 提供。有关可用功能的更多信息，请参阅运行时权限和 Linux 功能中的Docker 运行文档中）。

在上启动的任务AWS Fargate仅支持添加SYS_PTRACE内核功能。

请参阅下面的教程视频，该视频展示了如何通过 Sysdig 使用此功能法尔科项目。

#ContainersFromTheCouch-排查AWS Fargate使用 SYS_PTRACE 功能的任务

前面的视频中讨论的代码可以在 GitHub 上找到此处。

任务和容器安全

您应将容器映像视为抵御攻击的第一道防线。不安全、构建不良的映像可以让攻击者逃避容器的边界并获得对主机的访问权限。您应该执行以下操作来降低发生这种情况的风险。

Recommendations

建议您在设置任务和容器时执行以下操作。

创建最小的图像或使用无仿真图像

首先从容器映像中删除所有无关的二进制文件。如果您使用的是来自 Docker Hub 的不熟悉图像，请检查图像以引用每个容器图层的内容。您可以使用一个应用程序，如潜水执行此操作。

或者，您也可以使用无二探映像，只包含您的应用程序及其运行时依赖关系。它们不包含软件包管理器或 shell。无仿真图像改善了 “扫描仪的噪音信号，减轻了建立原点的负担，达到您所需要的目的”。有关更多信息，请参阅无二探。

Docker 有一个从保留的最小图像创建图像的机制，称为划痕。Forore 信息，请参阅创建一个简单的父映像划痕Docker 文档中。使用像 Go 这样的语言，您可以创建一个静态链接二进制文件并在 Docker 文件中引用它。以下示例演示如何实现此目标。

```
############################
# STEP 1 build executable binary
############################
FROM golang:alpine AS builder
# Install git.
# Git is required for fetching the dependencies.
RUN apk update && apk add --no-cache git
WORKDIR $GOPATH/src/mypackage/myapp/
COPY . .
# Fetch dependencies.
# Using go get.
RUN go get -d -v
# Build the binary.
RUN go build -o /go/bin/hello
############################
# STEP 2 build a small image
############################
```

```
FROM scratch
# Copy our static executable.
COPY --from=builder /go/bin/hello /go/bin/hello
# Run the hello binary.
ENTRYPOINT ["/go/bin/hello"]
This creates a container image that consists of your application and nothing else,
 making it extremely secure.
```

前面的示例也是多阶段构建的一个示例。从安全角度来看，这些类型的构建非常有吸引力，因为您可以使用它们最大限度地减少推送到容器注册表的最终映像的大小。没有构建工具和其他无关二进制文件的容器映像通过减少映像的攻击面来改善您的安全状态。有关多阶段构建的更多信息，请参阅创建多阶段构建。

扫描图像中的漏洞

与其虚拟机对应方类似，容器映像可能包含存在漏洞的二进制文件和应用程序库，或者随着时间的推移会出现漏洞。防止漏洞攻击的最佳方法是使用图像扫描仪定期扫描图像。存储在 Amazon ECR 中的图片可按推送或按需扫描（每 24 小时一次）。Amazon ECR 目前使用克莱尔，这是一个开源图像扫描解决方案。扫描图像后，结果将记录到亚马逊 EventBridge 中的亚马逊 ECR 事件流中。您还可以从 Amazon ECR 控制台中查看扫描结果，也可以通过调用描述图像查找结果API。具有`HIGH`或者`CRITICAL`漏洞应删除或重新构建。如果已部署的映像出现漏洞，则应尽快替换该漏洞。

Docker 桌面边缘版本 2.3.6.0或更高版本可以scan本地映像。扫描由斯奈克，一种应用程序安全服务。当发现漏洞时，Snyk 会识别与 Docker 文件中的漏洞的层和依赖关系。它还推荐安全的替代方案，例如使用更薄的基础映像，漏洞较少，或者将特定软件包升级到较新版本。通过使用 Docker 扫描，开发人员可以在将映像推送到注册表之前解决潜在的安全问题。

- 使用亚马逊 ECR 和AWS Security Hub介绍了如何显示来自 Amazon ECR 的漏洞信息AWS Security Hub，并通过阻止访问易受攻击的映像来自动修复。

从您的图像中删除特殊权限

访问权限标志`setuid`和`setgid`允许使用可执行文件的所有者或组的权限运行可执行文件。从映像中删除所有具有这些访问权限的二进制文件，因为这些二进制文件可用于提升权限。考虑删除所有 shell 和实用程序，如`nc`和`curl`，可用于恶意目的。您可以使用来找到文件`setuid`和`setgid`使用以下命令的访问权限。

```
find / -perm /6000 -type f -exec ls -ld {} \;
```

要从这些文件中删除这些特殊权限，请将以下指令添加到容器映像中。

```
RUN find / -xdev -perm /6000 -type f -exec chmod a-s {} \; || true
```

创建一组精心策划的图像

而不是允许开发人员创建自己的映像，而是为组织中的不同应用程序堆栈创建一组经过审查的图像。通过这样做，开发人员可以放弃学习如何撰写 Docker 文件，并专注于编写代码。随着更改合并到您的代码库中，CI/CD 管道可以自动编译资源，然后将其存储在工件存储库中。最后，将工件复制到相应的图像中，然后再将其推送到 Docker 注册表（如 Amazon ECR）。至少你应该创建一组基础映像帽子开发人员可以从创建自己的 Docker 文件。您应该避免从 Docker Hub 提取图像。您并不总是知道图像中的内容，而且前 1000 张图像中约有五分之一存在漏洞。有关这些映像及其漏洞的列表，请参阅https://vulnerablecontainers.org/。

扫描应用程序包和库中的漏洞

开放源代码库的使用现在很普遍。与操作系统和操作系统软件包一样，这些库可能存在漏洞。作为开发生命周期的一部分，应在发现关键漏洞时扫描和更新这些库。

Docker 桌面使用 Snyk 执行本地扫描。它还可用于查找开源库中的漏洞和潜在的许可问题。它可以直接集成到开发人员工作流程中，使您能够降低开源库带来的风险。有关更多信息，请参阅以下主题：

- 开源应用程序安全工具包含用于检测应用程序漏洞的工具列表。
- Docker 扫描作弊表

执行静态代码分析

您应该在构建容器映像之前执行静态代码分析。它针对您的源代码执行，并用于识别可能被恶意操作者利用的编码错误和代码（如故障注入）。SonarQube是静态应用安全测试 (SAST) 的流行选项，支持各种不同的编程语言。

以非 root 用户身份运行容器

您应该以非 root 用户身份运行容器。默认情况下，容器作为`root`用户，除非`USER`指令包含在您的码头文件中。Docker 分配的默认 Linux 功能限制可以作为`root`，但只是轻微的。例如，运行为`root`仍然不允许访问设备。

作为 CI/CD 管道的一部分，您应该让 Docker 文件查找`USER`指令，并且如果缺少构建失败。有关更多信息，请参阅以下主题：

- Dockerfile-林特是 RedHat 的一个开源工具，可用于检查文件是否符合最佳实践。
- 哈多林是构建符合最佳实践的 Docker 映像的另一个工具。

使用只读根文件系统

您应该使用只读根文件系统。默认情况下，容器的根文件系统是可写的。当您将容器配置为RO（只读）根文件系统，它会强制您明确定义数据可以保留的位置。这会减少您的攻击面，因为除非特别授予权限，否则容器的文件系统无法写入。

> **Note**
>
> 具有只读根文件系统可能会导致某些预期能够写入文件系统的操作系统包出现问题。如果您计划使用只读根文件系统，请事先进行彻底测试。

配置具有 CPU 和内存限制的任务（Amazon EC2）

您应配置具有 CPU 和内存限制的任务，以最大限度地降低以下风险。任务的资源限制为任务中的所有容器可以预留的 CPU 和内存量设置上限。如果未设置限制，则任务可以访问主机的 CPU 和内存。这可能会导致部署在共享主机上的任务可能会使系统资源的其他任务缺乏的问题。

> **Note**
>
> 上的 Amazon ECSAWS Fargate任务要求您指定 CPU 和内存限制，因为它将这些值用于计费目的。一个占用所有系统资源的任务不是 Amazon ECS Fargate 的问题，因为每个任务都在自己的专用实例上运行。如果您没有指定内存限制，Amazon ECS 会为每个容器分配至少 4MB。同样，如果没有为任务设置 CPU 限制，Amazon ECS 容器代理将至少分配 2 个 CPU。

将不可变标签与 Amazon ECR 结合使用

借助 Amazon ECR，您可以而且应该使用带有不可变标签的映像配置。这样可以防止将更改或更新的图像版本推送到具有相同标签的映像存储库。这可以防止攻击者将图像的受损版本推送到具有相同标签的映像上。通过使用不可变标签，您可以有效地强制自己为每次更改推送带有不同标签的新图像。

避免以特权方式运行容器（Amazon EC2）

您应该避免将容器作为特权运行。对于后台，容器作为`privileged`在主机上使用扩展权限运行。这意味着容器将继承分配给`root`在主机上。它的使用应该受到严格限制或禁止。我们建议设置

Amazon ECS 容器代理环境变量ECS_DISABLE_PRIVILEGED到true来防止容器作为privileged在特定主机上，如果privileged不需要。或者，您也可以使用AWS Lambda以扫描任务定义以使用privileged参数。

> **Note**
>
> 将容器运行为privileged上的 Amazon ECS 不支持AWS Fargate。

从容器中删除不必要的 Linux 功能

以下是分配给 Docker 容器的默认 Linux 功能的列表。有关各项功能的更多信息，请参阅Linux 功能概述。

```
CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_FOWNER, CAP_FSETID, CAP_KILL,
CAP_SETGID, CAP_SETUID, CAP_SETPCAP, CAP_NET_BIND_SERVICE,
CAP_NET_RAW, CAP_SYS_CHROOT, CAP_MKNOD, CAP_AUDIT_WRITE,
CAP_SETFCAP
```

如果一个容器不需要上面列出的所有 Docker 内核能力，请考虑将它们从容器中删除。有关各 Docker 内核功能的更多信息，请参阅内核功能。通过以下操作，您可以找出正在使用的功能：

- 安装操作软件包解放并运行pscap实用程序列出每个进程正在使用的功能。
- 您还可以使用Capsh来破译进程正在使用哪些功能。
- 请参阅Linux 功能了解更多信息。

使用客户托管密钥 (CMK) 对推送到 Amazon ECR 的映像加密

您应使用客户管理密钥 (CMK) 来强化推送到 Amazon ECR 的图片。推送到 Amazon ECR 的图像会自动静态加密，并使用AWS Key Management Service(AWS KMS) 托管密钥。如果您希望使用自己的密钥，Amazon ECR 现在支持AWS KMS使用客户管理密钥 (CMK) 进行加密。在使用 CMK 启用服务器端加密之前，请查看静态加密。

运行时安全

运行时安全性在您的容器运行时提供了主动保护。这个想法是检测和防止恶意活动发生你的容器。

使用安全计算 (seccomp) ，您可以防止容器化应用程序对底层主机操作系统的内核进行某些系统调用。虽然 Linux 操作系统有几百个系统调用，但其中大多数不是运行容器所必需的。通过限制容器可以进行哪些系统调用，可以有效地减少应用程序的攻击面。

要开始使用 seccomp，您可以使用`strace`来生成堆栈跟踪以查看您的应用程序正在进行的系统调用。您可以使用诸如`syscall2seccomp`以根据从堆栈跟踪收集的数据创建 seccomp 配置文件。有关更多信息，请参阅 。斯特勒和系统呼叫 2 分钟。

与 SELinux 安全模块不同，seccomp 无法将容器彼此隔离。但是，它可以保护主机内核免受未经授权的系统调用。它通过拦截系统调用功能，并且只允许允许列出的系统调用通过。Docker 有一个default (默认)seccomp 配置文件，适用于大多数通用工作负载。

> **Note**
>
> 也可以为需要额外权限的事物创建自己的配置文件。

AppArmor 是一个类似 seccomp 的 Linux 安全模块，但它限制了容器的功能，包括访问文件系统的部分。它可以在`enforcement`或者`complain`模式。由于构建 AppArmor 配置文件可能具有挑战性，因此，我们建议您使用涡根。有关 AppArmor 的更多信息，请参阅官方网站AppArmor页.

> **Important**
>
> 仅适用于 Ubuntu 和 Debian 的 Linux 发行版。

Recommendations

我们建议您在设置运行时安全性时采取下列操作。

使用第三方解决方案进行运行时防御

使用第三方解决方案进行运行时防御。如果您熟悉 Linux 安全性的工作原理，请创建和管理 seccomp 和 AppArmor 配置文件。两者都是开源项目。否则，请考虑使用其他第三方服务。大多数使用机器学习来阻止或警报可疑活动。有关可用的第三方解决方案的列表，请参阅AWS Marketplace用于容器的。

使用分析策略添加或删除 Linux 功能

使用 seccomp 可以更好地控制 Linux 功能并避免系统调用检查错误。Seccomp 作为一个系统调用过滤器，可撤消运行某些系统调用或使用特定调用的权限。

AWS合作伙伴

您可以使用以下任意AWS合作伙伴产品，为您的 Amazon ECS 工作负载增加额外的安全性和功能。有关更多信息，请参阅 。Amazon ECS 合作伙伴。

Aqua 安全

您可以使用Aqua 安全来保护从开发到生产的云原生应用程序。Aqua 云原生安全平台与您的云原生资源和编排工具集成，提供透明和自动化的安全性。它可以实时防止可疑活动和攻击，并有助于实施策略和简化法规遵从性。

Palo Alto Networks

Palo Alto Networks为云中以及整个开发和软件生命周期中的主机、容器和无服务器基础设施提供安全性和保护。

Twistlock 由帕洛阿尔托网络提供，可与亚马逊 ECS FireLens 集成。有了它，您可以访问高保真安全日志和事件，这些日志和事件无缝聚合到多个AWS服务。这些服务包括 Amazon CloudWatch、Amazon Athena 和 Amazon Kinesis。Twistlock 可保护部署在AWS容器服务。

系统迪格

您可以使用系统迪格在生产场景中运行安全且合规的云原生工作负载。Sysdig Secure DevOps 平台具有嵌入式安全和合规性功能，可保护您的云原生工作负载，并提供企业级可扩展性、性能和自定义。

亚马逊云服务器最佳实践指南的文档历史记录

下表介绍了 Amazon ECS 最佳实践指南的文档版本。

更新-历史记录-更改	更新-历史记录-描述	更新-历史记录-日期
安全最佳实践	添加了针对 Amazon ECS 工作负载的安全管理的最佳实践。	2021 年 5 月 26 日
自动扩展和容量管理最佳实践	添加了针对 Amazon ECS 工作负载的自动扩展和容量管理的最佳实践。	2021 年 5 月 14 日
持久性存储最佳实践	添加了针对 Amazon ECS 工作负载的持久存储的最佳实践。	2021 年 5 月 7 日
网络最佳实践	添加了针对 Amazon ECS 工作负载的网络管理的最佳实践。	2021 年 4 月 6 日
首次发布	首次发布亚马逊云服务器最佳实践指南	2021 年 4 月 6 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。